

GAttrinfo/mgatinf

int32 GAttrinfo(int32 *[ri, gr]_id*, int32 *attr_index*, char **name*, int32 **data_type*, int32 **length*)

<i>[ri, gr]_id</i>	IN:	Raster image or GR dataset identifier of the target object
<i>attr_index</i>	IN:	Index of the attribute
<i>name</i>	OUT:	Buffer for the name of the attribute
<i>data_type</i>	OUT:	Data type of the attribute
<i>length</i>	OUT:	Length of the attribute

Purpose Reads the specified attribute's data type and length.

Return value Returns `SUCCEED` (or 0) if successful and `FAIL` (or -1) otherwise.

Example This example illustrates the use of **GAttrinfo** in reading the attributes of a general raster image:

```
#include "hdf.h"

int32 ri_id, gr_id, stat, attr_index;
int32 *data_type, length;
int32 attr_values[2];
char name[MAX_GR_NAME];
...
ri_id = GRselect(gr_id, 0);
stat = GAttrinfo(gr_id, attr_index, name, data_type,
                 length);
stat = GRgetattr(ri_id, attr_index, attr_values)
...
```

FORTRAN

```
integer function mgatinf([ri, gr]_id, attr_index,
                        name, data_type, length)

integer [ri, gr]_id, data_type, num_attrs, length
character* (*) name
```

GRcreate/mgcreat

```
int32 GRcreate(int32 gr_id, const char *name, int32 ncomp, int32 data_type, int32
               interlace_mode, int32 dim_sizes[2])
```

<i>gr_id</i>	IN:	General raster interface identifier returned by GRstart
<i>name</i>	IN:	Name of the image to be created
<i>ncomp</i>	IN:	Number of components in each element of the image
<i>data_type</i>	IN:	Type of the image data
<i>interlace_mode</i>	IN:	Interlace mode of the image data
<i>dim_sizes</i>	IN:	Size of each dimension of the image

Purpose Creates a raster image via the general raster image interface.

Return value Returns a general raster image identifier if successful and `FAIL` (or `-1`) otherwise.

Example This example illustrates the use of **GRcreate**:

```
#include "hdf.h"

int32 gr_id, ri_id, file_id, stat;
char *name = "Image name";
int32 ncomp = 2;
int32 interlace_mode = MFGR_INTERLACE_PIXEL;
int32 data_type = DFNT_UINT16;

file_id = Hopen("myfile", DFACC_WRITE, 0);
gr_id = GRstart(file_id);
ri_id = GRcreate(gr_id, name, ncomp, data_type,
                interlace_mode, dim_sizes);
...
stat = Grendaccess(ri_id);
stat = Grend(gr_id);
Hclose(file_id);
```

FORTTRAN

```
integer function mgcreat(gr_id, name, ncomp, data_type
                        interlace_mode, dim_sizes)

integer gr_id, data_type, interlace_mode, dim_sizes(2)
character* (*) name
```

GRend/mgend

GRend/mgend

intn GRend(int32 *gr_id*)

gr_id IN: General raster interface identifier returned by **GRstart**.

Purpose Terminates the general raster interface session.

Return value Returns `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description This routine is used with the **GRstart** routine to define the extent of a general raster interface session. As with the start routines in the other interfaces, **GRend** disposes of the internal structures used in the remaining GR routines. Use the general purpose routines **Hopen** and **Hclose** to manage file access because the GR routines will not open and close HDF files.

Example This example illustrates the use of **GRstart** and **GRend** in initializing and terminating a GR interface session.

```
#include "hdf.h"
int32 gr_id, file_id, stat;

file_id = Hopen("myfile", DFACC_WRITE, 0);
gr_id = GRstart(file_id);
...
stat = GRend(gr_id);
Hclose(file_id);
```

FORTRAN integer function mgend(gr_id)

 integer gr_id

GRenderaccess/mgendac

intn GRenderaccess(int32 *ri_id*)

ri_id IN: General raster image identifier returned by **GRcreate** or **GRselect**

Purpose Terminates access to an general raster image.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description There should be one call to **GRenderaccess** for every *ri_id* returned from **GRselect** and/or **GRcreate**. Attempts to access *ri_id* after the call to **GRenderaccess** will result in an error condition.

Example This example illustrates the use of **GRenderaccess**:

```
#include "hdf.h"

int32 gr_id, ri_id, file_id, stat;
char *name = "Image name";
int32 ncomp = 2;
int32 interlace_mode = MFGR_INTERLACE_PIXEL;
int32 data_type = DFNT_UINT16;
int32 dim_sizes[2];

file_id = Hopen("myfile", DFACC_WRITE, 0);
gr_id = GRstart(file_id);
ri_id = GRcreate(gr_id, name, ncomp, data_type,
                interlace_mode, dim_sizes);

...
stat = GRenderaccess(ri_id);
stat = GRender(gr_id);
Hclose(file_id);
```

FORTTRAN integer function mgendac(ri_id)

 integer ri_id

GRfileinfo/mgfinfo

intn GRfileinfo(int32 *gr_id*, int32 **n_datasets*, int32 **n_file_attrs*)

gr_id IN: General raster interface identifier returned by **GRstart**

n_datasets OUT: Number of datasets in the file

n_file_attrs OUT: Number of global attributes in the file

Purpose Reports general information about the number of datasets and global attributes for the GR interface.

Return value Returns `SUCCEED` (or 0) if successful and `FAIL` (or -1) otherwise.

Description This routine is generally used to find the range of acceptable indices for **GRselect** calls.

Example This example illustrates the use of **GRfileinfo** to search through the general raster image datasets in a file:

```
#include "hdf.h"

int32 gr_id, n_datasets, n_file_attrs, gr_index;

...
stat = GRgetfileinfo(gr_id, &n_datasets, &n_file_attrs);
for(gr_index = 0; gr_index < n_datasets; gr_index++) {
    ri_id = GRselect(gr_id, gr_index);
    ...
}
```

FORTRAN integer function mgfinfo(gr_id, n_datasets, n_file_attrs)

integer gr_id, n_datasets, n_file_attrs

GRfindattr/mgfndat

int32 GRfindattr(int32 *[ri, gr]_id*, const char **attr_name*)

[ri, gr]_id IN: Identifier of the general raster image or the GR interface identifier returned by **GRstart**

attr_name IN: Name of the attribute to be searched for

Purpose Returns the index of the attribute with the specified name.

Return value Returns the index of the attribute if successful and `FAIL` (or `-1`) otherwise.

FORTTRAN integer function mgfndat(*[ri, gr]_id*, *attr_name*)

integer *[ri, gr]_id*
character* (*) *attr_name*

GRgetattr/mggnatt/mggcatt

GRgetattr/mggnatt/mggcatt

intn GRgetattr(int32 *[ri, gr]_id*, int32 *attr_index*, VOIDP *values*)

<i>[ri, gr]_id</i>	IN:	Identifier of the general raster image or the GR interface identifier returned by GRstart
<i>attr_index</i>	IN:	Index of the attribute(s)
<i>values</i>	OUT:	Buffer for the attribute values

Purpose Reads an attribute of a raster image, or all raster images, into a buffer.

Return value Returns `SUCCEED` (or 0) if successful and `FAIL` (or -1) otherwise.

Description **GRgetattr** is often used in conjunction with **GRselect** and **GRnametoindex**, or **GRselect** and **GRreftoindex**, to retrieve the attributes of a specific general raster image. If a *gr_id* is specified as the first parameter, the global attributes (applied to all general raster images in the file) are retrieved.

It is not possible to read a subset of the attribute values assigned to the target object with **GRgetattr** - all values will be read.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mggnatt**) and the other for buffered character data (**mggcatt**).

Example This example illustrates the use of **GRgetattr** to read the attributes of a general raster image:

```
#include "hdf.h"

int32 ri_id, gr_id, stat, attr_index;
int32 attr_values[2];
...
ri_id = GRselect(gr_id, 0);
attr_index = GRfindattr(ri_id, "Target attribute");
stat = GRgetattr(ri_id, attr_index, attr_values);
...
```

FORTRAN

```
integer function mggnatt([ri, gr]_id, attr_index,
                        values)

integer [ri, gr]_id, attr_index
<valid numeric data type> values(*)

integer function mggcatt([ri, gr]_id, attr_index,
                        values)

integer [ri, gr]_id, attr_index
character* (*) values
```

GRgetinfo/mggiinf

```
intn GRgetinfo(int32 ri_id, char *gr_name, int32 *ncomp, int32 *data_type, int32 *interlace,
               int32 dim_sizes[2], int32 *num_attrs)
```

<i>ri_id</i>	IN: General raster interface identifier returned by GRcreate or GRselect
<i>gr_name</i>	OUT: Buffer for the returned name of the general raster image
<i>ncomp</i>	OUT: Buffer for the returned number of components in the image
<i>data_type</i>	OUT: Buffer for the returned data type of the image data
<i>interlace</i>	OUT: Buffer for the returned interlace mode of the stored image data
<i>dim_sizes</i>	OUT: Buffer for the returned sizes of each image dimension
<i>num_attrs</i>	OUT: Buffer for the returned number of attributes assigned to the image

Purpose Reports general information about the specified raster image.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description Each of the arguments can be set to NULL, in which case the corresponding item of information will not be retrieved.

FORTTRAN

```
integer function mggiinf(ri_id, gr_name, ncomp, data_type,
                        interlace, dim_sizes, num_attrs)

integer ri_id, ncomp, data_type, interlace, num_attrs
integer dim_sizes[2]
character* (*) gr_name
```


GRgetlutid/mggltid

GRgetlutid/mggltid

int32 GRgetlutid(int32 *ri_id*, int32 *palette_index*)

ri_id IN: General raster image identifier returned by **GRcreate** or **GRselect**

palette_index IN: Index of the palette

Purpose Assigns a palette identifier to a general raster image dataset.

Return value Returns a palette identifier if successful and `FAIL` (or `-1`) otherwise.

Description This routine establishes the connection between a palette and a general raster image. Often used in conjunction with **GRwritelut**. Currently, only one palette can be assigned to a general raster image, which means that *palette_index* should always be set to `0`.

Example This example illustrates the use of **GRgetlutid** in creating a palette:

```
#include "hdf.h"

int32 pal_id, ri_id, gr_id, stat, image_index;
char pal_data[PALETTE_SIZE][3];
int32 ncomp = 3;
int32 data_type = DFNT_INT8;
int32 interlace = MFGR_INTERLACE_PIXEL;

...
image_index = GRnametoindex(gr_id, "Target image");
ri_id = GRselect(gr_id, image_index);
pal_id = GRgetlutid(ri_id, 0);
stat = GRwritelut(pal_id, ncomp, data_type, interlace,
    PALETTE_SIZE, pal_data);
...
```

FORTTRAN integer function mggltid(ri_id, palette_index)

integer ri_id, palette_index

GRgetlutinfo/mgglinf

```
intn GRgetlutinfo(int32 pal_id, int32 *ncomp, int32 *data_type, int32 *interlace, int32
                  *num_entries)
```

<i>pal_id</i>	IN: Palette identifier returned by GRgetlutid
<i>ncomp</i>	OUT: Number of components in the palette
<i>data_type</i>	OUT: Data type of the palette data
<i>interlace</i>	OUT: Interlace mode of the stored palette data
<i>num_entries</i>	OUT: Number of color lookup table entries in the palette

Purpose	Reports the data type, interlace mode and number of color lookup table entries of the specified palette.
----------------	--

Return value	Returns <code>SUCCESS</code> (or 0) if successful and <code>FAIL</code> (or -1) otherwise.
---------------------	--

Description	Each of the arguments can be set to <code>NULL</code> , in which case the corresponding item of information will not be retrieved.
--------------------	--

FORTRAN	<pre>integer function mgglinf(<i>pal_id</i>, <i>ncomp</i>, <i>data_type</i>, <i>interlace</i>, <i>num_entries</i>)</pre>
----------------	--

```
integer pal_id, ncomp, data_type, interlace, num_entries
```

GRidtoref/mgid2ref

GRidtoref/mgid2ref

uint16 GRidtoref(int32 *ri_id*)

<i>ri_id</i>	IN: General raster image identifier returned by GRselect or GRcreate
--------------	--

Purpose	Maps a general raster image identifier to a reference number.
----------------	---

Return value	Returns a general raster image identifier if successful and <code>FAIL</code> (or <code>-1</code>) otherwise.
---------------------	--

Description	This routine is commonly used for the purpose of annotating the image or including the image within a vgroup.
--------------------	---

Example	This example illustrates the use of GRidtoref in attaching an annotation to an image:
----------------	--

```
#include "hdf.h"

uint16 ref;
int32 an_id, ann_id, file_id;
ref = GRidtoref(ri_id);
an_id = ANstart(file_id);
ann_id = ANcreate(an_id, DFTAG_RI, ref, AN_DATA_LABEL);
```

FORTTRAN	<pre>integer function mgid2ref(ri_id) integer ri_id</pre>
----------	--

GRluttoref

uint16 GRluttoref(int32 *pal_id*)

<i>pal_id</i>	IN: Palette identifier returned from GRgetlutid
Purpose	Returns the reference number of the specified palette.
Return value	Returns the reference number of the palette if successful or <code>DFTAG_WILDCARD</code> (or 0) otherwise.
Description	This routine is commonly used for the purpose of annotating the palette or including the palette within a vgroup.

GRnametoindex/mgn2idx

GRnametoindex/mgn2idx

int32 GRnametoindex(int32 *gr_id*, const char **gr_name*)

<i>gr_id</i>	IN:	General raster interface identifier returned by GRstart
<i>gr_name</i>	IN:	Name of the target general raster image

Purpose Maps the name of a general raster image to an index.

Return value Returns the index of the image if successful and `FAIL` (or `-1`) otherwise.

Description In a manner similar to **GRreftoindex**, this routine is commonly used for the purpose of accessing an image in conjunction with a call to **GRselect**.

Example This example illustrates the use of **GRnametoindex**:

```
#include "hdf.h"

int32 gr_id, ri_id, index;
char *gr_name;
...
index = GRnametoindex(gr_id, gr_name);
ri_id = GRselect(gr_id, index);
...
```

FORTTRAN	<pre>integer function mgn2idx(gr_id, gr_name) integer gr_id character* (*) gr_name</pre>
----------	---

GRreadimage/mgrding/mgrcimg

```
intn GRreadimage(int32 ri_id, int32 start[2], int32 stride[2], int32 edge[2], VOIDP data)
```

<i>ri_id</i>	IN:	General raster image identifier returned by GRcreate or GRselect
<i>start</i>	IN:	Array containing the two-dimensional coordinate of the initial location for the read
<i>stride</i>	IN:	Array containing the number of data locations the current location is to be moved forward before each read
<i>edge</i>	IN:	Array containing the number of data elements that will be read along each dimension
<i>data</i>	OUT:	Buffer for the image data to be read

Purpose Reads a general raster image to the current HDF file.

Return value Returns `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description By setting the *start*, *stride* and *edge* parameters appropriately, **GRreadimage** will perform subsampling and image slab reads. Setting *stride* to `NULL` assumes a *stride* value of 1.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mgrding**) and the other for buffered character data (**mgrcimg**).

Example This example illustrates the use of **GRreadimage**:

```
#include "hdf.h"

int32 gr_id, ri_id, stat;
char *name = "Image name";
int32 ncomp = 0;
int32 interlace_mode = MFGR_INTERLACE_PIXEL;
int32 data_type = DFNT_UINT16;
int32 start[2], stride[2], edge[2];
VOIDP data[100];
int32 dim_sizes[2] = {50, 60};

file_id = Hopen("myfile", DFACC_READ, 0);
gr_id = GRstart(file_id);
ri_id = GRcreate(gr_id, name, ncomp, data_type,
                interlace_mode, dim_sizes);
...
start[0] = start[1] = 0;
stride[0] = stride[1] = 1;
edge[0] = edge[1] = 10;

stat = GRreadimage(ri_id, start, stride, edge, data);
```

GRreadimage/mgrdimg/mgrcimg

```
...
stat = Grendaccess(ri_id);
stat = Grend(gr_id);
Hclose(file_id);

FORTRAN      integer function mgrdimg(ri_id, start, stride, edge, data)

              integer ri_id, start(2), stride(2), edge(2)
              <valid numeric data type> data(*)

              integer function mgrcimg(ri_id, start, stride, edge, data)

              integer ri_id, start(2), stride(2), edge(2)
              character* (*) data
```

GRreadlut/mgrdlut/mgrclut

```
intn GRreadlut(int32 pal_id, VOIDP pal_data)
```

pal_id IN: Palette identifier of the target general raster image

pal_data OUT: Buffer for the palette data to be read

Purpose Reads the palette referred to by the given palette identifier.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description This routine is commonly used in conjunction with a call to **GRgetlutid**.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mgrdlut**) and the other for buffered character data (**mgrclut**).

Example This example illustrates the use of **GRreadlut**:

```
#include "hdf.h"

int32 pal_id, ri_id, gr_id, stat;
intn index;
char pal_data[PALETTE_SIZE];
...
index = GRnametoindex(gr_id, "Target image");
ri_id = GRselect(gr_id, index);
pal_id = GRgetlutid(ri_id, 0);
stat = GRreadlut(pal_id, pal_data);
...
```

FORTRAN integer function mgrclut(pal_id, pal_data)

integer pal_id
<valid numeric data type> pal_data(*)

integer function mgrdlut(pal_id, pal_data)

integer pal_id
character* (*) pal_data

GRreftoindex/mgr2idx

GRreftoindex/mgr2idx

int32 GRreftoindex(int32 *gr_id*, uint16 *ref*)

<i>gr_id</i>	IN:	General raster interface identifier returned by GRstart
<i>ref</i>	IN:	Reference number to be mapped

Purpose Maps the reference number of an image to an index.

Return value Returns the index of the image if successful and `FAIL` (or `-1`) otherwise.

Description In the same manner as **GRnametoindex**, this routine is commonly used for the purpose of accessing an image in conjunction with a call to **GRselect**.

Example This example illustrates the use of **GRreftoindex**:

```
#include "hdf.h"

int32 gr_id, ri_id, index;
uint16 ref;

...
index = GRreftoindex(gr_id, ref);
ri_id = GRselect(gr_id, index);
...
```

FORTTRAN

```
integer function mgr2idx(gr_id, ref)

integer gr_id, ref
```

GRreqimageil/mgrimil

intn GRreqimageil(int32 *ri_id*, intn *interlace_mode*)

<i>ri_id</i>	IN: General raster image identifier returned by GRcreate or GRselect
<i>interlace_mode</i>	IN: Interlace mode to be in effect during the next image operation: MFGR_INTERLACE_PIXEL (or 0), MFGR_INTERLACE_LINE (or 1), or MFGR_INTERLACE_COMPONENT (or 2)

Purpose Sets or resets the interlace mode to be in effect during the next image read operation.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description **GRreqimageil** can be called anytime before the image read operation.

Example This example illustrates the use of **GRreqimageil**:

```
#include "hdf.h"

int32 ri_id, stat;
intn interlace_mode = MFGR_INTERLACE_PIXEL;
int32 start[2], stride[2], edge[2];
uint16 data[100];

start[0] = start[1] = 0;
stride[0] = stride[1] = 1;
edge[0] = edge[1] = 10;
...
stat = GRreqimageil(ri_id, interlace_mode);
stat = GRreadimage(ri_id, start, stride, edge, data);
...
```

FORTRAN	integer function mgrimil(ri_id, interlace_mode)
	integer ri_id, interlace_mode

GRreqlutil/mgrltil

GRreqlutil/mgrltil

intn GRreqlutil(int32 *ri_id*, intn *interlace_mode*)

<i>ri_id</i>	IN:	General raster image identifier returned by GRcreate or GRselect
<i>interlace_mode</i>	IN:	Interlace mode of the next palette read operation: MFGR_INTERLACE_PIXEL (or 0), MFGR_INTERLACE_LINE (or 1), or MFGR_INTERLACE_COMPONENT (or 2)

Purpose	Sets or resets the interlace mode that will be in effect during the next palette read operation.
----------------	--

Return value	Returns <code>SUCCEED</code> (or 0) if successful and <code>FAIL</code> (or -1) otherwise.
---------------------	--

Description	GRreqlutil can to called anytime before the palette read operation.
--------------------	--

Example	This example illustrates the use of GRreqlutil :
----------------	---

```
#include "hdf.h"

int32 ri_id, pal_id, stat;
VOIDP palette_data;
intn interlace_mode = MFGR_INTERLACE_PIXEL;
...
stat = GRreqlutil(ri_id, interlace_mode);
stat = GRreadlut(pal_id, palette_data);
...
```

FORTTRAN	integer function mgrltil(ri_id, interlace_mode) integer ri_id, interlace_mode
----------	--

GRselect/mgselect

int32 GRselect(int32 *gr_id*, int32 *index*)

<i>gr_id</i>	IN: General raster interface identifier returned by GRstart
<i>index</i>	IN: Index of the general raster image in the file

Purpose Selects and returns the identifier for the general raster image identified by the index *index*.

Return value Returns the identifier of the selected general raster image if successful or `FAIL` (or -1) otherwise.

Description The index supplied by the parameter *index* is zero-based.

Example This example illustrates the use of **GRselect**:

```
#include "hdf.h"

int32 gr_id, ri_id, file_id, index, stat;

file_id = Hopen("myfile", DFACC_READ, 0);
gr_id = GRstart(file_id);
ri_id = GRselect(gr_id, index);
...
stat = Grendaccess(ri_id);
stat = Grend(gr_id);
Hclose(file_id);
```

FORTTRAN

```
integer function mgselect(gr_id, index)

integer gr_id, index
```

GRsetaccesstype/mgsactp

GRsetaccesstype/mgsactp

intn GRsetaccesstype(int32 *ri_id*, uintn *access_mode*)

ri_id IN: General raster image identifier returned by **GRcreate** or **GRselect**

access_mode IN: Access mode for the image data: DFACC_SERIAL or
 DFACC_PARALLEL

Purpose Specifies the access mode to be either parallel or serial I/O.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description **GRsetaccesstype** is designed specifically for use on computers that support parallel I/O. On all other computers, the standard serial mode I/O setting is recommended.

FORTRAN integer function mgsactp(*ri_id*, *access_mode*)

 integer *ri_id*, *access_mode*

GRsetattr/mgsnatt/mgscatt

intn GRsetattr(int32 [*ri*, *gr*]_{*id*}, const char **attr_name*, int32 *data_type*, int32 *count*, const VOIDP *values*)

<i>[ri, gr]_id</i>	IN:	Identifier of the raster image or the GR interface identifier returned by GRstart
<i>attr_name</i>	IN:	Name of the attribute(s)
<i>data_type</i>	IN:	Data type of the attribute(s)
<i>count</i>	IN:	Number of values in the attribute(s)
<i>values</i>	IN:	Buffer for attribute values

Purpose Assigns an attribute to one raster image, or all general raster images, in a file.

Return value Returns SUCCEED (or 0) if successful and FAIL (or -1) otherwise.

Description Currently, the only predefined attribute is the fill value, identified by the FILL_ATTR definition. If a *gr_id* is specified as the first parameter, the global attributes (applied to all general raster images in the file) are set.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mgsnatt**) and the other for buffered character data (**mgscatt**).

Example This example illustrates the use of **GRsetattr** to assign attributes to a general raster image:

```
#include "hdf.h"

int32 ri_id, gr_id, stat;
int32 attr_values[2];
...
attr_values[0] = 5;
attr_values[1] = 50;
ri_id = GRselect(gr_id, 0);
stat = GRsetattr(ri_id, "Value range", DFNT_INT32, 2
                 attr_values);
...
```

FORTTRAN

```
integer function mgsnatt([ri, gr]_id,
                        attr_name, data_type, count, values)

integer [ri, gr]_id, attr_name, data_type
integer count
<valid numeric data type> values(*)

integer function mgscatt([ri, gr]_id,
                        attr_name, data_type, count, values)

integer [ri, gr]_id, attr_name, data_type
```

GRsetattr/mgsnatt/mgscatt

integer count
character* (*) values

GRsetcompress

```
intn GRsetcompress(int32 ri_id, int32 comp_type, comp_info *c_info)
```

<i>ri_id</i>	IN:	General raster image identifier returned by GRcreate or GRselect
<i>comp_type</i>	IN:	Compression method for the image data: <code>COMP_CODE_RLE</code> , <code>COMP_CODE_DEFLATE</code> or <code>COMP_CODE_SKPHUFF</code>
<i>c_info</i>	IN:	Pointer to the <code>comp_info</code> union

Purpose	Specifies that the image data of a general raster dataset is a compressed special element.
----------------	--

Return value	Returns <code>SUCCESS</code> (or 0) if successful and <code>FAIL</code> (or -1) otherwise.
---------------------	--

Description	The <code>COMP_CODE_RLE</code> definition specifies run-length encoding, <code>COMP_CODE_DEFLATE</code> specifies Gnu ZIP (or GZIP) compression and <code>COMP_CODE_SKPHUFF</code> specifies skipping Huffman. The <code>comp_info</code> union contains algorithm-specific information for the library routines that perform the compression and is defined in the <code>hcomp.h</code> header file.
--------------------	---

GRsetexternalfile/mgsxfil

GRsetexternalfile/mgsxfil

int32 GRsetexternalfile(int32 *ri_id*, const char **filename*, int32 *offset*)

<i>ri_id</i>	IN:	General raster image identifier returned by GRcreate or GRselect
<i>filename</i>	IN:	Name of the file the external dataset will be stored in
<i>offset</i>	IN:	Offset, in bytes, from the beginning of the external file to the image data

Purpose Specifies that the image data of a general raster image dataset is a special element of an external element. Creates an external image array for a general raster dataset.

Return value Returns `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description An external image array is one that is stored in a file that is not the file containing the metadata for the image. **GRsetexternalfile** marks the image identified by *ri_id* as one whose data is to be written to the external image array. It can only be called once for each general raster dataset created.

FORTRAN

```
integer function mgsxfil(ri_id, filename, offset)

integer ri_id, offset
character* (*) filename
```

GRstart/mgstart

intn GRstart(int32 *file_id*)

file_id IN: File identifier returned by **Hopen**.

Purpose Initializes the general raster interface.

Return value Returns a general raster interface identifier if successful and `FAIL` (or `-1`) otherwise.

Description This routine is used with the **GRend** routine to define the extent of a general raster interface session. As with the start routines in the other interfaces, **GRstart** initializes the internal interface structures needed for the remaining GR routines. Use the general purpose routines **Hopen** and **Hclose** to manage file access. The GR routines will not open and close HDF files.

Example This example illustrates the use of **GRstart** and **GRend** in initializing and terminating a GR interface session.

```
#include "hdf.h"
int32 gr_id, file_id, stat;

file_id = Hopen("myfile", DFACC_WRITE, 0);
gr_id = GRstart(file_id);
...
stat = GRend(gr_id);
Hclose(file_id);
```

FORTTRAN integer function mgstart(file_id)

 integer file_id

GRwriteimage/mgwring/mgwcimg

GRwriteimage/mgwring/mgwcimg

intn GRwriteimage(int32 *ri_id*, int32 *start*[2], int32 *stride*[2], int32 *edge*[2], VOIDP *data*)

<i>ri_id</i>	IN:	General raster image identifier returned by GRcreate or GRselect
<i>start</i>	IN:	Array containing the two-dimensional coordinate of the initial location for the write
<i>stride</i>	IN:	Array containing the number of data locations the current location is to be moved forward before each write
<i>edge</i>	IN:	Array containing the number of data elements that will be written along each dimension
<i>data</i>	IN:	Buffer containing the image data to be written

Purpose Writes a general raster image to the current HDF file.

Return value Returns `SUCCESS` (or 0) if successful and `FAIL` (or -1) otherwise.

Description By setting the *start*, *stride* and *edge* parameters appropriately, **GRwriteimage** will perform subsampling and image slab writes. Setting *stride* to `NULL` assumes a stride value of 1.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mgwring**) and the other for buffered character data (**mgwcimg**).

Example This example illustrates the use of **GRwriteimage**:

```
#include "hdf.h"

int32 file_id, gr_id, ri_id, stat;
char *name = "Image name";
int32 ncomp = 0;
int32 interlace_mode = MFGR_INTERLACE_PIXEL;
int32 data_type = DFNT_UINT16;
int32 dim_sizes[2];
uint16 *data;

file_id = Hopen("myfile", DFACC_WRITE, 0);
gr_id = GRstart(file_id);
ri_id = GRcreate(gr_id, name, ncomp, data_type,
                interlace_mode, dim_sizes);
...
stat = GRwriteimage(ri_id, start, stride, edge, data);
...
stat = Grendaccess(ri_id);
stat = Grend(gr_id);
Hclose(file_id);
```

```
FORTRAN      integer function mgwring(ri_id, start, stride, edge, data)

              integer ri_id, start(2), stride(2), edge(2)
              <valid numeric data type> data(*)

              integer function mgwcimg(ri_id, start, stride, edge, data)

              integer ri_id, start(2), stride(2), edge(2)
              character* (*) data
```

GRwritelut/mgwrlut/mgwclut

GRwritelut/mgwrlut/mgwclut

intn GRwritelut(int32 *pal_id*, int32 *ncomp*, int32 *data_type*, int32 *interlace*, int32 *num_entries*,
VOIDP *pal_data*)

<i>pal_id</i>	IN:	Palette identifier to be assigned to the written data
<i>ncomp</i>	IN:	Number of color components in the palette
<i>data_type</i>	IN:	Data type of the palette data
<i>interlace</i>	IN:	Interlace mode of the stored palette data
<i>num_entries</i>	IN:	Number of entries in the palette
<i>pal_data</i>	IN:	Buffer for the palette data to be written

Purpose Writes palette data to a general raster image dataset.

Return value Returns `SUCCEED` (or 0) if successful and `FAIL` (or -1) otherwise.

Description This routine is commonly used in conjunction with a call to **GRgetlutid**.

Note that there are two Fortran-77 versions of this routine; one for buffered numeric data (**mgwrlut**) and the other for buffered character data (**mgwclut**).

Example This example illustrates the use of **GRwritelut**:

```
#include "hdf.h"

int32 pal_id, ri_id, gr_id, stat;
intn lut_index;
char pal_data[PALETTE_SIZE][3];
int32 ncomp = 3;
int32 data_type = DFNT_INT8;
int32 interlace = MFGR_INTERLACE_PIXEL;
...
index = GRnametoindex(gr_id, "Target image");
ri_id = GRselect(gr_id, index);
pal_id = GRgetlutid(ri_id, lut_index);
stat = GRwritelut(pal_id, ncomp, data_type, interlace,
    PALETTE_SIZE, pal_data);
...
```

FORTRAN

```
integer function mgwrlut(pal_id, ncomp, data_type,
    interlace, num_entries, pal_data)

integer pal_id, ncomp, data_type, interlace, num_entries
<valid numeric data type> pal_data(*)

integer function mgwclut(pal_id, ncomp, data_type,
    interlace, num_entries, pal_data)
```

```
integer pal_id, ncomp, data_type, interlace, num_entries  
character* (*) pal_data
```