

Space Telescope Imaging Spectrograph

IDL FITS Reader/Writer

D. Lindler
September 27, 1995

INTRODUCTION	1
READING FITS IMAGE FILES (THE BASICS)	2
RANDOM ACCESS OF AN INPUT FILE	3
BASIC USAGE OF THE FITS HEADER	4
EXTENSION/PRIMARY HEADER KEYWORDS	5
UPDATING FITS HEADERS	5
WRITING A FITS FILE.	7
APPENDING NEW EXTENSIONS TO AN EXISTING FITS FILE.	10
SCALED DATA IN FITS FILES	10
RANDOM GROUPS FORMAT FILES	11
IMPLICIT IMAGES	13
ERROR PROCESSING	14
READING ONLY A PORTION OF AN EXTENSION	14
WRITING UNSUPPORTED FITS EXTENSIONS	16
Appendix A: FITS Control Block	18
Appendix B: FITS_OPEN	19
Appendix C: FITS_HELP	20
Appendix D: FITS_READ	21
Appendix E: FITS_WRITE	24
Appendix F: FITS_CLOSE	26

Prepared under contract NAS5-32833 between Advanced Computer Concepts, Inc. and NASA/ Goddard Space Flight Center, Greenbelt, Maryland.

INTRODUCTION

The Flexible Image Transport System (FITS) format will be used for storing STIS science data and calibration reference data. Multiple images can be stored in a single file using FITS Image extensions. The basic layout of the fits file with extensions will be:

- Header (for the primary data unit, PDU)
- Primary data unit or image (this will normally be null for STIS data sets)
- Header for extension 1
- Image or data unit 1
- Header for extension 2
- Image or data unit 2
- ...
- Header for extension N
- Image or data unit N

The primary data unit (PDU) for STIS data will usually be a Null Image, identified by the keyword value `NAXIS = 0` in its header. The PDU header will contain other keyword parameters and history records that are applicable to all of the extension images. Keywords and history that are extension dependent will be stored in the extension headers. The multiple extensions may contain multiple readouts with the same instrument configuration (e.g. CR-SPLITS) and multiple data types for a single readout (i.e. a STIS image, an error image, and a data quality image).

The following IDL routines have been written for reading and writing these FITS data sets.

- FITS_OPEN - routine to open a FITS file
- FITS_HELP - print summary of the file contents
- FITS_READ - routine to read an extension
- FITS_WRITE - routine to write an extension
- FITS_CLOSE - routine to close a file

Routines available for manipulation of the FITS headers include:

- SXPAR - extract keyword values
- SXADDPAR - add or update keyword values
- SXDELPAR - delete keywords
- SXADDHIST - add history records
- HPRINT - print the header

READING FITS IMAGE FILES (THE BASICS)

To read a FITS file it must first be opened using:

```
FITS_OPEN, filename, FCB
```

where filename is the name of the file and FCB (FITS Control Block) is an output IDL structure variable containing information on the file. The user normally does not need to be concerned with the contents of FCB. However, FCB must be passed to subsequent routines. For more sophisticated users, the complete contents of the FCB is listed in Appendix A.

Once the file is opened, it can be read with FITS_READ:

```
FITS_READ, FCB, data, header
```

This will read the next extension in the file. On the first call it will read the PDU. If the PDU is a null image, the first call will read the first extension. Subsequent calls will read the following extensions in order. For users wishing to read all extensions in the file, the NEXTEND item of the FITS Control Block (FCB.NEXTEND) contains the number of extensions in the file.

Once you are finished with the file, it should be closed using FITS_CLOSE:

```
FITS_CLOSE, FCB
```

Read the first two images in a FITS image file

```
FITS_OPEN,'myfile.fits',FCB
FITS_READ,FCB,image1,header1
FITS_READ,FCB,image2,header2
FITS_CLOSE,FCB
```

Example: read and process all of the extensions in the file "myfile.fits"

```
FITS_OPEN, 'myfile.fits', FCB
n = fcb.nextend           ;number of extensions
for i = 0, n do begin     ;i=0 reads PDU
    FITS_READ,FCB, image, header
    Process the image
end
FITS_CLOSE,FCB
```

In many cases, the FITS file will contain only a single image. When running interactively, it is not convenient to call `FITS_OPEN`, `FITS_READ`, and then `FITS_CLOSE` to read the image. An alternative approach is to supply a filename to `FITS_READ` instead of the FCB.

```
FITS_READ, filename, image, header
```

In this case, `FITS_READ` will open the file, read the PDU (or if the PDU is null, the first extension), and close the file.

RANDOM ACCESS OF AN INPUT FILE

In many cases, you may not want to read the extensions in order, or you may want to skip extensions. The easiest way to do this is with the `EXTEN_NO` keyword parameter to `FITS_READ`. To read extension N of a file, type:

```
FITS_OPEN, filename, FCB  
FITS_READ, FCB, image, header, exten_no = N  
FITS_CLOSE, FCB
```

or equivalently

```
FITS_READ, filename, image,header, exten_no = N
```

To read the PDU, use `EXTEN_NO = 0`. If you are reading multiple extensions from the same data file, it is more efficient to open the file with `FITS_OPEN`, read the extensions, and then close the file with `FITS_CLOSE`. If you supply a filename to `FITS_READ`, the file will be opened and closed each time the routine is called.

An alternate method to read extensions is to use standard header keyword values that identify the extensions. These include:

```
XTENSION - extension type (e.g.. IMAGE, TABLE, etc.)  
EXTNAME - extension name (name of the extension, e.g. SCI, DQ, or ERR)  
EXTVER - sequential version number 1, 2, 3, etc.  
EXTLEVEL - extension level number
```

The `XTENSION` keyword is required in all extension headers. The other three are optional. In the current design of STIS data sets, `EXTNAME` and `EXTVER` will be used for identifying extensions.

```
READ_FITS examples  
Read a simple fits file, 'image.fits', containing an image.  
FITS_READ, 'image.fits', image, header  
  
Read first TABLE extension in fitstab.fits.  
FITS_OPEN, 'fitstab.fits', fcb  
READ_FITS, fcb, image, header, xtension = 'TABLE'  
  
Read first extension with EXTNAME = SCI and EXTVER = 5  
READ_FITS, fcb, image, header, extname = 'sci', extver = 5  
FITS_CLOSE, fcb
```

Searches for the correct EXTNAME and XTENSION are case insensitive. If you are running interactively and you do not know what extensions are in the file use:

```
FITS_HELP, FCB  
or  
FITS_HELP, filename
```

If you are writing a procedure that needs to know what extensions are in the file, use the information in the FITS control block (See Appendix A).

BASIC USAGE OF THE FITS HEADER

The output header from FITS_READ is a string array with 80 character records of the form:

```
<KEYWORD NAME> = VALUE
```

and COMMENT and HISTORY records. From IDL, you can print the header simply by typing:

```
PRINT, header
```

or better by typing:

```
HPRINT, header
```

To extract data values from the header into IDL variables, use the function SXPARG:

```
VALUE = SXPARG(header, keyword)
```

where KEYWORD is in the name of the keyword parameter. If the specified keyword is not in header then the system variable !ERR will be set to -1.

Example using SXPART to extract a header keyword value

```
FITS_READ,'mfile.fits',data,header  
det = sxpar(header, 'DETECTOR')  
if !err lt 0 then print,'Keyword DETECTOR not found'
```

EXTENSION/PRIMARY HEADER KEYWORDS

STIS data files will have a null primary data unit and the images following in "IMAGE" extensions. The primary data unit header will have keyword values and history which are applicable to all of the images in the file. Each extension header will have keyword values that change from one extension to another. When reading an image extension, FITS_READ combines header keywords from the primary data unit and the extension into a single output header. This avoids the need for internally maintaining and passing two separate headers to IDL routines. The structure of the output header of FITS_READ is:

*Required FITS keywords for the extension
(XTENSION, BITPIX, NAXIS, etc.)*

```
BEGIN MAIN HEADER -----  
  keywords from the primary data unit header  
BEGIN EXTENSION HEADER -----  
  keywords from the extension header  
END
```

In the event that duplicate keywords are in the primary and extension headers, SXPART will retrieve the last occurrence of the keyword value (the extension header keyword value) and print a warning that duplicate keyword names were found. If the PDU value of the duplicate keyword is needed, you can obtain it by reading the primary data unit with FITS_READ or by accessing the value in FCB by:

```
value = sxpar(fcb.hmain, keyword)
```

In the event that you may have a lot of duplicate keywords, you can read the extension header without the PDU header keywords included by adding the parameter, /NO_PDU, to the call to FITS_READ:

```
FITS_READ, FCB, image, header, /NO_PDU
```

UPDATING FITS HEADERS

Two routines can be used to update the FITS header, SXADDPAR and SXADDHIST. To update an existing keyword value or to add a new keyword value use:

SXADDPAR, header, keyword, value, comment

where KEYWORD is the name of the keyword (up to 8 characters) and VALUE is the value to be assigned to the keyword. Comment is an optional comment to be added to the header.

To add a single history line, or a string array containing multiple history lines to the header use:

SXADDHIST, history, header

where HISTORY is a string or string array containing the history.

The keyword parameter /PDU can be used to add a new keyword parameter or new history record to the primary data unit portion of a header.

SXADDPAR, header, keyword, value, /PDU
SXADDHIST, history, header, /PDU

If a keyword already exists in the extension header, it can not be added to the main primary data unit header. If you need to move a keyword from the extension header to the primary data unit portion of the header, the keyword must first be deleted from the header.

SXDELPAR, header, keyword.

Examples of updating a header stored in variable H.

Update the value of ROT1.

```
sxaddpar, h, 'ROT1', 55.5
```

Add new keyword parameter with a comment to the extension portion of the header.

```
sxaddpar, h, 'PROCDATE', 'August 20, 1995', 'I added this myself'
```

Add some history to the PDU portion of the header.

```
sxaddhist, 'Data Processed by procedure BIGPRO', h, /PDU
```

Move a keyword from the extension portion of the header to the PDU portion.

```
value = sxpar(h, 'MSM')           ; get value  
sxdelpar, h, 'MSM'                ; delete it from the extension portion  
sxaddpar, h, 'MSM', value, /PDU    ; add it to the PDU portion
```

WRITING A FITS FILE.

To create a FITS file it must first be opened for writing using `FITS_OPEN`.

```
FITS_OPEN, filename, fcb, /WRITE
```

The keyword `/WRITE` indicates that a new file is to be created. To write an image with the minimal required header, use:

```
FITS_WRITE, fcb, data  
FITS_CLOSE, fcb
```

The above will write the image into the primary data unit. This format can be read by virtually all FITS readers in other reduction systems. If `DATA` is not supplied or has a scalar value, `FITS_WRITE` will write a null image (i.e. `NAXIS=0`). If you want to add additional keywords to the output file's header, use:

```
FITS_WRITE, fcb, data, header
```

where `HEADER` contains the keywords to be added. `FITS_WRITE` will automatically insert or update the necessary required keywords in `HEADER`.

To write the image as the first extension instead of the PDU use:

FITS_WRITE, fcb
FITS_WRITE, fcb, image

The first call will write a null primary data unit. The second call will write the image to an "IMAGE" extension. The default extension type for FITS_WRITE is "IMAGE". To write other types of extensions, you can either supply a header to FITS_WRITE containing the keyword XTENSION or use the optional parameter XTENSION in the call to FITS_WRITE.

FITS_WRITE, fcb, image, /xtension = '<extension type>'

If the first call to FITS_WRITE after FITS_OPEN is for an extension, FITS_WRITE will automatically write a null PDU before writing the extension. If a header is supplied, FITS_WRITE will automatically separate the PDU header parameters and the extension header parameters. Additional calls to FITS_WRITE can then be used to write additional extensions. Any PDU header parameters supplied in a call to FITS_WRITE after the PDU has been written by a previous call will be ignored.

When all extensions are written, FITS_CLOSE should be used to close the file.

FITS_CLOSE, fcb

Examples of writing a single image, A, to a FITS file.

1) *A very basic file with A stored in the Primary Data Unit and no extensions.*

```
FITS_OPEN, 'newfile.fits', fcb, /write  
FITS_WRITE, fcb, a  
FITS_CLOSE, fcb
```

2) *A shorter method for example 1. FITS_WRITE opens and closes the file.*

```
FITS_WRITE, 'newfile.fits', a
```

3) *Write the image with some addition user supplied header information.*

```
h = ['END          '] ;null header (8 char.)  
SXADDPAR, h, 'targetid', 'ZETA-OPH'  
SXADDHIST, 'This was processed by routine process on '+!stime, h  
FITS_WRITE, 'newfile.fits', a, h
```

4) *Write A into an image extension instead of the primary data unit.*

```
FITS_OPEN, 'newfile.fits', fcb, /write  
FITS_WRITE, fcb ;null primary data unit  
FITS_WRITE, fcb, a ;image extension  
FITS_CLOSE, fcb
```

5) *A shorter way to do the same thing as number 4.*

```
FITS_WRITE, 'newfiles.fits', a, xtension = 'IMAGE'
```

6) *Write A to an image extension with a header that has separate PDU and EXTENSION keyword parameters. On the first call to FITS_WRITE, a scalar is supplied and a null image will be written with the PDU keywords extracted*

from

the header. On the second call, the data will be written with the extension keywords extracted from the header.

```
FITS_OPEN, 'newfile.fits', fcb, /write  
FITS_WRITE, fcb, 0, header  
FITS_WRITE, fcb, a, header  
FITS_CLOSE, fcb
```

7) *Short way to do number 6.*

```
FITS_WRITE, 'newfile.fits', a, header, xtension = 'IMAGE'
```

When writing multiple images to an output data set, XTENSION, EXTNAME, EXTVER, and EXTLEVEL can be used for identification. STIS currently plans to use EXTNAME and

EXTVER. The values written to an output file can come from two sources, the header supplied to FITS_WRITE, or keyword parameters in the call to FITS_WRITE.

```
FITS_WRITE, fcb, data, header, xtension = value, extname = value, extver = value, $
    extlevel = value
```

If XTENSION, EXTNAME, EXTVER, or EXTLEVEL, are present in both the header supplied to FITS_WRITE and as a keyword parameter in the calling sequence of FITS_WRITE, the calling sequence parameter will take precedence. If EXTNAME, EXTVER, or EXTLEVEL are not supplied by either method, the keywords will not be placed into the output header. If XTENSION is not supplied, it will have a value of "IMAGE" in the output file.

Examples for identifying FITS extensions in an output FITS file.

1) Extension identification supplied in the call to FITS_WRITE. XTENSION keyword is not supplied and will have the default value of "IMAGE"

```
FITS_OPEN, 'outfile.fits', fcb, /write
FITS_WRITE, fcb, FLUX1, extname = 'FLUX', extver = 1
FITS_WRITE, fcb, ERR1, extname = 'ERR', extver = 1
FITS_WRITE, fcb, DQ1, extname = 'DQ', extver = 1
FITS_WRITE, fcb, FLUX2, extname = 'FLUX', extver = 2
FITS_WRITE, fcb, ERR2, extname = 'ERR', extver = 2
FITS_WRITE, fcb, DQ2, extname = 'DQ', extver = 2
FITS_CLOSE, fcb
```

2) Process an input file and keep same extension identification

```
FITS_OPEN, 'infile.fits', fcbin
FITS_OPEN, 'outfile.fits', fcbout, /write

for i = 1, fcbin.nextend do begin
    FITS_READ, fcbin, data, header

    <process the data and header>

    FITS_WRITE, fcbout, data, header
endfor

FITS_CLOSE, fcbin
```

APPENDING NEW EXTENSIONS TO AN EXISTING FITS FILE.

Additional extensions can be added to an existing FITS file by opening the file with the APPEND keyword parameter.

```
FITS_OPEN, filename, fcb, /APPEND
```

Once a file is opened for APPEND, FITS_WRITE can be used to add additional extensions. The primary data unit and existing extensions remain unchanged.

Example: Adding an extension to an existing FITS file.

```
FITS_OPEN, 'myfile.fits', fcb, /append  
FITS_WRITE, fcb, data, header  
FITS_CLOSE, fcb
```

SCALED DATA IN FITS FILES

In the old days, floating point data were stored in FITS data files as scaled integers with the formula:

$$\text{Data_Value} = \text{BZERO} + \text{BSCALE} * \text{File_Value}$$

where BZERO, and BSCALE are header parameters, File_Value is the integer data value stored in the FITS file, and Data_Value is the unscaled data value. FITS_READ automatically scales the data to the floating point values when BZERO and BSCALE are present in the header. If you want the data left as unscaled integers, use the NOSCALE parameter to FITS_READ.

```
FITS_READ, fcb, data, header, /noscale
```

DATA will be returned in the original scaled form and HEADER will contain the values of BSCALE and BZERO. The data can be unscaled any time in IDL with.

$$\text{Data} = \text{sxpar}(\text{header}, \text{'BZERO'}) + \text{sxpar}(\text{header}, \text{'BSCALE'}) * \text{data}$$

FITS_WRITE does not perform scaling of the data before writing. Floating point data is written into the output file in the standard FITS IEEE format. If you wish to write the data as scaled integers, you must scale it before calling FITS_WRITE and add the scaling parameters to the header. The reverse process is:

$$\text{File_Value} = (\text{Data_Value} - \text{BZERO}) / \text{BSCALE}$$

Example: Writing an image as scaled integers

```
image = ROUND((image - 1000.0) / 25.0) ; 32 bit integers
image = FIX(image) ; 16 bit integers
SXADDPAR, header, 'BZERO', 1000.0
SXADDPAR, header, 'SCALE', 25.0
FITS_WRITE, 'scaled.fits', image, header
```

RANDOM GROUPS FORMAT FILES

FITS_READ can be used to read random group formatted FITS data files which have the following header keywords:

```
GROUPS = T
GCOUNT = number of groups
PCOUNT = number of parameters preceding each group
```

Random group data consists of a set of groups with the number of groups specified by GCOUNT. Each group consists of a number of parameters specified by PCOUNT followed by the data array with size specified by NAXIS, NAXIS1, etc. To read the Nth group of a FITS file use:

```
FITS_READ, fcb, data, header, group_par, group = N
```

where keyword GROUP specifies the group number to read (starting at 0 for the first group) and GROUP_PAR is a output vector containing the PCOUNT group parameters. If GROUP is not specified for a GROUP formatted file, the first group is returned. Three additional keywords can be used to describe each group parameters.

```
PTYPEn - character string giving the name of the Nth group parameter.
PSCALn - scale factor for the Nth parameter.
PZEROn - zero point for the Nth parameter.
```

FITS_READ does not apply the scale factors to the parameters. If needed, you can apply them by the formula:

$$\text{VALUE} = \text{PZEROn} + \text{PSCALn} * \text{group_parameter_value}$$

where group_parameter_value is the value recorded in the file and VALUE is the unscaled data value. It is important to note that the n in PTYPEn, PSCALn, and PZEROn starts at 1 and refers to the $n-1$ element of the IDL GROUP_PAR vector which starts with subscript 0.

Reading group 5 of a Random Groups formatted FITS file and printing the header parameters.

```
FITS_OPEN, 'groupfile.fits', fcb
FITS_READ, fcb, data, header, gpar, group = 5
FITS_CLOSE, fcb
pcount = sxpar(header, 'PCOUNT')

for i=1,pcount - 1 do begin
  string_i = strtrim(i,2)
  pscale = sxpar(header, 'PSCAL'+string_i)
  if !err lt 0 then pscale = 1.0
  pzero = sxpar(header, 'PZERO'+string_i)
  if !err lt 0 then pzero = 0.0
  print, pzero + pscale * gpar(i-1)
end
```

When reading multiple groups from the same file, It is inefficient to reread the header with each group. keyword /DATA_ONLY can be used in the call to FITS_READ to avoid rereading the header.

```
FITS_READ, fcb, data, header, gpar, /DATA_ONLY
```

Likewise if you want to only read the header:

```
FITS_READ, fcb, data, header, /HEADER_ONLY
```

In this call DATA is only a place holder in the calling sequence and is not read.

Looping on groups in a random groups formatted FITS file:

```
FITS_OPEN, 'groupfile.fits', fcb
FITS_READ, fcb, image, h, /HEADER_ONLY, exten_no = 0
n = sxpar(header, 'GCOUNT')
for i = 0, n-1 do begin
  FITS_READ, fcb, image, h, gpar, /DATA_ONLY, exten_no = 0
  <process the image>
end
```

It is also important when reading multiple groups is to explicitly specify the extension number to `FITS_READ`. Otherwise, `FITS_READ` will try to read the second group from the next extension.

`FITS_WRITE` does not support the writing of group formatted files. The use of random groups formatted files is not recommended. Better approaches to writing the data is to use FITS binary tables or FITS IMAGE extensions. Group parameters can then be put into columns of the binary table or as header parameters of the image extensions. If you must use random groups formatted output files, refer to the section, WRITING UNSUPPORTED FITS EXTENSIONS, for a very painful way of doing it.

IMPLICIT IMAGES

In some cases STIS will use "Implicit Images" in a FITS file. For example, if all of the data quality values for a raw image are 0 (i.e. no telemetry errors), it is a waste of disk space to store a 2048 x 2048 image of zeros. In this case an implicit image will be used as identified by header parameter `NAXIS=0` and three keyword parameters specifying an image with a constant value.

`NPIX1` - number of samples in the image
`NPIX2` - number of lines in the image
`PIXVALUE` - value of each pixel in the image

When this type of FITS header is encountered, `FITS_READ` will create an image with the size specified by `NPIX1` and `NPIX2` and a constant value specified by `PIXVALUE`. The fact that the image was not in the file is transparent to the user.

To write an implicit image, the user must populate the keywords `NPIX1`, `NPIX2`, and `PIXVALUE` in a new or existing header and call `FITS_WRITE` with a scalar supplied for the data.

Example: writing an implicit image to a FITS file. A raw image is written with an extension name 'SCI' followed by an implicit data quality image containing all zeros.

```
FITS_OPEN, 'newfile.fits', fcb,/write
FITS_WRITE, fcb, data, header, extname = 'SCI', extver=1
SXADDPAR, header, 'NPIX1', 1024
SXADDPAR, header, 'NPIX2', 1024
SXADDPAR, header, 'PIXVALUE', 0
FITS_WRITE, fcb, 0, header, extname = 'DQ', extver = 1
FITS_CLOSE, fcb
```

ERROR PROCESSING

When FITS_OPEN, FITS_READ, FITS_WRITE, or FITS_CLOSE encounters an irrecoverable error when processing a file, they print an error message and issue a RETALL. If you would prefer to handle the errors yourself (which should be the case if using these routines inside of a widget), a NO_ABORT keyword can be supplied to any of the routines. If NO_ABORT is specified, the FITS routines return control to the calling program along with the error message in a keyword parameter MESSAGE.

```
FITS_<xxxx>, ..., /NO_ABORT, MESSAGE = MESSAGE
```

where <xxxx> is OPEN, READ, WRITE, or CLOSE.

Sample code to interactively ask user for a FITS file name and read the image. If an error occurs the user is given another chance.

```
filename = "
start: read, 'Name of the FITS file?', filename
      FITS_READ,filename,data,header, /no_abort, message = text
;
; check for error reading the file
;
      if !err lt 0 then begin
          print, text
          goto,start
      endif
```

READING ONLY A PORTION OF AN EXTENSION

In some cases you may want to read only a portion of a FITS image or an extension. Two header parameters to `FITS_READ`, `FIRST` and `LAST`, allow you to specify which words of the extension to read. When `FIRST` and `LAST` are specified, `FITS_READ` returns the specified range of data as a one-dimensional vector (with the correct data type as specified by `BITPIX`). For partial reads, `FITS_READ` applies no scale factors (`BSCALE` and `BZERO`). Some applications of `FIRST` and `LAST` may be to:

- Read a selected range of lines in an image.
- Read a single image from a data cube
- Read a selected range of rows in a FITS binary table
- Read the heap area of a FITS extension.
- Read data from a non-standard extension.

Since `FITS_READ` returns the data as a vector, the IDL `REFORM` function may be needed to properly format multi-dimension results. The `NAXIS`, `AXIS`, and `PCOUNT` fields of the `FCB` structure (Appendix A) are useful for this task.

You must explicitly tell `FITS_READ` which extension to read when calling `FITS_READ` multiple times to read portions of the same extension. Otherwise, after each call to `FITS_READ`, it will attempt to read the next extension. Also when calling `FITS_READ` multiple times for a single extension, it is inefficient to read the extension header each time. The `/DATA_ONLY` keyword parameter can be set to skip reading the header. If you need the header, you can either read it on the first data read for the extension or you can call `FITS_READ` with the `/HEADER_ONLY` keyword to read the header separately.

Read and process each image in a data cube (one at a time). The first call to FITS_READ is used to read the header only.

```
FITS_OPEN, 'cube.fits', fcb
FITS_READ, fcb, data, header, exten_no=0, /header_only
ns = fcb.axis(0,0)
nl = fcb.axis(1,0)
nimages = fcb.axis(2,0)
;
; loop on images
;
  for i = 0, nimages-1 do begin
;
; read the data for the ith cube, don't read the header
;
    first = ns * nl * i
    last = first + ns*nl - 1
    FITS_READ, fcb, data, exten_no=0, /data_only, first=first, last=last
;
; Reformat the data as a 2-D image
;
    data = REFORM(data, ns, nl, /OVERWRITE)

    <process the data>

  endfor ; loop on images in the cube.
```

Example: Reading the heap area of a binary table

```
FITS_OPEN, 'btable.fits', fcb
;
; read the table
;
FITS_READ, fcb, table, header, exten_no = 2
;
; read the heap area
;
i1 = fcb.axis(0,2) * fcb.axis(1,2)
i2 = i1 + fcb.pcount(2) - 1
FITS_READ, fcb, heap, exten_no = 2, first = i1, last = i2, /data_only
FITS_CLOSE, fcb
```

WRITING UNSUPPORTED FITS EXTENSIONS

FITS_WRITE will only write a PDU or EXTENSION when PCOUNT is set to zero and the data in the whole extension is in an IDL variable. However, FITS_WRITE will allow the user to manually write an unsupported extension. To accomplish this the user must first create an extension header (which must be valid and must reflect the actual amount of data to be written). The header is written using WRITE_FITS and the /NO_DATA keyword parameter.

```
WRITE_FITS, fcb, 0, header, /NO_DATA
```

The data must be converted to external IEEE format and then written using WRITEU to the logical unit specified in FCB.UNIT.

```
HOST_TO_IEEE, data
WRITEU, fcb.unit, data
```

Multiple calls to WRITEU can be used for a single extension, provided the data are written in the correct order. When WRITE_FITS is called to begin a new extension, it will automatically pad the previous extension to complete a 2880 byte record.

This method of using WRITE_FITS can also be used to write a random group formatted FITS file. It is recommended that the use of this type of file is avoided.

Write a binary table to a FITS file. The table and heap area is in a byte array and *HOST_TO_IEEE* is not needed to convert it to external format. The header must be created by the user.

```
FITS_OPEN, 'btable.fits', fcb, /write
FITS_WRITE, fcb                      ;write null PDU
FITS_WRITE, fcb, 0, header, /no_data ;write user created header
WRITEU, fcb.unit, table               ;write the table
WRITEU, fcb, unit, heap               ; write the heap area
```

Example: writing a random groups formatted FITS file with the group parameters stored in the 2-D array *GPAR* where the second subscript gives the group number and the *DATA* arrays are stored in a 3-D array where the third subscript is the group number.

```
FITS_OPEN, 'outfile.fits', fcb, /write
size_data = size(data)
size_gpar = size(gpar)
h = ['END      ']
sxaddpar, h, 'SIMPLE', 'T'
sxaddpar, h, 'BITPIX', 32                ;longword data
sxaddpar, h, 'NAXIS', 2                  ;2-D images
sxaddpar, h, 'NAXIS1', size_data(1)
sxaddpar, h, 'NAXIS2', size_data(2)
sxaddpar, h, 'GROUPS', 'T', 'Random group data file'
sxaddpar, h, 'PCOUNT', size_gpar(1), 'Number of group parameters'
sxaddpar, h, 'GCOUNT', size_data(3), 'Number of groups'
    <optionally add PTYPEn, PSCALEn, PZERON for each paramter
    and any other desired keyword parameters or history>
;
; convert data to external format
;
    HOST_TO_IEEE, data
    HOST_TO_IEEE, gpar
;
; loop on groups
;
    for i=0,size_data(3)-1 do begin
        writeu,fcb.unit,gpar(*,i)
        writeu,fcb.unit,data(*,*,i)
    end
```

Appendix A

FITS CONTROL BLOCK

The FITS control block, FCB, returned by FITS_OPEN and used by FITS_READ, FITS_WRITE, FITS_HELP, and FITS_CLOSE, is an IDL structure. When FITS_OPEN is called without the /WRITE or /APPEND parameters, FCB contains:

FCB.FILENAME - name of the input file
.UNIT - unit number the file is opened to
.NEXTEND - number of extensions in the file.
.XTENSION - string array giving the extension type for each extension.
.EXTNAME - string array giving the extension name for each extension.
(null string if not defined the extension)
.EXTVER - vector of extension version numbers (0 if not defined)
.EXTLEVEL - vector of extension levels (0 if not defined)
.GCOUNT - vector with the number of groups in each extension.
.PCOUNT - vector with parameter count for each group
.BITPIX - bits per pixel for each extension with values
8 byte data
16 short word integers
32 long word integers
-32 IEEE floating point
-64 IEEE double precision floating point
.NAXIS - number of axes for each extension. (0 for null data units)
.AXIS - 2-D array where axis(*,N) gives the size of each axes for extension N
.START_HEADER - vector giving the starting byte in the file where each extension header begins
.START_DATA - vector giving the starting byte in the file where the data for each extension begins
.HMAIN - keyword parameters (less standard required FITS keywords) for the primary data unit.
.OPEN_FOR_WRITE - flag (0= open for read, 1=open for write, 2=open for update)
.LAST_EXTENSION - last extension number read.
.RANDOM_GROUPS - 1 if the PDU is random groups format, 0 otherwise

When FITS open is called with the /WRITE or /APPEND option, FCB contains:

FCB.FILENAME - name of the input file
.UNIT - unit number the file is opened to
.NEXTEND - number of extensions in the file.
.OPEN_FOR_WRITE - flag (1=open for write, 2=open for update)

Appendix B FITS_OPEN

Opens a FITS (Flexible Image Transport System) data file.

CALLING SEQUENCE:

FITS_OPEN, filename, fcb

INPUTS:

FILENAME : name of the FITS file to open

OUTPUTS:

FCB : (FITS Control Block) a IDL structure containing information concerning the file. It is an input to FITS_READ, FITS_WRITE and FITS_CLOSE

KEYWORD PARAMETERS:

/WRITE: Set this keyword to open a new file for writing.

/APPEND: Set to append to an existing file.

/NO_ABORT: Set to return to calling program instead of a RETALL when an I/O error is encountered. If set, the routine will return with !err=-1 and a message in the keyword MESSAGE. If not set, FITS_OPEN will print the message and issue a RETALL

MESSAGE = value: Output error message

/HPRINT - print headers with routine HPRINT as they are read.
(useful for debugging a strange file)

NOTES:

The output FCB should be passed to the other FITS routines (FITS_OPEN, FITS_READ, FITS_HELP, and FITS_WRITE). It has the structure given in Appendix A.

EXAMPLES:

Open a FITS file for reading:
FITS_OPEN,'myfile.fits',fcb

Open a new FITS file for output:
FITS_OPEN,'newfile.fits',fcb,/write

Appendix C

FITS_HELP

To print a summary of the primary data units and extensions in a FITS file.

CALLING SEQUENCE:

FITS_HELP,filename_or_fcb

INPUTS:

FILENAME_OR_FCB - name of the fits file or the FITS Control Block (FCB)
returned by FITS_OPEN.

OUTPUTS:

a summary of the fits file is printed.

EXAMPLES:

FITS_HELP,'myfile.fits'

FITS_OPEN,'anotherfile.fits',fcb
FITS_HELP,fcb

Appendix D FITS_READ

Reads a FITS file.

CALLING SEQUENCE:

FITS_READ, filename_or_fcb, data [,header, group_par]

INPUTS:

FILENAME_OR_FCB - this parameter can be the FITS Control Block (FCB) returned by FITS_OPEN or the file name of the FITS file. If a file name is supplied, FITS_READ will open the file with FITS_OPEN and close the file with FITS_CLOSE before exiting. When multiple extensions are to be read from the file, it is more efficient for the user to call FITS_OPEN and leave the file open until all extensions are read.

OUTPUTS:

DATA - data array. If /NOSCALE is specified, BSCALE and BZERO (if present in the header) will not be used to scale the data. If keywords FIRST and LAST are used to read a portion of the data or the heap portion of an extension, no scaling is done and data is returned as a 1-D vector. The user can use the IDL function REFORM to convert the data to the correct dimensions if desired. If /DATA_ONLY is specified, no scaling is done.

HEADER - FITS Header. If an extension is read, and the /NO_PDU keyword parameter is not supplied, the primary data unit header and the extension header will be combined. The header will have the form:

```
<required keywords for the extension: XTENSION, BITPIX,
NAXIS, ...>
BEGIN MAIN HEADER -----
<PDU header keyword and history less required keywords:
    SIMPLE, BITPIX, NAXIS, ...>
BEGIN EXTENSION HEADER -----
<extension header less required keywords that were
    placed at the beginning of the header.
END
```

The structure of the header is such that if a keyword is duplicated in both the PDU and extension headers, routine SXPAR will print a warning and return the extension value of the keyword. SXADDPAR and SXADDHIST will add new keywords and history to the extension portion of the header unless the

parameter /PDU is supplied in the calling sequence.
GROUP_PAR - Group parameter block for FITS random groups format files
Any scale factors in the header (PSCALEn and PZEROn) are not applied to the group parameters.

KEYWORD PARAMETERS:

/NOSCALE: Set to return the FITS data without applying the scale factors BZERO and BSCALE.
/HEADER_ONLY: set to read the header only.
/DATA_ONLY: set to read the data only. If set, if any scale factors are present (BSCALE or BZERO), they will not be applied.
/NO_PDU: Set to not add the primary data unit header keywords to the output header.
/NO_ABORT: Set to return to calling program instead of a RETALL when an I/O error is encountered. If set, the routine will return with ierr=-1 and a message in the keyword MESSAGE. If not set, FITS_READ will print the message and issue a RETALL
MESSAGE = value: Output error message
EXTEN_NO - extension number to read. If not set, the next extension in the file is read. Set to 0 to read the primary data unit.
XTENSION - string name of the xtension to read
EXTNAME - string name of the extname to read
EXTVER - integer version number to read
EXTLEVEL - integer extension level to read
FIRST - set this keyword to only read a portion of the data. It gives the first word of the data to read
LAST - set this keyword to only read a portion of the data. It gives the last word number of the data to read
GROUP - group number to read for GCOUNT>1. (Default=0, the first group)
NaNvalue - On non-IEEE floating point machines, it gives the value to place into words with IEEE NaN.
ENUM - Output extension number that was read.

NOTES:

Determination of which extension to read.
case 1: EXTEN_NO specified. EXTEN_NO will give the number of the extension to read. The primary data unit is referred to as extension 0. If EXTEN_NO is specified, XTENSION, EXTNAME, EXTVER, and EXTLEVEL parameters are ignored.
case 2: if EXTEN_NO is not specified, the first extension with the specified XTENSION, EXTNAME, EXTVER, and EXTLEVEL will be read. If any of the 4 parameters are not specified, they will not be used in the search. Setting EXTLEVEL=0, EXTVER=0, EXTNAME="", or XTENSION="" is the same as not supplying them.

case 3: if none of the keyword parameters, EXTEN_NO, XTENSION, EXTNAME, EXTVER, or EXTLEVEL are supplied. FITS_READ will read the next extension in the file. If the primary data unit (PDU), extension 0, is null, the first call to FITS_READ will read the first extension of the file.

The only way to read a null PDU is to use EXTEN_NO = 0.

If FIRST and LAST are specified, the data are returned without applying any scale factors (BSCALE and BZERO) and the data is returned in a 1-D vector. This will allow you to read any portion of a multiple dimension data set. Once returned, the IDL function REFORM can be used to place the correct dimensions on the data.

IMPLICIT IMAGES: FITS_READ will construct an implicit image for cases where NAXIS=0 and the NPIX1, NPIX2, and PIXVALUE keywords are present. The output image will be:

```
image = replicate(PIXVALUE,NPIX1,NPIX2)
```

EXAMPLES:

Read the primary data unit of a FITS file, if it is null read the first extension:

```
FITS_READ, 'myfile.fits', data, header
```

Read the first two extensions of a FITS file and the extension with EXTNAME = 'FLUX' and EXTVER = 4

```
FITS_OPEN, 'myfile.fits', fcb
FITS_READ, fcb,data1, header2, exten_no = 1
FITS_READ, fcb,data1, header2, exten_no = 2
FITS_READ, fcb,data3, header3, extname='flux', extver=4
FITS_CLOSE, fcb
```

Read the sixth image in a data cube for the fourth extension.

```
FITS_OPEN, 'myfile.fits', fcb
image_number = 6
ns = fcb.axis(0,4)
nl = fcb.axis(1,4)
i1 = (ns*nl)*(image_number-1)
i2 = i1 + ns*nl-1
FITS_READ,fcb,image,header,first=i1,last=i2
image = reform(image,ns,nl,/overwrite)
FITS_CLOSE
```

Appendix E

FITS_WRITE

Writes a fits primary data unit or extension.

CALLING SEQUENCE:

FITS_WRITE, filename_or_fcb, data, [header_in]

INPUTS:

FILENAME_OR_FCB: name of the output data file or the FITS control block returned by FITS_OPEN (called with the /WRITE or /APPEND) parameters.

OPTIONAL INPUTS:

DATA: data array to write. If not supplied or set to a scalar, a null image is written.

HEADER_IN: FITS header keyword. If not supplied, a minimal basic header will be created. Required FITS keywords, SIMPLE, BITPIX, XTENSION, NAXIS, ... are added by FITS_WRITE and do not need to be supplied with the header. If supplied, their values will be updated as necessary to reflect DATA.

KEYWORD PARAMETERS:

XTENSION: type of extension to write (Default="IMAGE"). If not supplied, it will be taken from HEADER_IN. If not in either place, the default is "IMAGE". This parameter is ignored when writing the primary data unit.

EXTNAME: EXTNAME for the extension. If not supplied, it will be taken from HEADER_IN. If not supplied and not in HEADER_IN, no EXTNAME will be written into the output extension.

EXTVER: EXTVER for the extension. If not supplied, it will be taken from HEADER_IN. If not supplied and not in HEADER_IN, no EXTVER will be written into the output extension.

EXTLEVEL: EXTLEVEL for the extension. If not supplied, it will be taken from HEADER_IN. If not supplied and not in HEADER_IN, no EXTLEVEL will be written into the output extension.

NaNvalue: data value in DATA to be replaced with IEEE NaN in the output file.

/NO_ABORT: Set to return to calling program instead of a RETALL when an I/O error is encountered. If set, the routine will return with !err=-1 and a message in the keyword MESSAGE. If not set, FITS_READ will print the message and issue a RETALL

MESSAGE: value of the error message for use with /NO_ABORT

HEADER: actual output header written to the FITS file.

`/NO_DATA`: Set if you only want `FITS_WRITE` to write a header. The header supplied will be written without modification and the user is expected to write the data using `WRITEU` to unit `FCB.UNIT`. When `FITS_WRITE` is called with `/NO_DATA`, the user is responsible for the validity of the header, and must write the correct amount and format of the data. When `FITS_WRITE` is used in this fashion, it will pad the data from a previously written extension to 2880 blocks before writing the header.

NOTES:

If the first call to `FITS_WRITE` is an extension, `FITS_WRITE` will automatically write a null image as the primary data unit.

Keywords and history in the input header will be properly separated into the primary data unit and extension portions when constructing the output header (See `FITS_READ` for information on the internal Header format which separates the extension and PDU header portions).

EXAMPLES:

Write an IDL variable to a FITS file with the minimal required header.

```
FITS_WRITE,'newfile.fits',ARRAY
```

Write the same array as an image extension, with a null Primary data unit.

```
FITS_WRITE,'newfile.fits',ARRAY,xtension='IMAGE'
```

Write 4 image extensions to the same file.

```
FITS_OPEN,'newfile.fits',fcb  
FITS_WRITE,fcb,data1,extname='FLUX',extver=1  
FITS_WRITE,fcb,err1,extname='ERR',extver=1  
FITS_WRITE,fcb,data2,extname='FLUX',extver=2  
FITS_WRITE,fcb,err2,extname='ERR',extver=2  
FITS_CLOSE,FCB
```

Appendix F FITS_CLOSE

Closes a FITS data file

CALLING SEQUENCE:

FITS_CLOSE, fcb

INPUTS:

FCB: fits control block returned by FITS_OPEN.

KEYWORD PARAMETERS:

/NO_ABORT: Set to return to calling program instead of a RETALL when an I/O error is encountered. If set, the routine will return with !err=-1 and a message in the keyword MESSAGE.

If not set, FITS_CLOSE will print the message and issue a RETALL
MESSAGE = value: Output error message

EXAMPLE:

Open a FITS file, read some data, and close it with FITS_CLOSE
FITS_OPEN, 'infile', fcb
FITS_READ, fcb, data
FITS_READ, fcb, moredata
FITS_CLOSE, fcb