DF24addimage/d2aimg

intn DF24addimage(const char **filename*, VOIDP *image*, int32 *width*, int32 *height*)

filename	IN:	Name of the file
image	IN:	Pointer to the image array
width	IN:	Number of columns in the image
height	IN	Number of rows in the image

Purpose Writes a 24-bit image to the specified file.

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

Description **DF24addimage** appends a 24-bit raster image set to the file. Array *image* is assumed to be width *x* height *x* 3 bytes. In Fortran-77, the dimensions of the array *image* must be the same as the dimensions of the image data.

The order in which dimensions are declared is different between C and Fortran-77. Ordering varies because Fortran-77 arrays are stored in column-major order, while C arrays are stored in row-major order. (Row-major order implies that the last coordinate varies fastest).

When **DF24addimage** writes an image to a file, it assumes row-major order. The Fortran-77 declaration that causes an image to be stored in this way must have the width as its first dimension and the height as its second dimension. In other words, the image must be built "on its side".

FORTRAN

```
integer function d2aimg(filename, image, width, height)

character* (*) filename
<valid numeric data type> image
integer width, height
```

DF24getdims/d2gdims

intn DF24getdims (const char **filename*, int32 **width*, int32 **height*, intn **il*)

filename	IN:	Name of the file
width	OUT:	Width of the image
height	OUT:	Height of the image
il	OUT:	File interlace mode of the image

Purpose Retrieves dimensions and interlace storage scheme of next image.

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

Description **DF24getdims** retrieves the dimensions and interlace of the image. If the file is being opened for the first time, **DF24getdims** returns information about the first image in the file. If an image has already been read, **DF24getdims** finds the next image. In this way, images are read in the same order in which they were written to the file.

If the dimensions and interlace of the image are known beforehand, there is no need to call **DF24getdims**. Simply allocate arrays with the proper dimensions for the image and invoke **DF24getimage** to read the images. If, however, you do not know the values of width and height, you must call **DF24getdims** to get them and then use them to determine the amount of memory to allocate for the image buffer.

Successive calls to **DF24getdims** and **DF24getimage** retrieve all of the images in the file in the sequence in which they were written.

The interlace mode codes are: `0` for pixel interlacing, `1` for scan-line interlacing and `2` for scan-plane interlacing.

FORTRAN

```
integer function d2gdims(filename, width, height, il)

character* (*) filename
integer width, height, il
```

DF24getimage/d2gimg

intn DF24getimage(const char **filename*, VOIDP *image*, int32 *width*, int32 *height*)

filename	IN:	Name of the HDF file
image	OUT:	Pointer to image buffer
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image

Purpose Retrieves an image from the next 24-bit raster image set.

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

Description **DF24getimage** retrieves the image and stores it in an array. If **DF24getdims** has not been called, **DF24getimage** finds the next image in the same way that **DF24getdims** does.

The amount of space allocated for the image should be width *x* height *x* 3 bytes.

To specify that the next call to **DF24getimage** should read the raster image using an interlace other than the interlace used to store the image in the file, first call **DF24reqil**.

FORTRAN

```
integer function d2gimg(filename, image, width, height)

character* (*) filename, image
integer width, height
```

DF24lastref/d2lref

uint16 DF24lastref( )

Purpose Retrieves the last reference number written to or read from a 24-bit raster image set.

Return value Returns the non-zero reference number if successful and `FAIL` (or `-1`) otherwise.

Description This routine is primarily used for attaching annotations to 24-bit images and adding 24-bit images to vgroups. **DF24lastref** returns the reference number of the last 24-bit raster image read or written.

FORTRAN `integer function d2lref( )`

DF24nimages/d2nimg

intn DF24nimages(const char **filename*)

filename	IN:	Name of the file

Purpose Counts the number of 24-bit raster images contained in an HDF file.

Return value Returns the number of 24-bit images in the file if successful and `FAIL` (or `-1`) otherwise.

Description **DF24nimages** counts the number of 24-bit images stored in the file.

FORTRAN

```
integer function d2nimg(filename)

character* (*) filename
```

DF24putimage/d2pimg

intn DF24putimage(const char **filename*, VOIDP *image*, int32 *width*, int32 *height*)

filename	IN:	Name of the file
image	IN:	Pointer to the image array
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image

Purpose Writes a 24-bit image as the first image in the file.

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

Description The array image is assumed to be width *x* height *x* 3 bytes. **DF24putimage** overwrites any information that exists in the HDF file. To append a new image to a file instead of overwriting an existing file, use **DF24addimage**.

FORTRAN

```
integer function d2pimg(filename, image, width, height)

character* (*) filename
<valid numeric data type> image
integer width, height
```

DF24readref/d2rref

intn DF24readref(const char **filename*, uint16 *ref*)

filename	IN:	Name of the file
ref	IN:	Reference number for the next call to **DF24getimage**

Purpose Specifies the reference number of the next image to be read when **DF24getimage** is next called.

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

Description **DF24readref** is commonly used in conjunction with **DFANlablist**, which returns a list of labels for a given tag together with their reference numbers. It provides a means of non-sequentially accessing 24-bit raster images in a file.

There is no guarantee that reference numbers appear in sequence in an HDF file. Therefore, it is not safe to assume that a reference number is the index of an image.

FORTRAN

```
integer function d2rref(filename, ref)

character* (*) filename
integer ref
```

DF24reqil/d2reqil

intn DF24reqil (intn *il*)

il	IN	Memory interlace of the next image read

Purpose Specifies the interlace mode for the next call to **DF24getimage** will use.

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

Description Regardless of what interlace scheme is used to store the image, **DF24reqil** causes the image to be loaded into memory and be interlaced according to the specification of *il*.

Because a call to **DF24reqil** may require a substantial reordering of the data, slower I/O performance could result than would be achieved if no change in interlace were requested.

The interlace mode codes are: `0` for pixel interlacing,`1` for scan-line interlacing and `2` for scan-plane interlacing.

FORTRAN

```
integer function d2reqil(il)

integer il
```

DF24restart/d2first

intn DF24restart( )

Purpose Specifies that the next 24-bit image read from the file will be the first one rather than the 24-bit image following the one most recently read.

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

FORTRAN `integer function d2first( )`

DF24setcompress/d2scomp

intn DF24setcompress(int32 *type*, comp_info **cinfo*)

type	IN:	Type of compression
cinfo	IN:	Pointer to compression information structure

Purpose Set the type of compression to use when writing the next 24-bit raster image.

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

Description This routines provides a method for compressing the next raster image written. The type can be one of the following values: COMP_NONE, COMP_JPEG, COMP_RLE, COMP_IMCOMP, COMP_NONE is the default for storing images if this routine is not called, therefore images are not compressed by default. COMP_JPEG compresses images with a JPEG algorithm, which is a lossy method. COMP_RLE uses lossless run-length encoding to store the image. COMP_IMCOMP uses a lossy compression algorithm called IMCOMP, and is included for backward compatibility only.

The comp_info union contains algorithm-specific information for the library routines that perform the compression and is defined in the hcomp.h header file as follows:

```
typedef union tag_comp_info
  {
      struct
       {
            intn    quality;
            intn    force_baseline;
        }
      jpeg;
      struct
        {
            int32   nt;
            intn    sign_ext;
            intn    fill_one;
            intn    start_bit;
            intn    bit_len;
        }
      nbit;
      struct
        {
            intn    skp_size;
        }
      skphuff;
      struct
        {
            intn    level;
        }
      deflate;
  }
comp_info;
```

This union is defined to provide future expansion, but is currently only used by the COMP_JPEG compression type. A pointer to a valid comp_info union is required for all compression types other than COMP_JPEG, but the values in the union are not used. The comp_info union is declared in the header file hdf.h and is shown here for informative purposes only, it should not be re-declared in a user program.

For COMP_JPEG compression, the quality member of the jpeg structure must be set to the quality of the stored image. This number can vary from 100, the best quality, to 0, terrible quality. All images stored with COMP_JPEG compression are stored in a lossy manner, even images stored with a quality of 100. The ratio of size to perceived image quality varies from image to image, some experimentation may be required to determine an acceptable quality factor for a given application. The force_baseline parameter determines whether the quantization tables used during compression are forced to the range 0-255. The force_baseline parameter should normally be set to 1 (forcing baseline results), unless special applications require non-baseline images to be used.

If the compression type is JPEG, **d2scomp** defines the default JPEG compression parameters to be used. If these parameters must be changed later, the **d2sjpeg** routine must be used. (See the Reference Manual entry for **d2sjpeg**)

FORTRAN

```
integer function d2scomp(type)

integer type
```

DF24setdims/d2sdims

intn DF24setdims(int32 *width*, int32 *height*)

width	IN:	Number of columns in the image
height	IN:	Number or rows in the image

Purpose Set the dimensions of the next image to be written to a file.

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

FORTRAN

```
integer function d2sdims(width, height)

integer width, height
```

DF24setil/d2setil

intn DF24setil(intn *il*)

il	IN:	Interlace mode

Purpose Specifies the interlace mode to be used on subsequent writes.

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

Description **DF24setil** sets the interlace mode to be used when writing out the raster image set for a 24-bit image by determining the interlace mode of the image data in memory. If **DF24setil** is not called, the interlace mode is assumed to be `0`.

The interlace mode codes are: `0` for pixel interlacing, `1` for scan-line interlacing and `2` for scan-plane interlacing.

FORTRAN

```
integer function d2setil(il)

integer il
```