University of Michigan
Space Physics Research Laboratory

TIDI Data IDL Reader	CAGE No.	0TK63
	Drawing No.	055-4231B
	Project	TIDI
	Contract No.	NASW-5-5049
	Page	1 of 10

REVISION RECORD

Rev	Description	Date	Author
B	Added section on installation issues	27 Jan 2003	mlc
A	Added example code	8 Jan 2002	mlc
	Initial Release	18 Dec 2002	mlc

Contents

1. References

1) Gell, D. A., "4191D-LOS VC File Format.doc", TIDI File 055-4191, 3 Nov 2002.

2) Gell, D. A., "3532I-Profile File Format.doc", TIDI File 055-3532, 9 Mar 2001.

3) Gell, D. A., "3933D-Vector File Format.doc", TIDI File 055-3933, 1 Aug 2001.

4) Gell, D. A., "4237-find_tidi_data.doc", TIDI File 055-4237, 29 Jan 2003.

2. Introduction

The purpose of this document is to describe the IDL code that reads the TIDI NetCDF data files (i.e. LOS, RAW–LOS, PRF, and VEC files).

3. Opening TIDI Data Files (open_tidi_file)

The routine `open_tidi_file` opens a TIDI NetCDF data file. It is a helper routine to simplify finding data files easier. You can also open the file using the IDL NetCDF library:

```
IDL> ncid = ncdf_open( file, /nowrite )
```

where `file` is the entire path and filename of the TIDI NetCDF file.

3.1 open_tidi_file Documentation

```
IDL> doc_library, 'open_tidi_file'

-------------------------------------------------------------------
----- Documentation for ./open_tidi_file.pro -----

pro open_tidi_file, ncid, file = file, date = date, type = type, $
                    version = version, revision = revision, $
                    source = source, filter = filter, debug = debug

  Purpose: Open a TIDI NetCDF file.

  Output:
    ncid   NetCDF file unit id number - returned for later use with:
           read_tidi_data, get_vars_info, get_tidi_atts, get_bin_info
           and close_tidi_file.

  Options:
    date = YYYYDOY
           For use with findTidiData; only finds data in production dirs.

    type = 'datafiletype' [i.e. 'LOS', 'RAW-LOS', 'PRF' or 'VEC']
           For use with findTidiData; only finds data in production dirs.
           Must also use date option!

    version = 'data_version_letter' [i.e. 'B1', 'C', etc.]
           For use with findTidiData; only finds data in production dirs.
           Must also use date option!

    source = 'data_source' [i.e. 'PB', 'RT', etc.]
           For use with findTidiData; only finds data in production dirs.
           Must also use date option!

    revision = 'revision_letter' [i.e. 'A', 'B', 'C', etc.]
           For use with findTidiData; only finds data in production dirs.
           Must also use date option!

    file = 'filename'
           Filename and path of the TIDI NetCDF file to read/plot
           (If not given, then a pop-up window will be used to choose.)
```

```
    filter = '*LOS*' <- default value
           If no filename is given then specify a wildcard to
           search for an LOS file in the variable: dirpath, given below.
           (use '/debug' option for choosing debug directory)
           Note: filter can also be used to pick a different directory
           (ex: filter='/homes/mlcooper/code/retrieve/debug/2002/*LOS')

    /debug   choose the debug directory instead of the production directory
    if ( KEYWORD_SET( debug ) ) then $
       dirpath = '/tidi/tidi_software/retrieve/debug/2002/' $
    else dirpath = '/data/tidi/'

------------------------------------------------------------------
```

3.2 open_tidi_file Examples

To open a TIDI NetCDF data file you can specify a file directly with the following:

```
IDL> open_tidi_file, ncid, file='/data/tidi/los/2002/TIDI_PB_2002100000100_2002100004414_C_A.LOS'
```

If you do not specify a file name, a dialog box will open allowing you to search directories. If you use the option 'filter', you can narrow down the search.

```
IDL> open_tidi_file, ncid, filter='/data/tidi/los/2002/TIDI_PB_2002*_C_A.LOS'
```

The previous command will give you a dialog box with all the files that match the wildcard.

You can also give a 'date' in `YYYYDOY` format and a '`type`' and the `find_tidi_data` IDL code (Ref 4) will be invoked. The entry for the option '`type`' is case insensitive.

```
IDL> open_tidi_file, ncid, date=2002100, type='RAW-LOS'
```

Note: in order to use the '`date`' option, the `find_tidi_data` routine (Ref 4) must be installed. See Section 11 for `find_tidi_data` installation issues (Section 11).

For now, only one TIDI NetCDF file can be opened at a time, if you are going to use the `read_tidi_data` subroutine .

4. Reading TIDI NetCDF data (read_tidi_data)

The routine `read_tidi_data` reads in any TIDI NetCDF data. Give the routine a time (in mission time format, DOY time format, i.e. `YYYYDOYHHMMSS`, or a two-dimensional array containing the YYYYDOY and seconds since 00:00 UT) and it will return an array of structures containing the data that is less than or equal to the requested time.

Although you are able to use either mission time, DOY time, or a two-dimensional time array to access data, once you select a format you may not change it. You can't switch to accessing the data with a DOY time format after getting the first data records using mission time and vice versa.

For now, only one TIDI NetCDF file can be opened at a time, if you are using the `read_tidi_data` subroutine .

4.1 read_tidi_data Documentation

```
IDL> doc_library, 'read_tidi_data'

----- Documentation for ./read_tidi_data.pro -----
------------------------------------------------------------------

pro read_tidi_data, ncid, time, tidi_rec, next_time, this_time, $
                    sec = sec, doy = doy, mtime = mtime, $
                    end_of_file = end_of_file, err = err, $
                    verbose = verbose, debug = debug
```

```
   Purpose: Reads TIDI NetCDF data from an already open file. Given a
            time, it returns the entire data record for the time that
            is less than or equal to the requested time.
            And also returns the time of the next record in the file.

   Inputs:
     ncid         NetCDF id for the open TIDI NetCDF file

     time         time of interest expressed in mission time format,
                  a DOY format string, or a 2-D array giving year and
                  day of year and seconds since midnight UT
                  i.e. time = 702432080 = missiontime (default)  OR
                       time = '2002100000107' = 'YYYYDOYHHMMSS'  OR
                       time(0) = 2002100 = YYYYDOY &
                       time(1) = 67 = seconds since midight UT

   Outputs:
     next_time    time of interest expressed in same format as time

     tidi_rec     structure of the TIDI data that's read in; will be
                  different for each type of TIDI data file

     this_time    time of that's found expressed in same format as time


   Options:
     /sec         use when time is expressed in YYYYDOY and seconds
                  since midnight UT format [where time is a 2-D array,
                  time(0) = YYYYDOY and time(1) = SECS]

     /doy         use when time is expressed in YYYYDOYHHMMSS format

     /mtime       use when time is expressed in mission time format,
                  seconds since Jan 6, 1980 [default format if none given]

     end_of_file  returns a 1, if the last set of records are read

     err          returns an errcode; 0=No error, 1=No spectra data,
                  2=Pre-version C, 3=No time format option given,
                  4=Format option does not match format of requested time

     verbose      turns on some informational messages for the user

     debug        turns on a whole bunch of messages of no interest to
                  the typical user

   Note: Use of this common block is for saving variables from call
   to call. The variables are: the mission time, UT date, UT time,
   the software name, variable names, TIDI data structure, and a flag
   for whether or not this is a new file being read for the first time.
    COMMON MISSIONTIME_BLOCK, missiontime, ut_date, ut_time, $
       sw_name, names, tididata, new_read_flag

-------------------------------------------------------------------
```

4.2 read_tidi_data Examples

The read_tidi_data routine is very versatile with the formats of requested time. As stated before there are three formats of time that one can utilize.

```
IDL> read_tidi_data, ncid, [2002100, 77], tidi_rec, next_time, this_time
```

The previous command will get you the TIDI data records that are less than or equal to 77 seconds after midnight UT on day 100 of year 2002 and return them in an array of structures stored in the variable `tidi_rec`. Note: the variables `this_time` and `next_time` will be given as two-dimensional arrays with YYYYDOY and seconds since midnight UT, since time was given in that format.

```
IDL> read_tidi_data, ncid, 0, tidi_rec, next_time, this_time
```

The previous command will get you the first data record in the TIDI file and return it in an array of structures stored in `tidi_rec`. Note: the variables `this_time` and `next_time` will be given in mission time format, if zero is used to find the first record.

```
IDL> read_tidi_data, ncid, '2002355000000', tidi_rec, next_time, this_time
```

The previous command will get you the first record that is less than or equal to the time 00:00:00 UT (on day 200355) in the currently open file. It will return the data in an array of structures stored in `tidi_rec`. Note: the variables `this_time` and `next_time` will be given in DOY time format (`YYYYDOYHHMMSS`), since time was given in that format.

5. Getting the Variable Information (get_var_info)

The `read_tidi_data` routine makes up the appropriate TIDI data structure "on the fly". The routine `get_var_info` helps with reading in all the variables in the TIDI NetCDF files and assigning a variable type to each one. This routine then stores the names, units, description, type, missing values and dimensions of each variable into a structure that's passed back to `read_tidi_data`. This routine is not normally invoked by the user.

```
----- Documentation for /tidi/tidi_idl/support/get_var_info.pro -----

pro get_var_info, ncid, varInfo, debug = debug
; get file attributes
theAttribs = ncdf_inquire( ncid )
;
; Define the structure of the variable information.
varInfo = replicate( {name:          ' ',   $
                      units:         ' ',   $
                      long_name:     ' ',   $
                      type:            0,   $
                      missing_value: {flt: 0.0, int2: 0, int4: 0L, str: ''}, $
                      ndims:           0,   $
                      dims: intarr(theAttribs.ndims)}, $
                     theAttribs.nvars)
-------------------------------------------------------------------
```

6. get_tidi_atts

The routine `get_tidi_atts` retrieves all of the global attributes for any of the TIDI NetCDF products (i.e. all LOS, RAW–LOS, PRF and VEC files).

6.1 get_tidi_atts Documentation

```
-------------------------------------------------------------------
----- Documentation for /tidi/tidi_idl/support/get_tidi_atts.pro -----

pro get_tidi_atts, atts, file = file, filter = filter, ncid = ncid, $
                   debug = debug, verbose = verbose

   Purpose: Reads a TIDI NetCDF file and stores the global
            attributes in the structure atts, i.e. atts.<attribute_name>

   NOTE: Attribute is filled with '--' if not defined for the file.

   atts      structure of global attributes, i.e. atts.<attribute_name>

   file      full filename of TIDI NetCDF file that's to be read in

   filter    wildcard for searching for the filename, if file not given

   ncid      if a NetCDF file is already open, you can just pass in
             the file unit number, ncid

   /debug    looks in debug directory first, if file not given

    if ( KEYWORD_SET( debug ) ) then $
       dirpath = '/tidi/tidi_software/retrieve/debug/2002/' $
    else dirpath = '/data/tidi/'

   /verbose  prints attributes and values to terminal

-------------------------------------------------------------------
```

6.2 get_tidi_atts Examples

The global attribute information can be retrieved two different ways. If a TIDI NetCDF file is not yet opened, you can give the entire path and filename and the file will be opened, the global attribute information will be read and then the file will be closed upon exiting the routine. The global attribute information will be returned in the structure `atts`.

```
IDL> get_tidi_atts, atts, file='/data/tidi/los/2002/TIDI_PB_2002363000101_2002363074553_C.RAW-LOS'
```

However, if a file is already open, then specifying a `ncid` (returned by `open_tidi_file`) instead of a filename is all that's required.

```
IDL> get_tidi_atts, atts, ncid = ncid
```

The routine will use the NetCDF file that's already open and will store the global attributes in the structure `atts`.

6.2.1 Description of the global attribute structure

```
IDL> help, atts, /struct
** Structure ATTS_STRUCT, 19 tags, length=152:
   TITLE           STRING    'Ancillary data and Line of Sight values'
   DATA_PRODUCT_TYPE
                   STRING    = 'ROUTINE, LEVEL1B'
   MISSION         STRING    'TIMED'
   SOURCE          STRING    'TIDI_POC'
   DATA_PRODUCT_VERSION
                   STRING    = '3.1'
   PRODUCT_FORMAT_VERSION
                   STRING    = '3.0'
   SOFTWARE_NAME   STRING    'RETRIEVE'
   SOFTWARE_VERSION
                   STRING    = '3.3'
   CALIBRATION_VERSION
                   STRING    = 'check cpf_filename'
   FILENAME        STRING    'TIDI_PB_2002363000101_2002363074553_C.RAW-LOS'
   INPUT_FILE      STRING    'TIDI_PB_2002363000101_2002363074553_O.TL0'
   CPF_FILENAME    STRING    'TIDI_2002044_2100366_002.CPF'
   PVAT_FILENAME   STRING    '/d4014/ftp/MDC/data_products/OrbitFiles/2002/'...
   DATE_CREATED    STRING    '2002365054029'
   MAGNETIC_LATITUDE_MODEL
                   STRING    = 'Altitude Adjusted Corrected Geomagnetic Coord'...
   SOLAR_BETA_ANGLE
                   STRING    = '41.1009'
   OBAND_RATIO_SOURCE
                   STRING    = '--'
   ATT_S_VAR       STRING    '-99.0000'
   ATT_H_VAR       STRING    '-99.0000'
```

7. Get Binning Table Information (get_bin_info)

The binning table information is contained in only the RETRIEVE products (i.e. LOS and RAW–LOS files)

7.1 get_bin_info Documentation

```
------------------------------------------------------------------------
----- Documentation for /tidi/tidi_idl/support/get_bin_info.pro -----

pro get_bin_info, ncid, bin_table_info

 Get binning table information from an already opened LOS or RAW-LOS file.

    bin_table_info = { bin_table_id:  lonarr(10),   $
                       initial_pixel: lonarr(5,75,10), $
                       final_pixel:   lonarr(5,75,10),     $
                       gain_values:   lonarr(5,75,10),     $
                       field_size:    lonarr(5,10) }
------------------------------------------------------------------------
```

7.2 get_bin_info Example

In order to get the binning table information (to make sense of the binning_id variable conatained in RETRIEVE products data structure), you would use the following:

```
IDL> get_bin_info, ncid, bin_table_info
```

8. Closing a TIDI NetCDF file (close_tidi_file)

This is the routine to close a TIDI NetCDF data file, if you've used read_tidi_data. There is an important flag to be set, if you'd like to close the current file and then open a new one.

```
----- Documentation for /tidi/tidi_idl/support/close_tidi_file.pro -----
pro close_tidi_file, ncid
;  author:   Marie L. Cooper
;  created:  12 Dec 2002
    COMMON MISSIONTIME_BLOCK
    ncdf_close, ncid
; Need to read in new values into the common block for the next TIDI file
    new_read_flag = 1
    return
END
------------------------------------------------------------------------
```

Note: If you haven't used read_tidi_data, you can close the TIDI NetCDF file by using the IDL NetCDF library command: `IDL> ncdf_close, ncid`.

9. Example Code (test_read_tidi_data.pro)

```
;+
pro test_read_tidi_data, verbose = verbose
;;
;;   Open a TIDI line-of-sight file. (RAW-LOS, LOS-FIT & LOS-BCK also work)
;;
open_tidi_file, ncid, date = 2002100, type = 'RAW-LOS', version = 'C'
;;
;;   Get the TIDI global attributes from the opened file.
;;
get_tidi_atts, atts, ncid = ncid
help, atts, /structure
;;
;;   Get the TIDI binning table information, if the file is a RETRIEVE product.
;;
if ( atts.software_name eq 'RETRIEVE' ) then begin
   get_bin_info, ncid, bin_table_info
   help, bin_table_info, /structure
   print, 'binning id: ', bin_table_info.bin_table_id
   print, 'field size: ', bin_table_info.field_size
endif
;;
```

```
;;   Setting the requested time variable to zero ensures that you get
;;   the first time of the file.
;;
requested_time = 0L
sc_lat = fltarr(1L)
sc_lon = fltarr(1L)
;;
;;   Read file until end of file is reached.
;;
no_more_data = 1
while ( no_more_data ) do begin
   read_tidi_data, ncid, requested_time, tidi_rec, $
      next_time, this_time, /mtime, end_of_file = end_of_file, err = err
;;
;;   Read errors will normally appear in the very first read
;;
   if ( err ne 0 ) then begin
      print, 'An error reading the data has occured.', err
      return
   endif
;;
;;   The variable 'tidi_rec' now contains the records with a mission
;;   time matching the first in the file. (In most LOS files, this means
;;   that tidi_rec is a five element array of structures.)
;;
   if ( requested_time eq 0L ) then begin
      print, 'Read data for time = ',strtrim(string(this_time),2)
      help, tidi_rec
      if (KEYWORD_SET(verbose)) then help,tidi_rec,/struct ;View data structure
   endif
   if ( KEYWORD_SET(verbose) or requested_time eq 0L ) then $
      print, 'time,date: ', tidi_rec[0].time, '   ',tidi_rec[0].ut_date, $
      '    req_time,next_time: ', requested_time, '   ',next_time
;;
;;   Store variable 'sc_lat' and 'sc_lon' for plotting. I'm only
;;   storing the value given for field of view (FOV) telescope 1 since
;;   the spacecraft latitude and longitude will not change based on FOV.
;;
   if ( requested_time eq 0L ) then begin
      sc_lat[0L] = tidi_rec[1].sc_lat
      sc_lon[0L] = tidi_rec[1].sc_lon
   endif else begin
      sc_lat = [ sc_lat, tidi_rec[1].sc_lat ]
      sc_lon = [ sc_lon, tidi_rec[1].sc_lon ]
   endelse
;;
;;   One way to end the read loop is with the option 'end_of_file'.
;;
   if ( end_of_file ) then begin
      print, 'The end_of_file flag is set <- Reached end of file.'
      no_more_data = 0
   endif
;;
;;   Another way to end the read loop is to check if 'this_time' is
;;   the same as 'next_time'.
;;
   if ( this_time eq next_time ) then begin
      print, 'This time equals next time  <- Reached end of file.'
      no_more_data = 0
   endif
;;
;;   Set the next time read to the 'next_time' given from read_tidi_data.
;;
   requested_time = next_time
endwhile
;;
;;   You must use close_tidi_data, if you'd like to open another TIDI
;;   file before exiting IDL. (Only one file open at a time for now.)
;;
close_tidi_file, ncid
;;
;;   Plot the stored values of spacecraft coordinates.
;;
plot, sc_lon, sc_lat, title = 'Spacecraft Latitude vs. Longitude', $
   xtitle = 'Longitude (degrees)', ytitle = 'Latitude (degrees)'
END
;-
```

10. Definition of the LOS structure returned from read_tidi_data

This section is only pertinent for those who will be accessing the LOS and RAW–LOS data (i.e. RETRIEVE products, also called Level 1 products).

The contents of the LOS and RAW–LOS data structure are described in TIDI document "4191B-LOS VC File Format.doc" (Ref 1). **NOTE!** There are two important changes to the format given in this document. The filter wheel 1 position and filter wheel 2 position are given as bytes (I1) in the document and they are defined as integers in the IDL structure. The other change is that the spectra and the variance of the spectra are not given as separate arrays indexed by `spec_index`, but instead they are given as float arrays of 75 values within the returned TIDI data structure.

11. find_tidi_data Installation Issues

11.1 Data Location

The `find_tidi_data` routine may be invoked when a specific data file is not supplied. This routine expects the LOS data files to be located in specific directories. The default directories may be overridden through the definition of a series of environment variables. The environment variables and the default values that they replace are listed in the following table:

Data Location Specification	
environment variable	default
`TIDI_ROOT`	`/data/tidi/`
`TIDI_LEVEL0_DATA`	`level0/`
`TIDI_LOS_DATA`	`los/`
`TIDI_PRF_DATA`	`profile/`
`TIDI_VEC_DATA`	`vector/`
`TIDI_ELO_DATA`	`event/`

Line of sight data is expected to be in the directory formed by concatenating the values of the `TIDI_ROOT` and `TIDI_LOS_DATA` environment variables and appending the year.

12. Code Distribution

The IDL code for these routines and those that they call is available on the TIDI data system in the directories beneath `/tidi/tidi_idl`. Compiled routines, in the form of an IDL save file are available for distribution to other sites on an as-is basis.