CORONAL DIAGNOSTIC SPECTROMETER

SOHO

CDS SOFTWARE NOTE No. 28

Version 2 29 December 1995

TAPE ARCHIVING PROCEDURES FOR THE SOHO/CDS

Donald G. Luttermoser
Applied Research Corporation
NASA/Goddard Space Flight Center
Code 682
Greenbelt, MD 20771

lutter@orpheus.nascom.nasa.gov

1 Introduction

This manual describes the technique for reading and writing archival tapes or CDROMS (note that CDROM input/output has not been tested yet) for the SOHO Coronal Diagnostic Spectrometer (CDS) data. In 1992, the SOHO Scientific Operations Working Group (SOWG) selected two standards for storing scientific data: the Standard Formatted Data Unit (SFDU) and the Flexible Image Transport System (FITS). FITS is a standard for formatting data into files in a computer–independent, self–describing fashion. Meanwhile, SFDU is a standard for documenting the contents and format of a data file, and for outlining the interdependencies of collections of files. A full description of the SFDU protocol can be found in the Consultative Committee for Space Data Systems' (CCSDS) Green Book (CCSDS 621.0-G-1), May 1992 entitled *Standard Formatted Data Units — A Tutorial*, and the Blue Book (CCSDS 620.0-B-2), May 1992 entitled *Standard Formatted Data Units — Structure and Construction Rules.*

SFDU works by associating a 20–byte label with each file which serves as a reference to documentation about the file format. This label can be prepended to the file, in which case it is called an *attached* SFDU label. Otherwise, it can be contained in a separate file on the archive medium, with a reference to the file it applies to, which is a *detached* label. In addition, each file can be associated with one or more additional files which describes its contents. Meanwhile, the FITS standard not only specifies how data is organized within a FITS file, but also how a FITS file is stored on disk or tape. FITS files, with their own *attached* labels, makes the attached SFDU label incompatible with the FITS format. As such, this project uses the **detached** SFDU header in a separate file, which will always be the first file on an archive tape or CDROM.

When writing data to a disk medium, there is generally a standard for how the files are organized and identified. For example, the ISO-9660 standard is the recommended way to write CDROMs, and these disks can then be easily read on just about all computer platforms. Performing a directory of the files on the disk, and extracting individual files are simple tasks. However, there is currently no comparable standard for writing tapes that provide the same facility of use. Although there is an ANSI standard for writing labeled tapes, the computer vendors such as Digital extend this standard for their own use, and Unix computers generally don't follow this standard at all, so that it is not a trivial matter to use a labeled tape written on one computer on another.

On the other hand, tapes written using the FITS specification can be read on any computer platform. This was, in fact, the primary goal of FITS. However, FITS tapes are fairly primitive, with each file separated from the next by only a filemark, and with no labels giving the names of the files. The contents of a FITS tape can only be determined by opening up each FITS file on the tape, and extracting its header. This is not only clumsy, but can take quite a bit of time to work through a modern tape cartridge with several gigabytes of storage.

There are then two goals for developing a tape format for SOHO data: First of all, the tape must be written to associate SFDU labels and descriptors with the data files, and at the same time allow the tape to be read in with standard FITS readers. Second, there needs to be a way of documenting the contents of the tape at the front of the tape, so that the specific data desired can be more rapidly retrieved. This manual describes a protocol that has been developed that combines these two goals into one, so that satisfying the first also satisfies the other.

This manual continues with §2 which gives a brief tutorial on how to run the tape archiving procedures. It essentially gives an easy recipe for running XWINTAPE and MAGTAPE. §3 summarizes the SFDU format and discusses how to revise the CDS/SFDU format with the CCSDS.

The FITS files are discusses in §4. Note that there are more detailed CDS Software Notes available (*e.g.*, #3, #4, and #11) concerning the FITS files. The main tape–driving procedure, MAGTAPE, is discussed in detail in §5, and likewise, the widget procedure, XWINTAPE, that drives MAGTAPE in an X–window environment, is presented in §6. §7 warns the operator of possible problems that may arise during a tape archiving run. Appendix A lists the IDL procedures used in during a tape archiving run. Finally note that *named* widget buttons have $\boxed{\text{frames}}$ around them in this manual and toggle–type buttons have a ◇ designation associated with the name.

2 Brief Tutorial

Before getting into the details of the archiving procedures of the SOHO/CDS telemetry, this section gives a brief tutorial for those anxious to use the software. All of the archiving software is written in IDL, and has been designed to operate in any operating system (*e.g.*, Unix, VMS, DOS, or MacOS — note that the DOS and MacOS sections of the code have not been fully implemented as of this writing). Also, the code has been designed to work under any version of IDL later than version 2.0. WARNING! This code has only been tested under IDL Version 3.2 and later, should any bugs be found for earlier IDL versions, please inform the CDS staff. The main driving procedure is called MAGTAPE. However, a widget program, called XWINTAPE, has been designed to drive MAGTAPE. The operation of XWINTAPE will be described in the next subsection and the operation of MAGTAPE from a dumb terminal is described in the subsequent subsection.

2.1 The Tape Drive Name

Before one enters the IDL session for tape archiving, one must inform IDL of the tape drive name. IDL has various procedures to access a tape drive, unfortunately these commands are only valid under the VMS operating system. To use these commands, IDL requires that the tape drive be assigned to a logical name MTn, where n is an integer between 0 and 9 (*i.e.*, MT0, MT1, ..., MT9). Tape drives connected to a machine running VMS have typical device names like "MUA0:" or "MKB500:". For your machine, enter SHOW DEV at the VMS prompt to get a list of all known devices connected to your machine. The magnetic tape drive should have a name similar to that listed above. One would then enter the following VMS commands before entering IDL:

ALL MUA0:	Allocate the tape drive to the user.
MOUNT/FOR MUA0:	Mount tape MUA0:
DEFINE MT3 MUA0:	Define the logical name MT3 as the tape drive name. In this example, the UNIT parameter below is then "3".

The SOHO/CDS group (in particular, Dr. William Thompson) has developed IDL procedures for the Unix operating system that mimic the IDL/VMS tape accessing commands. These procedures only can be used for IDL Version 3.1 or later running under Unix. These procedures have been set up so that the tape drive name must be assigned to an environment variable of type MTn, similar to that in VMS. Tape drive names in Unix are a bit more difficult to ascertain than VMS tape drives. In Unix, enter ls -l /dev/*mt* (Ultrix or OSF/1) or ls -l /dev/*st* (Sun Unix). This command should give you the following output:

crw-rw-rw-	1 root	system	9,20487	Jun 22	1994	/dev/nrmt0a	
crw-rw-rw-	1 root	system	9,20483	May 11	15:43	/dev/nrmt0h	⇐
crw-rw-rw-	1 root	system	9,20481	Jun 22	1994	/dev/nrmt0l	
crw-rw-rw-	1 root	system	9,20485	Jun 22	1994	/dev/nrmt0m	
crw-rw-rw-	1 root	system	9,20486	Jun 22	1994	/dev/rmt0a	
crw-rw-rw-	1 root	system	9,20482	Apr 20	11:45	/dev/rmt0h	⇐
crw-rw-rw-	1 root	system	9,20480	Jun 22	1994	/dev/rmt0l	
crw-rw-rw-	1 root	system	9,20484	Jun 22	1994	/dev/rmt0m	

Note that 2 entries have "times" instead of "years" listed in the column just before the device name. The prefix "n" simply means that the tape drive will not rewind automatically after every magnetic tape command given (*i.e.*, "nrmt0h" does not rewind the tape after use, while "rmt0h" does rewind the tape). One would then enter the following command before entering IDL:

setenv MT1 /dev/nrmt0h	The environment variable MT1 points to tape drive /dev/nrmt0h, hence UNIT = 1 in the examples below.

As of this writing, tape access on IBM compatible running DOS and Macintosh machines has not been fully implemented. Should the need arise, these platforms will be fully integrated into the software described below.

2.2 Archiving from an X–Terminal or X–Window Workstation

Upon entering IDL, type XWINTAPE (upper or lower case) or XWINTAPE, SFDU at the IDL prompt. In a few seconds, a large widget filled with subwidgets, as shown in Figure 1, will appear entitled: *XWINTAPE: Read/Write FITS/SFDU Tapes*. Step–by–step instructions directing you how to use this widget to read or write a tape can be found in the top Message: widget text–box. Note that for each message that is displayed in this box, a more detailed explanation of the directions can be found by clicking the Help widget button. See the *Widget Controls: XWINTAPE* section of this software note for a more detailed description of the XWINTAPE widget. To read/write an archive tape, follow this recipe:

1. If the tape does not contain a SFDU header file (for type Read) or you do not wish to place one on tape (for type Write), click on the ◇ No SFDU button in the Tape Header widget *before clicking any other buttons*. You then are telling XWINTAPE that this is a standard FITS tape I/O. **Note that for ALL SOHO/CDS tape archives, the tape must contain a SFDU header file.**
2. For normal operation of XWINTAPE, the first widget button to click is ◇ Read or ◇ Write in the Tape Access box.
3. Then if you wish to override the standard IDL tape accessing commands (see §7 *Help! — Archiving Problems that may Arise*), click the ◇ Unix mt & dd Commands button in the Access Override box. **Normally there will be no need to do this!**
4. The next button to click is the Tape Drive widget. Then select one of the values (*i.e.*, MT0 ... MT9). Note that before entering IDL, one should have set an environment variable (Unix) or logical name (VMS) of the form "MTn", where "n" is an integer between 0 and 9, that points to the name of the tape drive (see §2.1 *The Tape Drive Name*).

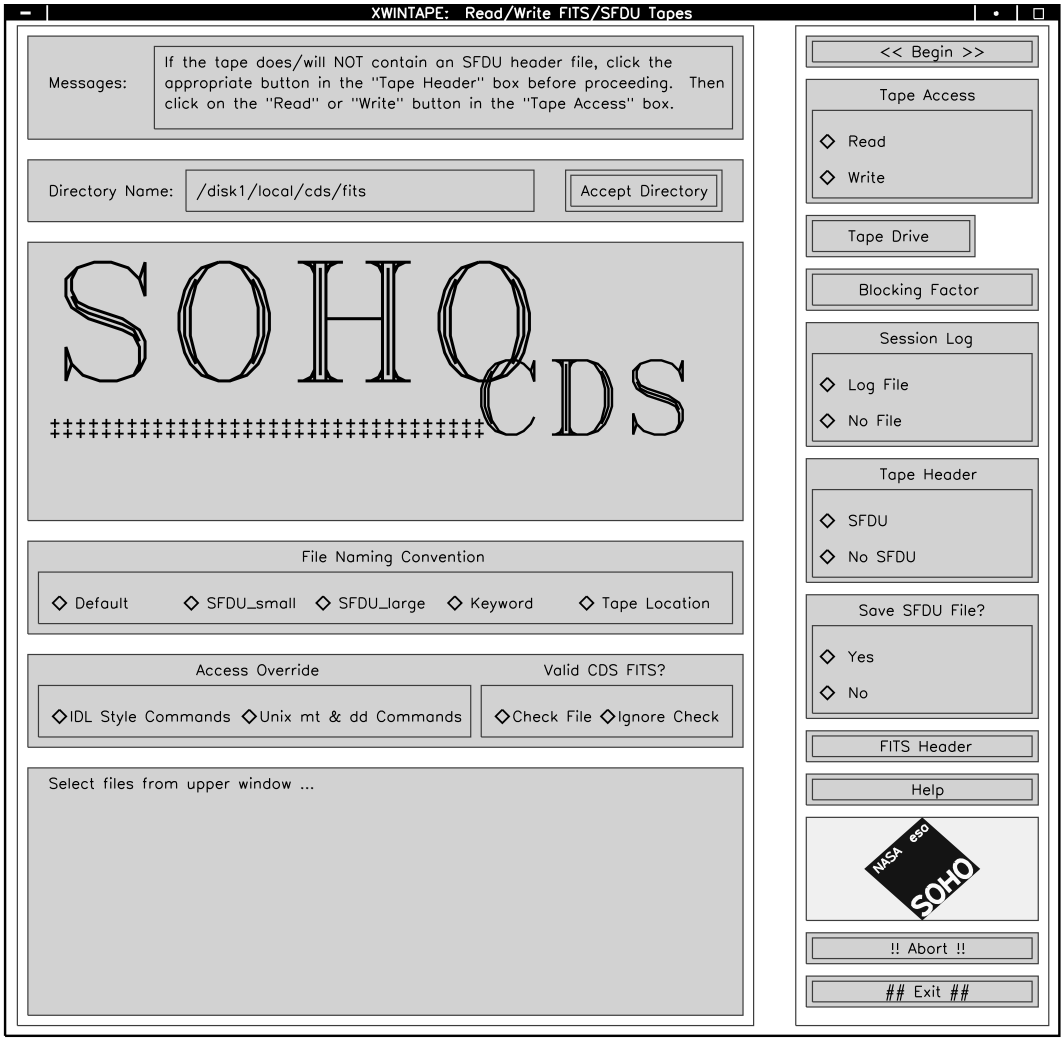

Figure 1: The XWINTAPE widget as it appears at the start of an archiving run.

5. Now click on the Directory Name: box and enter the directory that contains the files to be written to tape or where the files read from tape will be stored. Once this is typed, click on the Accept Directory button. The main widget becomes inactive and the cursor becomes an hourglass while the following occurs:

   (a) For type Write, all of the FITS files found in the entered directory will be listed in the large text widget box and the FITS headers from all of the files will be stored. Note that if no FITS files are found in this directory or an invalid directory was entered, XWINTAPE will tell you. If an incorrect directory was entered, just type in a new directory and re–accept it.

   (b) For type Read, the SFDU is read from the tape. Filenames and FITS headers are retrieved from the SFDU.

   Note that it takes a substantial amount of time to retrieve these FITS headers, especially if many files are to be accessed.

6. Once the files are listed in the upper large box, the main widget becomes active, including the FITS Header button. One can view the FITS headers associated with each file at anytime thereafter. One now selects the files that he/she wishes to read/write from/to the tape by either clicking on the filenames individually or by clicking <Select All Displayed Files>, which will move all of the FITS filenames from the top box to the bottom box. Note that if you make a mistake, files can be removed from the bottom box by simply clicking on the filename in that box.

7. If you wish to save the SFDU in a disk ASCII file, click ◇ Yes in the Save SFDU File? box. Note that also one could have retrieved the SFDU by passing a variable in the call to XWINTAPE (*e.g.*, XWINTAPE, SFDU), when XWINTAPE exits, the SFDU header is stored in the passed variable (*e.g.*, SFDU).

8. A default process for the *writing* archive procedures is to check the first FITS file in the selected directory to see if it conforms to the CDS standard. This means that array sizes and data type are consistent with those defined for the CDS software and that the FITS keywords are registered with the NSSDC (see §3). One can override this default validity check by "clicking" the appropriate button in the Valid CDS FITS? box.

9. Typically this is all you need to do before starting the archive. Next click on the << Begin >> button. Because this is a somewhat dangerous button, activating this button initiates another widget which asks you to confirm that you really wish to start the archive.

   (a) If so, click the YES, please continue! button to start the process.

   (b) If not, click the NO, please return to the main widget! button to cancel the call and return to the XWINTAPE widget.

10. Once MAGTAPE has been started, the XWINTAPE widget becomes inactive. All tape I/O messages appear in the upper large text box. Note however that tape I/O messages during a Unix mt & dd Commands run will be printed to the text window running XWINTAPE and not the widget.

    (a) If the CDS FITS validity check is made, a warning widget will appear if the FITS file does not meet the CDS standard. One has the option of quitting at that point or continuing with the archive. Whether or not one continues depends upon which invalid condition is found.

(b) When the tape I/O has successfully completed, the Help and ## Exit ## widgets become active and the cursor returns from its hourglass shape to its standard shape. Click ## Exit ## to leave XWINTAPE and destroy the widget.

Note that you may have an SFDU file (*e.g.*, sfdu19950518220807.head) and a log file (*e.g.*, tape19950518220801.log) waiting for your inspection, depending on the options set during your session.

2.3 Archiving from a Dumb Terminal

If archiving from a non–X–window environment, one must run MAGTAPE without the use of XWINTAPE. The call to MAGTAPE has the following form:

MAGTAPE, UNIT, TYPE, SFDU, BLFAC, FILES, XWSTR, NOSFDU=NOSFDU, DISK=DISK, NAME_TYPE=NAME_TYPE, OUTPUT=OUTPUT, UNIXMT=UNIXMT, CKFITS=CKFITS, XWIDGET=XWIDGET, ERRMSG=ERRMSG ,

see §5 *The MAGTAPE Procedure* for further descriptions of these parameters.

Let's assume that the magnetic tape name has been assigned to MT0, then UNIT = 0. The SFDU variable will contain a string array with the contents of the SFDU upon return of MAGTAPE. Let's now assume that we wish to read all of the FITS files off of the tape and place it in the current directory. Let's also keep a log file of the events that occur in MAGTAPE called "mytape.log". We then would perform the tape I/O with the following command:

MAGTAPE, 0, 'read', SFDU, OUTPUT='mytape.log' .

Now let's read FITS files #2, #8, and #17 (note that this corresponds to tape files #3, #9, and #18 when a SFDU header file exists on the tape, since this SFDU file is considered to be tape file #1) from the tape and place them in directory "/user/mydata" with filenames derived from the FILENAME keyword in each FITS file. Let MAGTAPE come up with its own name for the log file:

MAGTAPE, 0, 'read', SFDU, 2880, [2,8,17], DISK='/user/mydata', /OUTPUT, NAME_TYPE='kword:FILENAME' .

Note that we told MAGTAPE that the tape was written at 2880 bytes/record. But what if it was not? Assume that the tape was written at 14,400 (= 5×2880) bytes/record. The above command would not cause a problem! MAGTAPE would encounter a tape error and automatically adjust BLFAC (*i.e.*, the blocking factor) to the correct value (WARNING: this is not true if the /UNIXMT keyword was set).

We now wish to write all of the FITS files in directory '/user/fitsdata' to tape at 2×2880 bytes/record. We won't worry about keeping a log file, but we want any error messages that may be generated to be sent to the variable ERRMSG instead of being printed to the screen:

ERRMSG = "

MAGTAPE, 0, 'write', SFDU, 5770, DISK='/user/fitsdata', ERRMSG=ERRMSG

— the tape would now be written at 5770 bytes/record (including the SFDU header file).

3 Standard Formatted Data Unit

Interests by the space physics and solar science communities in *Standard Formatted Data Units* (SFDUs) has been greatly increased by the adoption of SFDUs as a standard by the GGS/ISTP (Global Geospace Science / International Solar Terrestrial Physics) project and by the pending adoption of SFDUs as a standard for NASA space science archiving generally by the National Space Science Data Center (NSSDC).

The basic SFDU building block is comprised of a LABEL field and a VALUE field, and is referred to as a Label–Value–Object (LVO). This structure is the fundamental structure element used to build SFDUs. The LVOs themselves are made up of a sequence of octets. SFDU data products are constructed from the basic LVO in one of two ways. If the VALUE field of the LVO contains purely user data, it is termed a *Simple LVO*. If, on the other hand, the VALUE field of the LVO contains purely LVOs, it is termed a *Compound LVO*.

SFDU products are always packaged in a special kind of Compound LVO called the *Exchange Data Unit* (EDU). Only EDUs may be interchanged between systems. Special types of Compound LVOs also can be used to package together application data (the *Application Data Unit* (ADU)) and data description data (the *Description Data Unit* (DDU)).

SFDUs work by associating a 20–byte LVO label with each file that serves as a reference to documentation about the file format. These types of labels will be referred to as *standard* SFDU labels from this point forward. The following represents a standard SFDU label:

CCSD3ZF0000100000001

which has the following meaning:

C	C	S	D	3	Z	F	0	0	0	0	1	0	0	0	0	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Bytes 1–4, 9–12:	ADID (Authority [CAID] and Descriptive Identifier [DDID]) (= CCSD0001)
Bytes 1–4:	CAID (= CCSD, can also be NSSD)
Bytes 9–12:	DDID (= 0001)

Byte 5: Version of SFDU label (= 3)

- 1 = Length in ASCII
- 2 = Length in Binary Integer
- 3 = Other Delimitation Technique

Byte 6: SFDU Class Identification (=Z)

- C = SFDU Identifier Service †
- D = Data Description Record †
- E = Data Element Dictionary †
- F = Description Data Unit (DDU) †

- I = (Primary) Data (of Interest) ‡
- K = Instance Identifier/Attributes ‡
- R = SFDU Reference Service ‡
- S = Supplementary Data ‡
- V = Volume Preparation Data ‡

- U = Related, Labeled Data Objects follow ⋆
- Z = "I'm an SFDU"; Labeled Object follows ⋆

Byte 7: Delimitation Type (= F)

- 0 = Spare (in Version 1 & 2, the rest are Version 3)
- S = Marker
- E = Sequential EOF(s)
- C = Consecutive EOF(s)
- F = System EOF

Byte 8: Spare (= 0)

Bytes 13–20: Description (= 00000001)

- Length of Object to follow (Version 1 & 2 Label)
- Marker (Version 3, Type S)
- EOF Count (Version 3, Type E & C)
- 00000001 (Version 3, Type F)

Under the *SFDU Class Identification* (Byte 6) in the above list,
† = Registered Metadata Objects, ‡ = Data Objects, ⋆ = SFDU Object Identifiers

Beside these standard 20–byte labels, one can have Compound LVOs containing *Parameter Value Language* (PVL). PVL lines resemble FITS headers, and as such, the FITS headers are used to construct PVL groups within the SFDU header. All PVL lines in the SFDU end with a ";" marker. Comments can also be included in the SFDU and take the form of comments in the C programming language (*i.e.*, text lines bracketed by "/*" and "*/" delimiters). All lines can extend out to 80 characters (*i.e.*, bytes) <u>including</u> the line ending <newline> character.

3.1 Sample SFDU

The following is an example of an SFDU header file for two FITS file written to tape. Imbedded comments help explain each entry.

```
CCSD3ZF0000100000001NSSD3VS00227MRK**001
/*                                                                        */
/* This data file is a "detached" Standard Formatted Data Unit (SFDU) file   */
/* referencing FITS files loaded to this archive tape.  It contains filename */
/* information and the primary FITS header for each file.  The FITS files    */
/* contain archival data from the Coronal Diagnostic Spectrometer (CDS)      */
/* onboard SOHO.  For further information concerning the tape archiving      */
/* procedure and SFDUs, see CDS Software Note #28.                            */
/*                                                                        */
/* In the VOL_INFORMATION block (NSSD3VS00227), the keywords have the         */
/* following meaning:                                                         */
/*     VOL_CREATION_TIME: The date and time the archive volume (i.e.,         */
/*                         tape/CD) was made -- CCSDS format.                 */
/*     VOL_BYTES_RECORD:  The number of bytes in each record for each file   */
/*                         on the tape/CD (= 2880*n, 1<=n<=10).               */
/*     TOTAL_VOL_BYTES:   Total number of bytes (including this SFDU header  */
/*                         file) on the tape/CD (i.e., FITS + SFDU sizes).    */
/*     TOTAL_FITS_FILES:  Total number of FITS files on the tape/CD.          */
/*     TOTAL_FITS_BYTES:  Total number of bytes associated with the FITS      */
/*                         files (i.e., TOTAL_VOL_BYTES - SFDU size).         */
/*     TOTAL_VMS_BLOCKS:  Total number of VMS blocks (1 block = 512 bytes)   */
/*                         associated with the FITS files. This number may    */
/*                         be useful for those working in a VMS environment. */
/*                                                                        */
/* Each file marker (CCSD3US00009) contains 2 submarkers:                     */
/*    CCSD3RS00003:  Replacement reference service -- file names associated  */
/*                    with the SFDU, where                                    */
/*                    CCSDS1 = File name (short version for PC/DOS machines) */
/*                              which follows the CCSDS1 specification,       */
/*                    CCSDS2 = File name (long version for VMS, Unix, and    */
/*                              Mac machines) which follows the CCSDS2        */
/*                              specification,                                */
/*                    CCSDS3 = Integer giving the position of the file on    */
/*                              the tape (note that the detached SFDU file    */
/*                              is designated as file #1 on the tape and the */
/*                              first FITS file is file #2 on the tape).      */
/*                              This follows the CCSDS3 specification.        */
/*                   Each file will logically be include at this spot in the */
/*                    SFDU and will logically include a LABEL (NSSD3IF00229) */
/*                    which identifies the file as a FITS file following the */
/*                    SOHO/CDS archive conventions.                           */
/*    NSSD3KS00228:  The FITS header associated with the file, which has     */
/*                    been placed in a PVL object labelled as FITS_HEADER.   */
```

```
/*                                                                          */

BEGIN_OBJECT = TAPE_INFORMATION;
     TAPE_CREATION_TIME = '1995-05-22T23:46:40.808Z';
     TAPE_BYTES_RECORD = 2880;
     TOTAL_TAPE_BYTES = 749584;
     TOTAL_FITS_FILES = 2;
     TOTAL_FITS_BYTES = 481744;
     TOTAL_VMS_BLOCKS = 941;
END_OBJECT;

CCSD$$MARKERMRK**001CCSD3US00009MRK**002CCSD3RS00003MRK**003

REFERENCETYPE = CCSDS0;
LABEL = NSSD3IF0022900000001;
REFERENCE = ( "CCSDS1 = CDS_1216.FIT"
            , "CCSDS2 = cds_window10_19940915_1216.fits"
            , "CCSDS3 = 2"
            );

CCSD$$MARKERMRK**003NSSD3KS00228MRK**004
BEGIN_OBJECT = FITS_HEADER;
     SIMPLE   = T;                /* Written by IDL: 28-Mar-1995 15:09:33.00 */
     BITPIX   = 32;                              /* Integer*4 (long integer) */
     NAXIS    = 2;
     NAXIS1   = 60;
     NAXIS2   = 120;
     DATE     = '28/03/95';
     SFDUADID = 'NSSD0229 ';                                     /* SFDU ADID */
     FILENAME = 'cds_window10_19940915_1216';
     ORIGIN   = 'SOHO-EOF';                /* Institute where file was written */
     TELESCOP = 'SOHO ';                    /* Solar Heliospheric Observatory */
     INSTRUME = 'CDS ';                    /* Coronal Diagnostic Spectrometer */
     DETECTOR = 'NIS ';            /* Normal or Grazing Incidence Spectrometer */
     STUDY_ID = 4;                                                /* Study ID */
     PROG_NUM = 5;                                           /* Study counter */
     SEQ_IND  = 0;                           /* Raster index within the study */
     RAS_ID   = -1;                                              /* Raster ID */
     DATE_OBS = '1994-09-15T12:16:53.023Z';/* Start date/time of observation */
     DATE_END = '1994-09-15T12:22:30.882Z';  /* End date/time of observation */
     OBT_TIME = 1158322642.023;                          /* Onboard start time */
     OBT_END  = 1158322979.882;                            /* Onboard end time */
     OBS_PROG = 'EMSQS ';         /* Emission Measure Study of the Quiet Sun */
     STUDYVAR = -1;                                  /* Study variation index */
     CATEGORY = 'Science ';                                 /* Study category */
     SCI_OBJ  = ' ';                                   /* Scientific objective */
     SCI_SPEC = ' ';                     /* More specific scientific objective */
     OBJECT   = ' ';                       /* Type of object or event observed */
```

```
     OBJ_ID   = ' ';                                         /* Object ID */
     TRACKING = F;                               /* Solar feature tracking */
     PROG_ID  = -1;                              /* Multi-study program ID */
     PROGNAME = ' ';                           /* Multi-study program name */
     CMP_NO   = -1;                             /* SOHO campaign ID number */
     CMP_NAME = ' ';                                     /* Campaign name */
     LL_ID    = -1;                                /* Line list ID number */
     LL_DESC  = ' ';                              /* Line list description */
     XCEN     = 42.8189;        /* Preliminary center in X of field-of-view */
     YCEN     = -3151.40;       /* Preliminary center in Y of field-of-view */
     ANGLE    = 0.00000;        /* Preliminary orientation of field-of-view */
     IXWIDTH  = 4.00000;         /* Preliminary width in X of field-of-view */
     IYWIDTH  = 226.000;         /* Preliminary width in Y of field-of-view */
     COMP_ID  = 6;                     /* Telemetry data compression scheme */
     COMP_OPT = 0;                     /* Telemetry data compression option */
     DW_ID    = 144;                      /* Data extraction window list ID */
     SER_ID   = 144;                         /* On-board sequence load ID */
     EXPCOUNT = 60;                    /* Number of exposures in the raster */
     EXPTIME  = 2.00000;                       /* Exposure time in seconds */
     NX       = 60;                       /* Number of steps in X direction */
     NY       = 1;                        /* Number of steps in Y direction */
     XSTEP    = 4.06400;                /* Step size in arcsec in X direction */
     YSTEP    = 0.00000;                /* Step size in arcsec in Y direction */
     OPSLBITS = '1011 ';              /* OPS L status bits at start of raster */
     OPS_L    = 2080;                   /* OPS L position at start of raster */
     OPSRBITS = '1010 ';              /* OPS R status bits at start of raster */
     OPS_R    = 2304;                   /* OPS R position at start of raster */
     SLIT_POS = 0;                       /* Slit position at start of raster */
     SLIT_NUM = 5;                                          /* Slit number */
     MIR_POS  = 68;                    /* Mirror position at start of raster */
     NWINDOWS = 11;                               /* Number of data windows */
     EV_ENAB  = F;                           /* Event recognition enabled */
     COMP_ERR = F;                       /* Data compression scheme error */
     VDS_ORI  = F;                 /* VDS telemetry data oriented by columns */
     VDS_ACC  = F;                                /* VDS accumulate mode */
     TTYPE1   = 'SOLAR_X ';              /* Solar X (cartesian west) axis */
     TTYPE2   = 'SOLAR_Y ';             /* Solar Y (cartesian north) axis */
     TUNIT1   = 'ARCSEC ';              /* Arcseconds from center of sun */
     TUNIT2   = 'ARCSEC ';              /* Arcseconds from center of sun */
     TRPIX1   = 1;                     /* Reference pixel along X dimension */
     TRPIX2   = 1;                     /* Reference pixel along Y dimension */
     TRVAL1   = 42.8189;            /* Reference position along X dimension */
     TRVAL2   = -3382.40;           /* Reference position along Y dimension */
     TDELT1   = 1.01600;                   /* Increments along X dimension */
     TDELT2   = 1.01600;                   /* Increments along Y dimension */
END_OBJECT;
CCSD$$MARKERMRK**004CCSD$$MARKERMRK**002
CCSD3US00009MRK**005NSSD3RS00229MRK**006
```

```
REFERENCETYPE = CCSDS0;
LABEL = NSSD3IF0022900000001;
REFERENCE = ( "CCSDS1 = CDS_1216.FIT"
            , "CCSDS2 = cds_window11_19940915_1216.fits"
            , "CCSDS3 = 3"
            );
CCSD$$MARKERMRK**006NSSD3KS00228MRK**007
BEGIN_OBJECT = FITS_HEADER;
     SIMPLE   = T;                 /* Written by IDL: 28-Mar-1995 15:09:33.00 */
     BITPIX   = 32;                               /* Integer*4 (long integer) */
     NAXIS    = 2;
     NAXIS1   = 60;
     NAXIS2   = 120;
     DATE     = '28/03/95';
     SFDUADID = 'NSSD0229 ';                                   /* SFDU ADID */
     FILENAME = 'cds_window11_19940915_1216';
     ORIGIN   = 'SOHO-EOF';               /* Institute where file was written */
     TELESCOP = 'SOHO ';                  /* Solar Heliospheric Observatory */
     INSTRUME = 'CDS ';                  /* Coronal Diagnostic Spectrometer */
     DETECTOR = 'NIS ';         /* Normal or Grazing Incidence Spectrometer */
     STUDY_ID = 4;                                              /* Study ID */
     PROG_NUM = 5;                                         /* Study counter */
     SEQ_IND  = 0;                        /* Raster index within the study */
     RAS_ID   = -1;                                            /* Raster ID */
     DATE_OBS = '1994-09-15T12:16:53.023Z';/* Start date/time of observation */
     DATE_END = '1994-09-15T12:22:30.882Z';  /* End date/time of observation */
     OBT_TIME = 1158322642.023;                       /* Onboard start time */
     OBT_END  = 1158322979.882;                         /* Onboard end time */
     OBS_PROG = 'EMSQS ';          /* Emission Measure Study of the Quiet Sun */
     STUDYVAR = -1;                                  /* Study variation index */
     CATEGORY = 'Science ';                                  /* Study category */
     SCI_OBJ  = ' ';                                   /* Scientific objective */
     SCI_SPEC = ' ';                     /* More specific scientific objective */
     OBJECT   = ' ';                       /* Type of object or event observed */
     OBJ_ID   = ' ';                                              /* Object ID */
     TRACKING = F;                                   /* Solar feature tracking */
     PROG_ID  = -1;                                   /* Multi-study program ID */
     PROGNAME = ' ';                                /* Multi-study program name */
     CMP_NO   = -1;                                 /* SOHO campaign ID number */
     CMP_NAME = ' ';                                           /* Campaign name */
     LL_ID    = -1;                                      /* Line list ID number */
     LL_DESC  = ' ';                                   /* Line list description */
     XCEN     = 42.8189;          /* Preliminary center in X of field-of-view */
     YCEN     = -3151.40;         /* Preliminary center in Y of field-of-view */
     ANGLE    = 0.00000;          /* Preliminary orientation of field-of-view */
     IXWIDTH  = 4.00000;           /* Preliminary width in X of field-of-view */
     IYWIDTH  = 226.000;           /* Preliminary width in Y of field-of-view */
     COMP_ID  = 6;                       /* Telemetry data compression scheme */
```

```
    COMP_OPT = 0;                          /* Telemetry data compression option */
    DW_ID    = 144;                           /* Data extraction window list ID */
    SER_ID   = 144;                                /* On-board sequence load ID */
    EXPCOUNT = 60;                         /* Number of exposures in the raster */
    EXPTIME  = 2.00000;                             /* Exposure time in seconds */
    NX       = 60;                            /* Number of steps in X direction */
    NY       = 1;                             /* Number of steps in Y direction */
    XSTEP    = 4.06400;                   /* Step size in arcsec in X direction */
    YSTEP    = 0.00000;                   /* Step size in arcsec in Y direction */
    OPSLBITS = '1011 ';                 /* OPS L status bits at start of raster */
    OPS_L    = 2080;                       /* OPS L position at start of raster */
    OPSRBITS = '1010 ';                 /* OPS R status bits at start of raster */
    OPS_R    = 2304;                       /* OPS R position at start of raster */
    SLIT_POS = 0;                           /* Slit position at start of raster */
    SLIT_NUM = 5;                                               /* Slit number */
    MIR_POS  = 68;                        /* Mirror position at start of raster */
    NWINDOWS = 11;                                    /* Number of data windows */
    EV_ENAB  = F;                                /* Event recognition enabled */
    COMP_ERR = F;                            /* Data compression scheme error */
    VDS_ORI  = F;                   /* VDS telemetry data oriented by columns */
    VDS_ACC  = F;                                      /* VDS accumulate mode */
    TTYPE1   = 'SOLAR_X ';                   /* Solar X (cartesian west) axis */
    TTYPE2   = 'SOLAR_Y ';                  /* Solar Y (cartesian north) axis */
    TUNIT1   = 'ARCSEC ';                    /* Arcseconds from center of sun */
    TUNIT2   = 'ARCSEC ';                    /* Arcseconds from center of sun */
    TRPIX1   = 1;                          /* Reference pixel along X dimension */
    TRPIX2   = 1;                          /* Reference pixel along Y dimension */
    TRVAL1   = 42.8189;                 /* Reference position along X dimension */
    TRVAL2   = -3382.40;                /* Reference position along Y dimension */
    TDELT1   = 1.01600;                         /* Increments along X dimension */
    TDELT2   = 1.01600;                         /* Increments along Y dimension */
END_OBJECT;
CCSD$$MARKERMRK**007CCSD$$MARKERMRK**005
```

3.2 SFDU Registration with the Consultative Committee for Space Data Systems (CCSDS)

The archiving procedures that have been developed for CDS are written in IDL. In particular, the SFDU for a set of FITS files is generated with the **sfdu_make.pro** procedure. This procedure makes use of the SFDU registration files in the */cs/data/sfdu* directory. The registration with the CCSDS (the controlling agency) took place in early December 1995. These files are described as follows:

File Name	ADIDNAME	Description
sfdu_vol_dict.reg0:	**NSSD0227**	Volume keyword dictionary.
sfdu_cds_cat_obj.reg0:	**NSSD0228**	Describes how the SFDU/PVL relates to the FITS headers.
sfdu_cds_fits_dict.reg0:	**NSSD0229**	SOHO/CDS FITS header dictionary.
sfdu_cds_desc.reg0:	**NSSD0238**	SOHO/CDS instrument description.

Note that the "0" in ".reg0" indicates that these are the original registration files. Future changes to these registration files will have filename extensions of ".reg1," ".reg2," etc.

The SFDU detached header that will appear as the first file on the tape or CD has various markers in it that point to the registration files listed above. Figure 2 shows a flow chart of the SFDU template used for the SOHO/CDS. The first line of the SFDU header has a "Z" class marker indicating that the file is an SFDU and that a Labeled Object follows. Imbedded in this marker (*i.e.*, the SFDU) is an "NSSD" *ADID* marker containing the Volume Preparation Data ("V" class) defined by the "0227" object description (*i.e.*, this marker points to the **NSSD0227** registration packet). This volume data is written in PVL. Following this initial volume data, which contains information concerning the tape or CD, a series of markers follow, containing the SFDU Reference Service. This reference service contains a LABEL pointing to the **NSSD0229** registration packet defining the FITS keywords — note that the LABEL on the sample SFDU above has NSSD3IF0022900000001, where "I" (the 6th character) indicates the file it is associated with is the primary data of interest and the "F" (the 7th character) indicates that the file ends with a tape (or CD) *eof* marker. The reference service also contains the name associated with the file (*i.e.*, the DOS version CCSDS1 protocol, the Unix/VMS/MacOS version CCSDS2 protocol, and the file's position on the tape [= CCSDS3 protocol]). Attached to this reference service is a Instance Identifier/Attributes ("K" class) which points to the **NSSD0228** registration packet. This area of the SFDU contains all of the keyword values and definitions of the FITS file headers, listed separately from the files in this SFDU header for the user's convenience. A closing marker follow each Reference Service entries.

The SFDU header file on the tape references the above mentioned registration files. In turn, these description and dictionary files point to each other depending on whether a DDRID (Data Description Record ID), a DEDID (Data Entity Dictionary ID), and/or a SUPID (Supplementary Metadata Data) entry is embedded (see Figure 3). For instance, the SUPID entry NSSD0238 points to a file that contains a summary of the SOHO mission description, and *in particular* the CDS instrument description. **Note that each instrument will have its own SUPID in its respective SFDU.** Additional DDRID entries not shown in Figure 3 include NSSD0006, which indicates that the data is in FITS format, and CCSD0006, which indicates that the Parameter Value Language (PVL) formalism is being used.

Each of the FITS files created from the CDS software will have an SFDUADID keyword in the header. For all CDS data, this SFDU ADID will be

$$\boxed{\text{SFDUADID = NSSD0229.}}$$

3.3 Revising the SFDU Registration Forms

Should any new descriptions or FITS keyword definitions need to be added the SFDU, only those individuals who have been designated on the previous SFDU registration (or update) forms (*i.e.*,

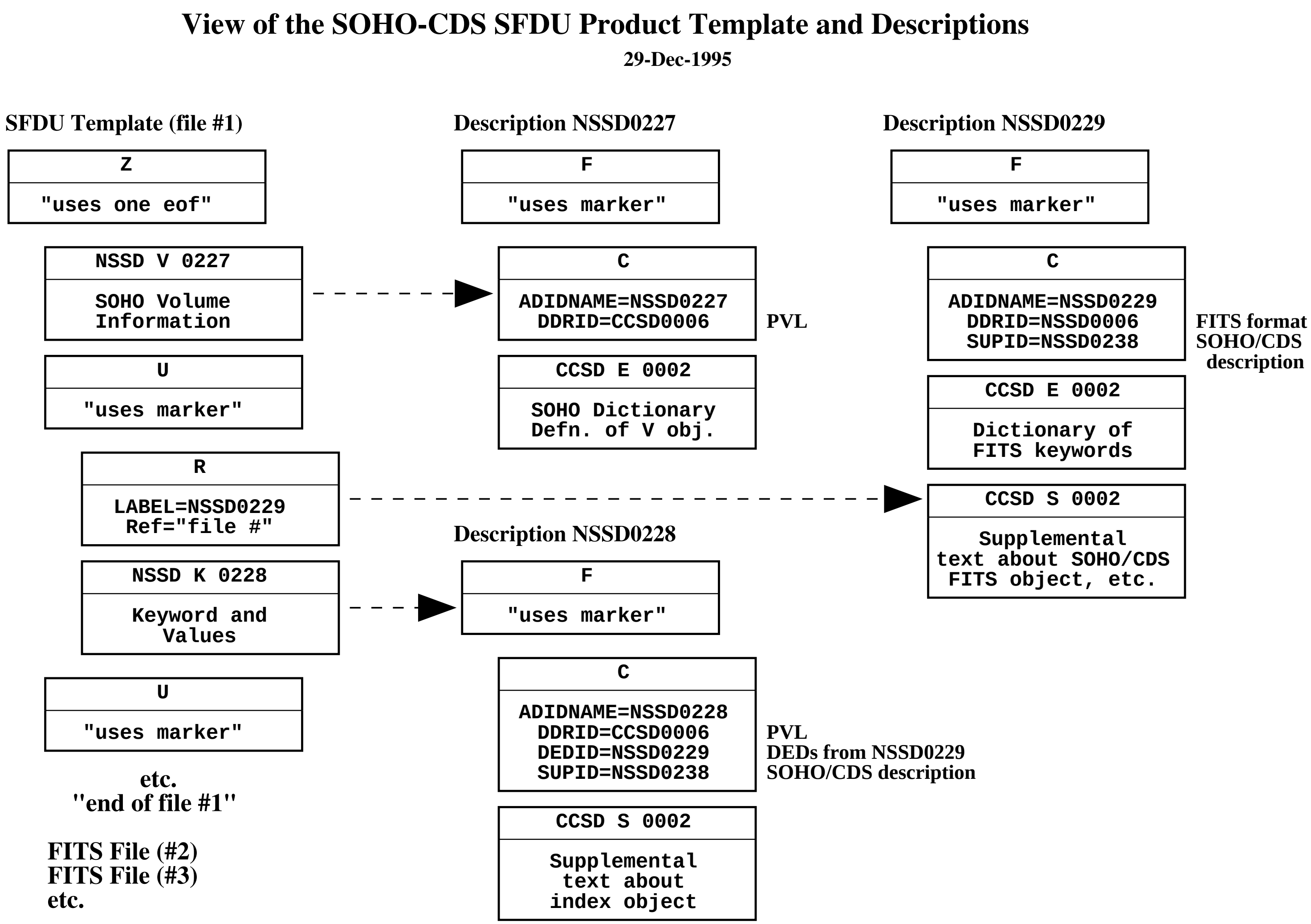

Figure 2: The SOHO/CDS SFDU product template flow chart as described in the text.

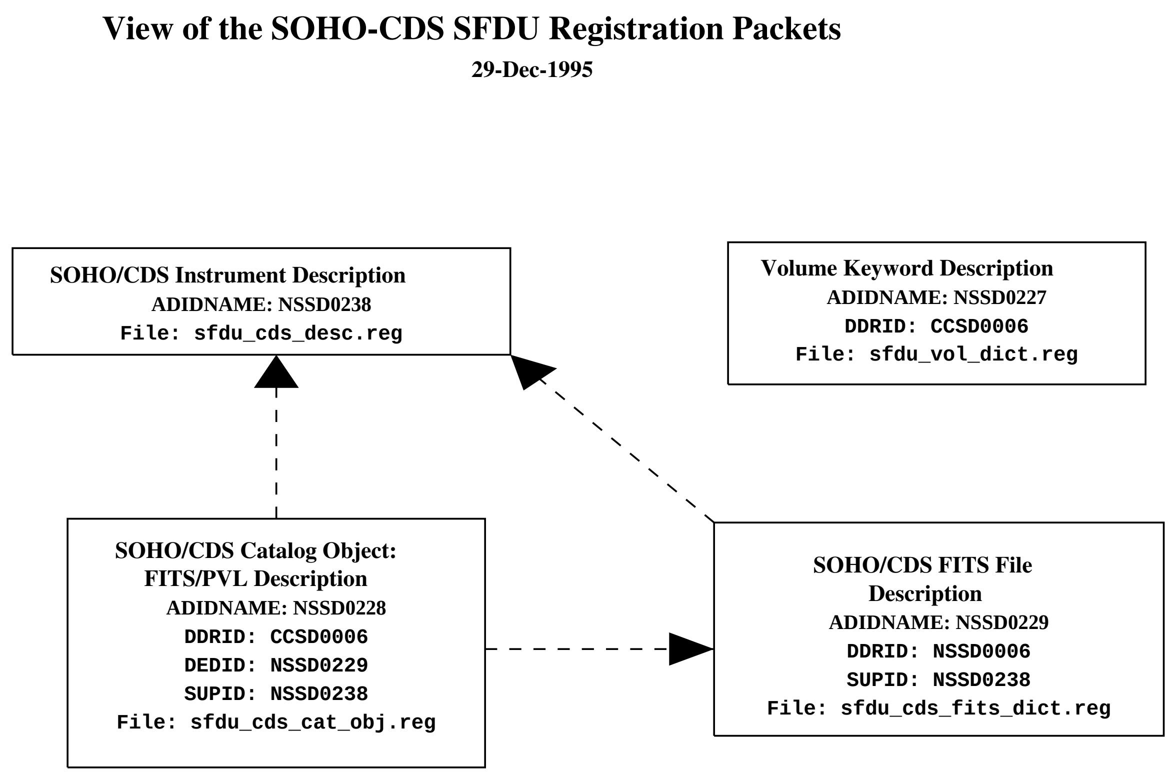

Figure 3: The relationship between the SOHO/CDS SFDU registration packets.

files) as a REVISOR can make revisions. Currently, only Drs. Art Poland, Richard Harrison, Bill Thompson, Dave Pike, and Don Luttermoser are designated as *revisors*. The original SFDU registration files have a *.reg0* suffix attached to the file name. To submit a revision, say the first revision from the original registration form, copy the four **.reg0* files to files ending with *.reg1*:

```
mv   sfdu_cds_cat_obj.reg0     sfdu_cds_cat_obj.reg1
mv   sfdu_cds_desc.reg0        sfdu_cds_desc.reg1
mv   sfdu_cds_fits_dict.reg0   sfdu_cds_fits_dict.reg1
mv   sfdu_vol_dict.reg0        sfdu_vol_dict.reg1
```

Then, one can make modifications to the *.reg1* files and resubmit them. The old files should not be deleted since they may be needed for reference in the future. In this way, it will be easy to keep track of the various revisions.

When submitting a revision to the SFDU registration files, one must be sure that the unique SOHO/CDS ADIDNAME is in the file. Before submitting an updated revision, be sure to modify the REVISOR group in the registration/update form if you wish to add or remove the names of those people who can make future revisions. Also note that the person who is making the revisions to the SFDU registration form must include his name in the REGISTRANT group section. Once the modifications to the registration/update form are complete, email the form to John Garrett of the NSSDC at

caoprod@nssdca.gsfc.nasa.gov

and he will reply when the revision has been recorded.

4 The FITS Files

As mentioned earlier, all telemetry data from the CDS will be stored in FITS format. A detailed description of these FITS files can be found in the CDS Software Notes #3 *Converting CDS Telemetry to FITS Files*, #4 *IDL Software for FITS Binary Tables*, and #11 *Converting CDS Calibration Telemetry to FITS*. The following tabulations contain the FITS keywords that have been registered with the CCSDS as of this printing. The value type that can be associated with each keyword is presented in square (*i.e.*, []) brackets: [INT] = integer, [LONG] = long integer, [FLOAT] = single precision real, [DOUBLE] = double precision real, [STRING] = string of ASCII characters, and [LOGICAL] = logical value ("T" for true and "F" for false).

FITS keywords that are always present:

ANGLE	Preliminary orientation of field-of-view (i.e., instrument) to solar north [FLOAT]
BITPIX	Number of bits per pixel for the data [INT]
CATEGORY	Study category: "T" for Test, "S" for science, "C" for calibration — implied by STUDY_ID [STRING]
CMP_NAME	Campaign name (if applicable) [STRING]
CMP_NO	SOHO campaign ID number (if applicable) [INT]
COMP_ERR	True (T) if data compression scheme error was encountered [LOGICAL]
COMP_ID	Telemetry data compression scheme ID [INT]
COMP_OPT	Telemetry data compression option [INT]
DATE	Date FITS file was created, dd/mm/yy [STRING]
DATE_OBS	Start date/time of observation (raster) – CCSDS format [STRING]
DATE_END	End date/time of observation [STRING]
DETECTOR	CDS detector: 'NIS' – Normal Incidence Spectrometer or 'GIS' – Grazing Incidence Spectrometer [STRING]
DW_ID	Data extraction window list ID [INT]
EV_ENAB	True (T) if event recognition enabled, false (F) otherwise [LOGICAL]
EXPCOUNT	Number of exposures in the raster [INT]
EXPTIME	Exposure time in seconds [FLOAT]
EXTNAME	'DATA' if FITS contents is scientific data, etc. [STRING]
FILENAME	FITS filename on originator machine w/o extension or path [STRING]
GCOUNT	Group count [INT]
INS_ROLL	Instrument roll angle (from calibration) [FLOAT]
INS_X0	Instrument origin pointing (legs) X-axis [FLOAT]
INS_Y0	Instrument origin pointing (legs) Y-axis [FLOAT]
INSTRUME	The instrument onboard SOHO for which the data were taken, *i.e.*, 'CDS' = Coronal Diagnostic Spectrometer [STRING]
IXWIDTH	Size of instrument field of view in X direction — calculated from the pointing information, data window tables, and binning factors [FLOAT]
IYWIDTH	Size of instrument field of view in Y direction [FLOAT]
LL_ID	Line list ID number — implied by DW_ID [INT]
LL_DESC	The name of the line list used — implied by DW_ID [STRING]
MIR_POS	Mirror position at start of raster [INT]
NAXIS	Number of axes for the data (=2 for image) [INT]
NAXIS1	Array size in X direction [LONG]
NAXIS2	Array size in Y direction [LONG]
NWINDOWS	Number of data extraction windows [INT]
NX	Number of steps in X direction [INT]
NY	Number of steps in Y direction [INT]
OBJ_ID	Object ID (if applicable) [INT]
OBJECT	Type of object or event observed [STRING]
OBS_MODE	Observing mode — a character string combining the values of DETECTOR, SLIT_NUM, RAS_ID, and EXPTIME, *e.g.*, 'NIS-S1-R3-E100' [STRING]
OBS_PROG	Name of the study — implied by STUDY_ID [STRING]
OBT_TIME	Onboard start time [DOUBLE]
OBT_END	Onboard end time [DOUBLE]

OPSLBITS	OPS L status bits at start of raster, as a character string of '0's and '1's [STRING]
OPSRBITS	OPS R status bits at start of raster [STRING]
OPS_L	OPS L position at start of raster [INT]
OPS_R	OPS R position at start of raster [INT]
ORIGIN	Institute where file was written: SOHO–EOF or RAL [STRING]
PCOUNT	Random parameter count [INT]
PROG_ID	Observing program ID, linking studies together [LONG]
PROG_IND	Repeated study index [LONG]
PROG_NUM	Study counter number [LONG]
PROGNAME	Observing program name, if applicable [STRING]
RAS_ID	Raster ID number — determined from STUDY_ID and SEQ_IND [LONG]
RAS_VAR	The raster variation index [INT]
SC_ROLL	Spacecraft roll [FLOAT]
SC_X0	Spacecraft X-direction pointing [FLOAT]
SC_Y0	Spacecraft Y-direction pointing [FLOAT]
SCI_OBJ	General science objective [STRING]
SCI_SPEC	More specific science objective [STRING]
SER_ID	Onboard sequence load ID — taken from the "Sequence ID" in the telemetry stream [LONG]
SEQ_IND	Raster index within the study [INT]
SEQ_NUM	Unique number of each file. This number is intended to serve as a unique identifier to a specific instance of a raster. Each data file generated by a CDS raster would have a unique value of SEQ_NUM [LONG]
SEQVALID	True (T) if the raster contains useful data. One possible use of this keyword is to signal whether or not the pointing was changed during the execution of the raster. If so, then the software will need to be able to distinguish between solar feature tracking and true pointing changes [LOGICAL]
SFDUADID	SFDU Authority and ADID label of INSTRUME, i.e., 'NSSD0229' for the CDS [STRING]
SIMPLE	First header line of all standard FITS files [LOGICAL]
SLIT_NUM	The number of the slit used [INT]
SLIT_POS	Slit position at start of raster [INT]
STUDYVAR	Study variation index [INT]
STUDY_ID	The study ID number of the program [LONG]
TELESCOP	Name of the telescope, i.e. 'SOHO' = Solar Heliospheric Observatory [STRING]
TFIELDS	Number of data columns in the binary tables [INT]
TITLE	Study title [STRING]
TITLE_ID	Study title ID [LONG]
TRACKING	Solar feature tracking: True (T) if solar feature tracking is used, otherwise false (F) [LOGICAL]
WAVEMAX	Maximum wavelength observed in Angstroms [FLOAT]
WAVEMIN	Minimum wavelength observed in Angstroms [FLOAT]
XCEN	Preliminary center in X–direction of field–of–view [FLOAT]
XSTEP	Step size in arcsec in X direction [FLOAT]
XTENSION	First header line of binary table FITS files, typically set to 'BINTABLE' [STRING]
YCEN	Preliminary center in Y–direction of field–of–view [FLOAT]
YSTEP	Step size in arcsec in Y direction [FLOAT]

Keywords present only for VDS data:

VDS_ACC	True if VDS is operated in accumulate mode [LOGICAL]
VDS_MODE	VDS readout mode [STRING]
VDS_ORI	True if VDS telemetry data oriented by columns [LOGICAL]
VDS_PMCP	VDS MCP programmable voltage setting [INT]

Keywords present only for GIS data:

BUGHV1	High voltage, detector 1 [FLOAT]
BUGHV2	High voltage, detector 2 [FLOAT]
BUGHV3	High voltage, detector 3 [FLOAT]
BUGHV4	High voltage, detector 4 [FLOAT]
BKGCHK1	Lookup table checksum, detector 1 [LONG]
BKGCHK2	Lookup table checksum, detector 2 [LONG]
BKGCHK3	Lookup table checksum, detector 3 [LONG]
BKGCHK4	Lookup table checksum, detector 4 [LONG]
BIGMCP1	High voltage current, detector 1 [FLOAT]
BIGMCP2	High voltage current, detector 2 [FLOAT]
BIGMCP3	High voltage current, detector 3 [FLOAT]
BIGMCP4	High voltage current, detector 4 [FLOAT]
BUGMCPB1	MCP F/face, detector 1 [FLOAT]
BUGMCPB2	MCP F/face, detector 2 [FLOAT]
BUGMCPB3	MCP F/face, detector 3 [FLOAT]
BUGMCPB4	MCP F/face, detector 4 [FLOAT]
BKGRCD1	Raw event count, detector 1 [LONG]
BKGRCD2	Raw event count, detector 2 [LONG]
BKGRCD3	Raw event count, detector 3 [LONG]
BKGRCD4	Raw event count, detector 4 [LONG]
BKGULCD1	Upper level discriminator count, detector 1 [LONG]
BKGULCD2	Upper level discriminator count, detector 2 [LONG]
BKGULCD3	Upper level discriminator count, detector 3 [LONG]
BKGULCD4	Upper level discriminator count, detector 4 [LONG]
BKGHV1L	High voltage last set value, detector 1 [FLOAT]
BKGHV2L	High voltage last set value, detector 2 [FLOAT]
BKGHV3L	High voltage last set value, detector 3 [FLOAT]
BKGHV4L	High voltage last set value, detector 4 [FLOAT]
BKGLDLD1	Last setting low level discriminator, detector 1 [FLOAT]
BKGLDLD2	Last setting low level discriminator, detector 2 [FLOAT]
BKGLDLD3	Last setting low level discriminator, detector 3 [FLOAT]
BKGLDLD4	Last setting low level discriminator, detector 4 [FLOAT]

Binary table column headers — note that the "n" in these keywords represent integers which associates the given keyword to the specific column "n", "n" always starts from 1:

TFORMn	Data format and number of data points [STRING]
TTYPEn	Label for data [STRING]
TDIMn	Dimensions [STRING]
TDESCn	Dimension labels [STRING]
TCUNIn	Units along each of the dimensions [STRING]
TRPIXn	Reference pixel position [STRING]
TRVALn	Axes values at reference point [STRING]
TDELTn	Pixel spacing along axes [STRING]
TROTAn	Rotation angle of dimensions [STRING]
TUNITn	Units of the data [STRING]
TNULLn	Data missing flag value [LONG]
TDMINn	Data minimum value [LONG or FLOAT depending on TUNITn]
TDMAXn	Data maximum value [LONG or FLOAT depending on TUNITn]
TDETXn	Start column on detector [INT]
TDETYn	Start row on detector (NIS detector only) [INT]
TBINXn	Detector binning across columns [INT]
TBINYn	Detector binning across rows [INT]
TWAVEn	Principle wavelength of first order [FLOAT]
TWMINn	Minimum wavelength of first order [FLOAT]
TWMAXn	Maximum wavelength of first order [FLOAT]
TWBNDn	Wavelength band (NIS: 1 or 2, GIS: 1 to 4) [INT]
TGORDn	Grating order [INT]

5 The MAGTAPE Procedure

This section describes the MAGTAPE procedure in detail. This routine is the main driver of the tape I/O. One would directly run this procedure from a non–X–window terminal. If you have access to an X–window terminal, it is recommended that you use XWINTAPE to drive MAGTAPE. The call to MAGTAPE has the following form:

```
MAGTAPE, UNIT, TYPE, SFDU, BLFAC, FILES, XWSTR, NOSFDU=NOSFDU,
  DISK=DISK, NAME_TYPE=NAME_TYPE, OUTPUT=OUTPUT, UNIXMT=UNIXMT,
  XWIDGET=XWIDGET, ERRMSG=ERRMSG
```

The following table describes each of these parameters, note that many of these parameters are optional (key: I = *input*, I* = *optional input*, O = *output*, O* = *optional output*, K = *optional keyword*):

UNIT: (I) Tape unit number. Must be set by the following commands prior to entering IDL (see §2.1 *The Tape Drive Name*:

VMS: `ALLOCATE MUA0:` (assuming MUA0: is a tape drive)
`MOUNT/FOR MUA0:`
`DEFINE MT1 MUA0:` (in this example, UNIT = 1, possible values: MT0 ... MT9, relate to UNIT = 0 ... 9)

Unix: `setenv MT0 /dev/nrmt0` (assuming /dev/nrmt0 is a tape drive, in this example, UNIT=0)

TYPE: (I) 'write': Write FITS files to the tape.
'read': Retrieve FITS files from the tape.

BLFAC: (I*) Blocking factor (1–10) = # of 2880 byte records per block [DEFAULT = 1].

FILES: (I*) This parameter can either be:

(1) For TYPE = 'read':
A vector of tape file numbers [INT] (or names [STR]) to be read off of the tape.

(a) If tape file numbers [INT] are given, then the disk filenames are constructed as described under the keyword "NAME_TYPE".
****
NOTE: The first file may contain the SFDU Header File. If so, consider its Tape File Number to be "0", so that the first FITS file is Tape File #1 (*i.e.*, input your tape file location integers as though this header file didn't exist → FITS File #1 = Tape File #1).
If NO SFDU header file exists on the tape, the first file (now a FITS file) is STILL Tape File #1.
****

(b) If tape file names [STR] are given, then the names supplied are compared with the names in the SFDU header file:
Form: SFDU_LG for Unix, VMS, and MacOS.
Form: SFDU_SM for DOS.

(c) If not supplied, the program will read the entire tape and place the contents in the directory associated with DISK using disk names as described by the keyword NAME_TYPE.

(2) For TYPE = 'write':
A string vector that contains all of the filenames from disk that will be written to tape. If not given, the program writes all files in the directory assigned to DISK and named '*.fits' (or '*.FIT' for DOS files) to the tape.

XWSTR:	(I*,O*)	A string array that contains informational text that is either displayed to the screen or to the FILENAME widget (set to XWIDGET internally) if XWINTAPE drives this routine. **Note that this variable is not needed for non–X–window runs.** Also note that MAGTAPE adds text to this array.
SFDU:	(O)	String array that contains the SFDU header created for or read from the tape. Set to 'NONE' if the /NOSFDU keyword has been set.
NOSFDU:	(K)	Do not use the Standard Formatted Data Unit labels. The default is to use them! MAGTAPE, 1, 'write', /NOSFDU
DISK:	(K)	Name of the directory where the files are to be read or written. MAGTAPE, 1, 'write', DISK='/cds/data'
NAME_TYPE:	(K)	Naming convention for TYPE 'read' only. Four possibilities exist: 'kword:XXX': Use information from the specified FITS keyword XXX, (i.e., if NAME_TYPE = 'kword:filename', then whatever name that is associated with the FILENAME keyword in the FITS header is used). If this keyword is used, the input filenames in FILES (if supplied) must contain integers of the file location on the tape. 'sfdu_sm': Use the "short name" from the SFDU header file [DEFAULT for reads in DOS]. 'sfdu_lg': Use the "long name" from the SFDU header file [DEFAULT for reads in Unix, VMS, or MacOS]. 'tapeloc': Use 'tapennnn.fits', where "nnnn" is the tape file number.
OUTPUT:	(K)	Filename of an output file that contains a listing of all events that takes place during the tape I/O. Using /OUTPUT in the call to MAGTAPE, forces MAGTAPE to generate its own unique filename (based on the current time) for this log.
UNIXMT:	(K)	Override the IDL tape reading routines and simply use the Unix *mt* and *dd* commands to access the tape (Unix systems only). Note this is the default under Unix/IDL versions earlier than 3.1.
XWIDGET:	(K)	If set, the widget program XWINTAPE is driving this procedure and any text that is normally printed to the screen will instead be printed to the widget. **Note that this variable is not needed for non–X–window runs.**

CKFITS:	(K)	If set, check the first file to see if the file conforms to the CDS level-1 binary FITS protocol. If so, the FITS file (and it is assumed that all other FITS files in this directory) is valid. The default is to check the first file for validity.
ERRMSG:	(K)	If defined and passed, then any error messages will be returned to the user in this parameter rather than being handled by the IDL MESSAGE utility. If no errors are encountered, then a null string is returned. In order to use this feature, the string ERRMSG must be defined first, e.g., ERRMSG = '' MAGTAPE, 1, 'read', SFDU, ERRMSG=ERRMSG IF ERRMSG(0) NE '' THEN ...

In the following two subsections, various examples will be presented for tape archive reading and writing. Other examples already have been introduced in §2 *Brief Tutorial*, refer to this section simple demonstrations. In the following subsections, assume that the tape drive name has been associated with UNIT = 1 and that the SFDU header is contained in the string array SFDU. **WARNING NOTE: Should MAGTAPE be accessed concurrently in a single IDL session, the passed SFDU variable should be undefined for all calls to MAGTAPE.** Otherwise, MAGTAPE assumes the SFDU has already been read from the current tape and uses the passed SFDU to access the FITS files on the tape.

5.1 Tape Reading

Example 1: Read all FITS files off of the tape and load to the current directory. Make a log file of the tape I/O and use the Unix *dd* and *mt* commands instead of the IDL tape commands.

MAGTAPE, 1, 'read', SFDU, /OUTPUT, /UNIXMT

The "/OUTPUT" keyword informs MAGTAPE to keep a log of all messages sent to the screen and dump them to ASCII file *tapennnnnnnnnnn.log*, where *nnnnnnnnnnn* is a string of numbers derived from the current date and time. The "/UNIXMT" keyword tells MAGTAPE to write and run a Unix script to access the tape drive using the *dd* and *mt* Unix commands.

Example 2: Read FITS files stored on tape with names in the string array FNAMES and load to directory "/cds/fits/readem" using the CCSDS1 (*i.e.*, DOS–like) naming protocol. Make a log file of the tape I/O called "fitsrdtape.log".

FNAMES = ['cds_window1_1215.fits', 'cds_window2_1500.fits', 'cds_window4_1215.fits']
MAGTAPE, 1, 'read', SFDU, 2880, FNAMES, OUTPUT='fitsrdtape.log',
  NAME_TYPE='sfdu_sm', DISK='/cds/fits/readem'

Since we wanted to select only a portion of the FITS files stored to tape, we had to include the tape blocking factor (BLFAC) in the call to MAGTAPE preceding the FNAME passed variable. For reading a tape, the blocking factor does not need to be known before a "read" is attempted. If the loaded tape was written at 28800 (= 10×2880) bytes/record, the passed BLFAC value of 2880 would <u>not</u> cause a crash. MAGTAPE would detect the error and make the appropriate change to BLFAC internally and continue reading the tape. Setting NAME_TYPE to 'sfdu_sm' tells MAGTAPE to use the DOS–style filenames associated with the "$1" keyword in the REFERENCE PVL parameter in the SFDU.

Example 3: Read FITS files #1–10 and #21–27 off of the tape and load to the current directory using the tape–file position numbers for the filenames. Don't keep a log file.

```
FNAMES = INDGEN(10) + 1
FNAMES = [FNAMES, INDGEN(7) + 21]
MAGTAPE, 1, 'read', SFDU, 2880, FNAMES, NAME_TYPE='tapeloc'
```

Once again, we wanted to select only a portion of the FITS files stored to the tape, so we included a *guessed* blocking factor (*i.e.*, 2880). FNAMES contains an array of integers (*i.e.*, [1, 2, ..., 10, 21, 22, ..., 27]) which is passed to MAGTAPE. Setting NAME_TYPE to 'tapeloc' tells MAGTAPE to make filenames associated with the "$5" keyword (*i.e.*, the tape location, first FITS file on tape = #1) in the REFERENCE PVL parameter in the SFDU. The stored files (17 total) will be loaded into the current directory with names: *tape0001.fits, tape0002.fits, ..., tape0027.fits.*

5.2 Tape Writing

Example 1: Write all FITS files from the current directory to the tape. Make a log file of the tape I/O and use the Unix *dd* and *mt* commands instead of the IDL tape commands.

```
MAGTAPE, 1, 'write', SFDU, /OUTPUT, /UNIXMT
```

The "/OUTPUT" keyword informs MAGTAPE to keep a log of all messages sent to the screen and dump them to ASCII file *tapennnnnnnnnnn.log*, where *nnnnnnnnnnn* is a string of numbers derived from the current date and time. The "/UNIXMT" keyword tells MAGTAPE to write and run a Unix script to access the tape drive using the *dd* and *mt* Unix commands. Since the tape blocking factor was not passed to MAGTAPE, the default value of 2880 bytes/record is used.

Example 2: Write the FITS files stored in directory "/cds/fits/writeit" with names in the string array FNAMES to tape. Make a log file of the tape I/O called "fitswrtape.log". Write the tape at 5×2880 bytes/record. Do <u>not</u> check to see if the FITS files conform to the CDS standard.

```
FNAMES = ['cds_window1_1215.fits', 'cds_window2_1500.fits', 'cds_window4_1215.fits']
MAGTAPE, 1, 'write', SFDU, 14400, FNAMES, OUTPUT='fitswrtape.log', CKFITS=0,
  DISK='/cds/fits/writeit'
```

Three files will be written to tape from directory "/cds/fits/writeit" (with their filenames stored in FNAMES) at 14,400 bytes/record using the IDL tape access commands. Upon return from MAGTAPE, an ASCII log file by the name of "fitswrtape.log" will exist in the current directory.

Example 3: Write the 3rd through 9th FITS files in the current directory as they appear in a directory listing. Don't write a log file and use all defaults in MAGTAPE. Pass any error message back through the ERRMSG variable.

```
ERRMSG = ''
MAGTAPE, 1, 'write', SFDU, 2880, [3, 4, 5, 6, 7, 8, 9], ERRMSG=ERRMSG
```

WARNING! No tape I/O will commence from this run! MAGTAPE will return with the following error message in ERRMSG:

Passed filenames must be "strings" when writing to tape!

In order to carry out the above example, the actual names of the files must be passed to MAGTAPE:

ERRMSG = ''
MAGTAPE, 1, 'write', SFDU, 2880, ['cds3.fits', 'cds4.fits', 'cds5.fits', 'cds6.fits', 'cds7.fits', 'cds8.fits', 'cds9.fits'], ERRMSG=ERRMSG

6 Widget Controls: XWINTAPE

Upon entering IDL (and assuming your session has the proper PATH environment, pointing to the IDL/CDS software directories, set in the system or user login file), type XWINTAPE (upper or lower case) or XWINTAPE, SFDU at the IDL prompt. In a few seconds, a large widget filled with subwidgets (see Figure 1) will appear entitled: *XWINTAPE: Read/Write FITS/SFDU Tapes*. The main widget is split into two halves: The left–hand–side (LHS) contains (from top to bottom):

- Message: text widget: Short and to–the–point information concerning the operation of the XWINTAPE widget.
- Directory Name: text and [Accept Directory] button widgets: Name of the directory where the FITS files are to be "written to" or "read from" and a button to *accept* the entered directory name.
- "Untitled" operational text widget: Initially contains a *SOHO/CDS* logo. Later it will contain information concerning the current operating status of the code and filenames to be written to or read from the tape.
- File Naming Convention toggle–button widget: There are 4 different mechanisms for naming FITS files read off of an archive tape. This widget allows you to select such a naming convention:
  - [◇ Default]: Let XWINTAPE decide on the form of the name for the FITS files read from the tape (see below).
  - [◇ SFDU_small]: Follow the CSSDS1 specification, where the filename can be no bigger than 8 characters (first character must be alphabetic) with an extension no bigger than 3 characters (*i.e.*, FIT). This is the default used under the DOS operating system.
  - [◇ SFDU_large]: Follow the CSSDS2 specification, where the filename can range anywhere between 1 and 30 characters including any extensions. Note that MAGTAPE has been designed to keep the total filename length at 30 characters or less including the mandatory ".FITS" extension. This is the default used under the Unix, VMS, and MacOS operating systems.
  - [◇ KEYWORD]: Use a string associated with a FITS keyword. When using this option from XWINTAPE, the keyword used is "FILENAME." Note that this option would probably never be used since the filename stored in the FILENAME keyword should be the same as that associated with the SFDU_large protocol.
  - [◇ Tape Location]: Use the name "tapennnn.fits" for the FITS file read from the tape, where "nnnn" is the tape number of the file (*i.e.*, the first FITS on the tape would be stored as "tape0001.fits").

- Access Override: IDL has "built-in" functions to access a tape drive. These IDL tape commands are only valid in a VMS environment. However, the SOHO/CDS group has written procedures that emulate these commands in a Unix environment, which have the same names as their VMS counterparts and pass the same parameters. However, the Unix versions of these functions will not work for versions of IDL earlier than 3.1 running in a Unix environment. When such a condition is met, MAGTAPE will automatically switch to the "Unix mt & dd Commands" by default. Also, the SOHO/CDS staff have noted that the IDL style commands under *DEC OSF/1 Unix* sometimes will not work — the code either "freezes" or "crashes." This is due to an IDL bug in the */NOSTDIO* keyword of the *OPENU* procedure.
  - ◇ IDL Style Commands: Use the IDL tape access commands. This is the default, except for versions of IDL earlier than 3.1 running in a Unix environment.
  - ◇ Unix mt & dd Commands: Use *mt* and *dd* to manipulate the tape drive in a Unix environment.
- Valid CDS FITS?: It is the default for MAGTAPE to check the first FITS file it encounters on a disk during a "write" for validity as a CDS level–1, binary extension FITS file. Checks are made to insure that the FITS keywords in the file are valid in that they were registered with the NSSDC. Further checks are made to the array sizes and data types for all columns of data in the binary extensions of the FITS file.
  - ◇ Check FITS Validity: See if the file is a valid CDS level–1, binary extension FITS file. This is the default.
  - ◇ Ignore FITS Check: Do not make the check before writing the tape or CDROM.
- "Untitled" selected files text widget: Initially contains a text "Select files from upper window ..." When files are selected from the upper text widget box, they are displayed in this box.

Meanwhile, the right–hand–side (RHS) of the main widget contains a variety of button and toggle–button widgets:

- << Begin >>: Click this button only after all of the options have been set. This is a **dangerous** button since it instructs XWINTAPE to access the tape drive through the procedure MAGTAPE. Due to its importance, clicking this button activates another widget that checks to be sure that you really want to access the tape at this time.
- Tape Access: Click either ◇ Read or ◇ Write, whichever is appropriate for the desired action. Note that this is a toggle switch — both cannot be depressed at the same time.
- Tape Drive: This is a "pull–down" menu listing MT0, MT1, ..., MT9. Select one of the 10 entries based upon the environment value you set for the tape drive name before entering IDL. The default is MT0.
- Blocking Factor: This is a "pull–down" menu listing the tape blocking factor: 1*2880 bytes/record, 2*2880 bytes/record, ..., 10*2880 bytes/record. The default is 1*2880 bytes/record for writing a tape and is determined from the tape itself for a read (except for the Unix mt & dd Commands access type, where it must be entered manually).
- Session Log: Make a log file of the current tape archiving session. Default is ◇ YES, make a log file.

- Tape Header: Will (or does) the tape contain a SFDU header file (*i.e.*, first file on the tape)? The default is $\boxed{\diamond \text{ YES}}$ it does. This switch allows you to read or write standard FITS tapes (*i.e.*, those without an SFDU header).
- Save SFDU File?: Do you wish to save the SFDU in a disk ASCII file? Default is $\boxed{\diamond \text{ NO}}$. Note that the SFDU can also be accessed by passing a variable in the call to XWINTAPE (*e.g.*, XWINTAPE, SFDU, where SFDU contains the SFDU header).
- $\boxed{\text{FITS Header}}$: Initially this widget button is inactive, but once the disk, where the FITS files are located, is selected (for type Write) or the SFDU header is obtained from the tape (for type Read), the button becomes active. It then can be pressed, which activates another widget containing the names of all the FITS files in the top *unnamed* text box. The FITS header for each file can be viewed by selecting the appropriate file. This may be useful when only certain types of files are desired.
- $\boxed{\text{Help}}$: Clicking this widget button starts another widget that contains a detailed description of the information displayed in the Message: widget box.
- "Untitled" picture widget: Contains the SOHO logo and remains inactive for the entire session.
- $\boxed{\text{!! Abort !!}}$: Press this button if you wish to stop XWINTAPE without accessing the tape.
- $\boxed{\text{\#\# Exit \#\#}}$: This widget button remains inactive until MAGTAPE has successfully returned to XWINTAPE. Click this button to leave the XWINTAPE widget session.

7 HELP! — Archiving Problems that may Arise

The use of SFDUs on the archive tapes requires that an archiving run cannot have multiple volume tapes or CDROMs (*i.e.*, if you fill up one tape, a second tape <u>cannot</u> be mounted for continuing the archive). That is, **each archive tape must contain an SFDU header file at the beginning of the tape.** But a question arises, How many FITS files, including the SFDU file, can be stored to a tape before filling it up? This is not an easy question to answer, especially on Unix machines, where their are a multitude of available tape drives, each with their own density. Exabyte tape drives come in a variety of densities, the most popular being 2.4 Gbyte, 5 Gbyte, and 7 Gbyte, with the 7 Gbyte capacity being more and more popular.

Although the procedure that creates the SFDU calculates the total number of bytes that will be loaded to the tape, their is currently no check to warn the user that a tape will be overfilled upon completion of the write. Once the archiving is in operation and archiving tape drives are defined, such a check will be installed in MAGTAPE in the future. Until that time, always be knowledgeable of the total bytes being sent to the tape.

The CDS staff has noted that the Unix IDL procedures that simulate the VMS IDL tape accessing commands (*i.e.*, TAPRD, TAPWRT, REWIND, SKIPF, WEOF), do not always work on Digital machines running OSF/1. MAGTAPE either "crashes" or "freezes" during such instances. This is due to a "bug" in the /NOSTDIO keyword in the OPENU command that is used to access the tape drive. Note that this does not seem to be a problem on Sun machines. Should you be operating in an OSF/1 or DEC–UNIX environment and you experience problems with the IDL–style tape accessing commands, exit the current IDL session, restart IDL, and rerun XWINTAPE or MAGTAPE. This time, however, use the Unit mt & dd Commands protocol to read or write the tape. Hopefully, future versions of IDL will fix this bug.

Acknowledgements. Portions of the introduction were written by Dr. William Thompson in his document *A Proposal for Storing FITS Files on Disk and Tape Archive and Distribution Media Using SFDUs.* I thank him for his permission for the use of this text. This work was supported by the NASA contract NAS 5–32350 to the Applied Research Corporation.

Appendix A: Procedures Needed for Tape Archiving

The table below summarizes all of the procedures and supplementary data files need for a tape archiving session. Run the IDL procedure DOC_LIBRARY to gain future information of each IDL procedure. If a procedure is identified as being called by XWINTAPE, then it is used only for the widget control program xwintape.pro.

magtape.pro	The main tape driver procedure (see §5).
sfdu_2fhd.pro	Extract the FITS header from the SFDU. Called by XWINTAPE.
sfdu_bytes.pro	Set various byte size parameters used for SFDU tape I/O. Called by MAGTAPE.
sfdu_fitshd.pro	Retrieve all primary FITS headers from disk files. Called by XWINTAPE.
sfdu_fsize.pro	Ascertain the total bytes in the FITS files for tape I/O. Called by MAGTAPE.
sfdu_getreg.dat	Data file containing the location of the SFDU registration files. Called by SFDU_GETREG.
sfdu_getreg.pro	Determine the name and location of the SFDU registration files. Called by MAGTAPE.
sfdu_help.pro	Return a detailed description of tape archiving. Called by XWINTAPE.
sfdu_make.pro	Generate an SFDU header from the passed filenames. Called by MAGTAPE.
sfdu_os.pro	Load the operating system information into variables. Called by both MAGTAPE and XWINTAPE.
sfdu_qwidg.pro	Make XWINTAPE widgets sensitive or insensitive.
sfdu_read.pro	Read the SFDU file off of the tape and return the filenames found. Called by both MAGTAPE and XWINTAPE.
sfdu_volhd.pro	Generate the Volume Description text for a SFDU header file. Called by MAGTAPE.
sfdu_write.pro	Write the SFDU file onto the tape. Called by MAGTAPE.
soho_cds.logo	ASCII text file containing the SOHO/CDS logo used in the "untitled" upper text widget. Used by XWINTAPE.
soho_lglogo.gif	GIF file containing the official SOHO logo and used in the "untitled" draw widget for DEC machines. Used by XWINTAPE.
soho_smlogo.gif	GIF file containing the official SOHO logo and used in the "untitled" draw widget for Sun machines. Used by XWINTAPE.
unixdd.pro	FITS tape I/O using the Unix *dd* and *mt* commands. Called by MAGTAPE.
xwintape.pro	Widget procedure that drives MAGTAPE (see §6).

Beside these specific archiving procedures, MAGTAPE also makes use of the standard FITS I/O routines. These routines are located in the util/fits and util/util directories and are summarized in the following table.

num2let.pro	Returns a letter ('a', 'b') from an integer (1, 2).
fxtaperead.pro	Copy FITS files from tape to disk.
fxtpio_read.pro	Copy FITS files from tape to disk — internal routine for FXTAPEREAD.
fxtapewrite.pro	Copy FITS files from disk to tape.
fxtpio_write.pro	Copy FITS files from disk to tape — internal routine for FXTAPEWRITE.