DF24addimage/d2aimg

intn DF24addimage(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 (char **filename*, int32 **width*, int32 **height*, intn **interlace_mode*)

filename	IN:	Name of the file
width	OUT:	Width of the image
height	OUT:	Height of the image
interlace_mode	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, interlace_mode)

character*(*) filename

integer width, height, interlace_mode
```

DF24getimage/d2gimg

intn DF24getimage(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(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(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(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
```

d2scomp

integer d2scomp(integer *quality*, integer *baseline*)

quality	IN:	JPEG quality specification
baseline	IN:	JPEG baseline specification

Purpose Fortran-specific routine that sets the parameters needed for the JPEG algorithm.

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

Description **d8sjpeg** changes the JPEG compression parameter settings set in the **d8scomp** routine.

d2sjpeg

integer d2sjpeg(integer *quality*, integer *baseline*)

quality	IN:	JPEG quality specification
baseline	IN:	JPEG baseline specification

Purpose Fortran-specific routine that sets the parameters needed for the JPEG algorithm.

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

Description **d2sjpeg** changes the JPEG compression parameter settings set in the **d2scomp** routine.

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
```

DFR8addimage/d8aimg

intn DFR8addimage(char **filename*, VOIDP *image*, int32 *width*, int32 *height*, uint16 *compress*)

filename	IN:	Name of the file
image	IN:	Array containing the image data
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image
compress	IN:	Type of compression to use, if any

Purpose **DFR8addimage** appends the RIS8 for the image to the file.

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

Description **DFR8addimage** is functionally equivalent to **DFR8putimage**, except that **DFR8putimage** cannot append image data; it only overwrites.

FORTRAN

```
integer function d8aimg(filename, image, width, height, compress)

character*(*) filename, image

integer width, height

integer compress
```

DFR8getdims/d8gdims

intn DFR8getdims(char **filename*, int32 **width*, int32 **height*, intn **ispalette*)

filename	IN:	Name of the HDF file
width	OUT:	Number of columns in the next image in the file
height	OUT:	Number of rows in the next image in the file
ispalette	OUT:	Indicator of the existence of a palette

Purpose Opens the file, finds the next image, retrieves the dimensions of the image, and determines whether there is a palette associated with the image.

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

Description **DFR8getdims** retrieves the dimensions of the image and indicates whether a palette is associated and stored with the image. If the file is being opened for the first time, **DFR8getdims** returns information about the first image in the file. If an image has already been read, **DFR8getdims** finds the next image. Thus, images are read in the same order in which they were written to the file.

Normally, **DFR8getdims** is called before **DFR8getimage** so that if necessary, space allocations for the image and palette can be checked, and the dimensions can be verified. If this information is already known, **DFR8getdims** need not be called.

Valid values of *ispalette* are: `1` if there is a palette, or `0` if not.

FORTRAN

```
integer function d8gdims(filename, width, height, ispalette)

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

DFR8getimage/d8gimg

intn DFR8getimage(char **filename*, uint8 **image*, int32 *width*, int32 *height*, uint8 **palette*)

filename	IN:	Name of the file
image	OUT:	Buffer for the returned image
width	IN:	Width of the image data buffer
height	IN:	Height of the image data buffer
palette	OUT:	Palette data

Purpose To retrieve the image and its palette, if it is present, and store them in the specified arrays.

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

Description In C, if *palette* is `NULL`, no palette is loaded, even if one is stored with the image. In FORTRAN-77, an array must be allocated to store the palette, even if no palette is expected to be stored. If the image in the file is compressed, **DFR8getimage** automatically decompresses it. If **DFR8getdims** has not been called, **DFR8getimage** finds the next image in the same way that **DFR8getdims** does.

The *width* and *height* parameters specify the number of columns and rows, respectively, in the array which you've allocated in memory to store the image. The image may be smaller than the allocated space.

The order in which you declare dimensions 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 horizontal coordinate varies fastest). When **d8gimg** reads an image from 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. To take this into account as you read image in your program, the image must be built “on its side”.

FORTRAN

```
integer function d8gimg(filename, image, width, height, palette)

character*(*) filename, image, palette

integer width, height
```

DFR8getpalref

intn DFR8getpalref(uint16 **pal_ref*)

pal_ref	OUT:	Reference number of the palette

Purpose Retrieves the reference number of the palette associated with the last image accessed.

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

Description Make certain that **DFR8getdims** is called before **DFR8getpalref**.

DFR8lastref/d8lref

uint16 DFR8lastref( )

Purpose Retrieves the last reference number written to or read from an RIS8.

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

Description This routine is primarily used for attaching annotations to images and adding images to vgroups. **DFR8lastref** returns the reference number of last raster image set read or written.

FORTRAN `integer function d8lref( )`

DFR8nimages/d8nims

intn DFR8nimages(char **filename*)

filename	IN:	Name of the HDF file

Purpose Retrieves the number of 8-bit raster images stored in the specified file.

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

FORTRAN

```
integer function d8nims(filename)

character*(*) filename
```

DFR8putimage/d8pimg

intn DFR8putimage(char **filename*, VOIDP *image*, int32 *width*, int32 *height*, uint16 *compress*)

filename	IN:	Name of the file to store the raster image in
image	IN:	Array with image to put in file
width	IN:	Number of columns in the image
height	IN:	Number of rows in the image
compress	IN:	Type of compression used, if any

Purpose Writes the RIS8 for the image as the first image in the file, overwriting any information previously in the file.

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

Description The *compress* parameter identifies the method to be used for compressing the data, if any. If IMCOMP compression is used, the image must include a palette.

DFR8putimage overwrites any information that exists in the HDF file. To write an image to a file by appending it, rather than overwriting it, use **DFR8addimage**.

In FORTRAN-77, the dimensions of the *image* array must be the same as the dimensions of the image itself.

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 horizontal coordinate varies fastest). When **DFR8putimage** 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, the reverse of the way it is done in C. To take this into account as you build your image in your FORTRAN-77 program, the image must be built “on its side”.

FORTRAN

```
integer function d8pimg(filename, image, width, height, compress)

character*(*) filename, image

integer width, height, compress
```

DFR8readref/d8rref

intn DFR8readref(char *filename, uint16 *ref*)

filename	IN:	Name of the file
ref	IN:	Reference number for next **DFR8getimage**

Purpose Specifies the reference number of the image to be read when **DFR8getimage** is next called.

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

Description **DFR8readref** is usually used in conjunction with **DFANlablist**, which returns a list of labels for a given tag together with their reference numbers. It provides, in a sense, a random access to images. 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 d8rref(filename, ref)

character*(*) filename

integer ref
```

DFR8restart/d8first

intn DFR8restart( )

Purpose **DFR8restart** causes the next get command to read from the first raster image set in the file.

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

FORTRAN `integer function d8first( )`

DFR8setcompress/d8scomp

intn DFR8setcompress(int32 *type*, comp_info **cinfo*)

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

Purpose Sets the compression type to be used when writing the next 8-bit raster image.

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

Description This routine 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 (refer to the header file for inline documentation):

```
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`. It 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, **d8scomp** defines the default JPEG compression parameters to be used. If these parameters must be changed later, the **d8sjpeg** routine must be used. (Refer to the Reference Manual page on **d8sjpeg**).

FORTRAN

```
integer function d8scomp(type)

integer type
```

d8scomp

integer d8scomp(integer *quality*, integer *baseline*)

quality	IN:	JPEG quality specification
baseline	IN:	JPEG baseline specification

Purpose Fortran-specific routine that sets the parameters needed for the JPEG algorithm.

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

Description **d8sjpeg** changes the JPEG compression parameter settings set in the **d8scomp** routine.

d8sjpeg

integer d8sjpeg(integer *quality*, integer *baseline*)

quality	IN:	JPEG quality specification
baseline	IN:	JPEG baseline specification

Purpose Fortran-specific routine that sets the parameters needed for the JPEG algorithm.

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

Description **d8sjpeg** changes the JPEG compression parameter settings set in the **d8scomp** routine.

DFR8setpalette/d8spal

intn DFR8setpalette(uint8 **palette*)

palette	IN:	Palette data

Purpose Indicate which palette, if any, is to be used for subsequent image sets.

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

Description The specified palette remains the default palette until changed by a subsequent call to **DFR8setpalette**.

FORTRAN

```
integer function d8spal(palette)

character*(*) palette
```

DFR8writeref/d8wref

intn DFR8writeref(char **filename*, uint16 *ref*)

filename	IN:	Name of the HDF file
ref	IN:	Reference number for next call to **DFR8putimage** or **DFR8addimage**

Purpose Specifies the reference number of the image to be written when **DFR8addimage** or **DFR8putimage** is next called.

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

Description It is unlikely that you will need this routine, but if you do, use it with caution. 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. In addition, using an existing reference number will overwrite the existing 8-bit raster image data.

FORTRAN

```
integer function d8wref(filename, ref)

character*(*) filename

integer ref
```

DFPaddpal/dpapal

intn DFPaddpal(char **filename*, VOIDP *palette*)

filename	IN:	Name of the HDF file
palette	IN:	Buffer containing the palette to be written

Purpose Appends a palette to a file.

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

Description If the named file does not exist, it is created and the palette written to it. The *palette* buffer should beat least 768 bytes in length.

FORTRAN

```
integer function dpapal(filename, palette)

character*(*) filename, palette
```

DFPgetpal/dpgpal

intn DFPgetpal(char **filename*, VOIDP *palette*)

filename	IN:	Name of the HDF file
palette	OUT:	Buffer for the returned palette

Purpose Retrieves the next palette from file and stores it in the buffer *palette*.

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

Description The *palette* buffer is assumed to be at least 768 bytes long. Successive calls to **DFPgetpal** retrieve the palettes in the sequence they are stored in the file.

FORTRAN

```
integer function dpgpal(filename, palette)

character*(*) filename. palette
```

DFPlastref/dplref

uint16 DFPlastref(void)

Purpose Returns the value of the reference number most recently read or written by a palette function call.

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

FORTRAN `integer function dplref( )`

DFPnpals/dpnpals

intn DFPnpals(char **filename*)

filename	IN:	Name of the file

Purpose	Indicates the number of palettes in the specified file.
Return value	Returns the number of palettes if successful and `FAIL` (or `-1`) otherwise.
FORTRAN	integer function dpnpals(filename)
	character*(*) filename

DFPputpal/dpppal

intn DFPputpal (char **filename*, VOIDP *palette*, intn *overwrite*, char **filemode*)

filename	IN:	Name of the file
palette	IN:	Buffer containing the palette to be written
overwrite	IN:	Flag identifying the palette to be written
filemode	IN:	File access mode

Purpose Writes a palette to the file.

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

Description This routine provides more control of palette write operations than **DFPaddpal**. Note that the combination *filemode*="w" and *overwrite*=`1` has no meaning and will result in an error condition. To overwrite a palette, *filename* must be the same filename as the last file accessed through the DFP interface.

Valid values for *overwrite* are: `1` to overwrite last palette; `0` to write a new palette.

Valid values for *filemode* are: "a" to append the palette to the file and "w" to create a new file.

The *palette* buffer must be at least 768 bytes in length.

FORTRAN

```
integer function dpppal(filename, palette, overwrite, filemode)

character*(*) filename, palette, filemode

integer overwrite
```

DFPreadref/dprref

intn DFPreadref(char **filename*, uint16 *ref*)

filename	IN:	Name of the file
ref	IN:	Reference number to be used in next **DFPgetpal** call

Purpose Retrieves the reference number of the palette to be retrieved next by **DFPgetpal**.

Return value Returns `SUCCEED` (or `0`) if the palette with the specified reference number exists and `FAIL` (or `-1`) otherwise.

Description Used to set the reference number of the next palette to be retrieved.

FORTRAN

```
integer function dprref(filename, ref)

character*(*) filename

integer ref
```

DFPrestart/dprest

intn DFPrestart( )

Purpose Specifies that **DFPgetpal** will read the first palette in the file, rather than the next unread palette.

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

FORTRAN `integer function dprest( )`

DFPwriteref/dpwref

intn DFPwriteref(char **filename*, uint16 *ref*)

filename	IN:	Name of the file
ref	IN:	Reference number to be assigned to the next palette written to a file

Purpose Determines the reference number of the next palette to be written.

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

Description The file name is ignored. The next palette written, regardless of the filename, is assigned the reference number *ref*.

FORTRAN

```
integer function dpwref(filename, ref)

character*(*) filename

integer ref
```

DFKNTsize

int DFKNTsize(int32 *data_type*)

data_type	IN:	Data type

Purpose Determines the size of the specified data type.

Return value Returns the size, in bytes, of the specified data type if successful and `FAIL` (or `-1`) otherwise.

DFUfptoimage/duf2im

int DFUfptoimage(int32 *hdim*, int32 *vdim*, float32 *max*, float32 *min*, float32 **hscale*, float32 **vscale*, float32 **data*, uint8 **palette*, char **outfile*, int *ct_method*, int32 *hres*, int32 *vres*, int *compress*)

hdim	IN:	Horizontal dimension of the input data
vdim	IN:	Vertical dimension of the input data
max	IN:	Maximum value of the input data
min	IN:	Minimum value of the input data
hscale	IN:	Horizontal scale of the input data (optional)
vscale	IN:	Vertical scale of the input data (optional)
data	IN:	Buffer containing the input data
palette	IN:	Pointer to the palette data
outfile	IN:	Name of the file the image data will be stored in
ct_method	IN:	Color transformation method
hres	IN:	Horizontal resolution to be applied to the output image
vres	IN:	Vertical resolution to be applied to the output image
compress	IN:	Compression flag

Purpose Converts floating point data to 8-bit raster image format and stores the converted image data in the specified file.

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

Description This routine is very similar to the utility `fptohdf`, which takes its input from one or more files, rather than from internal memory. Another difference is that this routine allows compression (run-length encoding), whereas `fptohdf` does not at present.

As this routine is meant to mimic many of the features of NCSA DataScope, much of the code has been taken directly from the DataScope source.

Valid values for *ct_method* are: `1` (or `EXPAND)` for expansion and `2` (or `INTERP)` for interpolation.

Valid values for *compress* are: `0` for no compression and `1` for compression enabled.

FORTRAN

```
integer function duf2im(hdim, vdim, max, min, hscale, vscale,
                        data, palette, outfile, ct_method,
                        hres, vres, compress)

integer hdim, vdim

real max, min, hscale, vscale, data

character*(*) palette, outfile

integer ctmethod, hres, vres, compress
```