FIRAS Explanatory Supplement Appendix J

Listings for FIRAS File Reading Programs

The reader should note that the complete IDL package of COBE analysis software, "UIDL", is available through the COBE home page

http://www.gsfc.nasa.gov/astro/cobe/cobe_home.html

The IDL library contains the complete set of COBE Guest Investigator analysis tools. A FORTRAN library is also available which contains only data I/O and coordinate conversion routines including those listed below.

This appendix contains informational headers for UIDL routines which are designed to read FIRAS FITS files. The reader is cautioned that these are top-level, NOT stand-alone, routines which must be used as part of the overall UIDL package.

DATAIN	(IDL)	Reads pixelized data
FIRASMOD	(IDL)	Reads other data

This appendix also contains listings for the following FIRAS FITS file reading programs:

FIRAS_READ	(FORTRAN)	Reads pixelized data

as well as listings for the following FIRAS VAX binary file reading programs:

READ_FSS.FOR	(FORTRAN)	Reads VAX native format files
READ_FSS.C	(C)	Reads VAX native format files

All programs listed above, with the exception of READ_FSS.C, are available at the addresses given, but again note that the IDL programs are part of the UIDL package and cannot stand alone.

```
;+NAME/ONE LINE DESCRIPTION OF ROUTINE:
;    DATAIN is a routine which reads in FITS and other format files in UIDL
;
;DESCRIPTION:
;    Datain will read FITS, IDL Save Sets, and output FITS or IDL Save Sets.
;    It should be noted that some of the parameters in the list below are
;    instrument specific. This procedure is meant to be used for fields
;    within a FITS file which have been pixelized.
;
;WARNINGS:
;   1) If the input file is an IDL Save Set then the output will be sent
;    directly to an UIMAGE object. If the user wants to get the save
;    set directly to UIDL command line simply use the RESTORE command.
;   2) This routine is to be used for data fields which have been
;      pixelized. Some checking is done for FIRAS PDS's file types
;      to see if none of the fields are pixelized. If this is true
;      then the routine FIRASMOD is called to read in the data.
```

```
;
;     IF YOU TRY AND READ IN A NON-PIXELIZED FIELD FOR A FILE THAT
;     CONTAINS SOME PIXELIZED FIELD THIS ROUTINE WILL BLOW UP DURING
;     THE PIXELIZATION.
;
;CALLING SEQUENCE: datain, dsname, [,format=format] [,intype=intype]
;                  [,face=face] [,dsfield=dsfield] [,subscr=subscr]
;               [,data=data] [,weights=weights] [,frequency=frequency]
;               [,instrume=instrume] [,badpixval=badpixval] [,res=res]
;               [,units=units] [,wunits=wunits] [,rms=rms] [,coord=coord]
;               [,nu_zero=nu_zero] [,delta_nu=delta_nu] [,num_freq=num_freq]
;               [,/dbsig] [,/dbqual] [,/simple]
;
;ARGUMENTS (I = input, O = output, [] = optional):
;    param              I/O     type           description
;    dsname              I      string         full name of file for input
;    format=format       I      string          FITS or CISS (default to FITS)
;    intype=intype       I      string          DIRBE, FIRAS, DMR, or XAB, not
;                                                needed for most functions
;    face=face           I      integer         face # 0-5, not needed for full
;                                                skymap
;    dsfield=dsfield     I      string          field to read in (if FITS BE)
;    subscr=subscr       I      string          subscripts to read in if FITS BE
;                                                format '4:7' (note: fields start
;                                                at zero (1).
;    dbsig               I      integer         return sigma frm photqual(dirbe)
;    dbqual              I      integer         return flag frm photqual(dirbe)
;    simple              I      integer         set flag if read in simple FITS
;                                                non-skymap, data
;    data=data           O      float           array for data to be returned to
;    weights=weights     O      float           array for weights
;    frequency=frequency O      float           array for frequency info
;    instrume=instrume   O      string          the instrument the data is for
;    badpixval=badpixval O      float           "no data" indicator
;    res=res             O      integer         SKYMAP resolution
;    units=units         O      string          units for data values
;    wunits=wunits       O      string          units for weights
;    rms=rms             O      float           DMR specific
;    coord=coord         O      string          DMR specific
;    nu_zero=nu_zero     O      float           FIRAS specific
;    delta_nu=delta_nu   O      float           FIRAS specific
;    num_freq=num_freq   O      float           FIRAS specific
;
;    DMR specific keywords
;    ---------------------
;    rms, coord
;
;    FIRAS specific keywords
;    -----------------------
;    delta_nu, nu_zero, num_freq
;
;    DMR and FIRAS specific keywords
;    -------------------------------
;    weights
```

```
;    wunits
;
;    DIRBE and FIRAS specific keywords
;    ---------------------------------
;    frequency, dbsig
;    subscr, dbqual
;
;EXAMPLE:
;        datain, 'cgis_fits:dmr_skymap_31a.fits',dsfield='SIGNAL',data=data,$
;                instrume=instrume
;
;        datain, 'cgis_fits:dmr_skymap_53a.fits',dsfield='SIGNAL+WEIGHTS',$
;                data=data,instrume=instrume
;
;        datain, 'cgis_fits:fip_sky_lhs.fits',dsfield='SIGNAL',data=data,$
;                frequency=frequency
;
;        datain, 'cgis_fits:fip_sky_lhs.fits',dsfield='SIGNAL',subscr='5:40',$
;                data=data,frequency=frequency
;
;        datain, '$CGIS_FITS/dirbe_galactic_plane_maps.fits',data=sigma,$
;                dsfield='photqual',face=0,/dbsig
;#
;COMMON BLOCKS: None
;
;PROCEDURE (AND OTHER PROGRAMMING NOTES):
;
;PERTINENT ALGORITHMS, LIBRARY CALLS, ETC.:
;
;MODIFICATION HISTORY
; SPR 11347 19-Aug-1993   Dalroy Ward,J. Newmark  initial routine
; SPR 11375 13-Oct-1993   Modify for decomposed PhotQual (DIRBE). J. Newmark
; SPR 11759 17-May-1994   Ingest FIRAS PDS's. J. Newmark
; SPR 11905 07-Sep-1994   Ingest DIRBE PDS's. J. Newmark
; SPR 12140 16-Mar-1995   Ingest FIRAS line maps. J. Newmark
;
;.TITLE
;Routine DATAIN
;-

;=============================================================================
;+NAME/ONE-LINE DESCRIPTION:
;     FIRASMOD: used to read in the FIRAS Calibration Model IPs,
;               FIRAS Line Profile PDSs, and other FIRAS PDS.
;
;DESCRIPTION: UIDL routine which reads in fields of the FIRAS IP/PDS
;           files. This is mainly used for non-skymap (i.e. not
;           pixilzed) fields.
;
;CALLING SEQUENCE:
;       firasmod, dsname, dsfield, data=data, units=units, message=message
;
;       dsname: the name of the data file to be read
;       dsfield: the name of the field to be read in (TTYPE field in
```

```
;               the fits extension header.  Pass this parameter in as
;               a question mark ('?') in order to get a list of
;               fieldnames.
;       data    array for the data that was read in to be returned to
;       units   var to hold the units of the returned data
;       message  Y (or not set) for messages to be printed as operations
;               are performed, N to be silent
;
;#
; SUBROUTINES CALLED: most of the FXB routines for handling the FITS
;                 I/O.
;
; COMMON BLOCKS:  None
;
; LIBRARY CALLS:  None
;
; WARNINGS:
;
; PROGRAMMING NOTES:
;
; MODIFICATION HISTORY: D. Ward   GSC   June 1993  Orginal program
; SPR 11759 17-May-1994   Ingest FIRAS PDS's. J. Newmark
;
;.TITLE
;Routine FIRASMOD
;-
```

==========

The FORTRAN program FIRAS_READ uses FITSIO calls to read in the pixel number, spectrum, and spectrum sigma fields. The fields are stored in pixel list order, i.e. not as a rasterized sky map, so additional calls to routines that convert quadcube pixel numbers into raster or sky coordinates would be necessary to display the data as an image.

```
      program firas_read

c     ------------------------------------------------------------
c      A simple program to read the FITS binary table data from the
c      FIRAS pixelized Project Data Set or Initial Product.  This
c      program uses the FITSIO package developed by HEASARC at Nasa GSFC.
c      This program will read in the PIXEL, REAL_SPE and SIGMAS fields.
c     ------------------------------------------------------------

      implicit        none

      integer maxdim
      parameter (maxdim = 20)

      character*30      errtxt,extnam
      character*30      ttype(maxdim), tform(maxdim), tunit(maxdim)
      character*80      filename
      character*80      comment
```

```
      character*12      c_nu_zero, c_delta_nu, c_num_freq, c_res

      logical   simple, extend, anyflg

      integer   iunit, status, bitpix, naxis, naxes(maxdim)
      integer   pcount, gcount
      integer   group, nelem, nrows
      integer   i, j
      integer   tfield, rwstat, bksize, vardat
      integer   colnum, frow, felem
      integer   hdutyp, inull
      integer   num_freq, res
c
c If reading the IP's the dimensions should be changed from 6144
c to 2000 (or 1347 for real_spe to signal) and 180 to 141.
c
      integer   pixel(6144)
      real     realspe(180,6144), sigmas(180,6144), enull
      real     nu_zero, delta_nu

      status = 0
      iunit = 15

c     ------------------------------------------------------------
c      open the existing   FITS file with readonly access
c     ------------------------------------------------------------
      rwstat = 0
      print *, 'Enter file name '
      read(*,2000) filename
2000     format(a)
c       filename = 'adbdisk:[ip_fits]fip_sky_lhs.fits'
      call ftopen(iunit, filename, rwstat, bksize, status)
      if (status .ne. 0) goto 1000

c     ------------------------------------------------------------
c      read in the required primary array keywords
c     ------------------------------------------------------------
      call ftghpr(iunit, maxdim, simple, bitpix, naxis, naxes,
     %     pcount,gcount, extend, status)
      if (status .ne. 0) goto 1000

c     ------------------------------------------------------------
c      Get in the COBE specific field, RES which defines the resolution
c      of the data.  This is used to calculate the pixel numbers which
c      correspond to a specific face of the skycube.
c     ------------------------------------------------------------
      call ftgkey(iunit, 'PIXRESOL', c_res, comment, status)

c     ------------------------------------------------------------
c      Read in the FIRAS specific keywords which define the starting
c      frequency, the delta for the freq array and the number of
c      frequencies that are in the data array.  Note that these
c      keywords are in the PRIMARY header for the FITS file.
c     ------------------------------------------------------------
```

```
      call ftgkey(iunit, 'NU_ZERO',  c_nu_zero, comment, status)
      call ftgkey(iunit, 'DELTA_NU', c_delta_nu, comment, status)
      call ftgkey(iunit, 'NUM_FREQ', c_num_freq, comment, status)
      if (status .ne. 0) goto 1000

c     ------------------------------------------------------------
c      convert everything to numeric from character
c     ------------------------------------------------------------
      read(c_res, '(i1)') res
      read(c_nu_zero(1:7),'(f7.0)') nu_zero
      read(c_delta_nu(1:6),'(f6.0)') delta_nu
      read(c_num_freq(1:3),'(i3)')   num_freq

      print*, 'This file is at resolution: ',res

      print*, 'nu_zero: ', nu_zero
      print*, 'delta_nu:', delta_nu
      print*, 'num_freq:', num_freq

c     ------------------------------------------------------------
c      now move onto the binary table extension
c     ------------------------------------------------------------
      call ftmahd(iunit, 2, hdutyp, status)
      if (status .ne. 0) goto 1000

c     ------------------------------------------------------------
c      get the binary table parameters
c     ------------------------------------------------------------
      call ftghbn(iunit, maxdim, nrows, tfield, ttype, tform,
     %                   tunit,extnam, vardat, status)
      if (status .ne. 0) goto 1000

      print *, 'nrows:  ',nrows

c     ------------------------------------------------------------
c      test that this is a FIRAS dataset, by looking at a few field names
c     ------------------------------------------------------------
      if (ttype(1) .ne. 'PIXEL' .or.
     %            ttype(4) .ne. 'REAL_SPE' .or.
     %            ttype(6) .ne. 'SIGMAS') then
            print *, 'This does''t seem to be the FIRAS PDS/IP file'
            print *, 'ttype(1) = ', ttype(1)
            print *, 'ttype(4) = ', ttype(4)
            print *, 'ttype(6) = ', ttype(6)
            goto 1000
      endif

c     ------------------------------------------------------------
c      read in the pixel, signal and serror fields
c     ------------------------------------------------------------
      write(6,*) 'reading in the data now'
      do 100 frow = 1, nrows
         felem = 1
         nelem = 1
```

```
          inull = 0

          colnum = 1
          call ftgcvj(iunit, colnum, frow, felem, nelem, enull,
     %                        pixel(frow), anyflg, status)
          if (status .ne. 0) write(6,*) '1frow:',frow,' status:',
     %                        status
          if (status .ne. 0) goto 1000

c        ---------------------------------------------------------------
c         note that for the signal field and the serror field we take
c         the number of elements to read in from the num_freq field
c         which defines the number of elements which actually have
c         data in them, rather than from the tform field which defines
c         the data type and the full width of the field
c        ---------------------------------------------------------------
          colnum = 4
          nelem = num_freq
          call ftgcve(iunit, colnum, frow, felem, nelem, enull,
     %                        realspe(1,frow), anyflg, status)
          if (status .ne. 0) write(6,*) '2frow:',frow,' status:',
     %          status
          if (status .ne. 0) goto 1000

          colnum = 6
          nelem = num_freq
          call ftgcve(iunit, colnum, frow, felem, nelem, enull,
     %                        sigmas(1, frow), anyflg, status)
          if (status .ne. 0) write(6,*) '3frow:',frow,' status:',
     %          status
          if (status .ne. 0) goto 1000
  100    continue
        write(6,*) 'done reading in the data'

c       ---------------------------------------------------------------
c        now close the table and quit
c       ---------------------------------------------------------------
        call ftclos(iunit, status)

   1000   if (status .le. 0) then
          print *, '*** Program completed successfully ***'
        else
c         ---------------------------------------------------------------
c          get the error description
c         ---------------------------------------------------------------
          call ftgerr(status, errtxt)
          print *, '*** Error, program did not run successfully ***'
          print *, 'status =',status,': ',errtxt
        endif

        end
```

==========

The VAX FORTRAN and C programs READ_FSS use the Record Definition Language file FSS_SSSKY.RDL to read FIRAS sky short sciene index records (see Section 12.4.1) in VAX binary format.

Note that the MAP-ENDMAP and UNION-ENDUNION constructs found in many of the RDL files may not be explicitly supported by some compilers. IDL in particular does not. (A MAP in RDL is simply a collection of variables contiguous in memory but not explicitly made into a substructure. UNION is a means of Equivalencing two or more MAPs.) If two or more of the aliases in a UNION are needed in user software, this can be handled by multiple structures, one containing each of the needed MAP collections in place of the entire UNION in the RDL.

Here are some VAX FORTRAN and VAX C equivalents of RDL declarations.
Notes:
1) C compilers on other machines may define the sizes of some data types differently.
2) While in VAX FORTRAN arrays, the FIRST index is least significant in memory location, while in C, the LAST index is least significant.
3) In C, a char array of n characters is terminated with an ASCII null (\0) and therefore declared as char X[n+1], but in these structures, only n bytes are allotted, so the declaration is as shown.

RDL	VAX FORTRAN	VAX C
scalar/byte X	Byte x	char x
scalar/word X	Integer*2 x	int x
scalar/long X	Integer*4 x	long x
scalar/adt X	Integer*4 x(2)	long x[2]
scalar/float X	Real*4 x	float x
scalar/double X	Real*8 x	double x
scalar/complex X	Complex*8 x	--Not Available--
scalar/text/length=n X	Character*n x	char x[n] (--not char[n+1])
array/byte/dim=(m,n) X	Byte x(m,n)	char x[n][m]
array/word/dim=(m,n) X	Integer*2 x(m,n)	int x[n][m]
array/long/dim=(m,n) X	Integer*4 x(m,n)	long x[n][m]
array/adt/dim=n X	Integer*4 x(2,n)	long x[n][2]
array/float/dim=(m,n) X	Real*4 x(m,n)	float x[n][m]
array/double/dim=(m,n) X	Real*8 x(m,n)	double x[n][m]
array/complex/dim=(m,n) X	Complex*8 x(m,n)	--Not Available--

```
c       Listing of:   READ_FSS.FOR

C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C                          READ_FSS.FOR
C
C   Purpose:  Count the number of valid coadd groups in an FSS_SSSKY file.
C
C   Real Purpose:  Open and read data from a FIRAS native format file
C                (FSS_SSSKY) using VAX FORTRAN.  This program is meant to be
C                used as an example format for writing codes to open and read
C                other FIRAS native format files.  Other languages that have
```

```
C               record structures such as C and IDL can be used similarly.
C
C   Note:  For simplicity no subroutines or functions are used.
C          The record structure for the data is created manually using
C          the RDL (Record Definition Language) file for the data type.
C
C   Author:  Larry P. Rosen, Hughes STX, 4 October 1994.
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      IMPLICIT NONE

      STRUCTURE /FSS_STRUCT/
        INTEGER*4     TIME(2)              ! Raw science IFG GMT
        BYTE          CHANNEL_ID           ! Channel ID number
        INTEGER*4     PIXEL_NO             ! Skycube pixel number
        BYTE          MTM_SCAN_SPEED       ! MTM scan speed
        BYTE          MTM_SCAN_LENGTH      ! MTM scan length
        BYTE          SCI_MODE             ! Science mode
        BYTE          ADDS_PER_GROUP       ! Adds per group
        BYTE          TRANSITION_FLAG      ! Flag denoting group for coadd
        CHARACTER*1   PIXEL_DEFINITION     ! Type of pixelization
        INTEGER*2     SKYMAP_INDEX         ! Type of skymap index content
        BYTE          DATA_QUALITY         ! Summary flag for instruments
        BYTE          ATTITUDE_QUALITY     ! Summary flag for attitude
        REAL*4        XCAL_TEMP            ! External calibrator temperature
        REAL*4        SKYHORN_TEMP         ! Skyhorn temperature
        REAL*4        REFHORN_TEMP         ! Reference horn temperature
        REAL*4        ICAL_TEMP            ! Internal calibrator temperature
        REAL*4        DIHEDRAL_TEMP        ! Dihedral temperature
        REAL*4        BOLOMETER_TEMP       ! Detector temperature
        BYTE          BOL_CMD_BIAS         ! Commanded bias
        INTEGER*2     EXC_GALACTIC_LAT     ! Exclude data within +/-
        INTEGER*2     SUN_ANGLE            ! Angle of sun from FIRAS skyhorn
        INTEGER*2     MOON_ANGLE           ! Angle of moon from FIRAS skyhorn
        INTEGER*2     EARTH_LIMB           ! Angle of Earth limb from FIRAS
        INTEGER*2     GALACTIC_LATITUDE    ! Galactic Latitude
        BYTE          SPARES(6)            ! Spares to pad record size 64b
      ENDSTRUCTURE
      INTEGER*2       RECLEN /16/          ! Length of record in long words
      RECORD /FSS_STRUCT/ FSS_REC          ! One FSS_SSSKY record
      CHARACTER*32    INFILE               ! Name of file to read
      INTEGER*4       RSTAT                ! Fortran function status
      INTEGER*4       LUNIN /10/           ! Logical unit number for infile
      INTEGER*4       REC                  ! Record number
      INTEGER*4       NUM_COADD_GROUPS     ! Number of coadd groups of ifgs
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C  Begin

      INFILE = 'FSS_SSSKY_LH.ED_8932800_8934301;'

C  Open file for sequential access.
C  (You could use direct access for direct read of a particular record.)

      OPEN ( UNIT=LUNIN, ACCESS='SEQUENTIAL', FORM='UNFORMATTED',
     *       FILE=INFILE, IOSTAT=RSTAT, RECL=RECLEN, STATUS='OLD' )
```

```
      IF (RSTAT .NE. 0) THEN
        WRITE (6,*) 'Error opening input file: ', INFILE
        WRITE (6,*) 'Open status = ', RSTAT
      ELSE

C  Read all the records in sequential order.

        REC = 1
        NUM_COADD_GROUPS = 0
        DO WHILE (RSTAT .EQ. 0)
          READ (LUNIN, IOSTAT=RSTAT) FSS_REC
          IF (RSTAT .EQ. 0) THEN
            IF (FSS_REC.TRANSITION_FLAG .EQ. 2) THEN
              NUM_COADD_GROUPS = NUM_COADD_GROUPS + 1
            ENDIF
            REC = REC + 1
          ENDIF
        ENDDO
        IF (RSTAT .GT. 0) THEN
          WRITE (6,*) 'Error reading record ',REC,' Status= ',RSTAT
        ELSE
          WRITE (6,*) 'Number of coadd groups = ', NUM_COADD_GROUPS
        ENDIF

C  Close the input data file.

        CLOSE (LUNIN)
      ENDIF
      END

------------------------------------------------------------------------------

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C                           READ_FSS.C
C
C   Purpose:  Count the number of valid coadd groups in an FSS_SSSKY file.
C
C   Real Purpose:  Open and read data from a FIRAS native format file
C                (FSS_SSSKY) using C.  This program is meant to be used as
C                AN example format for writing codes to open and read other
C                FIRAS native format files.  Other languages that have record
C                structures such as VAX FORTRAN and IDL can be used
C                similarly.
C
C   Note:  For simplicity no subroutines or functions are used.
C          The record structure for the data is created manually using
C          the RDL (Record Definition Language) file for the data type.
C
C   Author:  Larry P. Rosen, Hughes STX, 5 October 1994.
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

  #include  <stdio.h>
```

```
int main (void)
{
/* Record structure for data */

struct fss_struct {
        long     time[2];             /* raw science ifg gmt */
        char     channel_id;          /* channel id number */
        long     pixel_no;            /* skycube pixel number */
        char     mtm_scan_speed;      /* mtm scan speed */
        char     mtm_scan_length;     /* mtm scan length */
        char     sci_mode;            /* science mode */
        char     adds_per_group;      /* adds per group */
        char     transition_flag;     /* flag denoting group for coadd */
        char     pixel_definition;    /* type of pixelization */
        int      skymap_index;        /* type of skymap index content */
        char     data_quality;        /* summary flag for instruments */
        char     attitude_quality;    /* summary flag for attitude */
        float    xcal_temp;           /* external calibrator temperature */
        float    skyhorn_temp;        /* skyhorn temperature */
        float    refhorn_temp;        /* reference horn temperature */
        float    ical_temp;           /* internal calibrator temperature */
        float    dihedral_temp;       /* dihedral temperature */
        float    bolometer_temp;      /* detector temperature */
        char     bol_cmd_bias;        /* commanded bias */
        int      exc_galactic_lat;    /* exclude data within +/-  */
        int      sun_angle;           /* angle of sun from firas skyhorn */
        int      moon_angle;          /* angle of moon from firas skyhorn */
        int      earth_limb;          /* angle of earth limb from firas */
        int      galactic_latitude;   /* Galactic Latitude */
        char     spares[6];           /* Spares to pad record size 64b */
   };

/* Local */

   char              infile[] = "FSS_SSSKY_LH.ED_8932800_8934301;";
   FILE              *datafile;
   struct fss_struct  *rec_ptr;          /* Pointer to an FSS_SSSKY record */
   long              rec;              /* Record number */
   long              num_coadd_groups;   /* Number of coadd groups of ifgs */
   int               rstat = 0;         /* Return status of function */
   int               rec_size = 64;     /* Size of record in bytes */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/*  Begin */

/*  Open file for reading binary data. */

   datafile = fopen (infile, "rb");
   if (datafile == NULL)
      {
      printf ("ERROR ==> Data file not opened!\n     %s\n", infile);
      return 0;
      }
   else
      {
```

```
      printf ("File opened.\nReading: %s\n", infile);

  /*  Read all the records in sequential order. */

      rec = 1;
      num_coadd_groups = 0;
      rec_ptr = malloc (rec_size);
      while (!feof (datafile) && !ferror (datafile))
        {
        fread (rec_ptr, rec_size, 1, datafile);
        if (!ferror (datafile))
          {
          if (rec_ptr->transition_flag == 2)
            num_coadd_groups = num_coadd_groups + 1;
          rec = rec + 1;
          }
        }
      if (ferror (datafile))
        printf ("Error reading record %d\n", rec);
      else
        printf ("Number of coadd groups = %d\n", num_coadd_groups);

  /*  close the input data file. */

      fclose (datafile);
      }
    return(1);
    }
```