

Octave CFITSIO Toolkit 0.0.6

FITS file functionality for GNU Octave.

John Donoghue

Copyright © 2019-2023 John Donoghue

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the same conditions as for modified versions.

Distribution

The GNU Octave CFITSIO package is *free* software. Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. This means that everyone is free to use it and free to redistribute it on certain conditions. The GNU Octave CFITSIO package is not, however, in the public domain. It is copyrighted and there are restrictions on its distribution, but the restrictions are designed to ensure that others will have the same freedom to use and redistribute Octave that you have. The precise conditions can be found in the GNU General Public License that comes with the GNU Octave CFITSIO package and that also appears in [Appendix A \[Copying\]](#), page 28.

To download a copy of the GNU Octave CFITSIO package, please visit <https://gnu-octave.github.io/octave-cfitsio//index>.

Table of Contents

1	Installing and loading	1
1.1	Windows install	1
1.2	Online Direct install	1
1.3	Off-line install	1
1.4	Loading	1
2	Basic Usage Overview	2
2.1	Overview	2
2.2	Using the toolkit	2
2.2.1	Reading Data	2
2.2.2	Reading Information	2
2.2.3	Low level functionality	2
3	Function Reference	3
3.1	High Level File Functions	3
3.1.1	fitsdisp	3
3.1.2	fitsinfo	3
3.1.3	fitsread	4
3.1.4	fitswrite	4
3.2	Low Level File Functions	5
3.2.1	matlab.io.fits.closeFile	5
3.2.2	matlab.io.fits.createFile	5
3.2.3	matlab.io.fits.deleteFile	6
3.2.4	matlab.io.fits.fileMode	6
3.2.5	matlab.io.fits.fileName	6
3.2.6	matlab.io.fits.openDiskFile	7
3.2.7	matlab.io.fits.openFile	7
3.3	Low Level HDU Functions	7
3.3.1	matlab.io.fits.copyHDU	8
3.3.2	matlab.io.fits.deleteHDU	8
3.3.3	matlab.io.fits.getHDUnum	8
3.3.4	matlab.io.fits.getHDUoff	9
3.3.5	matlab.io.fits.getHDUtype	9
3.3.6	matlab.io.fits.getNumHDUs	9
3.3.7	matlab.io.fits.movAbsHDU	9
3.3.8	matlab.io.fits.movNamHDU	10
3.3.9	matlab.io.fits.movRelHDU	10
3.3.10	matlab.io.fits.writeChecksum	10
3.4	Low Level Keyword Functions	11
3.4.1	matlab.io.fits.deleteKey	11
3.4.2	matlab.io.fits.deleteRecord	11
3.4.3	matlab.io.fits.getHdrSpace	11
3.4.4	matlab.io.fits.readCard	11
3.4.5	matlab.io.fits.readKey	12
3.4.6	matlab.io.fits.readKeyCmplx	12
3.4.7	matlab.io.fits.readKeyDbl	12
3.4.8	matlab.io.fits.readKeyLongLong	12

3.4.9	matlab.io.fits.readKeyLongStr	13
3.4.10	matlab.io.fits.readKeyUnit	13
3.4.11	matlab.io.fits.readRecord	13
3.4.12	matlab.io.fits.writeComment	14
3.4.13	matlab.io.fits.writeDate	14
3.4.14	matlab.io.fits.writeHistory	14
3.4.15	matlab.io.fits.writeKey	14
3.4.16	matlab.io.fits.writeKeyUnit	15
3.5	Low Level Image Manipulation	15
3.5.1	matlab.io.fits.createImg	15
3.5.2	matlab.io.fits.getImgSize	15
3.5.3	matlab.io.fits.getImgType	16
3.5.4	matlab.io.fits.insertImg	16
3.5.5	matlab.io.fits.readImg	16
3.5.6	matlab.io.fits.setBscale	17
3.5.7	matlab.io.fits.setTscale	17
3.5.8	matlab.io.fits.writeImg	17
3.6	Low Level Utility Functions	18
3.6.1	matlab.io.fits.getConstantNames	18
3.6.2	matlab.io.fits.getConstantValue	18
3.6.3	matlab.io.fits.getOpenFiles	18
3.6.4	matlab.io.fits.getVersion	18
3.7	Low Level Compression Functions	19
3.7.1	matlab.io.fits.imgCompress	19
3.7.2	matlab.io.fits.isCompressedImg	19
3.7.3	matlab.io.fits.setCompressionType	19
3.7.4	matlab.io.fits.setHCompScale	19
3.7.5	matlab.io.fits.setHCompSmooth	20
3.7.6	matlab.io.fits.setTileDim	20
3.8	Low Level Binary and ASCII Tables	20
3.8.1	matlab.io.fits.createTbl	20
3.8.2	matlab.io.fits.deleteCol	21
3.8.3	matlab.io.fits.deleteRows	21
3.8.4	matlab.io.fits.getAColParms	21
3.8.5	matlab.io.fits.getBColParms	22
3.8.6	matlab.io.fits.getColName	22
3.8.7	matlab.io.fits.getColType	23
3.8.8	matlab.io.fits.getEqColType	23
3.8.9	matlab.io.fits.getNumCols	23
3.8.10	matlab.io.fits.getNumRows	23
3.8.11	matlab.io.fits.getRowSize	24
3.8.12	matlab.io.fits.insertATbl	24
3.8.13	matlab.io.fits.insertBTbl	24
3.8.14	matlab.io.fits.insertCol	25
3.8.15	matlab.io.fits.insertRows	25
3.8.16	matlab.io.fits.readATblHdr	25
3.8.17	matlab.io.fits.readBTblHdr	25
3.8.18	matlab.io.fits.readCol	26
3.8.19	matlab.io.fits.writeCol	26
3.9	Import functions	27
3.9.1	import_fits	27

Appendix A GNU General Public License 28

Index..... 38

1 Installing and loading

The GNU Octave CFITSIO toolkit must be installed and then loaded to be used.

It can be installed in GNU Octave directly from octave-cfitsio, or can be installed in an off-line mode via a downloaded tarball.

The toolkit has a dependency on the cfitsio library (<https://heasarc.gsfc.nasa.gov/fitsio/>), so it must be installed in order to successfully install the GNU Octave toolkit.

For Fedora: `yum install cfitsio-devel`

The toolkit must be then be loaded once per each GNU Octave session in order to use its functionality.

1.1 Windows install

If running in Windows, the package may already be installed, to check run:

```
pkg list cfitsio
```

Otherwise it can be installed by installing the requirements and then using the online or offline install method.

1.2 Online Direct install

With an internet connection available, the package can be installed from octave-cfitsio using the following command within GNU Octave:

```
pkg install https://sourceforge.net/projects/octave-cfitsio/files/v0.0.6/octave-cfitsio-
```

On GNU Octave 7 and higher, the package can be installed in the simpler form of:

```
pkg install -forge cfitsio
```

The latest released version of the toolkit will be downloaded and installed.

1.3 Off-line install

With the toolkit package already downloaded, and in the current directory when running GNU Octave, the package can be installed using the following command within GNU Octave:

```
pkg install octave-cfitsio-0.0.6.tar.gz
```

1.4 Loading

Regardless of the method of installing the toolkit, in order to use its functions the toolkit must be loaded using the pkg load command:

```
pkg load cfitsio
```

The toolkit must be loaded on each GNU Octave session.

2 Basic Usage Overview

2.1 Overview

The octave-cfitsio toolkit provides high and low level functionality for reading and writing FITS format files.

The high level functions provide base read and write of data to octave.

The low level functions almost direct access to the cfitsio API and are provided under the matlab.io.fits namespace.

Since GNU Octave does not support the matlab import command, a `import_fits` function is provided.

Running the statement:

```
import_fits
```

Is the equivalent of running in matlab:

```
import matlab.io.fits;
```

2.2 Using the toolkit

The package must be loaded each time a GNU Octave session is started:

```
pkg load cfitsio
```

After loading the toolkit, the toolkit functions are available.

2.2.1 Reading Data

To read the primary image data of a fits file, use the `fitsread` function:

```
imagedata = fitsread("thefitsfile.fits");
```

2.2.2 Reading Information

To read information about the content in a fits file, use the `fitsinfo` functions.

```
info = fitsinfo("thefitsfile.fits");
```

2.2.3 Low level functionality

Where functionality is required that is not met by the high level functions, most of the cfitsio functions are available in the matlab.io.fits namespace.

```
# import the fits functions so don't have to use the full namespace each time
import_fits;

# open the file
fd = fits.openFile('tst0012.fits');

# get number of hdus in the file
n = fits.getNumHDUs (fd);

# for each hdu, go to it, print out the type
for j = 1:n
    hdutype = fits.movAbsHDU (fd, j);
    printf ('HDU %d:  "%s"\n', j, hdutype);
endfor

# close the file
fits.closeFile (fd);
```

3 Function Reference

The functions currently available in the toolkit are described below:

3.1 High Level File Functions

3.1.1 fitsdisp

```
info = fitsdisp(filename)
info = fitsdisp(filename, propertyname, propertyvalue)
```

Display metadata about fits format file

Inputs

filename - filename to open.

propertyname, *propertyvalue* - property name/value pairs

Known property names are:

'Index' Value is a scalar or vector of hdu numbers to display

'Mode' display mode of 'standard' (default), 'min' or 'full'

'standard' display mode shows the standard keywords for the selected HDUs.

'min' display mode shows only the type and size of the selected HDUs.

'full' display shows all keywords for the selected HDU.

Outputs

info - the metadata of the file. If no output variable is provided, it displays to the screen.

Examples

```
filename = file_in_loadpath("demos/tst0012.fits");
fitsdisp(filename);
```

3.1.2 fitsinfo

```
info = fitsinfo(filename)
```

Read information about fits format file

Inputs

filename - filename to open.

Outputs

info - a struct containing the structure and information about the fits file.

Examples

```
filename = file_in_loadpath("demos/tst0012.fits");
info = fitsinfo(filename);
```

3.1.3 fitsread

```
data = fitsread(filename)
data = fitsread(filename, 'raw')
data = fitsread(filename, extname)
data = fitsread(filename, extname, index)
data = fitsread(filename, _____, propertyname, propertyvalue)
```

Read the primary data, or specified extension data. It scales the data and applied Nan to any undefined values.

Inputs

filename - filename to open.

exttype - can be 'primary', 'asciitable', 'binarytable', 'image', 'unknown'.

index - can be used to specify which table when more than one of a given type exists.

'raw'- If the 'raw' keyword is used, the raw data from the file will be used without replacing undefined values with Nan

Known property names are:

Info input info from fitsinfo call.

PixelRegion

pixel region to extract data for in an image. It expects a cell array of same size as the number of axis in the image. Each cell should be in vector format of: start, [start stop] or [start, increment, stop].

TableColumns

A list of columns to extract from a ascii or binary table.

TableRows

A list of rows to extract from an ascii or binary table.

Outputs

data - The read data from the table or image.

Examples

```
filename = file_in_loadpath("demos/tst0012.fits");

# read the primary image data
imagedata = fitsread(filename);

# read the 1st non primary image
imagedata = fitsread(filename, "image");

# read the first binary table, selected columns
tbldata = fitsread(filename, "binarytable", "TableColumns", [1 2 11]);

# read the first ascii table
atbldata = fitsread(filename, "asciitable");
```

3.1.4 fitswrite

```
fitswrite(data, filename)
fitswrite(data, filename, propertyname, propertyvalue)
```

Write image data *data* to FITS file *filename*. If the file already exists, overwrite it.

Inputs

data - imagedata to write to a file.

filename - filename to write to.

propertyname, *propertyvalue* - property name/value pairs

Additional properties can be set as *propertyname*, *propertyvalue* pairs. Known property names are:

WriteMode

Set mode for writing to image as 'overwrite' (default) or 'append' to append images.

Compression

Set compression type to use for image as 'none' (default), 'gzip', 'rice', 'hcompress' or 'plio'.

Outputs

None

Examples

```
filename = tempname();
X = double([1:3;4:6]);
fitswrite(X, filename);
```

3.2 Low Level File Functions

3.2.1 matlab.io.fits.closeFile

`closeFile(file)`

Close the opened fits file

This is the equivalent of the `fits_close_file` function.

Inputs

file - opened file returned from `openFile` or `createFile`.

Outputs

None

Examples

```
import_fits;
filename = file_in_loadpath("demos/tst0012.fits")

fd = fits.openFile(filename);
fits.closeFile(fd);
```

See also: `matlab.io.fits.createFile`, `matlab.io.fits.openFile`.

3.2.2 matlab.io.fits.createFile

`file = createFile(filename)`

Attempt to create a file of the given input name.

If the filename starts with ! and the file exists, it will create a new file, otherwise, if the file exists, the create will fail.

This is the equivalent of the `cfitsio fits_create_file` function.

Inputs

filename - filename to open.

Outputs

file - opened file identifier.

Examples

```
import_fits;

fd = fits.createFile("myfitsfile.fits");
fits.createImg(fd, 'uint16', [100 100]);
fits.closeFile(fd);
```

See also: `matlab.io.fits.openFile`.

3.2.3 `matlab.io.fits.deleteFile`

`deleteFile(file)`

Force a close and delete of a fits file.

This is the equivalent of the `fits_delete_file` function.

Inputs

file - opened fits file.

Outputs

None

3.2.4 `matlab.io.fits.fileMode`

`mode = fileMode(file)`

Return the file mode of the opened fits file.

This is the equivalent of the `fits_file_mode` function.

Inputs

file - opened fits file.

Outputs

mode - The mode will return as a string 'READWRITE' or 'READONLY'

3.2.5 `matlab.io.fits.fileName`

`filename = fileName(file)`

Return the file name of the opened fits file.

This is the equivalent of the `fits_file_name` function.

Inputs

file - opened fits file.

Outputs

filename - name of the fits file.

3.2.6 matlab.io.fits.openDiskFile

```
file = openDiskFile(filename)  
file = openDiskFile(filename, mode)
```

Attempt to open a file of the given input name, ignoring any special processing of the filename.

This is the equivalent of the cfitsio fits_open_diskfile function.

Inputs

filename - filename to open.

mode - If the option mode string 'READONLY' (default) or 'READWRITE' is provided, open the file using that mode.

Outputs

file - opened file identifier.

Examples

```
import_fits;  
filename = file_in_loadpath("demos/tst0012.fits")  
  
fd = fits.openDiskFile(filename, 'READONLY');  
fits.closeFile(fd);
```

See also: openFile, createFile.

3.2.7 matlab.io.fits.openFile

```
file = openFile(filename)  
file = openFile(filename, mode)
```

Attempt to open a file of the given input name.

This is the equivalent of the cfitsio fits_open_file function.

Inputs

filename - filename to open.

mode - If the option mode string 'READONLY' (default) or 'READWRITE' is provided, open the file using that mode.

Outputs

file - opened file identifier.

Examples

```
import_fits;  
filename = file_in_loadpath("demos/tst0012.fits")  
  
fd = fits.openFile(filename, 'READONLY');  
fits.closeFile(fd);
```

See also: matlab.io.fits.openDiskFile, matlab.io.fits.createFile.

3.3 Low Level HDU Functions

3.3.1 matlab.io.fits.copyHDU

`copyHDU(infile, outfile)`

Copy current HDU from one infile to another.

This is the equivalent of the cfitsio `fits_copy_hdu` function.

Inputs

filename - filename to open.

Outputs

infile - opened input file identifier.

outfile - opened output file identifier.

Examples

```
import_fits;

# open input and output files
infile = file_in_loadpath("demos/tst0012.fits");
infile = fits.openFile(infile);

outfile = fits.createFile("myfitsfile.fits");
# copy first hdu
fits.copyHDU(infile, outfile);
# move to and then copy 2nd hdu
fits.movAbsHDU(infile,2);
fits.copyHDU(infile, outfile);

# close files
fits.closeFile(infile);
fits.closeFile(outfile);
```

3.3.2 matlab.io.fits.deleteHDU

`type = deleteHDU(file)`

Delete the current HDU and go to next HDU.

Returns the newly current HDU type as a string.

This is the equivalent of the cfitsio `fits_delete_hdu` function.

Inputs

file - opened fits file.

Outputs

type - string value for type of the next HDU.

3.3.3 matlab.io.fits.getHDUnum

`num = getHDUnum(file)`

Return the index of the current HDU.

This is the equivalent of the cfitsio `fits_get_hdu_num` function.

Inputs

file - opened fits file.

Outputs

num - current hdu number.

3.3.4 matlab.io.fits.getHDUoff

```
[headtstart, datastart, dataend] = getHDUoff(file)
```

Return offsets of the current HDU.

This is the equivalent of the cfitsio fits_get_hduoff function.

Inputs

file - opened fits file.

Outputs

headtstart, *datastart*, *dataend* - offset information for the current HDU.

3.3.5 matlab.io.fits.getHDUtype

```
type = getHDUtype(file)
```

Return the current HDUs type as a string.

This is the equivalent of the cfitsio fits_get_hdu_type function.

Inputs

file - opened fits file.

Outputs

type - current hdu type

3.3.6 matlab.io.fits.getNumHDUs

```
num = getNumHDUs(file)
```

Return the count of HDUs in the file.

This is the equivalent of the cfitsio fits_get_num_hdus function.

Inputs

file - opened fits file.

Outputs

num - return the number of HDUs in the file.

Examples

```
import_fits;
testname = file_in_loadpath("demos/tst0012.fits");
fd = fits.openFile(testname);
hducoun = getNumHDUs(fd, 5);
fits.closeFile(fd);
```

3.3.7 matlab.io.fits.movAbsHDU

```
type = movAbsHDU(file, hdunum)
```

Go to absolute HDU index *hdunum*

Returns the newly current HDU type as a string.

This is the equivalent of the cfitsio fits_movabs_hdu function.

Inputs

file - opened fits file.

hdunum - HDU number to move to.

Outputs

type - hdu type of the now current HDU.

3.3.8 matlab.io.fits.movNamHDU

`hdutype = movNamHDU(file, hdutype, extname, extver)`

Go to HDU matching *hdutype*, *extname*, *extver*.

This is the equivalent of the cfitsio fits_movnam_hdu function.

Inputs

file - opened fits file.

hdutype - HDU number to move to. Valid *hdutype* values are 'IMAGE_HDU', 'ASCII_TBL', 'BINARY_TBL', 'ANY_HDU'.

extname, *extver* - EXTNAME and EXTVER keywords to match.

Outputs

hdutype - hdu type of the now current HDU.

3.3.9 matlab.io.fits.movRelHDU

`type = movRelHDU(file, hdunum)`

Go to relative HDU index *hdunum*.

Returns the newly current HDU type as a string.

This is the equivalent of the cfitsio fits_movrel_hdu function.

Inputs

file - opened fits file.

hdunum - relative HDU number to move to.

Outputs

type - hdu type of the now current HDU.

3.3.10 matlab.io.fits.writeChecksum

`writeChecksum(file)`

Recalculate the HDU checksum and if required, write the new value.

This is the equivalent of the cfitsio fits_write_chksum function.

Inputs

file - opened fits file.

Outputs

None

3.4 Low Level Keyword Functions

3.4.1 matlab.io.fits.deleteKey

`deleteKey(file, key)`

Delete a key in the fits file.

This is the equivalent of the cfitsio `fits_delete_key` function.

Inputs

file - opened fits file.

key - Key name to remove.

Outputs

None

3.4.2 matlab.io.fits.deleteRecord

`deleteRecord(file, keynum)`

Delete a key in the fits file.

This is the equivalent of the cfitsio `fits_delete_record` function.

Inputs

file - opened fits file.

keynum - Record number to remove.

Outputs

None

3.4.3 matlab.io.fits.getHdrSpace

`[numkeys, freekeys] = getHdrSpace(file)`

Get the number of keyword records used and available.

This is the equivalent of the cfitsio `fits_get_hdrspace` function.

Inputs

file - opened fits file.

Outputs

numkeys - number of existing keys.

freekeys - number of free key space.

3.4.4 matlab.io.fits.readCard

`card = readCard(file, recname)`

Read the keyword card for name *recname*

This is the equivalent of the cfitsio `fits_read_card` function.

Inputs

file - opened fits file.

recname - record name to read

Outputs

card - unparsed record value string

3.4.5 matlab.io.fits.readKey

`[keyvalue, keycomment] = readKey(file, recname)`

Read the keyword value and comment for name *recname*.

This is the equivalent of the cfitsio `fits_read_key_str` function.

Inputs

file - opened fits file.

recname - keyword name.

Outputs

keyvalue - string value of record.

keycomment - comment string

3.4.6 matlab.io.fits.readKeyCmplx

`[value, comment] = readKeyCmplx(file, recname)`

Read the key value *recname* as a complex double.

This is the equivalent of the cfitsio `fits_read_key_dblcmp` function.

Inputs

file - opened fits file.

recname - keyword name.

Outputs

value - complex value of record.

comment - comment string

3.4.7 matlab.io.fits.readKeyDbl

`[value, comment] = readKeyDbl(file, recname)`

[Function File]

Read the key value *recname* as a double.

This is the equivalent of the cfitsio `fits_read_key_dbl` function.\n \

Inputs

file - opened fits file.

recname - keyword name.

Outputs

value - double value of record.

comment - comment string

3.4.8 matlab.io.fits.readKeyLongLong

`[value, comment] = readKeyLongLong(file, recname)`

Read the key value *recname* as a long long.

This is the equivalent of the cfitsio `fits_read_key_lnglng` function.

Inputs

file - opened fits file.

recname - keyword name.

Outputs

value - int64 value of record.

comment - comment string

3.4.9 matlab.io.fits.readKeyLongStr

```
[value, comment] = readKeyLongStr(file, recname)
```

Read the key value *recname* as a string.

This is the equivalent of the cfitsio fits_read_key_longstr function.

Inputs

file - opened fits file.

recname - keyword name.

Outputs

value - string value of record.

comment - comment string

3.4.10 matlab.io.fits.readKeyUnit

```
keyunit = readKeyUnit(file, recname)
```

Read the physical key units value *recname*.

This is the equivalent of the cfitsio fits_read_key_unit function.

Inputs

file - opened fits file.

recname - keyword name.

Outputs

keyunit - units value of record.

3.4.11 matlab.io.fits.readRecord

```
rec = readRecord(file, recidx)
```

Read the keyword record at *recidx*.

This is the equivalent of the cfitsio fits_read_record function.

Inputs

file - opened fits file.

recidx - record number.

Outputs

rec - full keyword record

3.4.12 matlab.io.fits.writeComment

`writeComment(file, comment)`

Append a comment to to the fits file.

This is the equivalent of the cfitsio `fits_write_comment` function.

Inputs

file - opened fits file.

comment - comment to append

Outputs

None

3.4.13 matlab.io.fits.writeDate

`writeDate(file)`

Write the date keyword.

This is the equivalent of the cfitsio `fits_write_date` function.

Inputs

file - opened fits file.

Outputs

None

3.4.14 matlab.io.fits.writeHistory

`writeHistory(file, history)`

Append a history to to the fits file.

This is the equivalent of the cfitsio `fits_write_history` function.

Inputs

file - opened fits file.

history - history string.

Outputs

None

3.4.15 matlab.io.fits.writeKey

`writeKey(file, key, value)`

`writeKey(file, key, value, comment)`

`writeKey(file, key, value, comment, decimals)`

Append or replace a key in the fits file.

This is the equivalent of the cfitsio `fits_write_key` and `fits_update_key` function.

Inputs

file - opened fits file.

key - keyword name.

value - keyword value.

comment - keyword comment.

decimals - number of decimals.

Outputs

None

3.4.16 `matlab.io.fits.writeKeyUnit`

`writeKeyUnit(file, key, unit)`

Write a key unit to the fits file.

This is the equivalent of the cfitsio `fits_write_key_unit` function.

Inputs

file - opened fits file.

key - keyword name.

unit - keyword units as string.

Outputs

None

3.5 Low Level Image Manipulation

3.5.1 `matlab.io.fits.createImg`

`createImg(file, bitpix, naxis)`

create a new primary image or image extension.

This is the equivalent of the cfitsio `fits_create_imgll` function.

Inputs

file - file previously opened with `openFile`, `openDiskFile` or `createFile`.

bitpix - type for the data as a string in either matlab or cfitsio naming.

naxis - axis values for the image.

Outputs

None

Examples

```
import_fits;
fd = fits.createFile("test.fits");
fits.createImg(fd,'int16',[10 20]);
fits.close(fd);
```

3.5.2 `matlab.io.fits.getImgSize`

`size = getImgSize(file)`

Return size of a Image HDU.

This is the equivalent of the cfitsio `fits_get_img_size` function.

Inputs

file - opened fits file.

Outputs

size - vector containing the image dimensions.

3.5.3 matlab.io.fits.getImgType

```
type = getImgType(file)
```

Return datatype of a Image HDU

This is the equivalent of the cfitsio fits_get_img_type function.

Inputs

file - opened fits file.

Outputs

type - datatype as a string for the image type.

3.5.4 matlab.io.fits.insertImg

```
insertImg(file, bitpix, naxis)
```

Insert a new primary image or image extension at current HDU position.

This is the equivalent of the cfitsio fits_insert_imgll function.

Inputs

file - file previously opened with openFile, openDiskFile or createFile.

bitpix - type for the data as a string in either matlab or cfitsio naming.

naxis - axis values for the image.

Outputs

None

3.5.5 matlab.io.fits.readImg

```
data = readImg(file)
```

```
data = readImg(file, firstpix, lastpix)
```

```
data = readImg(file, firstpix, lastpix, inc)
```

Read Image data.

This is the equivalent of the cfitsio fits_read_subset function.

Inputs

file - opened fits file.

firstpix - first pixel coordinate

lastpix - last pixel coordinate

inc - pixel increment

Outputs

data - image data read

Examples

```
import_fits;
filename = file_in_loadpath("demos/tst0012.fits");
fd = fits.openFile(filename);
# read the image
imagedata = fits.readImg(fd);
# read a 70x80 part of the image
imagedata = fits.readImg(fd, [11 11],[80 90]);
fits.closeFile(fd);
```

3.5.6 matlab.io.fits.setBscale

`setBscale(file, bscale, bzero)`

Reset bscale and bzero to be used with reading and writing Images.

This is the equivalent of the cfitsio `fits_set_bscale` function.

Inputs

file - opened fits file.

bscale - bscale value

bzero - bzero value

Outputs

None

3.5.7 matlab.io.fits.setTscale

`setTscale(file, col, scale, zero)`

Reset scale and zero to be used with reading and writing table data.

This is the equivalent of the cfitsio `fits_set_tscale` function.

Inputs

file - opened fits file.

col - column number

scale - scale value

zero - zero value

Outputs

None

3.5.8 matlab.io.fits.writeImg

`writeImg(file, imagedata)`

`writeImg(file, imagedata, fpixel)`

write imagedata to a FITS file. The rows and column size must match the size of NAXIS, NAXIS etc

This is the equivalent of the cfitsio `fits_write_subset` function.

Inputs

file - opened fits file.

imagedata - Image data.

fpixel - start pixel to write from.

Outputs

None

Examples

Create a fits file and write a 10x10 image in the primary and image ext:

```
import_fits;
fd = fits.createFile("myfitsfile.fits");
```

```
data = int16(zeros(10,10));
# primary
fits.createImg(fd,class(data), size(data));
fits.writeImg(fd,data);
# image ext
fits.createImg(fd,class(data), size(data));
fits.writeImg(fd,data);
# close file
fits.closeFile(fd);
```

3.6 Low Level Utility Functions

3.6.1 matlab.io.fits.getConstantNames

```
namelist = getConstantNames()
```

Return the names of all known fits constants.

Inputs

None

Outputs

namelist - cell array of all known fits constant names

See also: getConstantValue.

3.6.2 matlab.io.fits.getConstantValue

```
value = getConstantValue(name)
```

Return the value of a known fits constant.

Inputs

name - name of the constant to retrieve value of.

Outputs

value - value of the constant

See also: getConstantNames.

3.6.3 matlab.io.fits.getOpenFiles

```
files = getOpenFiles()
```

Get the file handles of all open fits files.

Inputs

None

Outputs

files list of opened fits file handles.

See also: openFile.

3.6.4 matlab.io.fits.getVersion

```
ver = getVersion()
```

Return the version number of the cfitsio library used.

This is the equivalent of the cfitsio fits_get_version function.

Inputs

file - opened fits file.

Outputs

ver - version

3.7 Low Level Compression Functions

3.7.1 matlab.io.fits.imgCompress

`imgCompress(infile, outfile)`

Copy HDU and image data from one infile to another, using the outfile's compression type. This is the equivalent of the cfitsio `fits_img_compress` function.

Inputs

infile - opened input fits file.

outfile - opened writable output fits file.

Outputs

None

3.7.2 matlab.io.fits.isCompressedImg

`comp = isCompressedImg(file)`

Return true if image is compressed.

This is the equivalent of the cfitsio `fits_is_compressed_image` function.

Inputs

file - opened fits file.

Outputs

comp - boolean for whether image is compressed or not.

3.7.3 matlab.io.fits.setCompressionType

`setCompressionType(file, comptype)`

Set compression type for writing FITS images.

This is the equivalent of the cfitsio `fits_set_compression_type` function.

Inputs

file - opened fits file.

comptype - compression type. Valid *comptype* values are: 'GZIP', 'GZIP2', 'RICE', 'PLIO', 'HCOMPRESS' or 'NOCOMPRESS'.

Outputs

None

3.7.4 matlab.io.fits.setHCompScale

`setHCompScale(file, scale)`

Set scale to be used with HCOMPRESS compression.

This is the equivalent of the cfitsio `fits_set_hcomp_scale` function.

Inputs

file - opened fits file.

scale - scale value

Outputs

None

3.7.5 matlab.io.fits.setHCompSmooth

`setHCompSmooth(file, smooth)`

Set smooth value to be used with HCOMPRESS compression.

This is the equivalent of the cfitsio fits_set_hcomp_smooth function.

Inputs

file - opened fits file.

smooth - smooth value

Outputs

None

3.7.6 matlab.io.fits.setTileDim

`setTileDim(file, tiledims)`

Set compression tile dims for writing FITS images.

This is the equivalent of the cfitsio fits_set_tile_dim function.

Inputs

file - opened fits file.

tiledims - tile dimensions

Outputs

None

3.8 Low Level Binary and ASCII Tables**3.8.1 matlab.io.fits.createTbl**

`createTbl(file, tbltype, nrows, ttype, tform)`

`createTbl(file, tbltype, nrows, ttype, tform, tunit)`

`createTbl(file, tbltype, nrows, ttype, tform, tunit, extname)`

Create a new ASCII or bintable extension.

This is the equivalent of the cfitsio fits_create_tbl function.

Inputs

file - opened fits file.

tbltype - table type 'binary' or 'ascii'.

nrows - initial number of rows (normally 0)

ttype - cell array of column type

tform - cell array of column format

tunit - cell array of column units

extname - optional extension name

ttype, *tform*, *tunit* are expected to be the same size.

Outputs

None

Examples

```
import_fits;
fd = fits.createFile("test.fits");
ttype = {'Col1', 'Col2', 'Col3', 'Col4'};
tform = {'A9', 'A4', 'A3', 'A8'};
tunit = {'m', 's', 'kg', 'km'};
fits.createTbl(fd, 'binary', 0, ttype, tform, tunit, 'table-name');
fits.closeFile(fd);
```

3.8.2 matlab.io.fits.deleteCol

`deleteCol(file, colnum)`

Delete a column from a table.

This is the equivalent of the cfitsio `fits_delete_col` function.

Inputs

file - opened fits file.

colnum - Column to delete from current table.

Outputs

None

3.8.3 matlab.io.fits.deleteRows

`deleteRows(file, firstrow, numrows)`

Delete rows from a table.

This is the equivalent of the cfitsio `fits_delete_rows` function.

Inputs

file - opened fits file.

firstrow - Start row to delete.

numrows - Number of rows to delete.

Outputs

None

3.8.4 matlab.io.fits.getAColParms

```
[ttype, tbc, tunit, tform, scale, zero, nulstr, tdisp] =
    getAColParms(file, colnum)
```

Get ASCII table parameters.

This is the equivalent of the cfitsio `fits_get_acolparms` function.

Inputs

file - opened fits file.

colnum - Column to retrieve.

Outputs

ttype, tbc, tunit, tform, scale, zero, nulstr, tdisp column information in same format as provided by `fits_get_acolparms`.

3.8.5 `matlab.io.fits.getBColParms`

```
[ttype, tunit, typechar, repeat, scale, zero, nulval, tdisp] =  
    getBColParms(file, colnum)
```

Get binary table parameters.

This is the equivalent of the `cfitsio fits_get_bcolparms` function.

Inputs

file - opened fits file.

colnum - Column to retrieve.

Outputs

ttype, tunit, typechar, repeat, scale, zero, nulval, tdisp column information in same format as provided by `fits_get_bcolparms`.

3.8.6 `matlab.io.fits.getColName`

```
[colnum, colname] = getColName(file, template)  
[colnum, colname] = getColName(file, template, casesens)
```

Get column name.

This is the equivalent of the `cfitsio fits_get_colname` function.

Inputs

file - opened fits file.

template - template string for matching column name.

casesens - boolean whether to be case sensitive in match.

Outputs

colnum - column number of match.

colname - column name of match.

Examples

```
import_fits;  
filename = file_in_loadpath("demos/tst0012.fits");  
fd = fits.openFile(filename);  
fits.movAbsHDU(fd,2);  
[colnum, colname] = fits.getColName(fd,"C*");  
# returned 3, "COUNTS"  
fits.closeFile(fd);
```

3.8.7 matlab.io.fits.getColType

```
[dtype,repeat,width] = getColType(file, colnum)
```

Get column type.

This is the equivalent of the cfitsio fits_get_coltypell function.

Inputs

file - opened fits file.

colnum - Column to delete from current table.

Outputs

dtype,repeat,width - column information.

3.8.8 matlab.io.fits.getEqColType

```
[dtype,repeat,width] = getEqColType(file, colnum)
```

Get column type.

This is the equivalent of the cfitsio fits_get_eqcoltypell function.

Inputs

file - opened fits file.

colnum - Column number.

Outputs

dtype,repeat,width - column type

3.8.9 matlab.io.fits.getNumCols

```
ncols = getNumCols(file)
```

Get number of columns.

This is the equivalent of the cfitsio fits_get_num_cols function.

Inputs

file - opened fits file.

Outputs

ncols - the number of columns in the table.

3.8.10 matlab.io.fits.getNumRows

```
nrows = getNumRows(file)
```

Get number of rows.

This is the equivalent of the cfitsio fits_get_numrowsll function.

Inputs

file - opened fits file.

Outputs

nrows - the number of rows in in the current table.

3.8.11 matlab.io.fits.getRowSize

`nrows = getRowSize(file)`

Get optimum number of rows to read/write at one time.

This is the equivalent of the cfitsio `fits_get_rowsize` function.

Inputs

file - opened fits file.

Outputs

nrows - number of rows.

3.8.12 matlab.io.fits.insertATbl

`insertATbl(file, rowlen, nrows, ttype, tbc col, tform)`

`insertATbl(file, rowlen, nrows, ttype, tbc col, tform, tunit)`

`insertATbl(file, tbltype, nrows, ttype, tbc col, tform, tunit, extname)`

Insert a new ASCII table after current HDU.

This is the equivalent of the cfitsio `fits_insert_atbl` function.

Inputs

file - opened fits file.

rowlen - row length. If set to 0, the function will calculate size based on *tbc col* and *ttype*.

nrows - initial number of rows (normally 0)

ttype - cell array of column type

tbc col - array containing the start indices for each column.

tform - cell array of column format

tunit - cell array of column units

extname - optional extension name

Outputs

None

3.8.13 matlab.io.fits.insertBTbl

`insertBTbl(file, nrows, ttype, tform, tunit, extname, pcount)`

Insert a new bintable extension.

This is the equivalent of the cfitsio `fits_insert_btbl` function.

Inputs

file - opened fits file.

nrows - initial number of rows (normally 0)

ttype - cell array of column type

tform - cell array of column format

tunit - cell array of column units

extname - optional extension name

pcount - heap size.

ttype, *tform*, *tunit* are expected to be the same size.

Outputs

None

3.8.14 matlab.io.fits.insertCol

`insertCol(file, colnum, ttype, tform)`

Insert a column into a table.

This is the equivalent of the cfitsio `fits_insert_col` function.

Inputs

file - opened fits file.

colnum - Column to delete from current table.

ttype, tform - column type to insert

Outputs

None

3.8.15 matlab.io.fits.insertRows

`insertRows(file, firstrow, numRows)`

Insert rows into a table.

This is the equivalent of the cfitsio `fits_insert_rows` function.

Inputs

file - opened fits file.

firstrow - Start row to insert from.

numrows - Number of rows to add.

Outputs

None

3.8.16 matlab.io.fits.readATblHdr

`[rowlen, nrows, ttype, tbc, tform, tunit, extname] = readATblHdr(file)`

Get ASCII table parameters.

This is the equivalent of the cfitsio `fits_read_atblhdrll` function.

Inputs

file - opened fits file.

Outputs

rowlen, nrows, ttype, tbc, tform, tunit, extname - table properties

3.8.17 matlab.io.fits.readBTblHdr

`[nrows, ttype, tform, tunit, extname, pcount] = readBTblHdr(file)`

Get Binary table parameters.

This is the equivalent of the cfitsio `fits_read_btblhdrll` function.

Inputs

file - opened fits file.

Outputs

nrows, ttype, tform, tunit, extname, pcount] - table properties

3.8.18 matlab.io.fits.readCol

```
[coldata, nullval] = readCol(file, colnum)
[coldata, nullval] = readCol(file, colnum, firstrow, numRows)
  Get table row data.
```

This is the equivalent of the cfitsio fits_read_col function.

Inputs

file - opened fits file.

firstrow - Start row

numrows - Number of rows to read

Outputs

coldata - the column data rows

nulldata - the null value flags

Examples

```
import_fits;

# open file
filename = file_in_loadpath("demos/tst0012.fits");
fd = fits.openFile(filename);

# move to binary table and get column for flux
fits.movAbsHDU(fd, 2);
colnum = fits.getColName(fd, 'flux');

# read all rows in column
fluxdata = fits.readCol(fd, colnum);
# read data starting at 2nd value
fluxdata = fits.readCol(fd, colnum, 2);
# read rows 3 rows starting at row 2
fluxdata = fits.readCol(fd, colnum, 2, 3);
fits.closeFile(fd);
```

3.8.19 matlab.io.fits.writeCol

```
writeCol(file, colnum, firstrow, data)
```

Write elements to a table.

This is the equivalent of the cfitsio fits_write_col function.

Inputs

file - opened fits file.

colnum - column number.

firstrow - first row number.

data - data to write to column

Outputs

None

3.9 Import functions

3.9.1 `import_fits`

`import_fits`

Import the fits functions into a `fits.xxxxx` variable, to emulate importing the fits namespace.

Appendix A GNU General Public License

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works. The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable

section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a

consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

- c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d. Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS”

WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.

```
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands `'show w'` and `'show c'` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

Index

B

Basic Usage Overview 2

C

closeFile 5
 copyHDU 8
 copyright 28
 createFile 5
 createImg 15
 createTbl 20

D

deleteCol 21
 deleteFile 6
 deleteHDU 8
 deleteKey 11
 deleteRecord 11
 deleteRows 21

F

fileMode 6
 fileName 6
 fitsdisp 3
 fitsinfo 3
 fitsread 4
 fitswrite 4
 Function Reference 3

G

getAColParms 21
 getBColParms 22
 getColName 22
 getColType 23
 getConstantNames 18
 getConstantValue 18
 getEqColType 23
 getHdrSpace 11
 getHDUnum 8
 getHDUoff 9
 getHDUtype 9
 getImgSize 15
 getImgType 16
 getNumCols 23
 getNumHDUs 9
 getNumRows 23
 getOpenFiles 18
 getRowSize 24
 getVersion 18

H

High Level File Functions 3

I

imgCompress 19
 Import functions 27
 import_fits 27
 insertATbl 24
 insertBTbl 24
 insertCol 25
 insertImg 16
 insertRows 25
 Installing and loading 1
 isCompressedImg 19

L

Loading 1
 Low Level Binary and ASCII Tables 20
 Low Level Compression Functions 19
 Low Level File Functions 5
 Low level functionality 2
 Low Level HDU Functions 7
 Low Level Image Manipulation 15
 Low Level Keyword Functions 11
 Low Level Utility Functions 18

M

movAbsHDU 9
 movNamHDU 10
 movRelHDU 10

O

Off-line install 1
 Online install 1
 openDiskFile 7
 openFile 7
 Overview 2

R

readATblHdr 25
 readBTblHdr 25
 readCard 11
 readCol 26
 readImg 16
 Reading Data 2
 Reading Information 2
 readKey 12
 readKeyCmplx 12
 readKeyDbl 12
 readKeyLongLong 12
 readKeyLongStr 13
 readKeyUnit 13
 readRecord 13

S

setBscale	17
setCompressionType	19
setHCompScale	19
setHCompSmooth	20
setTileDim	20
setTscale	17

U

Using the toolkit	2
-------------------------	---

W

warranty	28
Windows install	1
writeChecksum	10
writeCol	26
writeComment	14
writeDate	14
writeHistory	14
writeImg	17
writeKey	14
writeKeyUnit	15