URL: http://cxc.harvard.edu/ciao3.4/readarf.html
Last modified: December 2006

*AHELP for CIAO 3.4*

# readarf

Context: varmm



## Synopsis

S−Lang function to read an Ancillary Response File (ARF)

## Syntax

```
Struct_Type readarf( filename )

Error Return Value: NULL

Argument:
filename is a String_Type variable
```

## Description

The readarf() function provides a high−level interface to reading in an Ancilary Response File (ARF) from a FITS binary table that follows the relevant OGIP standard. It can be called either directly or indirectly when using the readfile() function. The ahelp page for readfile describes the features of this routine that are common to all the "read" functions provided by the Varmm module. This page describes those features that are unique to the readarf() command.

The filename argument should be a string that contains the name of the file to be read in. Although it can include Data−Model filters it is best not to use them when reading in an ARF, since any filter may well remove needed information from the file. As an example of how the function is used:

```
  chips> arf = readarf( "src.arf" )
```

### What does the function return?

The function returns a structure whose fields contain the data read in from the file. If an error occurred − such as the file not being found, or it does not contain an ARF − then NULL is returned instead. The returned structure follows the format of the other "read" functions: metadata − i.e. information about the file − is stored in fields beginning with an underscore character followed by fields containing the image data and coordinate−transformation information. The initial fields are discussed in "ahelp readfile"; here we concentrate on those fields specific to ARFs.

**Fields specific to ARFs:**

| Field name: | Description: |
|---|---|
| _ncols | Gives the number of columns read from the ARF. |
| _nrows | Gives the number of rows read from the ARF. |
| ENERG_LO | The minimum energy of each bin. |
| ENERG_HI | The maximum energy of each bin. |
| BIN_LO | The minimum wavelength of each bin (for grating ARFs). |
| BIN_HI | The maximum wavelength of each bin (for grating ARFs). |
| SPECRESP | The effective area of the bin. |

For example,

```
chips> arf = readarf("source.arf")
chips> print( arf )
_filename        =  source.arf
_path            =  /data/analysis/
_filter          =  NULL
_filetype        =  9
_header          =  String_Type[180]
_ncols           =  3
_nrows           =  1090
ENERG_LO         =  Float_Type[1090]
ENERG_HI         =  Float_Type[1090]
SPECRESP         =  Float_Type[1090]
```

# Example 1

## Reading in an ARF using readarf()

```
chips> arf = readarf("source.arf")
chips> print( arf )
_filename        =  source.arf
_path            =  /data/analysis/
_filter          =  NULL
_filetype        =  9
_header          =  String_Type[180]
_ncols           =  3
_nrows           =  1090
ENERG_LO         =  Float_Type[1090]
ENERG_HI         =  Float_Type[1090]
SPECRESP         =  Float_Type[1090]
chips> x = 0.5 * (arf.ENERG_LO + arf.ENERG_HI)
chips> curve( x, arf.SPECRESP )
0
```

Here we read in the ARF and then plot it, using the mid−point of each bin.

# Example 2

## Reading in an ARF using readfile()

Since readfile() calls readarf() when given an ARF, the results of the following are the same as in the previous example.

```
chips> arf = readfile("source.arf")
chips> x = 0.5 * (arf.ENERG_LO + arf.ENERG_HI)
chips> curve( x, arf.SPECRESP )
0
```

# Example 3

## Reading in a grating ARF using readarf()

```
chips> garf = readarf("1801_1_LEG_garf.fits")
chips> print( garf )
_filename        =  1801_1_LEG_garf.fits
_path            =  /data/analysis/
_filter          =  NULL
_filetype        =  9
_header          =  String_Type[217]
_ncols           =  5
_nrows           =  16384
ENERG_LO         =  Float_Type[16384]
ENERG_HI         =  Float_Type[16384]
BIN_LO           =  Float_Type[16384]
BIN_HI           =  Float_Type[16384]
SPECRESP         =  Float_Type[16384]
```

## CHANGES IN CIAO 3.1

## Speed enhancements

The time taken to read in a tables which does not contain any array columns (i.e. more than one element per row of the column) has been reduced.

## Reading a file in a directory containing the string '::'

The routines no longer crash when reading a file within a directory whose name contains the string "::".

## Enhanced documentation

There are now separate ahelp files for the individual functions, rather than having all the information placed into this document.

## CHANGES IN CIAO 3.0.2

## Stack Underflow errors

It is now possible to use readfile() − or any of the other read functions described here − in an if statement. Prior to CIAO 3.0.2 you could not write something like

```
if ( NULL == readfile("evt2.fits") ) error("Failed to read file.");
```

since it would result in a "Stack Underflow" error message. This means that many routines that use readfile() − such as Sherpa's load_dataset() and related functions − can also now be used in an if statement such as:

```
if ( 1 != load_image(imgname) )
  verror( "Unable to load %s as an image.", imgname );
```

## CHANGES IN CIAO 3.0

### New field "_filetype"

A new field called "_filetype" has been added to the data structure which describes the type of the file read in. The contents of the field are described in the "Format of data structure" section in "ahelp readfile".

# Bugs

See the bugs page for the Varmm library on the CIAO website for an up−to−date listing of known bugs.

# See Also

*modules*
varmm
*varmm*
fits_bitpix, readascii, readbintab, readfile, readimage, readpha, readrdb, readrmf, writeascii, writefits

---



URL: http://cxc.harvard.edu/ciao3.4/readarf.html
Last modified: December 2006