CSC/SD-92/6028
560-7SUG/0290

International Solar-Terrestrial Physics (ISTP) Central Data Handling Facility (CDHF) Software System (ICSS) Programmer's Guide to Key Parameter Generation Software

Revision 7

Prepared for

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland

Under

Contract NAS5-31000
Subcontract HQ-001057
SLA 7 – ISTP

November 1996

CSC/SD-92/6028
560-7SUG/0290

International Solar-Terrestrial Physics (ISTP) Central Data Handling Facility (CDHF) Software System (ICSS) Programmer's Guide to Key Parameter Generation Software (KPGS)

Revision 7

Prepared for

GODDARD SPACE FLIGHT CENTER

by

COMPUTER SCIENCES CORPORATION

November 1996

Revised by: Computer Sciences Corporation
Contract NAS5-31000
Subcontract HQ-001057
SLA 7 – ISTP

Reviewed by:

G. Blackwell
KPGS Integration Test Team Lead, CSC

Quality Assured by:

S. Whisonant
ISTP CDHF Product Assurance Officer, CSC

Approved by:

S. Whitesell
Manager, Data Processing Systems, CSC

H. Williams
Mission Director, Mission Management Office
Code 510.2

TABLE OF CONTENTS

Section 1—Introduction

Section 2—Key Parameter Processing on the ISTP CDHF

Section 3—Coding Standards

Section 4—Support Routine Descriptions

Section 5—Near-Real-Time Key Parameter Generation

Section 6—Special Programming Notes

Appendix A—Sample Test Programs

Appendix B—Sample CDF Skeleton Tables

Appendix C—Error Messages

Appendix D—KPGS Delivery Form

Abbreviations and Acronyms

References

SECTION 1—INTRODUCTION

1.1 Purpose and Scope

This document describes the International Solar-Terrestrial Physics (ISTP) Central Data Handling Facility (CDHF) Software System (ICSS)-supplied support routines for use by the principal investigator (PI) key parameter generation software (KPGS) and the standards and procedures to be used in coding and testing the KPGS. Each support routine shows an explanation of the user-supplied input, the routine function, the routine output, and error conditions.

1.2 International Solar-Terrestrial Physics Key Parameter Generation Overview

One of the primary functions of the ISTP CDHF is to process ISTP level-zero and Geomagnetic Tail (GEOTAIL) Laboratory Scientific Information Retrieval Integrated Utilization System (SIRIUS) instrument datasets into key parameter datasets. Before launch, the ISTP instrument investigators deliver the software necessary to produce these key parameters from the level-zero or SIRIUS data. This software is integrated with the ICSS-supplied support routines and, after acceptance testing on the CDHF, will run as production jobs within the CDHF production environment. All production jobs are under configuration control.
The main purpose of the ICSS support routines is to provide an interface between the CDHF data management system and the PI-supplied KPGS. The ICSS support routines also ensure that the key parameters are generated in the CDHF standard format and that the proper accounting information is maintained in the CDHF catalog. These support routines serve as the interface between the KPGS and the CDHF system-level functions to be used for opening, reading, closing, or providing direct access to CDHF-cataloged files. The support routines perform all file access functions for CDHF online data files, as well as provide the PI with coordinate conversion routines.

1.3 Document Organization

This document is organized into six sections. Section 1 presents the document's purpose and scope, a general overview of ISTP key parameter generation, and document organization. Section 2 discusses the processing schema input and output required for key parameter generation on the ISTP CDHF and provides an overview of the development and test environment on the CDHF and guidelines for reprocessing key parameters. Section 3 addresses the coding standards the PI's KPGS will use. Section 4 details the support routines, including descriptions for each routine's interfaces, calling sequence, and error conditions. Section 5 describes near-real-time (NRT) data receipt and processing and NRT support routines. Section 6 provides special programming notes for the Interplanetary Physics Laboratory (WIND), Polar Plasma Laboratory (POLAR), and future missions that will be supported by the CDHF.
Appendix A contains information about nine sample KPGS programs. It provides the online location of these sample programs in the SYS$PUBLIC directory of the CDHF for easy access. The sample programs consist of FORTRAN and C versions that illustrate the way users should access the support routines within their program structures. These sample KPGS programs include code that illustrates KPGS programs for NRT key parameter generation, as well as special coding techniques unique to the Solar and Heliospheric Observatory (SOHO) mission. Appendix B provides information and the online location for sample common data format (CDF) skeleton tables used in the generation of key parameter output file(s). Appendix C contains error messages and definitions. Appendix D contains the online location of a sample delivery form used in the delivery of KPGS program and data files.

SECTION 2—KEY PARAMETER PROCESSING ON THE ISTP CDHF

2.1 Processing Schema

Key parameter processing, using playback data, generally takes place daily on the ISTP CDHF. For each calendar date, a key parameter dataset is generated for each instrument of each mission. Each key parameter dataset generation is viewed as an independent process; that is, previously generated key parameter datasets are not used as input for processing subsequent key parameter datasets. Therefore, key parameters may be processed in any order as long as the necessary input datasets are available (i.e., level-zero, housekeeping, orbit, attitude, spin-phase, and calibration). The ISTP CDHF will support the processing of data over multiple days for the generation of key parameters.

In special cases, key parameters can be generated on the CDHF in NRT using the real-time data acquired during scheduled Deep Space Network (DSN) passes.

The following sections describe the types of input and output required for key parameter generation on the ISTP CDHF using playback data. Key parameter generation using NRT data is described in Section 5.

2.2 Key Parameter Processing Input

The three sections that follow describe the various input to a key parameter processing program executing on the ISTP CDHF. Because normal production of key parameter processing on the CDHF is batch oriented, no interactive input is allowed.

2.2.1 Cataloged Input Files

All input files to a key parameter processing program must have entries in the ISTP CDHF catalog. Cataloged files are read-only files identified using attributes such as mission, instrument, data type, and calendar date. Each key parameter processing program produces a key parameter dataset from the input telemetry data. Normally, the timespan of the generated key parameter file corresponds to the timespan of the primary telemetry input file. Cataloged files available as input for key parameter generation include level-zero or SIRIUS instrument, housekeeping, orbit, attitude, calibration, and instrument parameter data. Orbit and attitude files are selected based on the best available files at the time of key parameter processing. In most cases, predictive files are used, unless definitive files are available. PIs may not use existing key parameter files as input for creating a new key parameter file. Each file in the catalog is associated with a calendar date.

2.2.2 Calibration Parameter Files

PIs may also require a set of files containing calibration parameters specific to their key parameter processing program. These files are delivered with the PI's KPGS and placed under configuration control, along with the corresponding software. The PI's software is responsible for providing access routines to the calibration parameters; that is, no support routines are provided to open or read from these calibration files. There are no project-specified format requirements for calibration parameter files. The ICSS_GET_CD support routine provides the names of the appropriate calibration files to the KPGS. Updates to the calibration files are handled in the same manner as updates to the KPGS. All calibration parameter files delivered to the CDHF are tracked by the CDHF catalog and placed under configuration control. Each instrument is allowed multiple calibration files, each of which is designated by a unique instrument component identifier, and multiple versions of each calibration file can be maintained online at the CDHF. The PI specifies the time range for which the calibration file is valid when the file is delivered to the CDHF; the PI also specifies the three- or four-character instrument component identifier to be used with the calibration file. Appendix B of the ISTP KPGS standards and conventions document (Reference 1) contains a dictionary defining valid instrument component identifiers. The CDHF uses this information to select the correct versions of the calibration files when key parameter generation is scheduled on the system.

2.2.3 PI Parameter File

PIs may also provide a PI parameter file as input to their KPGS. This file contains whatever controls the PI desires. For example, the parameter file may contain flags to initiate debugging within the KPGS. This file, like the calibration parameter files, is delivered with the PI's KPGS. The files are placed under configuration control and updated throughout the project on request. Each instrument is allowed access to a single parameter file. The ICSS_GET_PF support routine provides the name of the appropriate parameter file to the KPGS. The PI's software is responsible for providing access routines to the parameter file. As with calibration files, no project-specified requirements apply to PI input parameter files.

2.3 Key Parameter Processing Output

The four sections that follow describe the different categories of output from a key parameter processing program executing on the ISTP CDHF.

2.3.1 Cataloged Output (Key Parameter) Files

As described in the ISTP KPGS standards and conventions document, the key parameter processing program creates a key parameter file. The standard format for ISTP key parameter files is the CDF. The structure of the CDF key parameter file for each instrument reflects the parameters that are written to the file. The CDF key parameter file is defined by using a CDF skeleton table, which creates a properly formatted but empty CDF key parameter file for the instrument prior to key parameter generation. The KPGS writes key parameters to this existing file. The KPGS uses the CDF routines described in Section 4 to write data to the key parameter file. After key parameter processing completes, the ICSS_KPG_TERM support routine completes the standard formatted data unit (SFDU) header, closes all key parameter input and output files, and catalogs the new key parameter file(s). After a file is cataloged, it becomes read-only. The key parameter output files cannot be modified by any subsequent programs.

2.3.2 Scratch/Intermediate Files

The key parameter processing program may use scratch files during program execution. All scratch files are to be created in a scratch area defined by the logical name KP_SCRATCH (e.g., KP_SCRATCH:scratch_file.dat). The size and number of scratch files are established by each PI.

The creation and use of scratch files are under the control of the PI-supplied KPGS. These files are deleted from the scratch area on program termination. Appendix A contains sample code fragments illustrating the use of scratch files. A scratch file should be used for runtime processing only and opened with the delete option (i.e., STATUS='SCRATCH') to ensure that the file is deleted when the job ends.

2.3.3 User Message File

The user message file is available to KPGS applications for the logging of user-significant events, conditions, or data. This file is user-specific, created strictly for the use and convenience of the PI/user, and resides in an area accessible to the user. The logical name, USR_MSG_FILE, is assigned to the KPGS before its execution. The value for this logical filename is obtained from the database. It is set from the Static Mission Information Menu of the database interface system by the database administrator (DBA), as described in the CDHF user's guide (Reference 2). A user wishing to write data to his or her user message file must open the file and write, using the provided logical filename. The format and use of this file are under the control of the KPGS. Users can request their files be sent to them automatically after key parameter generation (using a flag set through the Interactive Interface); they can also use the User Data Services to select the file for transfer. Files can be sent automatically using either the mail facility or the copy command. Files that are sent automatically using the mail facility are sent to users specified on a distribution list. The distribution list is established by operations personnel on request from the PI. The copy command option for automatically receiving user message files is also set up by operations personnel. Files that are sent automatically using the copy command end up in a directory specified by the PI. PIs should specify the directory in their own area for which they expect the user message file to be copied (e.g., ISTP_USER:[GE_CPI.DUFFY.MSG]). If using the copy option to automatically receive the user message file, the PI should first execute the command file invoked by the symbol SET_UMSG. The system periodically deletes the old user message files from the CDHF user message file directory. The PI user message directory name should be included as part of the documentation that accompanies the KPGS delivery. Sample code fragments illustrating the creation and use of the user message file are available online (see Appendix A).

2.3.4 Error/Status Messages

PIs are responsible for maintaining processing control within their KPGS application. After every call to an ICSS support routine, it is the PI's responsibility to check the return status code. All ICSS support routines return the value SS$_NORMAL when they are successful. In some cases, the support routines will return a warning status that is not necessarily fatal (e.g., "end of file reached"). The end of file is such a warning. Section 3.3.7 contains a description of the error-handling procedures. The ICSS support routines record the error messages in the ISTP CDHF system message log. For a complete listing of error/status messages, refer to Appendix C. Appendix A contains sample programs that illustrate the proper handling of status codes.

2.4 CDHF Development Environment

The KPGS Integration Test Team (KITT) provides the PI teams with a development environment on the CDHF for the purpose of developing and testing the KPGS. Although the PI teams may develop the KPGS on their remote data analysis facilities (RDAFs), it is recommended that the KPGS be transferred to the CDHF and tested in the CDHF development environment before it is delivered. The development environment provides the PI teams with the computer resources, test data, and supporting software required to test the KPGS adequately before it is delivered to the KITT for integration testing. The development environment also provides a mechanism to deliver the KPGS to the CDHF electronically. This section describes the development environment, its use, and the mechanism by which the KPGS is delivered to the CDHF.

2.4.1 KPGS Development and Test

This section explains the way to apply for a development account and the way to use the resources of the development environment.

2.4.1.1 Getting an Account

To use the CDHF development and test environment, a user must first have an account on the CDHF system. If the user does not have an account, he or she can apply for one electronically. The user logs onto the system by doing one of the following:

For systems using the Digital Equipment Corporation's network (DECnet):

SET HOST ISTP1

or

SET HOST 15461

For systems using Transmission Control Protocol/Internet Protocol (TCP/IP):

telnet istp1.gsfc.nasa.gov

or

telnet 128.183.92.58

When the Username: prompt appears, the user should enter APPLY and press the carriage return. No password is required. The user will be logged into a captive account that prompts for information needed to set up a computer account on the CDHF. A day or two after making the application (and after the application has been approved), the user will receive confirmation that an account has been established.

2.4.1.2 Getting HELP

The CDHF contains an extensive set of online help files that describe operating system commands and utilities, software packages, and CDHF site-specific information. To get general help information on the operating system, the user types

HELP

To get CDHF site-specific information (e.g., available software packages, network information, operations information), the user types

HELP @SITE

Help is also available online to provide KPGS development information. Included in the KPGS help is information on the ICSS support routines, KPGS delivery, CDF, development environment, and KPGS build procedures. To get KPGS help information, the user types

HELP @KPGS

2.4.1.3 Developing and Testing KPGS

The CDHF development environment provides the resources for the PI team to develop and test the KPGS. To test the KPGS in this environment, the user must take the following steps:

1. Establish the test environment under the PI's account on the CDHF.
2. Compile the KPGS on the CDHF.
3. Link the KPGS with ICSS support routines and libraries.
4. Place test data supplied by the KPGS development team and the KITT in the appropriate directories, and have the KITT insert references to these files in the test database.
5. Create a file containing parameters for input to the KPGS.
6. Create the CDF skeleton table(s).
7. Create a command file to start the required mailboxes, define the appropriate logical names, create the output key parameter file, and run the KPGS program.
8. Execute the command file.

2.4.1.3.1 Establishing the Test Environment on the CDHF

Once the user has obtained an account in the KPGS development and test environment, he/she must contact the KITT and request to be made a valid user of the CDHF database. Requests can be submitted by telephone at (301) 794-2318 or by E-mail address at ISTP1::GBLACKWELL.

Before using the CDHF account for development and testing, the following command should be executed immediately after logging on for each session:

KPGS_DEV

This command defines the logical names and symbols required to set up the KPGS environment, to compile, link, and run the KPGS programs, as well as to perform other development and test activities. It also provides access to the KPGSIT version of the CDHF database.
The user should be in his/her KPGS_root directory before invoking this command. This command must be entered each time the user logs into the CDHF.
Note: If a user logs into the CDHF and establishes the development environment and then wishes to run the user interface to access the operational environment, he or she must log off and log in to the system again or start a second session. Any attempt to use the user interface after establishing the development and test environment via KPGS_DEV will result in conflicts of symbols and logical names between the test environment and the production environment.
Special database tables have been set up for use by the KPGS development team for testing their software on the CDHF. To access the test database tables, the user must execute the following command:

```
@KPGS_IT:[KPGSIT.SYSTEM]SET_USER_SYNONYMS
```

This command need only be executed once and should not be repeated on subsequent logins. The user is notified if a database update requires this command to be reexecuted.
To establish the development and test environment, the KITT requires that the user employ the same directory structure used in integrating his or her KPGS. This directory structure is created by typing the following command:

```
KPGS_DIR
```

This command sets up the required directory structure for the user and creates an object library. The library will be populated when the user runs the compile command file described in the following section. This command file needs to be run only once unless the user decides to delete the directory structure and re-create it.
The procedure creates the directory and the required subdirectories. The subdirectories created and the files to be placed in them are described as follows:

[<root directory>.SRC] = KPGS source files

[<root directory>.DOC] = KPGS design document and other documentation

[<root directory>.EXE] = Working executable

[<root directory>.TSTDATA] = Test input data, skeleton tables, calibration files, and parameter files

[<root directory>.COM] = Command files for compiling, linking, and running the KPGS

[<root directory>.kpgs_root.OBJ] = KPGS object files

where <root directory> is the directory from which the user wishes to set up the KPGS program.
The following sections provide information about the command files, namelist files, and other files required to establish the KPGS development and test environment. The user should contact the KITT for automatic tools that can be used to generate these files.

2.4.1.3.2 Compiling the KPGS

The user compiles the FORTRAN, C, and assembler source codes, moves the object codes into the object library created when KPGS_DIR is run, and deletes the object codes using the following command file:

```
$ SET DEF SRC_DIR
$ FOR_COMPILE :==FOR
$ C_COMPILE :==CC
$ MACRO_COMPILE :==MACRO
$ 'FOR_COMPILE' <KPGS source module name>
$ 'C_COMPILE' <KPGS source module name>
$ 'MACRO_COMPILE' <KPGS source module name>
$ LIB/REPLACE OBJ_DIR:OBJ_LIB.OLB *.OBJ
$ DELETE *.OBJ;*
$ SET DEF COM_DIR
```

If recompiles are done using this command file, the user should delete or rename the existing object library and create a new object library using the following command:

```
LIB/CREATE OBJ_DIR:OBJ_LIB.OLB
```

A sample copy of the compile command file can be found in the following directory:

KPGS_SUPPORT:SAMPLE_COMPILE_FILE.COM.

2.4.1.3.3 Linking With ICSS Support Routines, CDF Software, and Mathematics Libraries

The KPGS applications may be linked by using Virtual Address Extension (VAX) libraries for the object code or by simply using object modules. To link a FORTRAN application library with the ICSS support routines, the user executes the following command file:

```
LNPROFOR EXE_DIR:<my_kpgs> -
OBJ_DIR:<my_lib>/LIB/INCLUDE=<my_obj>,-
ICSS_OBJ:SR/LIB,-
CDF$LIB:LIBCDF/LIB,-
NAGD$DIR:NAG$LIBRARY/LIB,-
ICSS_OBJ:DMS/LIB,-
ICSS_OBJ:UTL/LIB,-
MSG_PTR T
```

where

<my_kpgs> is the name of the executable to be produced by the link.

<my_lib> is the name of the KPGS object library (OBJ_DIR:OBJ_LIB).

<my_obj> is the name of the top-level application module.

The link command file should be entered exactly as shown above; it is important that no extra spaces be added.

The name given to the executable image in the link command file (<my_kpgs>) should use the following naming convention:

<mission long name>_<instrument short name>_KP_<program version>
(e.g., GEOTAIL_MFG_KP_V2_5)

If a library is not used for the KPGS applications, then the user replaces the second line of the procedure with the object module name(s).

The Numerical Algorithms Group (NAG) Fortran Library is a comprehensive collection of Fortran 77 routines for the solution of numerical and statistical problems. The word “routine” is used to denote subroutine or function. The library is divided into chapters, each devoted to a branch of numerical analysis or statistics.

The last line of the procedure (MSG_PTR) may be followed by a “d” (separated from “MSG_PTR” by a space) to include debug code in the executable and/or by an “m” to produce a link map.

To link applications written in C, the user replaces “LNPROFOR” in the first line of the procedure with “LNPROC”.

To link applications with the NRT/ICSS support routines in the NRT environment, the user executes the following command file:

```
LINK/EXE=EXE_DIR:<my_kpgs> -
OBJ_DIR:<my_lib>/LIB/INCLUDE=<my_obj>,-
IMSL$DIR:SFUN.OLB/LIB,-  ! optional, include this library as needed
ICSS_OBJ:NRT_SR/LIB,-
ICSS_OBJ:SR/LIB,-
CDF$LIB:LIBCDF/LIB,-
NAGD$DIR:NAG$LIBRARY/LIB,-
ICSS_OBJ:DMS/LIB,-
ICSS_OBJ:UTL/LIB,-
MSG_PTR
```

where

<my_kpgs>, <my_lib>, and <my_obj> are as described above.

The LINK command in the first line of the procedure may be followed by /DEBUG to include debug code in the executable. Samples of both link command files can be found in KPGS_SUPPORT:SAMPLE_LINK_FILE.COM and KPGS_SUPPORT:SAMPLE_NRT_LINK_FILE.COM.

The International Mathematics and Statistics Library (IMSL) SFUN/LIBRARY is a collection of Fortran subroutines and functions useful in research and statistical analysis. Each routine is designed and documented to be used in research activities, as well as by technical specialists.

2.4.1.3.4 Supplying the KPGS Test Data Files

To test the KPGS, the appropriate data files must be placed in the proper directories and cataloged in the test database. These data files may be any of the following:

- Simulated Instrument Data Set (SIDS) data file [These PI-supplied files are converted to level-zero data files by the KITT; the level-zero data files are then cataloged by the KITT. See Appendix B of the ISTP CDHF KPGS Integration Test Plan (Reference 3)].
- SIRIUS data files (GEOTAIL only)
- Calibration parameter files
- PI parameter files

The PI teams are responsible for placing the required input files in the proper directory for access by the KITT. The KITT is responsible for cataloging these files after they are supplied and providing the level-zero (generated from SIDS data), housekeeping, attitude, and orbit files required for testing.
To place all input files in the proper directory, the PI teams copy these files to the logical directory "TST_DIR." After all the files have been provided, the KITT should be notified by electronic mail. The KITT specifies the time interval that will be used for testing.
After the KITT receives the notification, it inserts references to the input files in the database and provides the additional required data files. When this is done, the PI team is notified by E-mail.

2.4.1.3.5 Creating the Input Parameter File

To test the key parameter generation program, a file containing parameters for input to the KPGS must be created. This file identifies the input files and output files required by the key parameter generation program and is distinct from the PI parameter file described in Section 2.2.3. A template of the file containing parameters for input to the KPGS follows. A sample of this file can be found in KPGS_SUPPORT:SAMPLE_PARAMETER_FILE.DAT.

MISSION_NAME = "name of mission"
INSTRUMENT_NAME = "short name of instrument"
TL_INPUT_FILE_P = "primary level-zero or SIRIUS input file"
TL_INPUT_FILE_S = "secondary level-zero or SIRIUS input file"
HK_INPUT_FILE_P = "primary housekeeping input file"
HK_INPUT_FILE_S = "secondary housekeeping input file"
ORB_INPUT_FILE_P = "primary orbit input file"
ORB_INPUT_FILE_S = "secondary orbit input file"
ATT_INPUT_FILE_P = "primary attitude input file"
ATT_INPUT_FILE_S = "secondary attitude input file"
KPGS_PROGRAM_NAME = "KPGS program name"
KPGS_PROG_VER = "KPGS program version"
KP_OUTPUT_FILE_01 = "KP output filename"

This file has a fixed format. Parameters must appear in the order shown and all must be present. Parameters not applicable to the program are given NULL values (e.g., TL_INPUT_FILE_S = "" if no secondary level-zero file is required). A sample file containing parameters for input to the KPGS follows:

MISSION_NAME = "GEOTAIL"
INSTRUMENT_NAME = "PWI"
TL_INPUT_FILE_P = "GEOTAIL_LZ:GE_LZ_PWI_19920731_V01.SFDU"
TL_INPUT_FILE_S = ""
HK_INPUT_FILE_P = ""
HK_INPUT_FILE_S = ""
ORB_INPUT_FILE_P = "GEOTAIL_ORB:GE_OR_PRE_19920731_V01.SFDU"
ORB_INPUT_FILE_S = ""
ATT_INPUT_FILE_P = ""
ATT_INPUT_FILE_S = ""

```
KPGS_PROGRAM_NAME = “GEOTAIL_PWI_KP”
KPGS_PROG_VER = “V1.0”
KP_OUTPUT_FILE_01 = “GEOTAIL_KP:GE_K0_PWI_19910930_V01”
```

The values for the KPGS_PROGRAM_NAME and KPGS_PROG_VER parameters must be the same as the first two arguments passed to the ICSS_KPG_INIT support routine by the program (see Section 4.1). Logical names are used for the directory specification and must be included as shown.

2.4.1.3.6 Creating the CDF Skeleton Table

By referring to the sample CDF skeleton tables in Appendix B and the CDF user’s guide (Reference 4), the user may create a skeleton table that can be used to generate the instrument key parameter output file. Additional help for creating the skeleton table is available from Ramona Kessel at user ID NCF::KESSEL and the KITT at ISTP1:KPGSIT. Appendix A of Reference 4 provides a tutorial on creating the required skeleton table.

2.4.1.3.7 Creating the KPGS Run Command File

The KPGS run command file is used for setting up and running the KPGS. The following is the format of this file:

```
$define ICSS_INP_PARM <input parameter filename>
$@CDHF_DEV:[sdev.com]sim_def      ! Start simulated mailboxes
$@CDHF_DEV:[sdev.com]sim_ops
$@CDHF_DEV:[sdev.com]sim_mh
$@CDHF_DEV:[sdev.com]sim_cat
$inquire/nopun creat_skel “Do you want to create CDFSKELETON? (Y/N):”
$if creat_skel .eqs. “Y” .or. creat_skel .eqs. “y”
$       then
$               ! Delete the old .cdf file if present
$               if f$search (<KP output filename>) .nes. “”
$                       then
$                               delete <KP output filename>;*
$                       endif
$               ! Create the new cdf
$               cdfskeleton <skeleton filename> /CDFNAME=<KP output filename>
$       endif
$RUN KP_TOOLS:LOAD_QUEUE
        <KPGS program name>
        <KPGS program version>
        <Level-zero logical filename>
        <KP output filename>
$RUN <KPGS executable name>
$STOP <account>_1               !Stop simulated mailboxes
$STOP <account>_2
$STOP <account>_3
```

where

<input parameter filename> is the name (with path) of the input parameter file created in Section 2.4.1.3.5.

<KP output filename> is the name of the key parameter file that will be created by the run. This should be the same as the KP_OUTPUT_FILE_01 file specification in the input parameter file (see Section 2.4.1.3.5).

<skeleton filename> is the name of the skeleton table created in Section 2.4.1.3.6.

<KPGS program name> is the name of the KPGS program. This should be the same as the KPGS_PROGRAM_NAME parameter in the input parameter file (see Section 2.4.1.3.5).

<KPGS executable name> is the name of the KPGS executable created in Section 2.4.1.3.3.

<KPGS program version> is the version of the KPGS program. This should be the same as the KPGS_PROG_VER parameter in the input parameter file (see Section 2.4.1.3.5).

<Level-zero logical filename> is the name of the level-zero file being used as input for the run.

<account> is the user name of the account under which the KPGS test will be run.

STOP assumes no other spawn processes are running under <account>.

Invocation of the command file executes the KPGS program using the indicated input/output files.

LOAD_QUEUE is a program that initializes the ICSS ORACLE database for a KPGS program test run. It places entries in the data process queue, and removes entries of previous runs from the data process history and data transfer queue tables.

The LOAD_QUEUE program requires four input items, each on a separate line. The proper invocation of this program is as follows:

```
RUN KP_TOOLS:LOAD_QUEUE
<KPGS program name>
<KPGS program version>
<Level-zero logical filename>
<KP output filename>
```

<KPGS program name> and <KPGS program version> are the name and version of the KPGS program as defined in the input parameter file. <Level-zero logical filename> is the logical name (no path or extension) of the primary level-zero file that will be used as input for the run.

The following example is an invocation of the command that would initialize the data process queue for a test run of version 1.0 of the GEOTAIL Energy Particle and Ion Composition (EPIC) KPGS program using the level-zero file for September 21, 1992:

```
LOAD_QUEUE
GEOTAIL_EPI_KP
V1.0
GE_LZ_EPI_19920921_V01
GE_KP_EPI_19920921_V01
```

If this program is invoked with other than four parameters, an error message is returned and a help screen describing the correct invocation is displayed.

2.4.1.3.8 CREATE_KPGS_TEST Utility

The CREATE_KPGS_TEST utility creates the files necessary to run a playback mode KPGS program. This utility uses the CDHF database as the source for the information needed to create these files. It ensures that all of the required data files exist and are cataloged, and that all of the required database entries for the KPGS program have been made. If any discrepancies are noted during the running of this utility, they will be reported to the user and the test environment will not be created.

This utility will create a command file to run the KPGS program and an input parameter file to pass the names of input and output files to the KPGS program. The command file will be named

<Program Name>_RUN_YYMMDD.COM

where

<Program Name> is the name of the KPGS program, and YYMMDD is the date of the data that is used for the test.

The input parameter file is named

<Program Name>_YYMMDD.DAT

Both of these files are created in the default directory. Note that the command file can be renamed, but that the input parameter file, which is referenced by the command file, cannot be changed without also changing the reference in the command file.

Running the KPGS program will create a user message file with the name

<Program Name>_YYMMDD.LOG

This file can be changed by editing the command file.

The CREATE_KPGS_TEST utility is designed to be used by both the KITT and the PIs. For this reason, it is sensitive to the account in which it runs. The KITT usually runs KPGS programs with the ICSS system, while

the PIs use simulated mailboxes. If the CREATE_KPGS_ TEST utility is run in the KPGSIT account, it will create a command file that does not use the simulated mailboxes for the message handler, cataloger, and operator interface. If this utility is run in any other account, commands will be inserted into the command file to run the simulated mailboxes.

To create a KPGS test environment, the following steps must be performed:

1. Set the default directory to the root directory for the KPGS program that is to be tested.
2. Define the KPGS logicals by typing
   KPGS_DEV
3. Set the default directory to the [.COM] subdirectory by typing
   SD COM_DIR
4. Run the CREATE_KPGS_TEST utility by typing
   CREATE_KPGS_TEST
5. Respond to the prompts for program name and version with the identical entries that have been made in the PROCESS PROGRAM Table of the database (see Section 3).
6. Respond to the prompt for test date by entering the date of the input data that is to be used for the test.
7. The utility will now attempt to create the test environment. If the utility is successful in creating the test environment, a message will be displayed describing the command that will initiate the test. If for any reason the test environment cannot be established, a message will be displayed describing the problem and providing suggestions for resolving it.

2.4.2 KPGS, Calibration Data, and Parameter File Delivery

There are three methods for the delivery of KPGS and associated materials: electronic (the preferred method), hard media (e.g., magnetic tape and hardcopy), or a combination of electronic delivery and hard media (e.g., source code and test data delivered electronically and design documentation delivered on hard media). If a magnetic tape is used to make a delivery, it must be a nine-track, 6250-bit-per-inch (bpi) tape. VAX COPY or BACKUP format tapes are preferred. If the tape cannot be created as a VAX format tape, then a nonlabeled, fixed-length record format tape is acceptable. Instructions for reading the tape should be provided.

Electronic deliveries of the KPGS, calibration file, and/or PI parameter file are made from the directory structure created by KPGS_DIR.

When the delivery directory created by KPGS_DIR has been populated with the delivery items, the PI executes the delivery form (see Section 2.4.2.3) to initiate the delivery procedure. The delivery items are copied from the PI's delivery directory to a KPGS delivery holding area. The KITT is notified automatically by E-mail when the delivery form is entered in the system. The KITT then verifies that the delivery is complete and configures the delivery for integration test. Similarly, when they receive a delivery on hard media, the KITT verifies it for completeness and then configures it for integration test.

Electronic deliveries can also be made without using the delivery form. Instruction for making an electronic delivery without using the form can be accessed by entering DEL_PROC.

2.4.2.1 KPGS Delivery Items

Regardless of the means of delivery, each delivery of KPGS must consist of the following items:

- A list of items being delivered
- KPGS source code, CDF skeleton table, PI parameter file, and calibration file
- A working executable program
- Build instructions for the compilation and linking of the source code (including command files used to build the KPGS program)
- Test data
- Test procedures and test results

Whenever a change is made in one or more units of the KPGS, the entire program is delivered to the CDHF, not just the changed units. This simplifies configuration management (CM) of the KPGS on the CDHF and removes a potential source of errors.
If a change in the KPGS results in a change in the design of the KPGS, the resources required by the KPGS, or the key parameters generated, the KPGS design document must be updated and included with the delivery.

2.4.2.2 Calibration and PI Parameter Files Delivery Procedure

Deliveries of calibration and PI parameter files (if any) are treated independently of the source code deliveries, unless the source code has changed as a result of the change in the calibration/PI parameter files. Deliveries or updates to calibration or PI parameter file consists of the following:

- A list of items being delivered
- A new calibration and/or PI parameter file(s)
- Test data, when applicable
- Test procedures and test results

If a change in calibration or PI parameter files results in a change in the source code, a combined source/calibration/parameter delivery is made. If a change in the calibration/PI parameter files results in a change in the resources required by the KPGS, the KPGS design document must be updated and included with the delivery.

2.4.2.3 The Delivery Form

The KITT tracks all the software components that compose the various test and operational baselines. All new or revised KPGS programs and data must be delivered to the KITT. To facilitate software deliveries, an online form and database are used (deliveries on hard media include hardcopy listings of the delivery contents). Before a form is completed, all the associated files must be copied to the delivery subdirectory. In addition, the special instructions should indicate the data file types (mission, instrument, and data type) that are processed, used as input support files, and generated by a new or significantly revised KPGS program. Upon completion of the form, the associated files are automatically delivered to the KITT. A copy of the online delivery form is in Appendix D.

2.4.3 KPGS Problem Reporting

CCRs are the mechanism used by the Goddard Space Flight Center (GSFC) to track design changes and problem reports. Any member of the development, test, and management team may submit a CCR. Each CCR is analyzed to determine if a change is required and what type of change is necessary. The actual completed changes, the test results, and final closure of each CCR are also tracked.

2.4.4 SQL*Forms Operating Instructions

This section describes the procedure for using the online KPGS delivery form (KDF) to make an electronic delivery of the KPGS, calibration files, or PI parameter file.

2.4.4.1 Form Execution

The KDF is executed on the development database. After the operation database is created, the associated tables may be transferred. A user who does not have a development database account must contact the database administrator. To activate the KDF form from the Digital Command Language (DCL) prompt, the user types the following:

DFC (activation from DEC Terminal)
DFCE (activation from PC)

2.4.4.2 Basic SQL*Forms Operations

SQL*Forms is a user interface product for the ORACLE database management system (DBMS). It allows users to enter new information, delete information, query existing information, and update information. To perform these operations, users must master some basic operations:

- **Keypad Help**—Keypad mappings have been defined for DEC VT100 and VT220 compatible terminals. Based on terminal type, the RUNFORM command will automatically assign one of the two keypad mappings. At any point, a list of the keypad

mappings can be obtained by pressing the control key (Ctrl) and the letter "K" at the same time. If pulldown menus are preferable to function keys, the user must learn the MENU key to display the menu options. By using the return/enter key and the arrow keys, users can complete all operations with the pulldown menus.

- **Field Help**—A help message will automatically appear on the message line as the cursor enters each item. The help message will indicate the purpose of each item, whether the LIST key (LOV) is enabled to display the legal values, whether a special format is required, whether the field is mandatory (MAN), and whether the EDIT key (EDT) displays a popup window to view the entire field.
- **Basic Form Navigation**—The NEXT FIELD and PREVIOUS FIELD keys allow navigation between each item. The RIGHT and LEFT keys allow navigations within a field item. The UP and DOWN keys allow navigation to the previous and next entries (records). Some forms, such as KDF, have items grouped into blocks separated by a solid line on the form. To navigate from one block to another, the user must press the PREVIOUS BLOCK and NEXT BLOCK keys. Finally, many forms have special function keys that allow navigation to and from other form pages and popup windows. These special function keys are described at the bottom of the forms.
- **Performing Queries**—To retrieve all records, the user must press the EXECUTE QUERY key. In most cases, however, it is preferable to specify query criteria first rather than retrieve everything from the database. To specify query criteria, the user must press the ENTER QUERY key, which activates the query mode, and enter values in the items that should be used to restrict the query. A wildcard (%) can be used if the user cannot remember an entire word. After the criteria are entered, the user must press the EXECUTE QUERY key to retrieve the matching entries. The form will remain in query mode until entries are successfully retrieved. To exit query mode, the user should press the EXIT/CANCEL key. The ENTER QUERY message at the bottom of the screen indicates that the form is in query mode.
- **Performing Inserts**—To perform an insert, the user must find an empty entry and fill in the appropriate items. To create a blank entry, the user needs to press the INSERT RECORD key. After completing an entry, the user must press the COMMIT key to save the entry in the database.
- **Performing Updates**—To perform an update, the user must first perform a query to retrieve the applicable entries. Once the entries are retrieved, the user can type in any changes and save them to the database by using the COMMIT key.
- **Performing Deletes**—To perform a delete, the user must first perform a query to retrieve the applicable entries. Once the entries are retrieved, the user can delete them using the DELETE RECORD key. Although the record disappears immediately, the deletion does not occur in the database until the COMMIT key is used. The CLEAR FORM key can be used to undo a deletion before a commit. In many applications, deletes are restricted.
- **Performing Edits**—For longer fields, the form often will display a truncated version of the field. The EDIT key can be used to view the entire field. The help message for truncated fields will indicate that edit is available with the EDT code. The edit function is particularly useful for free-format text. Within an edit popup window, automatic word wrapping is enabled. For advanced users, a wide variety of edit functions such as cut, paste, copy, and search are available.
- **Exiting a Form**—To exit a form, the user must press the EXIT/CANCEL key. If in query mode, the user must press the key twice. If some changes are not committed, a warning message will be displayed to allow the user either to commit the changes or to ignore them.

2.5 Key Parameter Reprocessing

Errors in the KPGS or in the calibration file may require key parameter processing to be temporarily suspended and/or some key parameter data to be reprocessed. The ISTP CDHF user's guide describes the procedures for requesting the suspension of key parameter processing and for making key parameter reprocessing requests.

SECTION 3—CODING STANDARDS

This section describes the coding standards to be used in the development of the KPGS. Two compilers are supported on the CDHF for the KPGS: VAX FORTRAN and C. Each standard is described, along with the provided extensions, in the sections that follow. In addition, all appropriate standards and conventions discussed in Reference 1 should be adhered to.

3.1 CDHF FORTRAN Language Standards

VAX FORTRAN is an implementation of full language FORTRAN-77 conforming to American National Standards Institute (ANSI) FORTRAN, ANSI X3.9-1978. The KPGS may make use of any of the available VAX FORTRAN extensions.

3.2 CDHF C Language Standards

The VAX C language supported on the CDHF follows the ANSI C standard. The KPGS may make use of any of the available VAX C extensions.

3.3 KPGS Program Structure

To ensure the proper management and tracking of products generated by the KPGS, each KPGS application must conform to CDHF-defined rules concerning program structure. Specific variables and support routines must be included in every KPGS, and naming conventions must be followed. Appendix A contains sample KPGS program structures illustrating the conventions and requirements presented here.

3.3.1 Naming Conventions

The ICSS support routines are identifiable by their names. All CDHF support routines and declarations begin with “ICSS_” to signify that they are supplied by the ICSS. All CDF routines begin with “CDF_”, and database support routines begin with “DB_”. Consequently, the “ICSS_”, “CDF_”, and “DB_” prefixes are reserved by ICSS and should not be used by the PI within the KPGS for any PI-supplied routines or variables.

The naming conventions for the skeleton tables, calibration parameter files, and PI parameter files used on the ISTP CDHF follow the standards of logical names for other files on the CDHF. However, the date portion of the filenames has been omitted for these files, since the files are applicable over a range of dates. Skeleton tables shall be named as follows:

<Short Mission Name>_K<KP File #>_<Instrument Name>_V<Version>.SKELETON_TABLE

where

<Short Mission Name> is the two-character identifier for the applicable mission.
<KP File #> is a number between 0 and 9 identifying one of several K index value (KP) files for the same mission and instrument. The first KP file is always file “0.”
<Instrument Name> is the three- or four-character name of the applicable instrument.
<Version> is two numeric digits indicating the particular version of the skeleton table. The first version of a skeleton table is version “01.”

The following is an example of a valid filename for the first version of the skeleton table for the EPIC instrument on the GEOTAIL Mission:

GE_K0_EPI_V01.SKELETON_TABLE

Calibration files shall be named as follows:

<Short Mission Name>_CD_<Instrument Name>_V<Version>.<Instrument Component>

or

<Short Mission Name>_CD_<Instrument Name>_<Instrument Component without C>_V<Version>.<Instrument Component>

where

<Short Mission Name>, <Instrument Name>, and <Version> are as described above.
<Instrument Component> is a four-character extension that differentiates between different calibration files used for the same mission and instrument. This extension shall begin with a C followed by three numeric digits, i.e., C001.

The following is an example of a valid filename for the first version of the only calibration file for the EPIC instrument on the GEOTAIL Mission:

GE_CD_EPI_V01.C001

If more than one calibration file is needed, the instrument component number (three numeric digit, e.g., 001) should be used as part of the filename. The following are examples of valid filenames for the first version of two calibration files for the Comprehensive Plasma Composition (CPI) instrument on the GEOTAIL mission:

GE_CD_CPI_001_V01.C001

GE_CD_CPI_002_V01.C002

PI parameter files shall be named as follows:

<Short Mission Name>_PF_<Instrument Name>_V<Version>.DAT

where

<Short Mission Name>, <Instrument Name>, and <Version> are as described above.

The following is an example of a valid filename for the first version of the PI parameter file for the EPIC instrument on the GEOTAIL Mission:

GE_PF_EPI_V01.DAT

Optionally, a date in the form of YYMMDD can be used as part of the name for calibration and PI parameter files if these files are associated with a date (e.g., GE_PF_EPI_19950123_V01.DAT).

3.3.2 Required Definitions

KPGS applications require the following variable declarations. This is not a complete list of all declarations but, rather, those declarations requiring initial values by the PI. Declarations are given in FORTRAN and C.

FORTRAN declarations:

CHAR*20	KPGS_NAME
CHAR*5	KPGS_VERSION
DATA	KPGS_NAME/“name of the KPGS main program”/
DATA	KPGS_VERSION/“version number of the KPGS”/
INTEGER	CDF_ID(10) (used to store up to 10 unit numbers to be assigned to CDF key parameter output files)

C declarations:

AUTO $DESCRIPTOR (KPGS_NAME, “name of the KPGS main program”);
AUTO $DESCRIPTOR (KPGS_VERSION, “version number of the KPGS”);
INT CDF_ID[10]; (used to store up to 10 unit numbers to be assigned to CDF key parameter output files)

The KPGS program name contains up to 20 characters and follows the naming convention

Spacecraft designator + ‘_’ + Instrument designator + ‘_’ + ‘KP’

where

Spacecraft designator is a character code for the spacecraft (e.g., GEOTAIL, WIND, or POLAR).
Instrument designator is a three- or four-character code for the instrument.
‘KP’ indicates that the program generates key parameters.

For example, GEOTAIL_EFD_KP denotes the name of the KPGS for the GEOTAIL Electric Field Detector (EFD) instrument, and V1.0 in KPGS_VERSION denotes version 1 of the software.

3.3.3 Required Support Routines

All KPGS are required to call the following ICSS support routines:

ICSS_KPG_INIT ICSS_KPG_TERM

ICSS_KPG_INIT must be the first support routine called after the program declarations. If the returned status code yields an error, the KPGS application must terminate processing. ICSS_KPG_INIT records the error message in the ISTP CDHF system message log.
ICSS_KPG_TERM must be the last support routine called after key parameter processing completes (whether processing completed successfully or not). If an error occurs during its execution, ICSS_KPG_TERM records the error message in the ISTP CDHF system message log. Status is returned to the KPGS application.
The following support routines are available to the KPGS for key parameter processing. (The names of these routines are the same in both playback and NRT modes with the same argument list.)

ICSS_KPG_INIT
ICSS_KPG_TERM
ICSS_OPEN_ATT
ICSS_OPEN_HK
ICSS_OPEN_LZ
ICSS_OPEN_ORB
ICSS_OPEN_SD
ICSS_RET_ATT
ICSS_RET_HK
ICSS_RET_LZ
ICSS_RET_ORB
ICSS_RET_SD64
ICSS_RET_SD
ICSS_KPG_COMMENT
ICSS_TRANSF_ORB
ICSS_TRANSF_ATT
ICSS_CNVRT_TO_EPOCH
ICSS_GET_CD
ICSS_GET_PF
ICSS_SPINPH_SIRIUS
ICSS_GET_REFERENCE_FILES
ICSS_SPINPH_WIND_LZ
ICSS_PAYLOAD_TO_GSE
ICSS_CNVT_FROM_RP
ICSS_POS_OF_SUN
ICSS_VELOCITY_TRANS
ICSS_GCI_TO_GEODETIC
ICSS_TILT_ANGLE
ICSS_POS_VEL_OF_CELESTIAL
ICSS_GEODETIC_TO_GCI
ICSS_SD_BLK_TYP
ICSS_INDICES
ICSS_CNVRT_EPOCH_TO_PB5
ICSS_TRANSF_TO_MTC
ICSS_TRANSF_TO_MSPC
ICSS_GET_FILE
ICSS_COMPUTE_CGMLT
ICSS_COMPUTE_EDMLT
ICSS_OPEN_PO_SPINPH
ICSS_RET_PO_SPINPH
ICSS_OPEN_DP_ATT
ICSS_RET_DP_ATT
ICSS_ACF_TO_GCI
ICSS_OPEN_SOHO_LZ
ICSS_RET_SOHO_PACKETS
ICSS_OPEN_SOHO_HK
ICSS_RET_SOHO_HK
ICSS_OPEN_SOHO_ATT
ICSS_RET_SOHO_ATT
ICSS_NRT_ACTIVE
ICSS_WRITE_3DP_KP
ICSS_WRITE_MFI_KP
ICSS_WRITE_SWE_KP
ICSS_VAX_TO_IEEE
ICSS_UTC_TAI_OFFSET
ICSS_GSM_SM
ICSS_TSY

Sections 4 and 5 contain detailed descriptions of these routines.

3.3.4 CDF Routines

The CDF software library consists of a set of routines for describing, storing, and randomly accessing data files in a standard format. Because of the controlled nature of key parameter generation on the CDHF, only a subset of the full CDF library may be used within the KPGS.
The following CDF routines are available for use in the KPGS:

CDF_attr_entry_inquire
CDF_attr_get
CDF_attr_inquire
CDF_attr_num
CDF_attr_put
CDF_doc
CDF_error
CDF_inquire
CDF_lib
CDF_var_get
CDF_var_hyper_get
CDF_var_hyper_put
CDF_var_inquire
CDF_var_num
CDF_var_put
Compute_EPOCH
EPOCH_breakdown

For more details on these functions, see Reference 4.

3.3.5 FORTRAN Logical Unit Numbers

The logical unit numbers 10 through 99 may be used to open scratch, user message, PI parameter, or calibration files.

3.3.6 Required Include Files

The following include files must immediately follow the program declarations:
For FORTRAN:

```
CDF$INC:CDF.INC
ICSS_INC:ICSS_KP_FILL_VALUES.INC
ICSS_INC:ICSS_MESSAGES.INC
```

For C:

```
<descrip.h>
<cdf$inc:cdf.h>
<icss_inc:icss_kp_fill_values.h>
<icss_inc:icss_messages.h>
```

3.3.7 Required Error-Handling Procedures

All KPGS are required to check the status returned from each of the ICSS and CDF support routine calls and to take appropriate actions when such status indicates an error. All ICSS support routines return the value SS$_NORMAL when they are successful. Values other than SS$_NORMAL may indicate a warning, information, or fatal condition, as indicated in Section 4. The KPGS program is required to terminate when a fatal condition is indicated. The ICSS support routines will record the error messages in the ISTP CDHF system message log. Appendix C contains a complete listing of error/status messages. Appendix A contains descriptions of sample programs that illustrate the proper handling of codes. Proper status checking for CDF routines is illustrated in Reference 4.
When the KPGS abnormally aborts its processing, it must terminate (after first calling ICSS_KPG_TERM), using an EXIT statement, the format of which is as follows:
For FORTRAN:

```
Call EXIT(status)
```

For C:

```
EXIT(status);
```

where status is the returned value from the ICSS_KPG_TERM support routine.
Terminating the KPGS in this way, with a status other than SS$_NORMAL, signals the ICSS-supported exit handler to perform an alternate exit sequence that does not catalog the key parameter file. The EXIT statement should follow the call to the ICSS_KPG_TERM routine. The status argument whose value indicates the condition causing the abnormal termination is passed to the ICSS_KPG_TERM routine and the status returned from ICSS_KPG_TERM is passed to the ICSS-supported exit handler.

3.4 CDF Key Parameter File Conventions

The detailed definitions for the standard conventions related to the CDF key parameter files can be found in the ISTP KPGS standards and conventions document (Reference 1).

SECTION 4—SUPPORT ROUTINE DESCRIPTIONS

This section contains the detailed descriptions of the ICSS-supplied routines for supporting key parameter generation. Each support routine description includes input and output, sample calling sequences in FORTRAN and in C, and error conditions. As noted previously, routines that begin with ICSS_, CDF_, and DB_ are reserved. The KPGS developers should not use these prefixes for any PI-supplied routines. The support routines are under configuration control and consequently are not alterable by PIs or other CDHF users. The following list summarizes the complete set of ICSS support routines:

Unit Name	Description
ICSS_KPG_INIT	Initialize KPGS routine environment
ICSS_KPG_TERM	Terminate KPGS processing
ICSS_OPEN_ATT	Open attitude data file(s)
ICSS_OPEN_HK	Open housekeeping data file(s)
ICSS_OPEN_LZ	Open level-zero data file(s)
ICSS_OPEN_ORB	Open orbit data file(s)
ICSS_OPEN_SD	Open SIRIUS data file(s) (GEOTAIL only)
ICSS_RET_ATT	Return requested attitude data
ICSS_RET_HK	Return requested housekeeping data
ICSS_RET_LZ	Return requested level-zero data
ICSS_RET_ORB	Return requested orbit data
ICSS_RET_SD64	Return a block of 64 minor frames from a SIRIUS data file (GEOTAIL only)
ICSS_RET_SD	Return requested SIRIUS data (GEOTAIL only)
ICSS_KPG_COMMENT	Write a KPGS-supplied comment to SFDU header
ICSS_TRANSF_ORB	Transform orbit coordinates
ICSS_TRANSF_ATT	Transform attitude coordinates
ICSS_CNVRT_TO_EPOCH	Convert LZ major frame time to CDF epoch format
ICSS_GET_CD	Retrieve physical filename of a calibration file
ICSS_GET_PF	Retrieve physical filename of the PI parameter file
ICSS_SPINPH_SIRIUS	Calculate the spin-phase angle using SIRIUS data (GEOTAIL only)
ICSS_GET_REFERENCE_FILES	Obtain reference filenames from database [International Magnetosphere Physics (IMP)-8 only]
ICSS_SPINPH_WIND_LZ	Return the spin-phase angle (WIND only)
ICSS_PAYLOAD_TO_GSE	Transform nonspinning payload coordinates
ICSS_CNVT_FROM_RP	Convert a vector from rotating payload to fixed payload
ICSS_POS_OF_SUN	Return Sun position and velocity vectors
ICSS_VELOCITY_TRANS	Convert velocities from one coordinate system to another
ICSS_GCI_TO_GEODETIC	Convert an input Geocentric Celestial Inertial

	(GCI) vector to geodetic longitude, latitude, and height
ICSS_GEODETIC_TO_GCI	Convert an input geodetic longitude, latitude, and height to a vector in GCI coordinates
ICSS_SD_BLK_TYP	Determine whether a block of SIRIUS data contains a message block
ICSS_INDICES	Return solar indices
ICSS_CNVRT_EPOCH_TO_PB5	Convert date/time in epoch format to PB5 format
ICSS_TRANSF_TO_MTC	Transform orbit coordinates or any vectorized quantity to modified topographic coordinate (MTC) system
ICSS_TRANSF_TO_MSPC	Transform orbit coordinates or any vectorized quantity to modified spin plan coordinate (MSPC) system
ICSS_GET_FILE	Query the CDHF database for physical filename(s) and associated span start/stop date(s)
ICSS_COMPUTE_CGMLT	Compute the corrected geomagnetic local time (CGMLT)
ICSS_COMPUTE_EDMLT	Compute the eccentric-dipole magnetic local time (EDMLT)
ICSS_OPEN_PO_SPINPH	Open the POLAR spin-phase file(s) (POLAR only)
ICSS_RET_PO_SPINPH	Return requested POLAR spin-phase data (POLAR only)
ICSS_OPEN_DP_ATT	Open despun platform attitude data file(s) (POLAR only)
ICSS_RET_DP_ATT	Return despun platform attitude data (POLAR only)
ICSS_ACF_TO_GCI	Convert attitude vector in attitude control reference frame (ACF) to GCI reference frame (SOHO only)

Unit Name	Description
ICSS_OPEN_SOHO_LZ	Open SOHO level-zero data file(s) (SOHO only)
ICSS_RET_SOHO_PACKETS	Return SOHO packet data (SOHO only)
ICSS_OPEN_SOHO_HK	Open SOHO housekeeping data file(s) (SOHO only)
ICSS_RET_SOHO_HK	Return SOHO housekeeping data (SOHO only)
ICSS_OPEN_SOHO_ATT	Open SOHO attitude data file(s) (SOHO only)
ICSS_RET_SOHO_ATT	Return SOHO attitude data (SOHO only)
ICSS_UTC_TAI_OFFSET	Determine offset between UTC and TAI files
ICSS_GSM_SM	Converts the coordinate systems between Geocentric Celestial Magnetospheric (GSM) and Solar Magnetic (SM)
ICSS_TILT_ANGLE	Calculates the tilt angle
ICSS_TSY	Provides an ISTP interface to the Tsyganenko models
ICSS_POS_VEL_OF_ CELESTIAL	Retrieves the position and velocity vectors of a celestial body in GCI coordinate system

In addition, the following CDF routines are available for use by the KPGS for the generation of key parameter output file(s); the detailed descriptions of the CDF routines are documented in Reference 4.

Unit Name	Description
CDF_attr_entry_inquire	Inquire about a specific attribute entry
CDF_attr_get	Read an attribute entry from a CDF
CDF_attr_inquire	Inquire about a particular attribute
CDF_attr_num	Retrieve attribute number associated with a particular attribute
CDF_attr_put	Put an attribute entry to a CDF
CDF_doc	Retrieve documentation type information about a CDF
CDF_error	Retrieve text associated with a particular error code
CDF_inquire	Inquire about the basic characteristics of a CDF
CDF_lib	Perform all possible operations on a CDF
CDF_var_get	Read a single value from a variable
CDF_var_hyper_get	Read multiple values from a variable
CDF_var_hyper_put	Put multiple values to a variable
CDF_var_inquire	Inquire about a particular variable or the size of the buffers necessary to receive value(s)
CDF_var_num	Retrieve variable number associated with a particular variable
CDF_var_put	Put a single value to a variable

Unit Name	Description
Compute_EPOCH	Transform Gregorian time values into epoch values
EPOCH_breakdown	Convert an epoch value to Gregorian time

Appendix A contains descriptions of sample KPGS programs that illustrate the use of each of these routines, and the creation and use of scratch files and the user message file.

4.1 ICSS_KPG_INIT

4.1.1 Purpose

ICSS_KPG_INIT is the first support routine called within the KPGS. It initializes the key parameter files for key parameter processing.

4.1.2 Description

This routine verifies the key parameter input, opens the key parameter output files, and initializes the SFDU header for key parameter processing. The ISTP standards and conventions document (Reference 1) and the ISTP CDHF data format control document (Reference 5) contain details on the SFDU header.

4.1.3 Interfaces

Input to ICSS_KPG_INIT from the KPGS:

KPGS_NAME	The name of the KPGS application
KPGS_VERSION	The version number of the KPGS

Output from ICSS_KPG_INIT to the KPGS:

CDF_ID	CDF file IDs of key parameter output files (unit numbers)
KP_INIT_STATUS	The status, which indicates whether the program initialization was successful

Output from ICSS_KPG_INIT to the message handler:

operations_message	Message indicating status of the routine

4.1.4 Calling Sequence

For FORTRAN:

```
Call ICSS_KPG_INIT (KPGS_NAME, KPGS_VERSION, CDF_ID, KP_INIT_STATUS)
    CHARACTER*20    KPGS_NAME  !Program name of KPGS
    CHARACTER*5     KPGS_VERSION    !Version number of KPGS
    INTEGER*4       CDF_ID(10)      !CDF IDs of key parameter output files
                                    (see Section 3.3.2)
    INTEGER*4       KP_INIT_STATUS  !Key parameter initialization status
```

For C:

```
ICSS_KPG_INIT (&KPGS_NAME, &KPGS_VERSION, &CDF_ID,
        &KP_INIT_STATUS);
    auto        $DESCRIPTOR(KPGS_NAME, "GEOTAIL_EFD_KP");
    auto        $DESCRIPTOR(KPGS_VERSION, "V1.0");
    int         CDF_ID[10];
    int         KP_INIT_STATUS;
```

4.1.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_KPG_INIT:

ICSS_INIT_FAIL	Failure; unable to assign a mailbox channel
ICSS_INIT_FAIL_PUT_MSG	Failure to assign a mailbox channel; unable to call ICSS_PUT_MSG
ICSS_KPGS_INFILE_CLOSE_ERR	Error closing a file containing parameters for input to the KPGS
ICSS_KPGS_INFILE_OPEN_ERR	Error opening a file containing parameters for input to the KPGS
ICSS_KPGS_INFILE_READ_ERR	Error reading from a file containing parameters for input to the KPGS
ICSS_KPGS_INFILE_INV_REC	Invalid records in a file containing parameters for input to the KPGS
ICSS_KP_OUTFILE_OPEN_ERR	Error opening the key parameter output file

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after the call to ICSS_KPG_INIT. If the status indicates an error, the KPGS application must terminate processing.

4.2 ICSS_KPG_TERM

4.2.1 Purpose

ICSS_KPG_TERM is the last support routine called by the KPGS, just before application exit. It closes the key parameter input and output files.

4.2.2 Description

ICSS_KPG_TERM is responsible for completing the SFDU header, closing all the key parameter input files, closing the newly created key parameter output file(s), and cataloging the new file(s) in the CDHF catalog.

4.2.3 Interfaces

Input to ICSS_KPG_TERM from the KPGS:

INPUT_STATUS	Status sent from KPGS indicating if any errors were encountered prior to call; should be SS$_NORMAL when terminating normally, and another value when aborting

Output from ICSS_KPG_TERM to the KPGS:

KP_TERM_STATUS	The return status from ICSS_KPG_TERM

Output from ICSS_KPG_TERM to the message handler:

operations_message	Message indicating status of termination

4.2.4 Calling Sequence

For FORTRAN:

```
Call ICSS_KPG_TERM (INPUT_STATUS, KP_TERM_STATUS)
    INTEGER*4   INPUT_STATUS        !Status sent in from KPGS
    INTEGER*4   KP_TERM_STATUS      !Termination status
```

For C:

```
ICSS_KPG_TERM (&INPUT_STATUS, &KP_TERM_STATUS);
    int     INPUT_STATUS;
    int     KP_TERM_STATUS;
```

4.2.5 Error Conditions

The following fatal error conditions may be recorded in the system message log by ICSS_KPG_TERM:

ICSS_CATALOG_KP_FILE_ERROR	Failure to catalog a key parameter file. This is a fatal error and should be handled as such by the program.
ICSS_DB_ERR	Error connecting to database. The KPGS can discontinue processing if the program is unable to do any other processing at this time.

ICSS_DB_MISSING_ADI_NUM	ADI number is not in the database for the file.
ICSS_INPUT_FILE_CLOSERR	Input file cannot be closed.
ICSS_OUTPUT_FILE_CLOSERR	Output file cannot be closed. The KPGS can discontinue processing if the program is unable to do any other processing at this time.
ICSS_SFDU_HDR_CLOSE_ERR	SFDU header cannot be closed. An error message is sent to the message handler.
The new key parameter file(s) is not cataloged	and an error is returned to the KPGS.
The KPGS can discontinue processing if the program is unable to do any other processing at this time.	
ICSS_INV_START_TIME	Valid epoch time could not be found in the first record of the key parameter CDF file.
ICSS_INV_STOP_TIME	Valid epoch time could not be found in the last record of the key parameter CDF file.

4.3 ICSS_OPEN_ATT

4.3.1 Purpose

ICSS_OPEN_ATT opens the day of data and previous day's attitude files.

4.3.2 Description

ICSS_OPEN_ATT opens both the day of data and the previous day's attitude files, if specified. It also reads the global attributes of the CDF attitude files from those two files.

4.3.3 Interfaces

Input to ICSS_OPEN_ATT from the attitude file:

ATT_HEADER	The file label record for the attitude file

Output from ICSS_OPEN_ATT to the KPGS:

COMPLETION_STATUS	The return status from the routine

4.3.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_ATT (COMPLETION_STATUS)
    INTEGER*4    COMPLETION_STATUS        !Message number
```

For C:

```
ICSS_OPEN_ATT (&COMPLETION_STATUS);
    int     COMPLETION_STATUS;          /*Message number*/
```

4.3.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_ATT:

ICSS_INV_TIME_PR_ATT	Error obtaining file times from the header record in the primary attitude file
ICSS_INV_TIME_SC_ATT	Error obtaining file times from the header record in the secondary attitude file
ICSS_NO_PR_ATT	Primary attitude file does not exist
ICSS_OPEN_PR_ATT	Error opening the primary attitude file
ICSS_OPEN_SC_ATT	Error opening the secondary attitude file
ICSS_SC_AFTER_PR_ATT	Invalid secondary attitude file, begins after the primary file's start time

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.4 ICSS_OPEN_HK

4.4.1 Purpose

ICSS_OPEN_HK opens the day of data and previous day's housekeeping files.

4.4.2 Description

ICSS_OPEN_HK opens both the day of data and the previous day's housekeeping files, if specified. It also reads the file label record from those two files.

4.4.3 Interfaces

Input to ICSS_OPEN_HK from the housekeeping file:

HK_HEADER	The file label record for the housekeeping file

Output from ICSS_OPEN_HK to the KPGS:

COMPLETION_STATUS	The return status from this routine

4.4.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_HK (COMPLETION_STATUS)
    INTEGER*4   COMPLETION_STATUS       !Message number
```

For C:

```
ICSS_OPEN_HK (&COMPLETION_STATUS);
    int     COMPLETION_STATUS;          /*Message number*/
```

4.4.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_HK:

ICSS_GETLUN_PR_HK	Error obtaining a logical unit number for the primary housekeeping file
ICSS_GETLUN_SC_HK	Error obtaining a logical unit number for the secondary housekeeping file
ICSS_GET_HK_REC_SIZE	Error obtaining the data record size in the housekeeping file
ICSS_NO_PR_HK	Primary housekeeping file does not exist
ICSS_OPEN_PR_HK	Error opening the primary housekeeping file
ICSS_OPEN_SC_HK	Error opening the secondary housekeeping file
ICSS_READHDR_PR_HK	Error reading the header record from the primary housekeeping file
ICSS_READHDR_SC_HK	Error reading the header record from the secondary housekeeping file
ICSS_SC_AFTER_PR_HK	Invalid secondary housekeeping file, begins after the primary file's start time

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.5 ICSS_OPEN_LZ

4.5.1 Purpose

ICSS_OPEN_LZ opens the day of data and previous day's level-zero files.

4.5.2 Description

ICSS_OPEN_LZ opens both the day of data and the previous day's level-zero files, if specified. It also reads the file label record from those two files.

4.5.3 Interfaces

Input to ICSS_OPEN_LZ from the level-zero file:

LZ_HEADER	The file label record for the level-zero file

Output from ICSS_OPEN_LZ to the KPGS:

COMPLETION_STATUS	The return status from this routine

4.5.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_LZ (COMPLETION_STATUS)
    INTEGER*4    COMPLETION_STATUS        !Message number
```

For C:

```
ICSS_OPEN_LZ (&COMPLETION_STATUS);
    int      COMPLETION_STATUS;           /*Message number*/
```

4.5.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_LZ:

ICSS_GETLUN_PR_LZ	Error obtaining a logical unit number for the primary level-zero file
ICSS_GETLUN_SC_LZ	Error obtaining a logical unit number for the secondary level-zero file
ICSS_GET_LZ_REC_SIZE	Error obtaining the data record size in the level-zero file
ICSS_NO_PR_LZ	Primary level-zero file does not exist
ICSS_OPEN_PR_LZ	Error opening the primary level-zero file
ICSS_OPEN_SC_LZ	Error opening the secondary level-zero file
ICSS_READHDR_PR_LZ	Error reading the header record from the primary level-zero file
ICSS_READHDR_SC_LZ	Error reading the header record from the secondary level-zero file
ICSS_SC_AFTER_PR_LZ	Invalid secondary level-zero file, begins after the primary file's start time

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.6 ICSS_OPEN_ORB

4.6.1 Purpose

ICSS_OPEN_ORB opens the day of data and previous day's orbit files.

4.6.2 Description

ICSS_OPEN_ORB opens both the day of data and the previous day's orbit files, if specified. It also reads the global attributes of the CDF orbit files from those two files.

4.6.3 Interfaces

Input to ICSS_OPEN_ORB from the orbit file:

ORB_HEADER	The file label record for the orbit file

Output from ICSS_OPEN_ORB to the KPGS:

COMPLETION_STATUS	The return status from this routine

4.6.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_ORB (COMPLETION_STATUS)
    INTEGER*4    COMPLETION_STATUS        !Message number
```

For C:

```
ICSS_OPEN_ORB (&COMPLETION_STATUS);
    int      COMPLETION_STATUS;           /*Message number*/
```

4.6.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_ORB:

ICSS_INV_TIME_PR_ORB	Error obtaining file times from the header record in the primary orbit file
ICSS_INV_TIME_SC_ORB	Error obtaining file times from the header record in the secondary orbit file
ICSS_NO_PR_ORB	Primary orbit file does not exist
ICSS_OPEN_PR_ORB	Error opening the primary orbit file
ICSS_OPEN_SC_ORB	Error opening the secondary orbit file
ICSS_SC_AFTER_PR_ORB	Invalid secondary orbit file, begins after the primary file's start time

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.7 ICSS_OPEN_SD (GEOTAIL Support Only)

4.7.1 Purpose

ICSS_OPEN_SD opens the SIRIUS data files for the day of data and the previous day of data.

4.7.2 Description

ICSS_OPEN_SD opens both the day of data and the previous day's SIRIUS files, if specified. It also reads the message data block from all the files. ICSS_OPEN_SD opens SIRIUS files for a file group when there are multiple SIRIUS data files for a single day's worth of data.

4.7.3 Interfaces

Input to ICSS_OPEN_SD from the SIRIUS file:

SD_BLOCK	The message data block from the SIRIUS file

Output from ICSS_OPEN_SD to the KPGS:

START_YEAR	The year of the first major frame in the SIRIUS files
STOP_YEAR	The year of the last major frame in the SIRIUS files
COMPLETION_STATUS	The return status from this routine

4.7.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_SD (START_YEAR, STOP_YEAR, COMPLETION_STATUS)
    INTEGER*4   START_YEAR          !Year of first frame
    INTEGER*4   STOP_YEAR           !Year of last frame
    INTEGER*4   COMPLETION_STATUS   !Message number
```

For C:

```
ICSS_OPEN_SD (&START_YEAR, &STOP_YEAR, &COMPLETION_STATUS);
```

```
int     START_YEAR;          /*First frame year*/
int     STOP_YEAR;           /*Last frame year*/
int     COMPLETION_STATUS;   /*Message number*/
```

4.7.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_SD:

ICSS_GET_SD_FRAMES	Error getting the number of minor frames in the SIRIUS file
ICSS_GETLUN_SD	Error obtaining a logical unit number for the SIRIUS file
ICSS_NO_PR_SD	Primary SIRIUS files do not exist
ICSS_OPEN_FILE_SD	Error opening the SIRIUS file
ICSS_READMSGDATA_SD	Error reading the message data block from the SIRIUS file
ICSS_SC_AFTER_PR_SD	Secondary SIRIUS files have later time stamps than the primary SIRIUS files

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.
Note: Because it is possible for the SIRIUS data to span a year boundary, the key parameter generation program should monitor the date of each frame and determine when a year change has occurred.

4.8 ICSS_RET_ATT

4.8.1 Purpose

ICSS_RET_ATT extracts attitude data from the attitude files for spin-stabilized spacecraft.

4.8.2 Description

This routine locates and returns the attitude data that corresponds to a given request time. Linear interpolation is used to estimate the attitude request if an attitude point does not exist for a specific request time.

4.8.3 Interfaces

Input to ICSS_RET_ATT from the KPGS:

ATT_VEC_REQ_DATE	The day of year and time of day of the requested attitude data
ATT_REQ_COOR_SYS	The requested coordinate system of the returned attitude data

Output from ICSS_RET_ATT to the KPGS:

ATT_VECTOR	The interpolated right ascension, declination, and spin rate for the requested time
ATT_RET_STATUS	The status of the attitude vector request

4.8.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_ATT (ATT_VEC_REQ_DATE, ATT_REQ_COOR_SYS, ATT_VECTOR,
        ATT_RET_STATUS)
INTEGER*4       ATT_VEC_REQ_DATE(2)     !(1) YYYYDDD
                                        !(2) milliseconds of day
CHARACTER*3     ATT_REQ_COOR_SYS        !GSE,GSM,GCI
```

```
      REAL*4          ATT_VECTOR(3)          !(1) Right ascension in radians
                                             !(2) Declination in radians
                                             !(3) Spin rate in rpm
      INTEGER*4       ATT_RET_STATUS         !Corresponds to message
```

For C:

```
   ICSS_RET_ATT (&ATT_VEC_REQ_DATE, &ATT_REQ_COOR_SYS,
            &ATT_VECTOR, &ATT_RET_STATUS);
      int             ATT_VEC_REQ_DATE[2];   /*(0) YYYYDDD
                                               (1) milliseconds*/
      auto            $DESCRIPTOR(ATT_REQ_COOR_SYS, "GSE");
                                             /*GSE,GSM,GCI*/
      float           ATT_VECTOR[3];         /*Right ascension, declination,
      spin
                                               rate*/
      int             ATT_RET_STATUS;        /*Corresponds to message*/
```

4.8.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_TIME_OUTRANGE	Requested time out of range of files

The following errors may be recorded in the system message log by ICSS_RET_ATT:

ICSS_ERR_READ_CDF	Error reading the attitude CDF files
ICSS_CDF_INQ	Error on CDF inquire
ICSS_ATT_FILE_INV	Opened attitude files do not meet attitude data request
ICSS_COOR_SYS_INV	Invalid coordinate system request

All errors returned from this support routine, except ICSS_TIME_OUTRANGE, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates a fatal error, the KPGS application must terminate processing.

4.9 ICSS_RET_HK

4.9.1 Purpose

ICSS_RET_HK extracts housekeeping data records from the housekeeping files.

4.9.2 Description

This support routine returns the housekeeping data records. The calling routine indicates whether the request is made by offset, by time, or in sequence. If the request is by offset, the record returned is the record with a start time closest to the start time of the last record read plus the millisecond offset. The offset may be a positive or negative number. If the request is by time, the record returned is the record that occurred closest to the given millisecond time. The time may be a positive or negative number. Negative numbers indicate times within the previous day's file. If the request is in sequence, the record returned is the next record sequentially within the file. If the first call to ICSS_RET_HK is in sequence, the first data record of the primary file is returned. If the first call is by offset, the time is calculated using the start time of the first data record in the primary file.

Only requests in sequence allow detection of gaps in the file. If a gap occurs, an informational status is returned along with the number of missing major frames.

4.9.3 Interfaces

Input to ICSS_RET_HK from the KPGS:

HK_REQ_TYPE	A flag indicating whether the housekeeping record request is being made by offset, by time, or in sequence
HK_REC_REQ	The housekeeping record request. If HK_REQ_TYPE indicates that request is by time, this variable contains

an elapsed-millisecond-of-day time. A negative number indicates a request for data from the previous day's file; i.e., -350000 requests the record at 86400000-350000, or 86050000. If HK_REQ_TYPE indicates that the record is by offset, HK_REC_REQ contains the millisecond offset (positive or negative) from the last data record read. This field is ignored if the request is in sequence.

Output from ICSS_RET_HK to the KPGS:

HK_RECORD	The requested housekeeping record
HK_RET_STAT	The return status from ICSS_RET_HK
MISSING_FRAMES	This output contains the number of major frames missing whenever a gap occurs. This number is always zero unless the request (HK_REQ_TYPE) is in sequence and a gap exists.

4.9.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_HK (HK_REQ_TYPE, HK_REC_REQ, HK_RECORD, HK_RET_STAT,
        MISSING_FRAMES)
    INTEGER*4   HK_REQ_TYPE                 !0 = In sequence
                                            !1 = By offset
                                            !2 = By time
    INTEGER*4   HK_REC_REQ          !Millisecond time
    INTEGER*4   HK_RECORD (REC_SIZE)    !Housekeeping data record; REC_SIZE
                                        !should be replaced with the number
                                        !of words in the instrument data record
    INTEGER*4   HK_RET_STAT             !Message number
    INTEGER*4   MISSING_FRAMES          !Number of frames lost in gap
```

For C:

```
ICSS_RET_HK (&HK_REQ_TYPE, &HK_REC_REQ, &HK_RECORD,
        &HK_RET_STAT, &MISSING_FRAMES);
    int     HK_REQ_TYPE;                /*0 = In sequence*/
                                        /*1 = By offset*/
                                        /*2 = By time*/
    int     HK_REC_REQ;                 /*Millisecond time*/
    int     HK_RECORD[REC_SIZE]         /*Housekeeping data record; REC_SIZE
                                          should be replaced with the maximum
                                          number of words in the instrument
    data
                                          record*/
    int     HK_RET_STAT;                /*Message number*/
    int     MISSING_FRAMES;             /*Number of frames lost in gap*/
```

4.9.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_EOF_PR_HK End of file reached reading the primary housekeeping file

The following informational message may be recorded in the system message log:

ICSS_GAP_IN_FILE A gap has occurred since the last major frame read

The following fatal errors may be detected by this routine:

ICSS_INV_HK_REQTYPE Invalid housekeeping request type; request must be by offset, by time, or in sequence

ICSS_INV_HK_REQTIME Invalid requested time; the time is not in the range of the housekeeping files

ICSS_READ_PR_HK Error reading the primary housekeeping file data record

ICSS_READ_SC_HK Error reading the secondary housekeeping file data record

If the ICSS_RET_HK returns an ICSS_GAP_IN_FILE, the record returned is the first housekeeping record following the data gap. The program should check the MISSING_FRAMES parameter for the number of missing frames. If this condition is unacceptable, the KPGS application should treat this condition as a fatal error.

The KPGS application must check the return status immediately after every support routine call. If the status indicates a fatal error, the KPGS application must terminate processing.

4.10 ICSS_RET_LZ

4.10.1 Purpose

ICSS_RET_LZ extracts level-zero data records from the level-zero files.

4.10.2 Description

This support routine returns the level-zero data records. The calling routine indicates whether the request is made by offset, by time, or in sequence. If the request is by offset, the record returned is the record with a start time closest to the start time of the last record read plus the millisecond offset. The offset may be a positive or negative number. If the request is by time, the record returned is the record that occurred closest to the given millisecond time. The time may be a positive or negative number. Negative numbers indicate times within the previous day's file. If the request is in sequence, the record returned is the next record sequentially within the file. If the first call to ICSS_RET_LZ is in sequence, the first data record of the primary file is returned. If the first call is by offset, the time is calculated using the start time of the first data record in the primary file.

Only requests in sequence allow detection of gaps in the file. If a gap occurs, an informational status is returned along with the number of missing major frames.

4.10.3 Interfaces

Input to ICSS_RET_LZ from the KPGS:

LVLZ_REQ_TYPE A flag indicating if the level-zero data record request is being made by offset, by time, or in sequence

LVLZ_REC_REQ The level-zero data record request. If LVLZ_REQ_TYPE indicates that request is by time, this variable contains an elapsed-millisecond-of-day time. A negative number indicates a request for data from the previous day's file; i.e., -350000 requests the record 86400000-350000, or 86050000. If LVLZ_REQ_TYPE indicates that the record is by offset, LVLZ_REC_REQ contains the millisecond offset (positive or negative) from the last data record

read.
This field is ignored if the request is in sequence.

Output from ICSS_RET_LZ to the KPGS:

LVLZ_RECORD	The requested level-zero data record
LVL_RET_STAT	The return status from ICSS_RET_LZ
MISSING_FRAMES	This output contains the number of major frames missing whenever a gap occurs. This number is always zero unless the (LVLZ_REQ_TYPE) is in sequence and a gap exists.

4.10.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_LZ (LVLZ_REQ_TYPE, LVLZ_REC_REQ, LVLZ_RECORD,
        LVLZ_RET_STAT, MISSING_FRAMES)
    INTEGER*4   LVLZ_REQ_TYPE           !0 = In sequence
                                        !1 = By offset
                                        !2 = By time
    INTEGER*4   LVLZ_REC_REQ            !Millisecond time
    INTEGER*4   LVLZ_RECORD(LZ_REC_SIZE) !Level-zero data record;
                                        !LZ_REC_SIZE should be
    replaced
                                        !by the maximum number of
    words
                                        !in the instrument data record
    INTEGER*4   LVLZ_RET_STAT           !Message number
    INTEGER*4   MISSING_FRAMES          !Number of frames lost in gap
```

For C:

```
ICSS_RET_LZ (&LVLZ_REQ_TYPE, &LVLZ_REC_REQ, &LVLZ_RECORD,
        &LVLZ_RET_STAT, &MISSING_FRAMES);
    int     LVLZ_REQ_TYPE;              /*0 = In sequence
                                          1 = By offset
                                          2 = By time*/
    int     LVLZ_REC_REQ;               /*Millisecond time*/
    int     LVLZ_RECORD[LZ_REC_SIZE];   /*Level-zero data record;
                                          LZ_REC_SIZE should be
    replaced
                                          by the maximum number of
    words
                                          in the instrument data
    record*/
    int     LVLZ_RET_STAT;              /*Message number*/
    int     MISSING_FRAMES;             /*Number of frames lost in
    gap*/
```

4.10.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_EOF_PR_LZ	End of file reached reading the primary level-zero file

The following informational message may be recorded in the system message log:

ICSS_GAP_IN_FILE	A gap occurred since the last major frame read

The following fatal errors may be detected by this routine:

ICSS_INV_LZ_REQTYPE	Invalid level-zero request type; request must be by offset, by time, or in sequence
ICSS_INV_LZ_REQTIME	Invalid requested time; the time is not in the range of the level-zero files
ICSS_READ_PR_LZ	Error reading the primary level-zero file data record
ICSS_READ_SC_LZ	Error reading the secondary level-zero file data record

If the ICSS_RET_LZ returns an ICSS_GAP_IN_FILE, the record returned is the first level-zero data record following the data gap. The program should check the MISSING_FRAMES parameter for the number of missing frames. If this condition is unacceptable, the KPGS application should treat this condition as a fatal error.

The KPGS application must check the return status immediately after every support routine call. If the status indicates a fatal error, the KPGS application must terminate processing.

4.11 ICSS_RET_ORB

4.11.1 Purpose

ICSS_RET_ORB extracts orbit data from the orbit files.

4.11.2 Description

This routine locates and returns the orbit vector that corresponds to a given request time. Cubic-spline interpolation is used to estimate the orbit point if an orbit point does not exist for a specific request time.

4.11.3 Interfaces

Input to ICSS_RET_ORB from the KPGS:

ORB_VEC_REQ_DATE	The day of year and time of day of the requested orbit vector
ORB_REQ_COOR_SYS	The requested coordinate system of the returned orbit vector

Output from ICSS_RET_ORB to the KPGS:

ORB_POS_VEC	The orbit position vector that corresponds to the request time
ORB_VEL_VEC	The orbit velocity vector that corresponds to the request time
ORB_RET_STATUS	The status of the orbit vector request

4.11.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_ORB (ORB_VEC_REQ_DATE, ORB_REQ_COOR_SYS,
        ORB_POS_VEC, ORB_VEL_VEC, ORB_RET_STATUS)
INTEGER*4    ORB_VEC_REQ_DATE(2)      !(1) YYYYDDD
                                      !(2) milliseconds of day
CHARACTER*3  ORB_REQ_COOR_SYS         !GSE,GSM,GCI
REAL*8       ORB_POS_VEC(3)           !Position vector (x,y,z) (km)
REAL*8       ORB_VEL_VEC(3)           !Velocity vector (dx,dy,dz)
(km/sec)
INTEGER*4    ORB_RET_STATUS           !Corresponds to message
```

For C:

```
ICSS_RET_ORB (&ORB_VEC_REQ_DATE, &ORB_REQ_COOR_SYS,
        &ORB_POS_VEC, &ORB_VEL_VEC, &ORB_RET_STATUS);
int          ORB_VEC_REQ_DATE[2];     /*(0) YYYYDDD
                                        (1) milliseconds of day*/
```

```
auto        $DESCRIPTOR(ORB_REQ_COOR_SYS, "GSE");
                                              /*GSE,GSM,GCI*/
double      ORB_POS_VEC[3];
double      ORB_VEL_VEC[3];
int         ORB_RET_STATUS;
```

4.11.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_TIME_OUTRANGE Requested time out of range of files

The following errors may be recorded in the system message log by ICSS_RET_ORB:

ICSS_ERR_READ_CDF Error reading orbit CDF files
ICSS_CDF_INQ Error from CDF inquiry
ICSS_ORB_FILE_INV Opened orbit files do not meet orbit data request
ICSS_COOR_SYS_INV Invalid coordinate system request

All errors returned from this support routine, except ICSS_TIME_OUTRANGE, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.12 ICSS_RET_SD64 (GEOTAIL Support Only)

4.12.1 Purpose

ICSS_RET_SD64 returns a block of 64 minor frames from the SIRIUS files.

4.12.2 Description

This support routine returns SIRIUS minor frames in a data block. A maximum of 64 frames are returned. The last block in a SIRIUS file may contain less than 64 minor frames. A user may request a block of minor frames offset from the last minor frame returned. Only forward offsets are allowed, i.e., no backward time jumps. To obtain the next contiguous block of minor frames, the user requests a block with an offset of 1. Only the primary SIRIUS file is used.

The SIRIUS blocks are returned in International Business Machines, Inc. (IBM) format, i.e., extended binary-coded decimal interchange code (EBCDIC) and IBM integers. Only minor frames are returned; the message data block and the control block are never returned.

4.12.3 Interfaces

Input to ICSS_RET_SD64 from the KPGS:

MINOR_FRAME_OFFSET

The offset instructs the routine to skip a number of minor frames ahead. Usually, this number is 1 to return the next contiguous block of 64 minor frames. Note that this number is ***not*** the minor frame number within a SIRIUS block. If the last minor frame returned on the previous call were 50 (in the data block), an offset of 51 would return the 101st minor frame within the data block. The offset may be any positive number; it may not be 0 or negative.

Output from ICSS_RET_SD64 to the KPGS:

SD_BLOCK A block of 64 contiguous minor frames

MINOR_FRAMES_RET

The number of minor frames returned within the SD_BLOCK. Usually this number is 64; however, the last block of minor frames within the file may contain fewer than 64 minor frames.

SD64_RET_STAT Status returned from the routine

4.12.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_SD64 (MINOR_FRAMES_OFFSET, SD_BLOCK,
        MINOR_FRAMES_RET, SD64_RET_STAT)
    INTEGER*4   MINOR_FRAMES_OFFSET     !Greater than 0, usually = 1
    BYTE        SD_BLOCK(9216)          !SIRIUS data block
    INTEGER*4   MINOR_FRAMES_RET        !Number of minor frames
    returned
    INTEGER*4   SD64_RET_STAT           !Message number
```

For C:

```
ICSS_RET_SD64 (&MINOR_FRAME_OFFSET, &SD_BLOCK,
        &MINOR_FRAMES_RET, &SD64_RET_STAT);
    int     MINOR_FRAMES_OFFSET;  /*Greater than 0, usually = 1*/
    int     SD_BLOCK[2304];       /*SIRIUS data block*/
    int     MINOR_FRAMES_RET;     /*Number of minor frames
                                    returned*/
    int     SD64_RET_STAT;        /*Message number*/
```

4.12.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_EOF_SD	End of file reached reading the last primary SIRIUS file

The following fatal error may be detected by this routine:

ICSS_INV_SD_OFF	Invalid SIRIUS minor frame offset specified (must be >0)

The KPGS application must check the return status immediately after every support routine call. If the status indicates a fatal error, the KPGS application must terminate processing.

4.13 ICSS_RET_SD (GEOTAIL Support Only)

4.13.1 Purpose

ICSS_RET_SD returns a data block from the SIRIUS files (160 minor frames).

4.13.2 Description

This support routine returns SIRIUS data blocks from the SIRIUS data files. The calling routine indicates whether the request is made by offset, by time, or in sequence. If the request is by offset, the record returned is the record with a start time closest to the start time of the last record read plus the millisecond offset. The offset may be a positive or negative number. If the request is by time, the record returned is the record that occurred closest to the given millisecond time. The time may be a positive or negative number. Negative numbers indicate times within the previous day's files. The returned data block has a time stamp equal to or before the requested time. If the request is in sequence, the record returned is the next record sequentially within the file. When the routine gets to the end of a SIRIUS file, it advances to the next file in the sequence. If the first call is by offset, the time is calculated using the start time of the message data block in the first primary file. The message data block from a SIRIUS file is returned whenever a request is made by time and the data for the requested time falls within the first record of the file, or when the request is made by sequence and the next record is the first record. The return of a message data block is indicated by a warning status returned to the caller. The control block of a SIRIUS file is never returned.

The SIRIUS data blocks are returned in IBM format, i.e., EBCDIC and IBM integers.

4.13.3 Interfaces

Input to ICSS_RET_SD from the KPGS:

SD_REQ_TYPE — A flag indicating if the SIRIUS data block request is being made by offset, by time, or in sequence

SD_REC_REQ — The SIRIUS data block request. If SD_REQ_TYPE indicates that request is by time, this variable contains a millisecond time. A negative number indicates a request for data from the previous day's files; i.e., -350000 requests the block at 86400000-350000

or

86050000. SD_REQ_TYPE indicates that the record is by

offset,

SD_REC_REQ contains the millisecond offset (positive or negative) from the last data block read. This field is ignored if

the

request is in sequence.

Output from ICSS_RET_SD to the KPGS:

SD_BLOCK — The requested SIRIUS data block

SD_RET_STAT — The return status from ICSS_RET_SD

4.13.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_SD (SD_REQ_TYPE, SD_REC_REQ, SD_BLOCK, SD_RET_STAT)
    INTEGER*4    SD_REQ_TYPE              !0 = In sequence
                                          !1 = By offset
```

```
                                        !2 = By time
    INTEGER*4   SD_REC_REQ          !Millisecond time
    BYTE        SD_BLOCK(23040)         !SIRIUS data block
    INTEGER*4   SD_RET_STAT             !Message number
```

For C:

```
ICSS_RET_SD (&SD_REQ_TYPE, &SD_REC_REQ, &SD_BLOCK, &SD_RET_STAT);
    int         SD_REQ_TYPE;            /*0 = In sequence
                                          1 = By offset
                                          2 = By time*/
    int         SD_REC_REQ;             /*Millisecond time*/
    int         SD_BLOCK[5760];         /*SIRIUS data block*/
    int         SD_RET_STAT;            /*Message number*/
```

4.13.5 Error Conditions

The following warning conditions may be detected by this routine:

ICSS_EOF_SD	End of file reached reading the last primary SIRIUS file
ICSS_MSG_BLK_RET_SD	The data block returned contains a SIRIUS file message data block

The following fatal errors may be detected by this routine:

ICSS_INV_SD_REQTYPE	Invalid SIRIUS requested type; request must be by offset, by time, or in sequence
ICSS_INV_SD_REQTIME	Invalid SIRIUS requested time; the time is not in the range of the SIRIUS files
ICSS_READ_SD	Error reading the SIRIUS file
ICSS_SD_CBLK_ERR	Error in the control block from the file

All errors returned from this support routine, except ICSS_MSG_BLK_RET_SD and ICSS_EOF_SD, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates a fatal error, the KPGS application must terminate processing.

4.14 ICSS_KPG_COMMENT

4.14.1 Purpose

ICSS_KPG_COMMENT writes a PI-supplied comment to the SFDU header associated with the key parameter output files.

4.14.2 Description

ICSS_KPG_COMMENT writes a PI-supplied comment to the SFDU header associated with the key parameter output files. This routine can be called multiple times as desired.

Note: The maximum number of bytes for KP_COMMENT is 240. Also, because the SFDU header should not contain null characters, it is important to initialize the KP_COMMENT buffer to blanks prior to writing any comments to it.

4.14.3 Interfaces

Input to ICSS_KPG_COMMENT from the KPGS:

KP_COMMENT (maximum length is 240 bytes; 80 bytes are recommended)	The comment to be written to the SFDU header

Output from ICSS_KPG_COMMENT to the KPGS:

COMMENT_STATUS	The status indicating whether the comment was successfully written to the SFDU header or whether there was an error

Output from the ICSS_KPG_COMMENT to the SFDU:

KP_COMMENT	The PI-supplied comment

4.14.4 Calling Sequence

For FORTRAN:

```
Call ICSS_KPG_COMMENT (KP_COMMENT, COMMENT_STATUS)
CHARACTER*240    KP_COMMENT                !PI-supplied comment
INTEGER*4        COMMENT_STATUS            !Message number
```

For C:

```
ICSS_KPG_COMMENT (&KP_DESC, &COMMENT_STATUS);
char             KP_COMMENT[240]={“THIS IS A COMMENT”};
                                           /*PI-supplied
comment*/
auto             $DESCRIPTOR (KP_DESC, KP_COMMENT);
int              COMMENT_STATUS;           /*Message number*/
```

4.14.5 Error Conditions

The following warning message may be recorded in the system message log:

ICSS_SFDU_BUF_FULL	SFDU buffer is full; no comment was written

4.15 ICSS_TRANSF_ORB

4.15.1 Purpose

ICSS_TRANSF_ORB transforms orbit coordinates or any vectorized quantity.

4.15.2 Description

This routine transforms orbit position data or any vectorized quantity to and from one of the following coordinate systems:

- Geocentric Solar Ecliptic (GSE)
- Geocentric Solar Magnetospheric (GSM)
- Geocentric Celestial Inertial (GCI)
- Geographic (GEO)

The data and the desired coordinate transformation code are passed to the routine from the KPGS, and the transformed orbit (or vector) data are returned. Any errors in performing the transformation are reported to the KPGS in the return status.

Note that the KPGS programs that use the ICSS_TRANSF_ORB routine must be linked with the NAG double-precision math library. (See Section 2.4.1.3.3.)

4.15.3 Interfaces

Input to ICSS_TRANSF_ORB from the KPGS:

ORB_SRC_SYS	The current coordinate system of the vector to be transformed
ORB_TARGET_SYS	The requested coordinate system for the transformed

orbit vector

ORB_POS	Orbit position or other vector to be transformed
ORB_POS_TIME	Time of orbit vector to be transformed

Output from ICSS_TRANSF_ORB to the KPGS:

TRANS_ORB_POS	The transformed orbit or other vector
ROTATION_MATRIX	The transformation rotation matrix
TRANS_ORB_STAT	The status of the coordinate transformation

The relationship between the nine-element single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX(1) = Position (1,1) [row, column]
ROTATION_MATRIX(2) = Position (2,1)
ROTATION_MATRIX(3) = Position (3,1)
ROTATION_MATRIX(4) = Position (1,2)
ROTATION_MATRIX(5) = Position (2,2)
ROTATION_MATRIX(6) = Position (3,2)
ROTATION_MATRIX(7) = Position (1,3)
ROTATION_MATRIX(8) = Position (2,3)
ROTATION_MATRIX(9) = Position (3,3)

4.15.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TRANSF_ORB (ORB_SRC_SYS, ORB_TARGET_SYS, ORB_POS,
        ORB_POS_TIME, TRANS_ORB_POS, ROTATION_MATRIX,
        TRANS_ORB_STAT)
CHARACTER*3     ORB_SRC_SYS          !GSE, GSM, GCI
CHARACTER*3     ORB_TARGET_SYS       !GSE, GSM, GCI, GEO
REAL*8          ORB_POS(3)           !Source position vector (x,y,z)
INTEGER*4       ORB_POS_TIME(2)      !(1) YYYYDDD
                                     !(2) milliseconds of day
REAL*8          TRANS_ORB_POS(3)  !Transformed position vector (x,y,z)
REAL*8          ROTATION_MATRIX(9)   !The transformation rotation
matrix
INTEGER*4       TRANS_ORB_STAT       !Corresponds to message
```

Note: From FORTRAN, ICSS_TRANSF_ORB may be called using a 3-by-3 array for the rotation matrix.

For C:

```
ICSS_TRANSF_ORB (&ORB_SRC_SYS, &ORB_TARGET_SYS, &ORB_POS,
        &ORB_POS_TIME, &TRANS_ORB_POS, &ROTATION_MATRIX,
        &TRANS_ORB_STAT);
auto    $DESCRIPTOR(ORB_SRC_SYS, "GSE");          /*GSE, GSM,
GCI*/
auto    $DESCRIPTOR(ORB_TARGET_SYS, "GSE");       /*GSE, GSM,
GCI,
                                                    GEO*/
double  ORB_POS[3];                               /*Position
vector*/
int     ORB_POS_TIME[2];                          /*(0)
YYYYDDD*/
```

```
                                                                    /*(1)
milliseconds of
                                                                      day*/
double      TRANS_ORB_POS[3];                   /*Transformed
                                                                    position
vector
                                                                    (x,y,z)*/
double      ROTATION_MATRIX[9];                 /*Rotation
matrix*/
int         TRANS_ORB_STAT;                     /*Corresponds to
                                                                    message*/
```

4.15.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_INV_SRC_SYS	Invalid source coordinate system
ICSS_INV_SRC_TARGET_SYS	Invalid source and target coordinate systems
ICSS_INV_TARGET_SYS	Invalid target coordinate system
ICSS_SLP_LUN	Error getting a logical unit number for the Solar Lunar Planetary (SLP) file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_ERR_READ_SLP	Error reading from the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the timing coefficients file
ICSS_EREAD_TIMCOEF_FILE	Error reading from the timing coefficients file

The following informational error may be returned to the caller, and output may be returned for the requested coordinate system:

ICSS_SRC_EQ_TARGET	Source and target coordinate systems are identical

The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application should terminate processing.

4.16 ICSS_TRANSF_ATT

4.16.1 Purpose

ICSS_TRANSF_ATT transforms spin-stabilized attitude coordinates.

4.16.2 Description

This routine transforms spin-stabilized attitude data to and from one of the following coordinate systems: GSE, GSM, or GCI.
The attitude data and the desired coordinate transformation code are passed to the routine from the KPGS, and the transformed attitude data are returned. Any errors in performing the transformation are reported to the KPGS in the return status.
Note that KPGS programs that use the ICSS_TRANSF_ATT routine must be linked with the NAG double-precision math library. (See Section 2.4.1.3.3.)

4.16.3 Interfaces

Input to ICSS_TRANSF_ATT from the KPGS:

ATT_SRC_SYS	The current coordinate system of the vector to be transformed
ATT_TARGET_SYS	The requested coordinate system for the transformed attitude vector
ATT_POS	Attitude position vector to be transformed
ATT_POS_TIME	Time of attitude vector to be transformed

Output from ICSS_TRANSF_ATT to the KPGS:

TRANS_ATT_POS	The transformed attitude vector
ROTATION_MATRIX	The transformation rotation matrix
TRANS_ATT_STAT	The status of the coordinate transformation

The relationship between the nine-element single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX(1) = Position (1,1) [row, column]
ROTATION_MATRIX(2) = Position (2,1)
ROTATION_MATRIX(3) = Position (3,1)
ROTATION_MATRIX(4) = Position (1,2)
ROTATION_MATRIX(5) = Position (2,2)
ROTATION_MATRIX(6) = Position (3,2)
ROTATION_MATRIX(7) = Position (1,3)
ROTATION_MATRIX(8) = Position (2,3)
ROTATION_MATRIX(9) = Position (3,3)

4.16.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TRANSF_ATT (ATT_SRC_SYS, ATT_TARGET_SYS, ATT_POS,
        ATT_POS_TIME, TRANS_ATT_POS, ROTATION_MATRIX,
        TRANS_ATT_STAT)
    CHARACTER*3     ATT_SRC_SYS          !GSE, GSM, GCI
    CHARACTER*3     ATT_TARGET_SYS       !GSE, GSM, GCI
    REAL*8          ATT_POS(3)           !(1) Spin rate in rpm
                                         !(2) Right ascension in radians
                                         !(3) Declination in radians
    INTEGER*4       ATT_POS_TIME(2)      !(1) YYYYDDD
                                         !(2) milliseconds of day
    REAL*8          TRANS_ATT_POS(3)  !Transformed position vector that
                                         !indicates array elements
                                         !(1) Spin rate in rpm
                                         !(2) Right ascension in radians
                                         !(3) Declination in radians
    REAL*8          ROTATION_MATRIX(9)   !The transformation rotation
    matrix
    INTEGER*4       TRANS_ATT_STAT       !Corresponds to message
```

Note: From FORTRAN, ICSS_TRANSF_ATT may be called using a 3-by-3 array for the rotation matrix.

For C:

```
ICSS_TRANSF_ATT (&ATT_SRC_SYS, &ATT_TARGET_SYS, &ATT_POS,
        &ATT_POS_TIME, &TRANS_ATT_POS, &ROTATION_MATRIX,
        &TRANS_ATT_STAT);
    auto    $DESCRIPTOR(ATT_SRC_SYS, "GSE");          /*GSE, GSM, GCI*/
    auto    $DESCRIPTOR(ATT_TARGET_SYS, "GSE"); /*GSE, GSM, GCI*/
    double  ATT_POS[3];                               /*Spin rate, right
    ascension,
                                                        and declination*/
```

```
int     ATT_POS_TIME[2];                    /*(0) YYYYDDD
                                              (1) milliseconds of
day*/
double  TRANS_ATT_POS[3];                   /*Transformed position
                                              vector that indicates
array
                                              elements: spin rate,
right
                                              ascension, and
                                              declination*/
double  ROTATION_MATRIX[9];                 /*Rotation matrix*/
int     TRANS_ATT_STAT;                     /*Corresponds to message*/
```

4.16.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_INV_SRC_SYS	Invalid source coordinate system
ICSS_INV_SRC_TARGET_SYS	Invalid source and target coordinate systems
ICSS_INV_TARGET SYS	Invalid target coordinate system
ICSS_SLP_LUN	Error getting a logical unit number for the SLP file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_ERR_READ_SLP	Error reading from the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the timing coefficients file
ICSS_EREAD_TIMCOEF_FILE	Error reading from the timing coefficients file

The following informational error may be returned to the caller, and output may be returned for the requested coordinate system:

ICSS_SRC_EQ_TARGET	Source and target coordinate systems are identical

The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application should terminate processing.

4.17 ICSS_CNVRT_TO_EPOCH

4.17.1 Purpose

This routine converts level-zero major frame time to CDF epoch format.

4.17.2 Description

This routine returns the equivalent epoch time with a given year, day of year, milliseconds, and microseconds.

4.17.3 Interfaces

Input to ICSS_CNVRT_TO_EPOCH from the KPGS:

YEAR	Year
DAYOFYEAR	Day of year
MILLISEC	milliseconds
MICROSEC	Microseconds

Output from ICSS_CNVRT_TO_EPOCH to the KPGS:

EPOCH	Time in CDF epoch format
STATUS	Status of the time reformatting

4.17.4 Calling Sequence

For FORTRAN:

```
Call ICSS_CNVRT_TO_EPOCH (YEAR, DAYOFYEAR, MILLISEC, MICROSEC,
        EPOCH,
    STATUS)
    INTEGER*4       YEAR        !Year
    INTEGER*4       DAYOFYEAR   !Day of year
    INTEGER*4       MILLISEC    !milliseconds
    INTEGER*4       MICROSEC    !Microseconds
    REAL*8          EPOCH       !CDF epoch format
    INTEGER*4       STATUS      !Status
```

For C:

```
ICSS_CNVRT_TO_EPOCH (&YEAR, &DAYOFYEAR, &MILLISEC, &MICROSEC,
        &EPOCH, &STATUS);
    int         YEAR;
    int         DAYOFYEAR;
    int         MILLISEC;
    int         MICROSEC;
    double      EPOCH;
    int         STATUS;
```

4.17.5 Error Conditions

The following error may be recorded in the system message log by ICSS_CNVRT_TO_EPOCH:

ICSS_ERR_CON_EPOCH	There is an error converting to epoch time standard

This is a fatal error. The KPGS application must check the return status immediately after the call to ICSS_CNVRT_TO_EPOCH. If the status indicates an error, the KPGS application must terminate processing.

4.18 ICSS_GET_CD

4.18.1 Purpose

ICSS_GET_CD returns the physical filename of the optional calibration file that is associated with the current version of the KPGS.

4.18.2 Description

This support routine queries the CDHF database for a calibration file, using the process program name and the instrument component (see Section 2.2.2) specified as an argument to the routine. The filename returned is the one listed in the database for the date contained in the primary level-zero data filename.

4.18.3 Interfaces

Input to ICSS_GET_CD from the KPGS:
The level-zero filename initialized by the ICSS_KPG_INIT support routine is used to derive the logical filename of the calibration file.

CD_INSTRUMENT_COMP name	The instrument component of the calibration file, the of which is being retrieved; unique calibration filename used to differentiate multiple calibration files for the instrument (e.g., C001 or C002)

Output from ICSS_GET_CD to the KPGS:

CD_FILENAME	The physical filename of the calibration file
CD_STATUS	The return status from ICSS_GET_CD

4.18.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GET_CD (CD_INSTRUMENT_COMP, CD_FILENAME, CD_STATUS)
    CHARACTER*4     CD_INSTRUMENT_COMP    !Logical file extension
    CHARACTER*44    CD_FILENAME           !Physical filename
    INTEGER*4       CD_STATUS             !Completion status
```

For C:

```
ICSS_GET_CD (&CD_INSTRUMENT_COMP, &CD_DESC, &CD_STATUS);
    auto    $DESCRIPTOR (CD_INSTRUMENT_COMP, "EPI1");
    char    CD_FILENAME [44];
    auto    $DESCRIPTOR (CD_DESC, CD_FILENAME);
    int     CD_STATUS;
```

Note: CD_FILENAME is returned as a nonterminated string. For C programs, a null character must be inserted into the last character of the string before it can be used in a function call to open the file.

4.18.5 Error Conditions

The following error may be recorded in the system message log by ICSS_GET_CD:

ICSS_NO_CD_FILE	There is no calibration file for this mission and instrument with the specified instrument component or for this time period

This error is fatal. The KPGS application must check the return status immediately after the call to ICSS_GET_CD. If the status indicates an error, the KPGS application must terminate processing.

4.19 ICSS_GET_PF

4.19.1 Purpose

ICSS_GET_PF returns the physical filename of the optional PI parameter file that is associated with the current version of the KPGS.

4.19.2 Description

This support routine queries the CDHF database for a PI parameter file using the processing program name. The filename returned is the one listed in the database for the date contained in the primary level-zero data filename.

4.19.3 Interfaces

Input to ICSS_GET_PF from the KPGS:

Level-zero filename initialized by the ICSS_KPG_INIT support routine

Output from ICSS_GET_CD to the KPGS:

PF_FILENAME	The physical filename of the PI parameter file
PF_STATUS	The return status from ICSS_GET_PF

4.19.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GET_PF (PF_FILENAME, PF_STATUS)
    CHARACTER*44    PF_FILENAME     !Physical filename
    INTEGER*4       PF_STATUS       !Completion status
```

For C:

```
ICSS_GET_PF (&PF_DESC, &PF_STATUS);
```

```
char        PF_FILENAME [44];
auto        $DESCRIPTOR (PF_DESC, PF_FILENAME);
int         PF_STATUS;
```

Note: PF_FILENAME is returned as a nonterminated string. For C programs, a null character must be inserted into the last character of the string before it can be used in a function call to open the file.

4.19.5 Error Conditions

The following error may be recorded in the system message log by ICSS_GET_PF:

ICSS_NO_PF_FILE There is no PI parameter file for this mission and instrument

This error is fatal. The KPGS application must check the return status immediately after the call to ICSS_GET_PF. If the status indicates an error, the KPGS application must terminate processing.

4.20 ICSS_SPINPH_SIRIUS (GEOTAIL Support Only)

4.20.1 Purpose

ICSS_SPINPH_SIRIUS returns the spin-phase angle of the spacecraft.

4.20.2 Description

This support routine calculates the spin-phase angle of the spacecraft using SIRIUS and attitude data. Before calling this support routine, the attitude files must have been opened prior to the first call to this unit. Furthermore, the block of SIRIUS data that is used for calculation in this routine must have been retrieved by ICSS_RET_SD or ICSS_RET_SD64.

4.20.3 Interfaces

Input to ICSS_SPINPH_SIRIUS from the KPGS:

TIME An array specifying the time for which the spin phase is to be calculated

TOLER Time tolerance in milliseconds for which the search is performed

SIRIUS_DATA_BLOCK The SIRIUS data block used for computing the spin phase

SIRIUS_SIZE The size of the SIRIUS data block received in frames

Output from ICSS_SPINPH_SIRIUS to the KPGS:

SPIN_PHASE Calculated spin-phase angle of the spacecraft

SPIN_RATE Calculated spin rate in rpm

STATUS Fault indicator status returned:

0 = Data returned is reliable.

1 = First data point in data block was invalid; second value was valid.

2 = First and second values were invalid; third or fourth value was valid.

3 = No spin values matched the nominal spin rate. Spin phase set to fill value.

4 = No data were present in data block for requested time. Spin phase set to fill value.

4.20.4 Calling Sequence

For FORTRAN:

```
Call ICSS_SPINPH_SIRIUS (TIME, TOLER, SIRIUS_DATA_BLOCK, SIRIUS_SIZE,
        SPIN_PHASE, SPIN_RATE, STATUS)
    INTEGER*4   TIME(2)                         !(1) YYYYDDD
                                                !(2) Millisecond of day
    INTEGER*4   TOLER                           !milliseconds
    BYTE        SIRIUS_DATA_BLOCK(23040)        !For full SIRIUS data block
                                or
                SIRIUS_DATA_BLOCK(9216) !For 64-frame SIRIUS data block
    INTEGER*4   SIRIUS_SIZE
    REAL*4      SPIN_PHASE                      !Angle in radians
    REAL*4      SPIN_RATE                       !In rpm
    INTEGER*4   STATUS                          !Returned status
```

For C:

```
ICSS_SPINPH_SIRIUS (&TIME, &TOLER, &SIRIUS_DATA_BLOCK, &SIRIUS_SIZE,
        &SPIN_PHASE, &SPIN_RATE, &STATUS);
    int         TIME[2];
    int         TOLER;
    int         SIRIUS_DATA_BLOCK(5760);        /*For full SIRIUS data block*/
                                or
                SIRIUS_DATA_BLOCK(2304);        /*For 64-frame SIRIUS data
    block*/
    int         SIRIUS_SIZE;
    float       SPIN_PHASE;             /*Angle in radians*/
    float       SPIN_RATE;                      /*In rpm*/
    int         STATUS;
```

4.20.5 Error Conditions

This support routine has no error paths.

4.21 ICSS_GET_REFERENCE_FILES (IMP-8 Support Only)

4.21.1 Purpose

ICSS_GET_REFERENCE_FILES returns the physical filename(s), span start/stop date(s), and the number of input filenames when KPGS opens and reads telemetry files directly.

4.21.2 Description

This support routine queries the CDHF database for the physical filename(s) and span start/stop date(s) for the input telemetry files. The filename(s) and date(s) returned are the ones listed in the reference section of the SFDU header.

4.21.3 Interfaces

Input to ICSS_GET_REFERENCE_FILES from the KPGS:

Level-zero filename initialized by the ICSS_KPG_INIT support routine

Output from ICSS_GET_REFERENCE_FILES to the KPGS:

REF_FILES	The physical filename(s) and span start/stop date(s) of the level-zero logical filename
NUM_REF_FILES	The number of physical filenames returned
RETURN_STATUS	The return status from ICSS_GET_REFERENCE_FILES

4.21.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GET_REFERENCE_FILES (REF_FILES, NUM_REF_FILES,
        RETURN_STATUS)
    CHARACTER*44    REF_FILES (3,100)   !Physical filename(s) and span start/stop
                                        !date
    INTEGER*4       NUM_REF_FILES       !Number of filenames returned from
                                        !database
    INTEGER*4       RETURN_STATUS       !Completion status
```

For C:

```
ICSS_GET_REFERENCE_FILES (&REF_FILES, &NUM_REF_FILES,
        &RETURN_STATUS);
    char        REF_FILES[100] [3] [44];
    auto        $DESCRIPTOR (FILE_DESC, REF_FILES);
    int         NUM_REF_FILES;
    int         RETURN_STATUS;
```

Note: The above status values are not defined in ICSS_MESSAGE.INC and must be declared externally to be used.

4.21.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_GET_REFERENCE_FILES:

X_NO_REFERENCE	No reference filenames/dates could be obtained from the database
X_DBA_E_MSG	Database error occurred when obtaining reference filenames/dates

These are fatal errors. The KPGS application must check the return status immediately after the call to ICSS_GET_REFERENCE_FILES. If the status indicates an error, the KPGS application should terminate processing if it is determined that no further processing can be performed.

4.22 ICSS_SPINPH_WIND_LZ (WIND Support Only)

4.22.1 Purpose

ICSS_SPINPH_WIND_LZ returns the spin-phase angle at a user input time for the WIND spacecraft.

4.22.2 Description

This support routine calculates the spin-phase angle of the WIND spacecraft using housekeeping data. Before calling this support routine, the block of housekeeping data that will be used for calculation in this routine must have been retrieved by ICSS_RET_HK.

4.22.3 Interfaces

Input to ICSS_SPINPH_WIND_LZ from the KPGS:

TIME	An array specifying the time for which the spin phase is to be calculated
HK_REC	Housekeeping data record containing the time for which the spin-phase calculation is desired

Output from ICSS_SPINPH_WIND_LZ to the KPGS:

SPIN_PHASE	Calculated spin-phase angle of the spacecraft
AVG_SPIN_RATE	Calculated average spin rate for the entire housekeeping data record
STNDEV_SPIN_RATE	Calculated standard deviation of the spin rate for the entire housekeeping data record
NUM_PNT	Number of points used to compute the average and standard deviation of the spin rate

INDICATOR	Fault-level status indicator: 0 = No error 1 = Invalid telemetry mode in housekeeping data record, fill values for spin phase, average spin rate, and standard deviation of spin rate 2 = User requested time not in housekeeping data record, fill values returned 3 = Cannot compute accurate spin rate and/or standard deviation of spin rate, fill values returned
RETURN_STATUS	The return status from ICSS_SPINPH_WIND_LZ

4.22.4 Calling Sequence

For FORTRAN:

```
Call ICSS_SPINPH_WIND_LZ (TIME, HK_REC, SPIN_PHASE, AVG_SPIN_RATE,
        STNDEV_SPIN_RATE, NUM_PNT, INDICATOR, RETURN_STATUS)
    INTEGER*4   TIME(2)                 !(1) YYYYDDD
                                        !(2) milliseconds of day
    INTEGER*4   HK_REC(REC_SIZE)        !REC_SIZE should be replaced with
                                        !maximum number of words in the
                                        !housekeeping record
    REAL*4      SPIN_PHASE              !In radians
    REAL*4      AVG_SPIN_RATE           !In rad/sec
    REAL*4      STNDEV_SPIN_RATE        !In rad/sec
    INTEGER*4   NUM_PNT
    INTEGER*4   INDICATOR
    INTEGER*4   RETURN_STATUS
```

For C:

```
ICSS_SPINPH_WIND_LZ (&TIME, &HK_REC, &SPIN_PHASE, &AVG_SPIN_RATE,
        &STNDEV_SPIN_RATE, &NUM_PNT, &INDICATOR,
        &RETURN_STATUS);
    int     TIME[2];                    /*(1) YYYYDDD*/
                                        /*(2) milliseconds of day*/
    int     HK_REC[REC_SIZE];  /*REC_SIZE should be replaced with number of
                                 words in the housekeeping data record*/
    float   SPIN_PHASE;         /*In radians*/
    float   AVG_SPIN_RATE;      /*In rad/sec*/
    float   STNDEV_SPIN_RATE;   /*In rad/sec*/
    int     NUM_PNT;
    int     INDICATOR;
    int     RETURN_STATUS;
```

4.22.5 Error Conditions

The following warning condition may be detected by this subroutine:

ICSS_INVALID_MODE	Invalid telemetry mode found in WIND housekeeping data record header

The following fatal errors may be recorded in the system message log by ICSS_SPINPH_WIND_LZ:

ICSS_AVG_SPIN_RATE_ERR	Calculated average spin rate housekeeping data record is equal to zero
ICSS_REQ_TIME_ERR	User-requested time not in WIND housekeeping data record
ICSS_SPIN_RATE_ERR	Cannot compute accurate spin rate and/or standard deviation of spin rate for WIND housekeeping data record

The KPGS application must check the return status immediately after the call to ICSS_SPINPH_WIND_LZ. If the status indicates an error, the KPGS application can discontinue processing, extrapolate from previously known values, or reset the status to normal and continue processing.

4.23 ICSS_PAYLOAD_TO_GSE

4.23.1 Purpose

ICSS_PAYLOAD_TO_GSE transforms nonspinning payload coordinates.

4.23.2 Description

This support routine transforms nonspinning payload coordinates to GSE coordinates. The payload coordinates and GSE attitude angles are passed to the routine from the KPGS, and the transformation rotation matrixes are returned.

This routine uses PAYLOAD_POS coordinates to determine the GSE_POS. It returns the GSE_POS or the GSE_POS with the SC_VEL vector added in, as a function of whether or not the PAYLOAD_POS coordinates are position coordinates or velocity coordinates.

The user is responsible for knowing whether PAYLOAD_POS is a position or a rate. If the PAYLOAD_POS is a position, the routine is called with the VEL_FLAG set to zero. If the PAYLOAD position is a rate, the routine is called with the VEL_FLAG set to a nonzero value and the SC_VEL set to a GSE spacecraft velocity value. In either case the ROTATION_MATRIX is not affected.

Note that KPGS programs that use the ICSS_PAYLOAD_TO_GSE routine must be linked with the NAG double-precision math library. (See Section 2.4.1.3.3.)

4.23.3 Interfaces

Input to ICSS_PAYLOAD_TO_GSE from the KPGS:

PAYLOAD_POS	The payload coordinates to be transformed
GSE_ATT_POS	GSE attitude angles (right ascension and declination) at the epoch time of the spacecraft coordinates

Output from ICSS_PAYLOAD_TO_GSE to the KPGS:

ROTATION_MATRIX	The transformation rotation matrix
GSE_POS	The transformed GSE coordinates

The relationship between the nine-element single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX (1) = Position (1,1) [row, column]
ROTATION_MATRIX (2) = Position (2,1)
ROTATION_MATRIX (3) = Position (3,1)
ROTATION_MATRIX (4) = Position (1,2)
ROTATION_MATRIX (5) = Position (2,2)
ROTATION_MATRIX (6) = Position (3,2)
ROTATION_MATRIX (7) = Position (1,3)
ROTATION_MATRIX (8) = Position (2,3)
ROTATION_MATRIX (9) = Position (3,3)

4.23.4 Calling Sequence

For FORTRAN:

```
Call ICSS_PAYLOAD_TO_GSE (PAYLOAD_POS, GSE_ATT_POS,
        ROTATION_MATRIX, GSE_POS, VEL_FLAG, SC_VEL)
    REAL*8          PAYLOAD_POS (3)     !The payload coordinates
    (x,y,z) to
                                        !be transformed
    REAL*4          GSE_ATT_POS (2)     !(1) Right ascension in radians
                                        !(2) Declination in radians
    REAL*8          ROTATION_MATRIX (9) !The transformation rotation
    matrix
    REAL*8          GSE_POS (3)         !The transformed GSE
    coordinates
    INTEGER*4       VEL_FLAG            !Flag to show whether the
    quality
                                        !being transformed is a velocity.
                                        !Value = 0 is not a velocity; do
    not
                                        !add translation vector. Value >
    0 is
                                        !a velocity; add in spacecraft
                                        !velocity vector
    REAL*8          SC_VEL(3)           !Spacecraft velocity in GSE
                                        !coordinate; used when
    VEL_FLAG
                                        !is nonzero
```

For C:

```
ICSS_PAYLOAD_TO_GSE (&PAYLOAD_POS, &GSE_ATT_POS,
        &ROTATION_MATRIX, &GSE_POS, &VEL_FLAG, &SC_VEL);
    double      PAYLOAD_POS[3];
    float       GSE_ATT_POS[2];
    double      ROTATION_MATRIX[9];
    double      GSE_POS[3];
    int         VEL_FLAG;
    double      SC_VEL[3];
```

4.23.5 Error Conditions

This support routine has no error paths.

4.24 ICSS_CNVT_FROM_RP

4.24.1 Purpose

ICSS_CNVT_FROM_RP converts a vector in the spacecraft rotating coordinate frame to a vector in the fixed payload coordinate frame.

4.24.2 Description

Given the spin-phase angle and a vector in the spacecraft rotating coordinate frame, ICSS_CNVT_FROM_RP converts the vector to the fixed payload coordinate frame.

4.24.3 Interfaces

Input to ICSS_CNVT_FROM_RP from the KPGS:

SPIN_PHASE The spin-phase angle in radians
SC_VECTOR The XYZ vector in the spacecraft rotating coordinate frame

Output from ICSS_CNVT_FROM_RP to the KPGS:

PAYLOAD_VECTOR The XYZ vector in the fixed payload coordinate frame
MATRIX The transformation matrix used in converting the vector

The order of the nine elements of the (MATRIX) array is as follows: (1,1), (2,1), (3,1), (1,2), (2,2), (3,2), (1,3), (2,3), and (3,3).

4.24.4 Calling Sequence

For FORTRAN:

```
Call ICSS_CNVT_FROM_RP (SPIN_PHASE, SC_VECTOR, PAYLOAD_VECTOR,
        MATRIX)
  REAL*4    SPIN_PHASE       !The spin-phase angle in radians
  REAL*4    SC_VECTOR(3)     !The XYZ vector in spacecraft rotating
                             !coordinate
```

```
      REAL*4      PAYLOAD_VECTOR(3)     !The XYZ vector in fixed payload
                                        !coordinate
      REAL*8      MATRIX(9)             !The transformation matrix
```

For C:

```
   ICSS_CNVT_FROM_RP (&SPIN_PHASE, &SC_VECTOR, &PAYLOAD_VECTOR,
         &MATRIX);
      float       SPIN_PHASE;
      float       SC_VECTOR[3];
      float       PAYLOAD_VECTOR[3];
      double      MATRIX[9];
```

4.24.5 Error Conditions

This routine has no error handling.

4.25 ICSS_POS_OF_SUN

4.25.1 Purpose

ICSS_POS_OF_SUN returns the Sun position and velocity vectors.

4.25.2 Description

This routine returns the Sun position and velocity vectors that correspond to a given request time.

4.25.3 Interfaces

Input to ICSS_POS_OF_SUN from the KPGS:

SUN_VEC_REQ_DATE	The day of year and time of day of the requested Sun vectors

Output from ICSS_POS_OF_SUN to the KPGS:

SUN_POS_VEC	The Sun position vector that corresponds to the request time
SUN_VEL_VEC	The Sun velocity vector that corresponds to the request time
SUN_RET_STATUS	The status of the Sun vector request

4.25.4 Calling Sequence

For FORTRAN:

```
   Call ICSS_POS_OF_SUN (SUN_REQ_DATE, SUN_POS_VEC, SUN_VEL_VEC,
         SUN_RET_STATUS)
      INTEGER*4   SUN_REQ_DATE(2)       !(1) YYYYDDD
                                        !(2) milliseconds of day
      REAL*8      SUN_POS_VEC(3)        !Sun position vector (x,y,z)(km)
      REAL*8      SUN_VEL_VEC(3)        !Sun velocity vector (dx,dy,dz)(km/sec)
      INTEGER*4   SUN_RET_STATUS        !Corresponds to message
```

For C:

```
   ICSS_POS_OF_SUN (&SUN_REQ_DATE, &SUN_POS_VEC, &SUN_VEL_VEC,
         &SUN_RET_STATUS);
      int         SUN_REQ_DATE[2];
      double      SUN_POS_VEC[3];
      double      SUN_VEL_VEC[3];
      int         SUN_RET_STATUS;
```

4.25.5 Error Conditions

The following fatal errors may be recorded in the system message log by ICSS_POS_OF_SUN:

ICSS_SLP_LUN	Error getting a logical unit number for the SLP file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_ERR_READ_SLP	Error reading from the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the timing coefficients file
ICSS_EREAD_TIMCOEF_FILE	Error reading from the timing coefficients file

The KPGS application must check the return status immediately after the call to ICSS_POS_OF_SUN. If the status indicates an error, the KPGS application must terminate processing.

The following warning condition may be detected by this subroutine:

ICSS_TIME_OUTRANGE	Requested time out of range of files

4.26 ICSS_VELOCITY_TRANS

4.26.1 Purpose

ICSS_VELOCITY_TRANS converts velocities from one coordinate system to another.

4.26.2 Description

The vectors in GCI, GSE, or GSM coordinates are accepted and transformed into the user's choice of GCI, GSE, GSM, or GEO coordinates.

4.26.3 Interfaces

Input to ICSS_VELOCITY_TRANS from the KPGS:

ORB_SRC_SYS	The current coordinate system of the vector to be transformed
ORB_TARGET_SYS	The requested coordinate system for the transformed orbit vector
ORB_POS_VEL	Orbit vectors (position and velocity) to be transformed

ORB_POS_TIME Time of vector to be transformed
VELOCITY_REQ A flag showing that velocities are to be transformed

Output from ICSS_VELOCITY_TRANS to the KPGS:

TRANS_ORB_POS_VEL Transformed orbit vector
ROTATION_MATRIX Rotation matrix
TRANS_ORB_STAT The status of the coordinate transformation

The relationship between the nine-element, single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 dimensional transformation rotation matrix is described in Section 4.15.3.

4.26.4 Calling Sequence

For FORTRAN:

```
Call ICSS_VELOCITY_TRANS (ORB_SRC_SYS, ORB_TARGET_SYS,
        ORB_POS_VEL, ORB_POS_TIME, TRANS_ORB_POS_VEL,
        ROTATION_MATRIX, VELOCITY_REQ, TRANS_ORB_STAT)
    CHARACTER*3     ORB_SRC_SYS          !GSE, GSM, GCI
    CHARACTER*3     ORB_TARGET_SYS       !GSE, GSM, GCI, GEO
    REAL*8          ORB_POS_VEL(6)       !Position and velocity vectors
                                         !(x-pos, y-pos, z-pos, x-vel,
                                         !y-vel, and z-vel)
    INTEGER*4       ORB_POS_TIME(2)      !Time of orbit vector
                                         !(1) YYYYDDD
                                         !(2) milliseconds of day
    REAL*8          TRANS_ORB_POS_VEL(6)
    REAL*8          ROTATION_MATRIX(9)   !The transformation rotation
                                         !matrix
    INTEGER*4       VELOCITY_REQ         !Flag to show if velocity
                                         !transformations are needed
    INTEGER*4       TRANS_ORB_STAT       !Corresponding to message
                                         !0 = Velocity computations are
                                         !not done
                                         !1 = Velocity computations are
    done
```

For C:

```
ICSS_VELOCITY_TRANS (&ORB_SRC_SYS, &ORB_TARGET_SYS,
        &ORB_POS_VEL, &ORB_POS_TIME, &TRANS_ORB_POS_VEL,
        &ROTATION_MATRIX, &VELOCITY_REQ, &TRANS_ORB_STAT);
    auto        $DESCRIPTOR(ORB_SRC_SYS, "GSE");
    auto        $DESCRIPTOR(ORB_TARGET_SYS, "GSE");
    double      ORB_POS_VEL[6];
    int         ORB_POS_TIME[2];
    double      TRANS_ORB_POS_VEL[6];
    double      ROTATION_MATRIX[9];
    int         VELOCITY_REQ;
    int         TRANS_ORB_STAT;
```

4.26.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_INV_SRC_SYS Invalid source coordinate system
ICSS_INV_SRC_TARGET_SYS Invalid source and target coordinate systems
ICSS_INV_TARGET_SYS Invalid target coordinate system
ICSS_SLP_LUN Error getting a logical unit number for the SLP file

ICSS_OPEN_SLP	Error opening the SLP file
ICSS_ERR_READ_SLP	Error reading from the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for timing coefficients file
ICSS_OPEN_TIMCOEF	Error opening the timing coefficients file
ICSS_EREAD_TIMCOEF_FILE	Error reading from the timing coefficients file

The following informational error may be returned to the caller, and output may be returned for the requested coordinate system:

ICSS_SRC_EQ_TARGET	Source and target coordinate systems are identical

The KPGS application must check the return status immediately after every subroutine call. If the status indicates an error, the KPGS application should terminate processing.

4.27 ICSS_GCI_TO_GEODETIC

4.27.1 Purpose

ICSS_GCI_TO_GEODETIC converts an input GCI vector to geodetic longitude, latitude, and height.

4.27.2 Description

This routine converts a mean-of-date GCI vector to true-of-date geodetic coordinates.

4.27.3 Interfaces

Input to ICSS_GCI_TO_GEODETIC from the KPGS:

idtfTime	Time for the calculation in IDTF format
gciVector	The mean-of-date GCI vector in kilometers

Output from ICSS_GCI_TO_GEODETIC to the KPGS:

latitude	True-of-date geodetic latitude in radians
longitude	True-of-date geodetic longitude in radians
height	True-of-date geodetic height in kilometers

4.27.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GCI_TO_GEODETIC (idtfTime, gciVector, latitude, longitude, height)
    INTEGER*4   idtfTime(2)          !(1) YYYYDDD
                                     !(2) milliseconds of day
    REAL*8      gciVector(3)         !x,y,z
    REAL*8      latitude, longitude, height
```

For C:

```
ICSS_GCI_TO_GEODETIC (&idtfTime, &gciVector, &latitude, &longitude, &height);
    int         idtfTime[2];
    double      gciVector[3];
    double      latitude, longitude, height;
```

4.27.5 Error Conditions

If iteration for the latitude and height fails to converge, the latitude, longitude, and height are returned as zeros.

4.28 ICSS_GEODETIC_TO_GCI

4.28.1 Purpose

ICSS_GEODETIC_TO_GCI converts geodetic coordinates to a vector in GCI coordinates.

4.28.2 Description

This routine converts an input geodetic longitude, latitude, and height to a GCI vector.

4.28.3 Interfaces

Input to ICSS_GEODETIC_TO_GCI from the KPGS:

idtfTime	Time for the calculation in IDTF format
latitude	True-of-date geodetic latitude in radians
longitude	True-of-date geodetic longitude in radians
height	True-of-date geodetic height in kilometers

Output from ICSS_GEODETIC_TO_GCI to the KPGS:

gciVector	The mean-of-date GCI vector in kilometers

4.28.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GEODETIC_TO_GCI (idtfTime, latitude, longitude, height, gciVector)
    INTEGER*4   idtfTime(2)             !(1) YYYYDDD
                                        !(2) milliseconds of day
    REAL*8      latitude, longitude, height
    REAL*8      gciVector(3)            !x,y,z
```

For C:

```
ICSS_GEODETIC_TO_GCI (&idtfTime, &latitude, &longitude, &height, &gciVector);
    int         idtfTime[2];
    double      latitude, longitude, height;
    double      gciVector[3];
```

4.28.5 Error Conditions

This routine has no error handling.

4.29 ICSS_SD_BLK_TYP

4.29.1 Purpose

ICSS_SD_BLK_TYP is used to determine whether a block of SIRIUS data retrieved by the ICSS_RET_SD support routine contained a file message block.

4.29.2 Description

ICSS_SD_BLK_TYP returns a status that indicates whether the last block of SIRIUS data retrieved by the ICSS_RET_SD support routine contained a file message block.

4.29.3 Interfaces

There are no inputs to the ICSS_SD_BLK_TYP routine.

Output from ICSS_SD_BLK_TYP to the KPGS:

SD_RET_STATUS	The status of the last SIRIUS data block retrieved by the ICSS_RET_SD support routine

4.29.4 Calling Sequence

For FORTRAN:

```
Call ICSS_SD_BLK_TYP(SD_RET_STATUS)
    INTEGER*4    SD_RET_STATUS      ! = ICSS_MSG_BLK_RET_SD when last block
                                    !    contained message block
                                    ! = SS$_NORMAL otherwise
```

For C:

```
ICSS_SD_BLK_TYP(&SD_RET_STATUS);
    int          SD_RET_STATUS;
```

4.29.5 Error Conditions

There are no error conditions associated with the ICSS_SD_BLK_TYP support routine.

4.30 ICSS_INDICES

4.30.1 Purpose

ICSS_INDICES returns the observed 10.7-centimeter (cm) flux, 90-day mean 10.7-cm flux, estimated average daily A index value (AP), and estimated KPs for the 3-hour period.

4.30.2 Description

This support routine queries the CDHF database for solar indices using an observation date. The solar indices returned are those listed in the database for the observation date.

4.30.3 Interfaces

Input to ICSS_INDICES from the KPGS:

OBS_DATE	Observation date

Output from the ICSS_INDICES to the KPGS:

OBS_10_7_FLUX	Observed 10.7-cm flux
MEAN_10_7_FLUX	90-day mean 10.7-cm flux
AP_EST	Estimated AP
KP_EST	Estimated KPs for each 3-hour period
STATUS	Return status

Note: A negative 1 is returned from ICSS_INDICES if no data is found for the corresponding output argument. The KPGS application should check each variable individually (i.e., OBS_10_7_FLUX, MEAN_10_7_FLUX, AP_EST, and KP_EST) for negative 1 after calling ICSS_INDICES.

4.30.4 Calling Sequences

For FORTRAN:

```
Call ICSS_INDICES (OBS_DATE, OBS_10_7_FLUX, MEAN_10_7_FLUX, AP_EST,
        KP_EST, STATUS)
    INTEGER*4    OBS_DATE(2) !Observation date
                             !(1) YYYYDDD
                             !(2) milliseconds of day
    INTEGER*4    OBS_10_7_FLUX   !Observed 10.7-cm flux
    INTEGER*4    MEAN_10_7_FLUX  !Mean 10.7-cm flux
    INTEGER*4    OBS_AP          !Estimated daily AP
    INTEGER*4    KP_EST(8)       !Estimated KPs
    INTEGER*4    STATUS          !Status
```

For C:

```
ICSS_INDICES (&OBS_DATE, &OBS_10_7_FLUX, &MEAN_10_7_FLUX, &AP_EST,
        &KP_EST, STATUS);
```

```
int         OBS_DATE[2];
int         OBS_10_7_FLUX;
int         MEAN_10_7_FLUX;
int         OBS_AP;
int         KP_EST[8];
int         STATUS;
```

4.30.5 Error Conditions

The following error may be recorded in the system message log by ICSS_INDICES:

ICSS_DB_ERR Error accessing the database

The KPGS application must check the returned status immediately after the call to ICSS_INDICES. If the status indicates an error, the KPGS application can discontinue processing if the program is unable to do any other processing at this time.

4.31 ICSS_CNVRT_EPOCH_TO_PB5

4.31.1 Purpose

ICSS_CNVRT_EPOCH_TO_PB5 converts a date/time in the epoch format to one in the PB5 format.

4.31.2 Description

This support routine transforms a date/time that is formatted in the epoch format to one that is in the PB5 format.

4.31.3 Interfaces

Input to the ICSS_CNVRT_EPOCH_TO_PB5 from the KPGS:

EPOCH_DATE/TIME An epoch-formatted date/time

Output from the ICSS_CNVRT_EPOCH_TO_PB5 to the KPGS:

PB5_DATE/TIME A PB5-formatted date/time

4.31.4 Calling Sequence

For FORTRAN:

```
Call ICSS_CNVRT_EPOCH_TO_PB5 (EPOCH_TIME, PB5_TIME, RETURN_STATUS)
    REAL*8      EPOCH_TIME !Epoch-formatted date/time
    INTEGER*4   PB5_TIME(3)        !PB5-formatted date/time
                                   !(1) Year
                                   !(2) Day of year
                                   !(3) milliseconds of day
    INTEGER*4   RETURN_STATUS
```

For C:

```
ICSS_CNVRT_EPOCH_TO_PB5 (&EPOCH_TIME, &PB5_TIME,
        &RETURN_STATUS);
    double      EPOCH_TIME;
    int         PB5_TIME[3];
    int         RETURN_STATUS;
```

4.31.5 Error Conditions

The following fatal error may be returned to the caller:

ICSS_ERR_CON_EPOCH Epoch-formatted date/time was invalid

4.32 ICSS_TRANSF_TO_MTC

4.32.1 Purpose

ICSS_TRANSF_TO_MTC transforms orbit coordinates or any vectorized quantity. MTCs are as follows:

- Y along the satellite spin axis (approximately longitudinal)
- X along the projection of X topographic onto the satellite spin plane (approximately northward)
- Z completes the right-handed coordinate system

Note: X topographic is defined as on any spherical surface concentric with the Earth, X is parallel to this surface with positive direction northward.

4.32.2 Description

ICSS_TRANSF_TO_MTC transforms orbit position data or any vectorized quantity to the MTC system from any of the following coordinate systems:

- GSE
- GSM
- GCI

The data is passed to the routine from the KPGS, and the transformed orbit (or vector) data is returned. Any errors in performing the transformation are reported to the KPGS in the return status.
Note that the KPGS programs that use the ICSS_TRANSF_TO_MTC routine must be linked with the NAG double-precision math library (see Section 2.4.1.3.3).

4.32.3 Interfaces

Input to ICSS_TRANSF_TO_MTC from the KPGS:

REFERENCE_FRAME	Current coordinate system of the vector to be transformed
SPIN_AXIS_ATT	Spacecraft spin axis orientation

TIME Time of orbit vector to be transformed
INPUT_VECTOR Orbit position or other vector to be transformed

Output from ICSS_TRANSF_TO_MTC to the KPGS:

OUTPUT_VECTOR The transformed orbit or other vector
TRANSFORM_MATRIX The transformation rotation matrix
RETURN_STATUS The return status from the routine

The relationship between the nine-element single-dimension array returned (TRANSFORM_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX(1) = Position (1,1) [row, column]
ROTATION_MATRIX(2) = Position (2,1)
ROTATION_MATRIX(3) = Position (3,1)
ROTATION_MATRIX(4) = Position (1,2)
ROTATION_MATRIX(5) = Position (2,2)
ROTATION_MATRIX(6) = Position (3,2)
ROTATION_MATRIX(7) = Position (1,3)
ROTATION_MATRIX(8) = Position (2,3)
ROTATION_MATRIX(9) = Position (3,3)

4.32.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TRANSF_TO_MTC (REFERENCE_FRAME, SPIN_AXIS_ATT, TIME,
        INPUT_VECTOR, OUTPUT_VECTOR, TRANSFORM_MATRIX,
        RETURN_STATUS)
    CHARACTER *3    REFERENCE_FRAME !GCI, GSE, GSM
    REAL*4          SPIN_AXIS_ATT(3)    !Spacecraft spin axis orientation
                                        !(1)Right ascension in radians
                                        !(2)Declination in radians
                                        !(3)Spin rate in rpm
    INTEGER*4       TIME(2)             !(1) YYYYDDD
                                        !(2) milliseconds of day
    REAL*8          INPUT_VECTOR(3)     !Source position (x, y, z)
    REAL*8          OUTPUT_VECTOR(3) !Transformed position (x, y, z)
    REAL*8          TRANSFORM_MATRIX(9) !Transformation rotation matrix
    INTEGER*4       RETURN_STATUS       !Status of call
```

Note: From FORTRAN, ICSS_TRANSF_TO_MTC may be called using a 3-by-3 array for the rotation matrix.

For C:

```
ICSS_TRANSF_TO_MTC (&REFERENCE_FRAME,& SPIN_AXIS_ATT, &TIME,
        &INPUT_VECTOR, &OUTPUT_VECTOR, &TRANSFORM_MATRIX,
        &RETURN_STATUS)
    auto        $DESCRIPTOR(REFERENCE_FRAME, “GSE”);    /*GCI, GSE,
    GSM*/
    single      $SPIN_AXIS_ATT[3];              /*Spacecraft spin axis
                                                  orientation*/
    int         TIME[2];                        /*(0)
    YYYYDDD*/
                                                /*(1)
    milliseconds of
    day*/
    double      INPUT_VECTOR[3];                /*Source
    vector*/
    double      OUTPUT_VECTOR[3];               /*Transformed
                                                  vector*/
    double      TRANSFORM_MATRIX[9];            /*Rotation matrix*/
```

```
int RETURN_STATUS; /*Status of call*/
```

Assumption: The same reference frame is used for both the spin axis vector and the input vector.

4.32.5 Error Conditions

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.33 ICSS_TRANSF_TO_MSPC

4.33.1 Purpose

ICSS_TRANSF_TO_MSPC transforms orbit coordinates or any vectorized quantity. MSPC for POLAR is defined as follows:

- Z along the satellite spin vector
- X along the projection of satellite velocity onto the spin plane of the satellite
- Y completes the right-handed coordinate system

4.33.2 Description

ICSS_TRANSF_TO_MSPC transforms orbit position data or any vectorized quantity to the modified SPC coordinate system from any of the following coordinate systems:

- GSE
- GSM
- GCI

The data is passed to the routine from the KPGS, and the transformed orbit (or vector) data is returned. Any errors in performing the transformation are reported to the KPGS in the return status.
Note that the KPGS programs that use the ICSS_TRANSF_TO_MSPC routine must be linked with the NAG double-precision math library (see Section 2.4.1.3.3).

4.33.3 Interfaces

Input to ICSS_TRANSF_TO_MSPC from the KPGS:

REFERENCE_FRAME	Current coordinate system of the vector to be transformed
SPIN_AXIS_ATT	Spacecraft spin axis orientation
VELOCITY_VECTOR	MSPC reference unit vector
TIME	Time of orbit vector to be transformed
INPUT_VECTOR	Orbit position or other vector to be transformed

Output from ICSS_TRANSF_TO_MTC to the KPGS:

OUTPUT_VECTOR	The transformed orbit or other vector
TRANSFORM_MATRIX	The transformation rotation matrix
RETURN_STATUS	The return status from the routine

The relationship between the nine-element single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX(1) = Position (1,1) [row, column]
ROTATION_MATRIX(2) = Position (2,1)
ROTATION_MATRIX(3) = Position (3,1)
ROTATION_MATRIX(4) = Position (1,2)
ROTATION_MATRIX(5) = Position (2,2)
ROTATION_MATRIX(6) = Position (3,2)
ROTATION_MATRIX(7) = Position (1,3)
ROTATION_MATRIX(8) = Position (2,3)
ROTATION_MATRIX(9) = Position (3,3)

4.33.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TRANSF_TO_MSPC (REFERENCE_FRAME, SPIN_AXIS_ATT,
        VELOCITY_VECTOR, TIME, INPUT_VECTOR, OUTPUT_VECTOR,
        TRANSFORM_MATRIX, RETURN_STATUS)
    CHARACTER *3    REFERENCE_FRAME !GCI, GSE, GSM
    REAL*4          SPIN_AXIS_ATT(3)        !Spacecraft spin axis orientation
                                            !(1)Right ascension in radians
                                            !(2)Declination in radians
                                            !(3)Spin rate in rpm
    REAL*8          VELOCITY_VECTOR(3)      !MSPC reference unit velocity
    (x, y,
                                            z)
    INTEGER*4       TIME(2)                 !(1) YYYYDDD
                                            !(2) milliseconds of day
    REAL*8          INPUT_VECTOR(3)         !Source position (x, y, z)
    REAL*8          OUTPUT_VECTOR(3) !Transformed position (x, y, z)
    REAL*8          TRANSFORM_MATRIX(9)     !Transformation rotation matrix
    INTEGER*4       RETURN_STATUS           !Status of call
```

Note: From FORTRAN, ICSS_TRANSF_TO_MSPC may be called using a 3-by-3 array for the rotation matrix.

For C:

```
ICSS_TRANSF_TO_MSPC (&REFERENCE_FRAME, &SPIN_AXIS_ATT,
        &VELOCITY_VECTOR, &TIME, &INPUT_VECTOR, &OUTPUT_VECTOR,
        &TRANSFORM_MATRIX, &RETURN_STATUS)
    auto        $DESCRIPTOR(REFERENCE_FRAME, “GSE”);    /*GCI, GSE,
    GSM*/
    float       $SPIN_AXIS_ATT[3];                  /*Spacecraft spin axis
                                                      orientation*/
    double      VELOCITY_VECTOR[3];                 /*MSPC
    reference
                                                      unit velocity*/
    int         TIME[2];                            /*(0)
    YYYYDDD*/
                                                    /*(1)
    milliseconds of
                                                      day*/
    double      INPUT_VECTOR[3];                    /*Source
    vector*/
    double      OUTPUT_VECTOR[3];                   /*Transformed
                                                      vector*/
    double      TRANSFORM_MATRIX[9];            /*Rotation matrix*/
    int         RETURN_STATUS;                      /*Status of
    call*/
```

Assumption: The same reference frame is used for both the spin axis vector and the input vector.

4.33.5 Error Conditions

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.34 ICSS_GET_FILE

4.34.1 Purpose

ICSS_GET_FILE returns an array of physical filename(s), span start date(s), and span stop date(s).

4.34.2 Description

This support routine queries the CDHF database for physical filename(s), associated span start date(s), and span stop date(s) based on an input mission, datatype, descriptor, and a primary/secondary flag. Based on the input primary/secondary flag, this routine returns a primary or secondary file. This routine also builds the logical file identifier and adds it to the reference section of the SFDU file.

4.34.3 Interfaces

Input to ICSS_GET_FILE from the KPGS:

MISSION_NAME	The mission name
DATA_TYPE	The data type
DESCRIPTOR	The descriptor
PRIM_SEC_FLAG	The primary/secondary flag

Output from the ICSS_GET_FILE to the KPGS:

INPUT_FILES	An array of physical filename(s)
START_DATE	An array of span start date(s)
STOP_DATE	An array of span stop date(s)
NUM_FILES	The number of files listed in input_files (up to 99)
STATUS	Status returned from this routine

4.34.4 Calling Sequences

For FORTRAN:

```
Call ICSS_GET_FILE (MISSION_NAME, DATA_TYPE, DESCRIPTOR,
        PRIM_SEC_FLAG, INPUT_FILES, START_DATE, STOP_DATE,
        NUM_FILES, STATUS)
    CHARACTER*2     MISSION_NAME      !Mission name
    CHARACTER*2     DATA_TYPE         !Data type
    CHARACTER*4     DESCRIPTOR  !Descriptor
    CHARACTER*1     PRIM_SEC_FLAG     !Primary/secondary flag
                                      !P = Primary
                                      !S = Secondary
    CHARACTER*44    INPUT_FILES(99)   !Physical filename
    INTEGER*4       START_DATE(2,99)  !Span start date
    INTEGER*4       STOP_DATE(2,99)   !Span stop date
                                      !Date formats
                                      !(1) YYYYDDD
                                      !(2) milliseconds of day
    INTEGER*4       NUM_FILES         !The number of input files
    INTEGER*4       STATUS            !Return status
```

For C:

```
ICSS_GET_FILE (&MISSION_DESC, &DATA_TYPE_DESC, &DESCRIPTOR_DESC,
        &PRIM_SEC_DESC, &INPUT_FILE_DESC, &START_DATE,
        &STOP_DATE, &NUM_FILES, &STATUS);
    auto    $DESCRIPTOR(MISSION_DESC, "SO");          /*Mission short name*/
    auto    $DESCRIPTOR(DATA_TYPE_DESC, "LZ");   /*Input data type*/
    auto    $DESCRIPTOR(DESCRIPTOR_DESC, "ERNE");     /*Instrument
                                                        descriptor*/
```

```
auto    $DESCRIPTOR(PRIM_SEC_DESC, "P");          /*P or S*/
char    INPUT_FILES[99] [44];                     /*Filenames*/
auto    $DESCRIPTOR(INPUT_FILE_DESC, INPUT_FILES;
int     START_DATE[99] [2];                  /*File start dates*/
int     STOP_DATE[99] [2];                        /*File end dates*/
int     NUM_FILES;                                /*Number of input
files*/
int     STATUS;
```

Note: The filenames returned by this routine are not null terminated. A null character should be inserted into INPUT_FILE[i] [43] before using it as an argument to the fopen function.

4.34.5 Error Conditions

The following fatal errors may be recorded in the system message log by ICSS_GET_FILE:

X_DBA_E_MSG	Database error occurred obtaining reference filenames/ dates
X_NO_REFERENCE	No filenames/dates could be obtained from the database

The KPGS application must check the returned status immediately after the call to ICSS_GET_FILE. If the status indicates an error, the KPGS application must terminate processing.

4.35 ICSS_COMPUTE_CGMLT

4.35.1 Purpose

ICSS_COMPUTE_CGMLT computes the CGMLT for a given date and time and geodetic orbit position.

4.35.2 Description

This support routine uses the algorithm that has been supplied by Kile Baker of the Johns Hopkins Applied Physics Laboratory to compute the CGMLT.

4.35.3 Interfaces

Input to the ICSS_COMPUTE_CGMLT from the KPGS:

CGMLT_REQ_TIME	Day of year and time of day of requested CGMLT
GEO_ORB_POSITION	Spacecraft position in geodetic reference frame

Output from the ICSS_COMPUTE_MLT to the KPGS:

CGMLT_Time	Computed CGMLT
RETURN_STATUS	Status of call

4.35.4 Calling Sequence

For FORTRAN:

```
Call ICSS_COMPUTE_CGMLT (CGMLT_REQ_TIME, GEO_ORB_POS, CGMLT_TIME,
    RETURN_STATUS)
    INTEGER*4       CGMLT_REQ__TIME(2)      !Date/time of requested
    CGMLT
                                            !(1) YYYYDDD
                                            !(2) milliseconds
    REAL*8          GEO_ORB_POS(3)          !Orbit position vector in the
                                            !geodetic reference frame
                                            !(1) Latitude (radians)
                                            !(2) Longitude (radians)
                                            !(3) Altitude (kilometers)
    REAL*4          CGMLT_TIME      !Calculated CGMLT
                                            !in degrees divided by 15 (0,
    24)
    INTEGER*4       RETURN_STATUS           !Status of calculation
```

For C:

```
ICSS_COMPUTE_CGMLT(&CGMLT_REQ_TIME, &GEO_ORB_POS,
        &CGMLT_TIME, &RETURN_STATUS);
    int         CGMLT_REQ_TIME[2]
    double      GCI_ORB_POS[3];
    float       CGMLT_TIME
    int         RETURN_STATUS;
```

4.35.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_INV_IDTFYR	Invalid year in CGMLT_REQ_TIME
ICSS_INV_IDTFDAY	Invalid day in CGMLT_REQ_TIME
ICSS_INV_IDTFMSEC	Invalid milliseconds in CGMLT_REQ_TIME
ICSS_COOR_SYS_INV	Invalid GEO_ORB_POS element

4.36 ICSS_COMPUTE_EDMLT

4.36.1 Purpose

ICSS_COMPUTE_EDMLT computes the eccentric-dipole magnetic local time and associated parameters.

4.36.2 Description

ICSS_COMPUTE_EDMLT calculates the eccentric-dipole magnetic local time (EDMLT), magnetic latitude (ED_MLAT) in eccentric-dipole (ED) frame, and L-shell (ED_L) in ED frame at a particular time and spacecraft position. The calculation is based on the formula $$ED\_L = \frac{R}{\cos^2(ED\_MLAT)}$$

The EDMLT algorithm has been supplied by

Dr. Mauricio Peredo
Hughes STX Corporation
NASA/Goddard Space Flight Center
Building 26/Room G1
Greenbelt, Maryland 20771
peredo@istp1.gsfc.nasa.gov

4.36.3 Interfaces

Input to ICSS_COMPUTE_EDMLT from the KPGS:

MLT_REQ_TIME	Date/time for which calculation is made
GEO_ORB_POS	Orbit position of spacecraft in geographic coordinate frame (in km)

Output from ICSS_COMPUTE_EDMLT to the KPGS:

EDMLT	Eccentric-dipole magnetic local time Units = degrees divided by 15 (0, 24)
MAG_LATITUDE	Eccentric-dipole magnetic latitude (in degrees)
INV_LATITUDE	Invariant latitude (in degrees)
L_SHELL	Eccentric-dipole L-shell parameter
STATUS	Status of calculation

4.36.4 Calling Sequence

For FORTRAN:

```
CALL ICSS_COMPUTE_EDMLT(MLT_REQ_TIME, GEO_ORB_POS, EDMLT,
    MAG_LATITUDE, INV_LATITUDE, L_SHELL, STATUS)
    INTEGER*4   MLT_REQ_TIME(2)         !(1) YYYYDDD
                                        !(2) Milliseconds of day
```

```
        REAL*8 GEO_ORB_POS          !Orbit position of spacecraft in geographic
                                            !coordinate frame (in km)
        REAL*4 EDMLT                !Eccentric-dipole magnetic local time
                                            !Units = degrees divided by 15
                                            !( 0, 24)
        REAL*4 MAG_LATITUDE             !Magnetic latitude (in degrees)
        REAL*4 INV_LATITUDE         !Invariant latitude (in degrees)
        REAL*4 L_SHELL                  !L-shell parameter
        INTEGER*4  STATUS               !Status of calculation
```

For C:

```
ICSS_COMPUTE_EDMLT(&MLT_REQ_TIME, &GEO_ORB_POS, &EDMLT,
        &MAG_LATITUDE, &INV_LATITUDE, &L_SHELL, &STATUS)
        int         MLT_REQ_TIME[2]         /*(1) YYYYDDD
                                              (2) Milliseconds of day*/
        double      GEO_ORB_POS             /*Orbit position of spacecraft in geographic
                                              coordinate frame (in km)*/
        float       EDMLT                   /*Eccentric-dipole magnetic local time
                                              Units = degrees divided by 15
                                              (0, 24)*/
        float       MAG_LATITUDE            /*Magnetic latitude (in degrees)*/
        float       INV_LATITUDE            /*Invariant latitude (in degrees)*/
        float       L_SHELL                 /*L-Shell parameter*/
        int         STATUS                  /*Status of calculation*/
```

4.36.5 Error Condition

The following warning condition may be detected by this routine:

ICSS_TIME_OUTRANGE	Requested date is outside range of SLP file

The following fatal errors may be returned to the caller:

ICSS_SLP_LUN	Error getting logical unit for SLP file
ICSS_OPEN_SLP	Error opening SLP file
ICSS_ERR_READ_SLP	Error reading SLP file
ICSS_SLP_NO_BODY	SLP file does not contain body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for timing coefficient file
ICSS_EREAD_TIMCOEF_FILE	Error reading from timing coefficient file

4.37 ICSS_OPEN_PO_SPINPH (POLAR Support Only)

4.37.1 Purpose

ICSS_OPEN_PO_SPINPH opens the day of data and previous day's POLAR spin-phase files.

4.37.2 Description

ICSS_OPEN_PO_SPINPH opens both the day of data and the previous day's POLAR spin-phase files, if specified. It also reads the start and stop time global attributes of the CDF POLAR spin-phase files from the two files.

4.37.3 Interfaces

Input to ICSS_OPEN_PO_SPINPH from the POLAR spin-phase header:

POLAR_FILE_HEADER	The header for the POLAR spin-phase file

Output from ICSS_OPEN_PO_SPINPH to the KPGS:

STATUS	The return status from this routine

4.37.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_PO_SPINPH (STATUS)
    INTEGER*4    STATUS        !Status of call
```

For C:

```
ICSS_OPEN_PO_SPINPH (&STATUS);
    int          STATUS;       /*Status of call*/
```

4.37.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_PO_SPINPH:

ICSS_DB_GET_PHY_ERR	Error obtaining the physical filename from the database for the logical identifier
ICSS_DB_GET_VERSION_ERR	Error obtaining the latest version from the database for the POLAR spin-phase file
ICSS_NO_FILE_ERR	POLAR spin-phase file does not exist
ICSS_SC_AFTER_PR_PSP	Invalid secondary file; begins on or after the start time of the primary file

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.38 ICSS_RET_PO_SPINPH (POLAR Support Only)

4.38.1 Purpose

ICSS_RET_PO_SPINPH extracts POLAR spin-phase data from the POLAR spin-phase files.

4.38.2 Description

ICSS_RET_PO_SPINPH locates and returns the POLAR spin-phase data that corresponds to a given request time. Linear interpolation is used to estimate the POLAR spin-phase request if a POLAR spin-phase point does not exist for a specific request time.

4.38.3 Interfaces

Input to ICSS_RET_PO_SPINPH from the KPGS:

REQ_DATE	The year, day of year, and time of day of the requested POLAR spin-phase data

Output from ICSS_RET_PO_SPINPH to the KPGS:

SPIN_PHASE	The interpolated spin-phase point for the requested time
AVG_SPIN_RATE	The interpolated average spin rate for the requested time
STAN_DEV	The standard deviation of the requested time
RETURN_STAT	The status of the requested POLAR spin-phase data

4.38.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_PO_SPINPH (REQ_DATE, SPIN_PHASE, AVG_SPIN_RATE,
        STAN_DEV, RETURN_STAT)
    INTEGER*4        REQ_DATE(2) !(1) YYYYDDD
                                 !(2) milliseconds of day
    REAL*4           SPIN_PHASE  !In radians
```

```
      REAL*4          AVG_SPIN_RATE    !In radians/second
      REAL*4          STAN_DEV         !In radians/second
      INTEGER*4       RETURN_STAT      !Status of call
```

For C:

```
ICSS_RET_PO_SPINPH (&REQ_DATE, &SPIN_PHASE, &AVG_SPIN_RATE,
        &STAN_DEV, &RETURN_STAT);
      int             REQ_DATE[2];     /*(1) YYYYDDD
                                           (2) milliseconds of day*/
      float           SPIN_PHASE;  /*In radians*/
      float           AVG_SPIN_RATE;   /*In radians/second*/
      float           STAN_DEV;        /*In radians/second*/
      int             RETURN_STAT;     /*Status of call*/
```

4.38.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_TIME_OUTRANGE Requested time out of range of files

The following error may be recorded in the system message log by ICSS_RET_PO_SPINPH:

ICSS_NO_FILE_ERR File does not exist

The error returned from this support routine is fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.39 ICSS_OPEN_DP_ATT (POLAR Support Only)

4.39.1 Purpose

ICSS_OPEN_DP_ATT opens the day of data and previous day's despun platform attitude files.

4.39.2 Description

ICSS_OPEN_DP_ATT opens both the day of data and the previous day's despun platform attitude files, if specified. It also reads the global attributes of the CDF attitude files from the two files.

4.39.3 Interfaces

Input to ICSS_OPEN_DP_ATT from the despun platform attitude file:

DPATT_HEADER The file label record for the attitude file

Note: This routine derives the despun platform attitude filename from the file date of the level-zero filename.

Output from ICSS_OPEN_DP_ATT to the KPGS:

COMPLETION_STATUS The return status from the routine

4.39.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_DP_ATT (COMPLETION_STATUS)
    INTEGER*4                COMPLETION_STATUS      !Message number
```

For C:

```
ICSS_OPEN_DP_ATT (&COMPLETION_STATUS);
    int                      COMPLETION_STATUS;     /*Message number*/
```

4.39.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_DP_ATT:

ICSS_DB_GET_PHYS_ERR	Error obtaining the physical filename from the database for the logical identifier
ICSS_DB_GET_VERSION_ERR	Error obtaining the latest version from the database for the despun platform attitude file
ICSS_NO_FILE_ERR	Despun platform attitude file does not exist
ICSS_SC_AFTER_PR_ATT	Invalid secondary attitude file; begins on or after start time of primary file
ICSS_OPEN_FILE_ERR	Error opening the CDF file
ICSS_READTIMES_ERR	Error reading the file times from the CDF file
ICSS_INV_TIME_ERR	Error obtaining the file times for the CDF file

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.40 ICSS_RET_DP_ATT (POLAR Support Only)

4.40.1 Purpose

ICSS_RET_DP_ATT extracts despun platform attitude data from the despun platform attitude files.

4.40.2 Description

ICSS_RET_DP_ATT locates and returns the despun platform attitude data that corresponds to a given request time. Linear interpolation is used to estimate the despun platform attitude request if an attitude point does not exist for a specific request time.

4.40.3 Interfaces

Input to ICSS_RET_DP_ATT from the despun platform attitude file:

REQ_DATE	The day of year and time of day of the requested despun platform attitude data
COOR_SYS	The requested coordinate system of the returned despun platform attitude data

Output from ICSS_RET_DP_ATT to the KPGS:

EUL_ANG	The interpolated pitch, roll, and yaw for the requested time
ROTATION_MATRIX	The attitude matrix for the euler angles at the requested time
QUALITY_FLAG	The quality of the interpolated data: 0 = good, 1 = bad, 2 = intermediate (i.e., interpolation of one good value and one bad value)
RETURN_STATUS	The return status from the routine

The relationship between the nine-element single-dimension array returned (ROTATION_MATRIX) and the 3-by-3 two-dimensional transformation rotation matrix is as follows:

ROTATION_MATRIX(1) = Position (1,1) [row, column]
ROTATION_MATRIX(2) = Position (2,1)
ROTATION_MATRIX(3) = Position (3,1)
ROTATION_MATRIX(4) = Position (1,2)
ROTATION_MATRIX(5) = Position (2,2)
ROTATION_MATRIX(6) = Position (3,2)
ROTATION_MATRIX(7) = Position (1,3)
ROTATION_MATRIX(8) = Position (2,3)
ROTATION_MATRIX(9) = Position (3,3)

4.40.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_DP_ATT (REQ_DATE, COOR_SYS, EUL_ANG, ROTATION_MATRIX,
    QUALITY_FLAG, RETURN_STATUS)
    INTEGER*4           REQ_DATE(2)         !(1) YYYYDDD
                                            !(2) milliseconds of day
    CHARACTER*3         COOR_SYS            !ORB, GSE, GSM, GCI
    REAL*4              EUL_ANG(3)          !(1) Pitch in radians
                                            !(2) Roll in radians
                                            !(3) Yaw in radians
    REAL*8              ROTATION_MATRIX (9) !Attitude matrix
    INTEGER*4           QUALITY_FLAG        !Quality flag
    INTEGER*4           RETURN_STATUS       !Message number
```

For C:

```
ICSS_RET_DP_ATT (&REQ_DATE, &COOR_SYS, &EUL_ANG,
        &ROTATION_MATRIX, &QUALITY_FLAG, &RETURN_STATUS);
    int             REQ_DATE[2];                    /*(0)
    YYYYDDD
                                                    (1) milliseconds
    of
                                                        day*/
    auto            $DESCRIPTOR(COOR_SYS,"GSE");    /*ORB, GSE, GSM,
                                                        GCI*/
    float           EUL_ANG[3];             /*(1) Pitch
                                                (2) Roll
                                                (3) Yaw*/
    double          ROTATION_MATRIX[9];     /*Attitude matrix*/
    int             QUALITY_FLAG;           /*Quality flag*/
    int             RETURN_STATUS;          /*Message number*/
```

4.40.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_RET_DP_ATT:

ICSS_COOR_SYS_INV	Invalid coordinate system
ICSS_TIME_OUTRANGE	Requested time not in valid range
X_CDF_F_MSG	Error reading from CDF file

All errors returned from this support routine, except ICSS_TIME_OUTRANGE, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.41 ICSS_ACF_TO_GCI (SOHO Support Only)

4.41.1 Purpose

ICSS_ACF_TO_GCI converts an attitude vector in ACF to a vector in the GCI reference frame.

4.41.2 Description

This support routine transforms an attitude vector in the ACF spacecraft reference frame to an attitude vector in the GCI reference frame.

4.41.3 Interfaces

Input to the ICSS_ACF_TO_GCI from the KPGS:

IDTF_TIME	Date/time in internal day time format (IDTF)
GCI_ORB_POSITION	The spacecraft orbital position vector
ACF_ATT_VECTOR	The attitude vector in ACF

Output from the ICSS_ACF_TO_GCI to the KPGS:

GCI_ATT_VECTOR	The spacecraft attitude vector in the GCI reference frame
TRANSFORM_MATRIX	The transformation matrix used to transform a vector in ACF to a vector in the GCI reference frame
STATUS	The status of the transformation

4.41.4 Calling Sequence

For FORTRAN:

```
Call ICSS_ACF_TO_GCI (IDTF_TIME, GCI_ORB_POSITION, ACF_ATT_VECTOR,
                      GCI_ATT_VECTOR, TRANSFORM_MATRIX, RETURN_STATUS)
    INTEGER*4   IDTF_TIME(2)          !IDTF-formatted date/time
                                      !(1) YYYYDDD
                                      !(2) milliseconds of day
    REAL*8      GCI_ORB_POSITION(3)   !Orbit position vector
                                      !(x,y,x) at IDTF_TIME
    REAL*4      ACF_ATT_VECTOR(3)     !Spacecraft attitude vector in
ACF
    REAL*4      GCI_ATT_VECTOR(3)     !Spacecraft attitude vector in
the                                         !GCI reference frame
    REAL*8      TRANSFORM_MATRIX(3,3) !ACF to GCI transformation matrix
    INTEGER*4   STATUS                !The status of the
transformation
```

For C:

```
ICSS_ACF_TO_GCI (&IDTF_TIME, &GCI_ORB_POSITION, &ACF_ATT_VECTOR,
                 &GCI_ATT_VECTOR, &TRANSFORM_MATRIX, &RETURN_STATUS);
    int         IDTF_TIME[2];         /*(0) YYYYDDD
                                        (1) milliseconds of day*/
    double      GCI_ORB_POSITION[3];  /*GCI orbit position vector*/
    float       ACT_ATT_VECTOR[3];    /*ACF attitude vector*/
    float       GCI_ATT_VECTOR[3]     /*GCI attitude vector*/
    double      TRANSFORM_MATRIX[9];  /*ACF to GCI transformation
                                        matrix*/
    int         RETUN_STATUS;         /*The status of transformation*/
```

4.41.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_SLP_LUN	Error getting a logical unit for the SLP file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_ERR_READ_SLP	Error reading the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the timing coefficients file
ICSS_EREAD_TIMECOEF_FILE	Error reading from the timing coefficients file
ICSS_TIME_OUTRANGE	Requested date is outside the range of the SLP file

4.42 ICSS_OPEN_SOHO_LZ (SOHO Support Only)

4.42.1 Purpose

This routine opens a SOHO level-zero telemetry file.

4.42.2 Description

This routine opens a SOHO level-zero file so that subsequent calls to the ICSS_RET_SOHO_ PACKETS support routine may read that file. The name of the file is passed to this routine as the first argument. A status is returned indicating whether the file was opened successfully.

4.42.3 Interfaces

Input to ICSS_OPEN_SOHO_LZ from the KPGS:

SOHO_FILE_NAME	Full name (including path and extension) of the file to be opened

Output from ICSS_OPEN_SOHO_LZ to the KPGS:

STATUS	Status of opened file

4.42.4 Calling Sequence

For FORTRAN

```
Call ICSS_OPEN_SOHO_LZ (%REF(FILE_NAME), %REF(STATUS))
    CHARACTER*44        FILENAME
    INTEGER*4           STATUS
```

For C:

```
ICSS_OPEN_SOHO_LZ (FILE_NAME, STATUS);
    char                *FILE_NAME;
    int                 *STATUS;
```

4.42.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_SOHO_LZ:

ICSS_OPEN_PR_LZ	End opening the primary level-zero file
ICSS_READHDR_PR_LZ	Error reading the primary level-zero file header

All errors returned from this support routine are fatal. The KPGS must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.43 ICSS_RET_SOHO_PACKETS (SOHO Support Only)

4.43.1 Purpose

This routine reads a number of packets from the SOHO level-zero telemetry file and returns these packets along with quality accounting information to the caller.

4.43.2 Description

This routine reads a number of packets from the SOHO level-zero file. The buffer into which the packets are to be returned is passed as the first argument. This routine returns the entire packet, including the packet header and spacecraft time field. The quality and accounting information for the packets is returned in a second buffer, which is passed as the second argument to this routine. There is one 32-bit word (containing error flags, frames with Reed-Solomon (R-S) correction, and location of packet fill) in this buffer for each packet that is returned.

The number of packets to be read is passed to this routine as the third argument. The actual number of packets that were read is returned in the fourth argument. A status is returned indicating whether the file was read successfully and whether the end of file has been encountered.

4.43.3 Interfaces

Input to ICSS_RET_SOHO_PACKETS from the KPGS:

NO_PACKETS_TO_READ	The number of packets to be read from the SOHO level-zero file

Output from ICSS_RET_SOHO_PACKETS to the KPGS:

PACKET_BUFFER	Buffer containing the packets that have been read
QAC_BUFFER	Buffer containing 32 bits of packet quality information for each packet returned
PACKETS_READ	Number of packets that are being returned
STATUS	Status of packet reads

4.43.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_SOHO_PACKETS (%REF(PACKET_BUFFER), %REF(QAC_BUFFER),
        %REF(NO_PACKETS_TO_READ), %REF(PACKETS_READ),
        %REF(STATUS))
    BYTE        PACKET_BUFFER(*)  !Buffer for packets
    INTEGER*4   QAC_BUFFER(*)     !Buffer for quality accounting
                                  !Bits 0-7—Error type flags
                                  !Bits 8-15—Frames with R-S correction
                                  !Bits 16-31—Packet fill start location
    INTEGER*4   NO_PACKETS_TO_READ  !How many packets to return
    INTEGER*4   PACKETS-READ        !Number of packets returned
    INTEGER*4   STATUS              !Status of packet reads
```

For C:

```
ICSS_RET_SOHO_PACKETS (PACKET_BUFFER, QAC_BUFFER,
        NO_PACKETS_TO_READ, PACKETS_READ, STATUS);
    char        *PACKET_BUFFER;
    int         *QAC_BUFFER;
```

```
int             *NO_PACKETS_TO_READ;
int             *PACKETS_READ;
int             *STATUS;
```

4.43.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_EOF_PR_LZ	End of the file reached reading the primary level-zero file

The following error may be recorded in the system message log by ICSS_RET_SOHO_PACKETS:

ICSS_READ_PR_LZ	Error reading the primary level-zero file data record

All errors returned from this support routine, except ICSS_EOF_PR_LZ, are fatal. The KPGS must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.44 ICSS_OPEN_SOHO_HK (SOHO Support Only)

4.44.1 Purpose

This routine opens a SOHO housekeeping telemetry file.

4.44.2 Description

This routine opens a SOHO housekeeping file so that subsequent calls to the ICSS_RET_SOHO_HK support routine may read that file. The name of the file is passed to this routine as the first argument. A status is returned indicating whether the file was opened successfully.

4.44.3 Interfaces

Input to ICSS_OPEN_SOHO_HK from the KPGS:

SOHO_FILE_NAME	Full name (including path and extension) of the file to be opened

Output from ICSS_OPEN_SOHO_HK to the KPGS:

STATUS	Status of opened file

4.44.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_SOHO_HK (%REF(FILE_NAME), %REF(STATUS))
    CHARACTER*44        FILENAME
    INTEGER*4           STATUS
```

For C:

```
ICSS_OPEN_SOHO_HK (FILE_NAME, STATUS);
    char                *FILE_NAME;
    int                 *STATUS;
```

4.44.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_SOHO_HK:

ICSS_OPEN_PR_HK	Error opening the primary housekeeping file
ICSS_READHDR_PR_HK	Error reading the primary housekeeping file header

All errors returned from this support routine are fatal. The KPGS must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.45 ICSS_RET_SOHO_HK (SOHO Support Only)

4.45.1 Purpose

This routine reads a number of packets from the SOHO housekeeping file and returns these packets along with quality accounting information to the caller.

4.45.2 Description

This routine reads a number of packets from the SOHO housekeeping file. The buffer into which the packets are to be returned is passed as the first argument. This routine returns the entire packet, including the packet header and spacecraft time field. The quality and accounting information for the packets is returned in a second buffer, which is passed as the second argument to this routine. There is one 32-bit word (containing error flags, frames with R-S correction, and location of packet fill) in this buffer for each packet that is returned.

The number of packets to be read is passed to this routine as the third argument. The actual number of packets that were read is returned in the fourth argument. A status is returned indicating whether the file was read successfully and whether the end of file has been encountered.

4.45.3 Interfaces

Input to the ICSS_RET_SOHO_HK from the KPGS:

NO_PACKET_TO_READ	The number of packets to be read from the SOHO housekeeping file

Output from ICSS_RET_SOHO_HK to the PKGS:

PACKET_BUFFER	Buffer containing the packets that have been read
QAC_BUFFER	Buffer containing 32 bits of packet quality information for each packet returned

PACKETS_READ	Number of packets that are being returned
STATUS	Status of packet reads

4.45.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_SOHO_HK (%REF(PACKET_BUFFER), %REF(QAC_BUFFER),
        %REF(NO_PACKETS_TO_READ), %REF(PACKETS_READ),
        %REF(STATUS))
    BYTE        PACKET_BUFFER(*)  !Buffer for packets
    INTEGER*4   QAC_BUFFER(*)     !Buffer for quality accounting
                                  !Bits 0-7—Error type flags
                                  !Bits 8-15—Frames with R-S correction
                                  !Bits 16-31—Packet fill start location
    INTEGER*4   NO_PACKETS_TO_READ  !How many packets to return
    INTEGER*4   PACKETS_READ        !Number of packets returned
    INTEGER*4   STATUS              !Status of packet reads
```

For C:

```
ICSS_RET_SOHO_HK (PACKET_BUFFER, QAC_BUFFER,
        NO_PACKETS_TO_READ, PACKETS_READ, STATUS);
    char    *PACKET_BUFFER;
    int     *QAC_BUFFER;
    int     *NO_PACKETS_TO_READ;
    int     *PACKETS_READ;
    int     *STATUS;
```

4.45.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_EOF_PR_HK	End of file reached reading the primary housekeeping file

The following error may be recorded in the system message log by ICSS_RET_SOHO_HK:

ICSS_READ_PR_HK	Error reading the primary housekeeping file data record

All errors returned from this support routine, except ICSS_EOF_PR_HK, are fatal. The KPGS must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.46 ICSS_OPEN_SOHO_ATT (SOHO Support Only)

4.46.1 Purpose

ICSS_OPEN_SOHO_ATT opens the day of data and previous day's SOHO attitude files.

4.46.2 Description

ICSS_OPEN_SOHO_ATT opens both the day of data and the previous day's SOHO attitude files, if specified. It also reads the start and stop time global attributes of the CDF SOHO attitude files from the two files.

4.46.3 Interfaces

Input to ICSS_OPEN_SOHO_ATT from the SOHO attitude file:

SOHO_FILE_HEADER	The header for the SOHO attitude file

Output from ICSS_OPEN_SOHO_ATT to the KPGS:

STATUS	The return status from this routine

4.46.4 Calling Sequence

For FORTRAN:

```
Call ICSS_OPEN_SOHO_ATT (STATUS)
    INTEGER*4    STATUS;      !Status of call
```

For C:

```
ICSS_OPEN_SOHO_ATT (&STATUS);
    int          STATUS;      /*Status of call*/
```

4.46.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_OPEN_SOHO_ATT:

ICSS_DB_GET_PHY_ERR	Error obtaining physical filename from database for logical identifier
ICSS_DB_GET_VERSION_ERR	Error obtaining latest version from database for SOHO attitude file
ICSS_NO_FILE_ERR	SOHO attitude file does not exist
ICSS_SC_AFTER_PR_SA	Invalid secondary file; begins on or after the start time of the primary file

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.47 ICSS_RET_SOHO_ATT (SOHO Support Only)

4.47.1 Purpose

ICSS_RET_SOHO_ATT extracts SOHO attitude data from the SOHO spin-phase files

4.47.2 Description

ICSS_RET_SOHO_ATT locates and returns the SOHO attitude data that corresponds to a given request time. Linear interpolation is used to estimate the SOHO attitude request if a SOHO attitude point does not exist for a specific request time.

4.47.3 Interfaces

Input to ICSS_RET_SOHO_ATT from the KPGS:

REQ_DATE	The year, day of year, and time of day of the requested SOHO attitude data
COORD_SYS	The desired coordinate system of the SOHO attitude data

Output from ICSS_RET_SOHO_ATT to the KPGS:

OUTPUT_VECTOR	The interpolated pitch, roll, and yaw for the requested time
STD_DEV	The standard deviation of the pitch, roll, and yaw for the requested time
STATUS	The status of the requested SOHO attitude data

4.47.4 Calling Sequence

For FORTRAN:

```
Call ICSS_RET_SOHO_ATT (REQ_DATE, COORD_SYS, OUTPUT_VECTOR,
        STD_DEV, STATUS)
    INTEGER*4       REQ_DATE(2)          !(1) YYYYDDD
                                         !(2) milliseconds of day
    CHARACTER*3     COORD_SYS            !(GCI, GSE, GSM, SAT)
    REAL*4          OUTPUT_VECTOR(3) !In radians
                                         !(1) Pitch
                                         !(2) Roll
                                         !(3) Yaw
```

```
      REAL*4          STAN_DEV(3)     !In radians/second
                                      !(1) Pitch
                                      !(2) Roll
                                      !(3) Yaw
      INTEGER*4       STATUS          !Status of call
```

For C:

```
   ICSS_RET_SOHO_ATT (&REQ_DATE, &COORD_SYS, &OUTPUT_VECTOR,
         &STD_DEV, &STATUS);
      int             REQ_DATE[2];    /*(1) YYYYDDD
                                        (2) milliseconds of day*/
      auto            $DESCRIPTOR     /*(GCI, GSE, GSM, SAT)*/
                      (COORD_SYS, "GCI");
      float           OUTPUT_VECTOR[3]; /*In radians
                                        (1) Pitch
                                        (2) Roll
                                        (3) Yaw*/
      float           STAN_DEV[3];    /*In radians/second
                                        (1) Pitch
                                        (2) Roll
                                        (3) Yaw*/
      int             STATUS;         /*Status of call*/
```

4.47.5 Error Conditions

The following warning condition may be detected by this routine:

ICSS_TIME_OUTRANGE	Requested time out of range of files

The following errors may be recorded in the system message log by ICSS_RET_SOHO_ATT:

ICSS_NO_FILE_ERR	File does not exist
ICSS_COORD_SYS_INV	Invalid coordinate system

All errors returned from this support routine, except ICSS_TIME_OUTRANGE, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

4.48 ICSS_UTC_TAI_OFFSET

4.48.1 Purpose

ICSS_UTC_TAI_OFFSET determines the offset between universal time coordinated (UTC) and international atomic time (TAI).

4.48.2 Description

This routine determines the number of seconds that must be added to TAI to derive UTC on a specific date.

4.48.3 Interfaces

Input to ICSS_UTC_TAI_OFFSET from the KPGS:

TAI_TIME	Date on which offset is determined

Output from ICSS_UTC_TAI_OFFSET to the KPGS:

OFFSET	Offset (in seconds) between UTC and TAI times at DATE (UTC-TAI)
STATUS	Status of offset determination

4.48.4 Calling Sequences:

For FORTRAN:

```
Call ICSS_UTC_TAI_OFFSET(TAI_TIME, OFFSET, STATUS)
    INTEGER*4       TAI_TIME(2)     !(1) = year/day of year (YYYYDDD)
                                    !       Year is between 1990 and 2020
                                    !       Day is between 1 and 366
                                    !(2) = milliseconds of day is between 0
    and
                                    !       86400000
    INTEGER*4       OFFSET          !Offset (in seconds) between UTC and
    TAI
                                    !times at DATE (UTC-TAI)
    INTEGER*4       STATUS          !SS$_NORMAL if successful;
    otherwise
                                    !DATE was invalid
```

For C:

```
ICSS_UTC_TAI_OFFSET(&TAI_TIME, &OFFSET, &STATUS)
    int             TAI_TIME(2)  /*(1) = year/day of year (YYYYDDD)
                                          Year is between 1990 and 2020
                                          Day is between 1 and 366*/
                                 /*(2) = milliseconds of day is between 0
    and
                                          86400000*/
    int             OFFSET          /*Offset (in seconds) between UTC and
    TAI
                                       times at DATE (UTC-TAI)*/
    int             STATUS          /*SS$_NORMAL if successful;
    otherwise
                                      DATE was invalid*/
```

4.48.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_UTC_TAI_OFFSET:

ICSS_INV_IDTFYR	Year in MLT_REQ_TIME is invalid
ICSS_INV_IDTFDAY	Day in MLT_REQ_TIME is invalid
ICSS_INV_IDTFMSEC	Milliseconds in MLT_REQ_TIME are invalid

All errors returned from this routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

NOTE: The TDIFF routine that is used to calculate the UTC-TAI offset attempts to calculate the offset regardless if the requested date is outside the range of the time coefficient file. However, dates outside this range may not be as accurate as those with the range of the file.

```
INTEGER*4   TAI_TIME(2)     !Date at which offset is to be determined
INTEGER*4   OFFSET          !Seconds that must be added to UTC to get TAI
INTEGER*4   STATUS          !Return status
REAL*8      R_OFFSET        !Offset returned by T_DIFF subroutine
INTEGER*4   YEAR            !Year component of TAI_TIME
INTEGER*4   DAY_OF_YEAR     !Day of year component of TAI_TIME
```

```
INTEGER*4   MONTH       !Month of year derived from DAY_OF_YEAR
INTEGER*4   DAY         !Day of month derived from DAY_OF_YEAR
INTEGER*4   IJUL_DATE   !Julian date of TAI_TIME
REAL*8      JUL_DATE    !Real*8 copy of IJUL_DATE
```

4.49 ICSS_GSM_SM

4.49.1 Purpose

ICSS_GSM_SM converts the coordinate systems .

4.49.2 Description

ICSS_GSM_SM converts the coordinate systems between GSM and SM.

4.49.3 Interfaces

Input to ICSS_GSM_SM from the KPGS:

REQ_SYS	Coordinate system of transformed vector
SRC_POS	Source vector
SRC_POS_TIME	Time of source vector

Output from ICSS_GSM_SM to the KPGS:

TRANS_POS	Transformed vector
ROTATION_MATRIX	Rotation matrix
RETURN_STATUS	Status of call

4.49.4 Calling Sequence

For FORTRAN:

```
Call ICSS_GSM_SM(REQ_SYS, SRC_POS, SRC_POS_TIME, TRANS_POS,
                 ROTATION_MATRIX, RETURN_STATUS)

INTEGER*4   REQ_SYS             !1 = Converts SM vector to GSM vector
                                !2 = Converts GSM vector to SM vector
REAL*8      SRC_POS(2)          !(1) X-component
                                !(2) Y-component
                                !(3) Z-component
INTEGER*4   SRC_POS_TIME(2)     !(1) YYYYDDD
                                !(2) Milliseconds of day
REAL*8      TRANS_POS(3)        !(1) X-component
                                !(2) Y-component
                                !(3) Z-component
REAL*8      ROTATION_MATRIX(9)  !Rotation from source to target
                                !coordinate system, 3x3 row-ordered
                                !matrix
INTEGER*4   RETURN_STATUS       !Return status
```

For C:

```
ICSS_GSM_SM(&REQ_SYS, &SRC_POS, &SRC_POS_TIME, &TRANS_POS,
            &ROTATION_MATRIX, &RETURN_STATUS)

int         REQ_SYS             /*1 = Converts SM vector to GSM
    vector*/
                                /*2 = Converts GSM vector to SM
    vector*/
double      SRC_POS[2]          /*(1) X-component*/
                                /*(2) Y-component*/
                                /*(3) Z-component*/
```

```
int         SRC_POS_TIME[2]     /*(1) YYYYDDD*/
                                /*(2) Milliseconds of day*/
double      TRANS_POS[3]        /*(1) X-component*/
                                /*(2) Y-component*/
                                /*(3) Z-component*
double      ROTATION_MATRIX[9]  /*Rotation from source to target
    coordinate                                  system, 3x3 row-
    ordered matrix*/
int         RETURN_STATUS       /*Return status*/
```

4.49.5 Error Conditions

The following fatal errors may be returned to the caller:

```
SS$_BADPARAM REQ_SYS is something other than 1 or 2
SS$_SSFAIL   An error occurred while trying to calculate the tilt angle
```

4.50 ICSS_TILT_ANGLE

4.50.1 Purpose

ICSS_TILT_ANGLE calculates the tilt angle, which is an angle between the direction of the Earth's magnetic field and the direction perpendicular to the Earth's Sun line.

4.50.2 Description

ICSS_TILT_ANGLE calculates the tilt angle based on the requested time.

4.50.3 Interfaces

Input to ICSS_TILT_ANGLE from the KPGS:

```
ORB_POS_TIME    Request time
```

Output from ICSS_TILT_ANGLE to the KPGS:

```
TILT_ANGLE      Tilt angle (Rad)
DIPOLE_VECTOR   GEO dipole vector
RETURN_STATUS   Return status
```

4.50.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TILT_ANGLE(ORB_POS_TIME, TILE_ANGLE, DIPOLE_VECTOR,
        RETURN_STATUS)
INTEGER*4   ORB_POS_TIME        !(1) YYYYDDD
                                !(2) Milliseconds of day
REAL*8      TILT_ANGLE          !Tilt angle (Rad)
REAL*8      DIPOLE_VECTOR(3)    !(1) X-component
                                !(2) Y-component
                                !(3) Z-component
INTEGER*4   RETURN_STATUS       !Return status
```

For C:

```
ICSS_GSM_SM(&ORB_POS_TIME, &TILE_ANGLE, &DIPOLE_VECTOR,
        &RETURN_STATUS)
int         ORB_PO_TIME[2]      /*1 = Converts SM vector to GSM
    vector
                                  2 = Converts GSM vector to SM
    vector*/
```

```
double          TILT_ANGLE          /*Tilt angle (Rad)*/
double          DIPOLE_VECTOR[3]    /*(1) X-component
                                        (2) Y-component
                                        (3) Z-component*/
int             RETURN_STATUS       /*Return status*/
```

4.50.5 Error Conditions

This support routine has no error handling.

4.51 ICSS_TSY

4.51.1 Purpose

ICSS_TSY provides an ISTP interface to the Tsyganenko models.

4.51.2 Description

ICSS_TSY computes the total magnetic field and the magnetic field line traces (down to 100 km altitude) using the Tsyganenko model T89C with kp = 3-, 3, 3+.

4.51.3 Interfaces

Input to ICSS_TSY from the KPGS:

POS_TIME	Time and date
GSM_POS	GSM position of satellite (Re = Earth radii)
IITRACE	Trace option

Output from ICSS_TSY to the KPGS:

LN	Number of points along fieldline traced northward
LS	Number of points along fieldline traced southward
B_GSM	GSM magnetic field at satellite point (nT)
H_GSM	GSM IGRF field at satellite point (nT)
N_END	GSM position of northern footpoint (Re) at 100 km altitude
S_END	GSM position of southern footpoint (Re)
N_TRACE	GSM position of points along fieldline traced northward (Re)
S_TRACE	GSM position of points along fieldline traced northward (Re)
RETURN_STATUS	Return status

4.51.4 Calling Sequence

For FORTRAN:

```
Call ICSS_TSY(POS_TIME, GSM_POS, IITRACE, LN, LS, B_GSM, H_GSM, N_END,
              S_END, N_TRACE, S_TRACE, RETURN_STATUS)
  INTEGER*4     POS_TIME(2)     !(1) YYYYDDD
                                !(2) Milliseconds of day
  REAL*8        GSM_POS(3)      !(1) X-component
                                !(2) Y-component
                                !(3) Z-component
  INTEGER*4     IITRACE         != 1, magnetic field at input location
                                != 2, magnetic field at input location,
                                !     and northern footpoint
                                != 3, magnetic field at input location,
                                !     and southern footpoint
                                != 4, magnetic field at input location,
                                !     and both northern and southern
                                !     footpoints
  INTEGER*4     LN              !Number of points along fieldline traced
                                !northward
  INTEGER*4     LS              !Number of points along fieldline traced
                                !southward
  REAL*8        B_GSM(3)        !(1) Bx-component
                                !(2) By-component
                                !(3) Bz-component
  REAL*8        H_GSM(3)        !(1) Hx-component (IGRF)
                                !(2) Hy-component (IGRF)
                                !(3) Hz-component (IGRF)
```

```
    REAL*8          N_END(3)            !(1) X-component
                                        !(2) Y-component
                                        !(3) Z-component
    REAL*8          S_END(3)            !(1) X-component
                                        !(2) Y-component
                                        !(3) Z-component
    REAL*8          N_TRACE(3,800)      !(I,K), I = cartesian component(X,Y,Z)
                                        !       K = K'th point in trace
                                        !(1:3,1) is same as input position
                                        !(1:3,LN) is same as N_END
    REAL*8          S_TRACE(3,800)      !(I,K), I = cartesian component (X,Y,Z)
                                        !       K = K'th point in trace
                                        !(1:3,1) is same as input position
                                        !(1:3,LS) is same as S_END
    INTEGER*4       RETURN_STATUS       !Return status
```

For C:

```
    ICSS_TSY(&POS_TIME, &GSM_POS, &IITRACE, &LN, &LS, &B_GSM, &H_GSM,
                    &N_END, &S_END, &N_TRACE, &S_TRACE,
        &RETURN_STATUS)
        int         POS_TIME[2]     /*(1) YYYYDDD
                                        (2) Milliseconds of day*/
        double      GSM_POS[3]      /*(1) X-component
                                        (2) Y-component
                                        (3) Z-component*/
        int         IITRACE         /*= 1, magnetic field at input location
                                      = 2, magnetic field at input location,
                                           and northern footpoint
                                      = 3, magnetic field at input location,
                                           and southern footpoint
                                      = 4, magnetic field at input location,
                                           and both northern and southern
                                           footpoints*/
        int         LN              /*Number of points along fieldline
        traced
                                        northward*/
        int         LS              /*Number of points along fieldline
        traced
                                        southward*/
        double      B_GSM(3)        /*(1) Bx-component
                                        (2) By-component
                                        (3) Bz-component*/
        double      H_GSM(3)        /*(1) Hx-component (IGRF)
                                        (2) Hy-component (IGRF)
                                        (3) Hz-component (IGRF)*/
```

```
double      N_END(3)          /*(1) X-component
                                (2) Y-component
                                (3) Z-component*/
double      S_END(3)          /*(1) X-component
                                (2) Y-component
                                (3) Z-component*/
double      N_TRACE(3,800)    /*(I,K), I = cartesian component(X,Y,Z)
                                      K = K'th point in trace
                                (1:3,1) is same as input position
                                (1:3,LN) is same as N_END*/
double      S_TRACE(3,800)    /*(I,K), I = cartesian component
(X,Y,Z)
                                      K = K'th point in trace
                                (1:3,1) is same as input position
                                (1:3,LS) is same as S_END*/
int         RETURN_STATUS     /*Return status*/
```

4.51.5 Error Conditions

The following fatal error may be returned to the caller:

SS$_BADPARAM Returned when an invalid TRACE function is requested

4.52 ICSS_POS_VEL_OF_CELESTIAL

4.52.1 Purpose

ICSS_POS_VEL_OF_CELESTIAL retrieves the position and velocity vectors of a celestial body in GCI coordinates.

4.52.2 Description

This routine returns the position and velocity of a celestial body at a requested time.

4.52.3 Interfaces

Input to ICSS_POS_VEL_OF_CELESTIAL from the KPGS:

ORB_POS_TIME	Requested time
CEL_INDEX	Index for the celestial body

Output from ICSS_POS_VEL_OF_CELESTIAL to the KPGS:

CEL_POSITION	Position of the celestial body (km)
CEL_VELOCITY	Velocity of the celestial body (km)
PV_STATUS	Status of getting the position and velocity

4.52.4 Calling Sequences

For FORTRAN:

```
Call ICSS_POS_OF_CELESTIAL (ORB_POS_TIME, CEL_INDEX, CEL_POSITION,
          CEL_VELOCITY, PV_STATUS)
 INTEGER*4      ORB_POS_TIME(2)     !(1) YYYYDDD
                                    !(2) Milliseconds of day
 INTEGER*4      CEL_INDEX           !Index for the celestial body
                                    != 2, Moon
                                    != 3, Sun
```

REAL*8 (km)	CEL_POSITION(3)	!Position of the celestial body
REAL*8 (km)	CEL_VELOCITY(3)	!Velocity of the celestial body
INTEGER*4 velocity	PV_STATUS	!Status of getting position and

For C:

```
ICSS_POS_OF_CELESTIAL (&ORB_POS_TIME, &CEL_INDEX, &CEL_POSITION,
                       &CEL_VELOCITY, &PV_STATUS);
```

int	ORB_POS_TIME(2)	/*(1) YYYYDDD (2) Milliseconds of day*/
int	CEL_INDEX	/*Index for celestial body = 2, Moon = 3, Sun*/
double (km)*/	CEL_POSITION(3)	/*Position of the celestial body
double (km)*/	CEL_VELOCITY(3)	/*Velocity of the celestial body
int	PV_STATUS	/*Status of getting the position and velocity*/

4.52.5 Error Conditions

The following fatal errors may be returned to the caller:

ICSS_SLP_LUN	Error getting a logical unit number for the SLP file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the Timing Coefficient file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested

SECTION 5—NEAR-REAL-TIME KEY PARAMETER GENERATION

The ISTP CDHF supports the generation of key parameters for NRT telemetry data received from the Generic Data Capture Facility (GDCF). The NRT processing is performed on a standalone system and the results are transmitted via network services to the appropriate destination.

This section describes the NRT data receipt and processing and the special considerations and support routines for the KPGS to be executed in the NRT environment.
The ISTP CDHF supports the receipt, processing, and distribution of NRT data for the WIND and POLAR missions. NRT data is received from the GDCF on a dedicated set of computers in the ISTP CDHF.

5.1 NRT Server System

The NRT Server System supports the real-time distribution of WIND and POLAR NRT level-zero data and NRT key parameter data (future). Client software (see below) is used to connect to the NRT Server System and request data. Authorized clients will then receive the requested data as it becomes available. The NRT level-zero and key parameter data is packetized and sent to the client via a TCP socket connection. The format of the packets is described in the ISTP Data Format Control Document (DFCD) (Reference 5).

5.1.1 NRT Client Routines

The client requires four routines to connect, log in, request, and receive data from the NRT server. These routines are described below. All client routines are written in C language and may be obtained from the ISTP CDHF (istp1.gsfc.nasa.gov) via anonymous File Transfer Protocol (FTP). The NRT client routines are located in the NRT subdirectory. They may also be obtained directly on the ISTP CDHF from the SYS$PUBLIC:[NRT] directory.

5.1.1.1 RTC_CONNECT

5.1.1.1.1 Purpose

RTC_CONNECT connects the client to the NRT Server System.

5.1.1.1.2 Description

RTC_CONNECT establishes a socket connection between the client and the NRT Server System. Status messages will be reported to the standard output device.

5.1.1.1.3 Interfaces

Input to RTC_CONNECT from the client program:

SERVER_NAME	The address of the NRT server
FPTR	Log file pointer

Output from RTC_CONNECT to the client program:

STATUS	0 indicates success
	-1 indicates an error in connecting to the server

5.1.1.1.4 Calling Sequence

For C:

```
STATUS = RTC_CONNECT (SERVER_NAME, FPTR);
        int     STATUS;
        char    SERVER_NAME;    /* Typically istp6.gsfc.nasa.gov */
        FILE    *FPTR;          /* Log file pointer*/
```

5.1.1.1.5 Error Conditions

The following error messages may be returned by this routine:

[RTC_CONNECT] host unknown	Specified host is unknown
[RTC_CONNECT] error issuing socket command	Error returned from socket call; implies a network or TCP/IP problem
[RTC_CONNECT] error issuing connect command	Error returned from connect call; implies a network or TCP/IP problem
[RTC_CONNECT] unable to connect error	Unable to connect to the host; implies a network or TCP/IP problem

5.1.1.2 RTC_LOGINUSER

5.1.1.2.1 Purpose

RTC_LOGINUSER verifies the account information of the user.

5.1.1.2.2 Description

RTC_LOGINUSER verifies that the specified user has a valid account on the ISTP CDHF system. Status messages will be reported to the standard output device.

5.1.1.2.3 Interfaces

Input to RTC_LOGINUSER from the client program:

USERNAME	The username for the user's account on the NRT system
PASSWORD	The associated account password

Output from RTC_LOGINUSER to the client program:

STATUS	0 indicates success
	-1 indicates an error in authorization

5.1.1.2.4 Calling Sequence

For C:

```
STATUS = RTC_LOGINUSER (USERNAME, PASSWORD);
    int         STATUS;
    char        USERNAME[32], PASSWORD[256];
```

5.1.1.2.5 Error Conditions

The following error messages may be returned by this routine:

[RTC_LOGINUSER] error reading username command	An error occurred reading the username string.
[RTC_LOGINUSER] error sending username command	An error occurred sending the username string to or receiving the username string from the server.
[RTC_LOGINUSER] error sending password command	An error occurred sending the password string to or receiving the password string from the server.
[RTC_LOGINUSER] error from socket_read read	An error occurred on a socket operation.
[RTC_LOGINUSER] error from socket_write write	An error occurred on a socket operation.
[RTC_LOGINUSER] error sending WAIT command WAIT	An error occurred sending the command.
[RTC_LOGINUSER] error sending WTG ACK WTG	An error occurred sending the acknowledgment.

All of the above errors imply a network or TCP/IP error.

5.1.1.3 RTC_SETPACKET

5.1.1.3.1 Purpose

RTC_SETPACKET selects the data packet types to be received.

5.1.1.3.2 Description

RTC_SETPACKET selects the data packet types to be received. Selections are validated against the user's access privileges as specified in the ISTP database. (See the "ISTP Users and Privileges Form" in the Database Interface System.)

RTC_SETPACKET is called once for each packet type to be selected and then one final time with the packet type string set to "END" to signal the end of the selection process. The NRT server supports the distribution of multiple types of packets within the same mission and datatype only. For example, one session may request level-zero data from multiple instruments on WIND but may not also request data from POLAR or key parameter data. To receive data packets from multiple missions or from different data types (level-zero or key parameter) separate sessions must be established.

The valid packet type strings and their sizes are listed in Table 5–1. Note that the system currently provides data from the WIND and POLAR spacecraft only because these are the missions for which the CDHF provides real-time support. Requests for data from unsupported missions will generate an error.

Status messages will be reported to the standard output device. A return status of -1 indicates that the requested packet type is invalid or that the server software is not functional. If this return status is

encountered, check the PACKET_STATUS string to see whether the request was "Denied." If so, verify that the packet type string is correct and that the access privileges (as listed by the ISTP Users and Privileges Form in the Database Interface System) for the specified username allow access to this data type.

5.1.1.3.3 Interfaces

Input to RTC_SETPACKET from the client program:

PACKET_TYPE	Packet type specification string or 'END'

Output from RTC_SETPACKET to the client program:

PACKET_STATUS	String indicating status of packet request "Accepted"— Request was accepted "Denied"— Request was denied "Remote System Not operational"— The remote server software is not running
STATUS	0 indicates success -1 indicates an error in the packet type request or other error

5.1.1.3.4 Calling Sequence

For C:

```
STATUS = RTC_SETPACKET (PACKET_TYPE, PACKET_STATUS);
                int     STATUS;
                char    *PACKET_TYPE;
                char    *PACKET_STATUS;
```

5.1.1.3.5 Error Conditions

The following error messages may be returned by this subroutine:

Remote system not operational	Message when the server is not operational.
Denied	Access to desired data is denied.

Table 5–1. NRT Data Packet Specifications

Packet Type	Description	Packet Header + Data Record + Packet Trailer (bytes)
PO_LZ_CAM	CAMMICE level-zero packet	24 + 4552 + 8
PO_LZ_CEP	CEPPAD level-zero packet	24 + 8552 + 8
PO_LZ_EFI	EFI level-zero packet	24 + 6552 + 8
PO_LZ_HYD	HYDRA level-zero packet	24 + 12052 + 8
PO_LZ_MFE	MFE level-zero packet	24 + 2800 + 8
PO_LZ_PIX	PIXIE level-zero packet	24 + 6300 + 8
PO_LZ_PWI	PWI level-zero packet	24 + 10800 + 8
PO_LZ_SCR	Spacecraft housekeeping level-zero packet	24 + 13300 + 8
PO_LZ_TID	TIDE level-zero packet	24 + 10800 + 8
PO_LZ_TIM	TIMAS level-zero packet	24 + 10552 + 8
PO_LZ_UVI	UVI level-zero packet	24 + 14800 + 8
PO_LZ_VIS	VIS level-zero packet	24 + 14052 + 8
WI_LZ_3DP	3D-Plasma level-zero packet	24 + 12800 + 8
WI_LZ_EPA	EPAC level-zero packet	24 + 6300 + 8
WI_LZ_KON	KONUS level-zero packet	24 + 2792 + 8
WI_LZ_MFI	MFI level-zero packet	24 + 6552 + 8
WI_LZ_SCR	Spacecraft housekeeping level-zero packet	24 + 17300 + 8
WI_LZ_SMS	SMS level-zero packet	24 + 10800 + 8
WI_LZ_SWE	SWE level-zero packet	24 + 11552 + 8
WI_LZ_TGR	TGRS level-zero packet	24 + 5552 + 8
WI_LZ_WAV	WAVES level-zero packet	24 + 11552 + 8
ISTP NRT key parameter distribution is not yet implemented		
WI_K0_3DP	3D-Plasma NRT key parameters	
WI_K0_MFI	MFI NRT key parameters	
WI_K0_SWE	SWE NRT key parameters	

5.1.1.4 RTC_GETPACKET

5.1.1.4.1 Purpose

RTC_GETPACKET receives packets of data from the NRT server.

5.1.1.4.2 Description

RTC_GETPACKET receives packets of data from the NRT server, placing them in a user supplied buffer. Level-zero data is buffered by the server and all requested packets are sent at once and appear in the buffer in the order requested. Key parameter packets are sent individually as they become available. Therefore, for level-zero data, the supplied buffer must be large enough to hold the total size of the packets requested. For key parameter data, the buffer need only be large enough to hold the largest packet requested.
Each packet consists of a 24-byte packet header followed by the data (either a level-zero instrument major frame or a key parameter record) and by an 8-byte packet trailer. (see
Table 5–1).
The end-of-pass is signaled by a packet containing the 9-character string “ENDOFPASS” in the message body.

5.1.1.4.3 Interfaces

Input to RTC_GETPACKET from the client program:

BUFFER	The address of the buffer to receive the data
BUFFER_SIZE	The size (in bytes) of the buffer

Output from RTC_GETPACKET to the client program:

STATUS	>0 = number of bytes received
	-1 = error in receiving the data

5.1.1.4.4 Calling Sequence

For C:

```
NBYTES = RTC_GETPACKET (&BUFFER, (LONG*) SIZEOF(BUFF));
                int     NBYTES;
                char    BUFFER(large enough to hold data packets);
```

5.1.1.4.5 Error Conditions

The following error/status message may be returned by this subroutine:

Socket read error—socket failed	TCP/IP error

5.1.2 Sample Clients

Sample client software is available from istp1.gsfc.nasa.gov via anonymous FTP in the NRT directory or directly on the CDHF in the SYS$PUBLIC:[NRT] directory.

5.2 NRT Key Parameter Generation

The NRT key parameters are generated from the NRT telemetry data received by the CDHF NRT system from the GDCF. Telemetry data are received in real time and processed into instrument major frames. The edit and decommutation process will put the instrument major frames, along with the spacecraft housekeeping major frames, into a shared memory area on the NRT computer system. Access to the instrument data is made transparent to the KPGS by providing a support routine interface that mimics the routines used in processing playback data. Predicted orbit, predicted attitude, and, optionally, the user-supplied calibration and PI parameter files are also available as input to the NRT key parameter generation.
Key parameters generated on the NRT system are written to files on the CDHF and as a backup are written locally on the NRT system. The NRT key parameters are not put into CDF. They are created as Institute of Electrical and Electronics Engineers (IEEE) 754 binary files. Users can be notified via E-mail when key parameter files have been created, as with playback data. These files are accessible on the CDHF. Currently, the NRT key parameter generation is supported for the WIND Magnetic Fields Investigation (MFI), Solar Wind Experiment (SWE), and 3-Dimensional Plasma (3-DP) instruments only.

Note: For the Solar Wind Interplanetary Mission (SWIM) investigation, the NRT key parameters for the WIND MFI, SWE, and 3-D Plasma instruments are transmitted in NRT over the network to designated sites.

5.3 NRT Key Parameter Generation Software

The NRT environment for the KPGS has been made to look and behave as much like the KPGS environment for playback data as possible. The KPGS developed for the processing of playback data will run in the NRT environment with minimal change. The KPGS can be designed and coded in such a way that the same code will run in both environments. There are no level-zero or housekeeping data files, no SFDU headers, and no CDF key parameter files. No files are cataloged. Decommutated level-zero and housekeeping data are passed from the NRT task to the KPGS task through global sections. Event flags are used to signal when data are available. Section 5.2.2 discusses the design considerations for NRT KPGS.

5.3.1 NRT and Playback Environment Support Routine Differences

The following table summarizes the differences between the NRT and playback versions of the KPGS support routines. There are no changes in the calling sequences of any of the routines. However, some routines have restrictions placed on their use.

ICSS Support Routine	Playback/NRT Difference
ICSS_KPG_INIT	No change in use
ICSS_KPG_TERM	No change in use
ICSS_OPEN_ATT	No change in use
ICSS_OPEN_LZ	No change in use
ICSS_OPEN_ORB	No change in use

ICSS Support Routine	Playback/NRT Difference
ICSS_OPEN_SD	Used for GEOTAIL only
ICSS_RET_ATT	No change in use
ICSS_RET_HK	Sequential reads only
ICSS_RET_LZ	Sequential reads only
ICSS_RET_ORB	No change in use
ICSS_RET_SD64	Used for GEOTAIL only
ICSS_RET_SD	Used for GEOTAIL only
ICSS_KPG_COMMENT	No change in use but comments not logged
ICSS_TRANSF_ORB	No change in use
ICSS_TRANSF_ATT	No change in use
ICSS_CNVRT_TO_EPOCH	No change in use
ICSS_GET_CD	No change in use
ICSS_GET_PF	No change in use
ICSS_SPINPH_SIRIUS	Used for GEOTAIL only
ICSS_GET_REFERENCE_FILES	Used for IMP-8 only
ICSS_SPINPH_WIND_LZ	Used for WIND only
ICSS_PAYLOAD_TO_GSE	No change in use
ICSS_CNVT_FROM_RP	No change in use
ICSS_POS_OF_SUN	No change in use
ICSS_VELOCITY_TRANS	No change in use
ICSS_GCI_TO_GEODETIC	No change in use
ICSS_GEODETIC_TO_GCI	No change in use
ICSS_SD_BLK_TYP	No change in use
ICSS_INDICES	No change in use
ICSS_CNVRT_EPOCH_TO_PB5	No change in use
ICSS_TRANSF_TO_MTC	No change in use
ICSS_TRANSF_TO_MSPC	No change in use
ICSS_COMPUTE_CGMLT	No change in use
ICSS_COMPUTE_EDMLT	No change in use
ICSS_GET_FILE	No change in use
ICSS_OPEN_PO_SPINPH	Used for POLAR only
ICSS_RET_PO_SPINPH	Used for POLAR only
ICSS_OPEN_DP_ATT	Used for POLAR only
ICSS_RET_DP_ATT	Used for POLAR only
ICSS_ACF_TO_GCI	Used for SOHO only
ICSS_OPEN_SOHO_LZ	Used for SOHO only
ICSS_RET_SOHO_PACKETS	Used for SOHO only
ICSS_OPEN_SOHO_HK	Used for SOHO only
ICSS_RET_SOHO_HK	Used for SOHO only
ICSS_OPEN_SOHO_ATT	Used for SOHO only
ICSS_RET_SOHO_ATT	Used for SOHO only

ICSS Support Routine	Playback/NRT Difference
ICSS_NRT_ACTIVE	Used by KPGS running in NRT and playback environments
ICSS_WRITE_3DP_KP	For NRT processing only
ICSS_WRITE_MFI_KP	For NRT processing only
ICSS_WRITE_SWE_KP	For NRT processing only
ICSS_VAX_TO_IEEE	For NRT processing only
ICSS_UTC_TAI_OFFSET	Used for SOHO only
ICSS_GSM_SM	No change in use
ICSS_TILT_ANGLE	No change in use
ICSS_TSY	No change in use
ICSS_POS_VEL_OF_CELESTIAL	No change in use
ICSS_RTC_CONNECT	For NRT processing only
ICSS_RTC_LOGINUSER	For NRT processing only
ICSS_RTC_GETPACKET	For NRT processing only
ICSS_RTC_SETPACKET	For NRT processing only

5.3.2 NRT KPGS Design Considerations

Designing code to run in the NRT environment requires that certain considerations be made. This section describes the things that should be kept in mind when designing and coding the KPGS that is to run in the NRT environment.

- Writing and reading information from disk files is slow and adds overhead to the system. Store temporary data arrays in memory rather than writing to disk files.
- In the NRT system, the major frames of level-zero and housekeeping data arrive in real time. They are presented to the KPGS program through the ICSS_RET_LZ and ICSS_RET_HK routines as though the data are being read sequentially. It is impossible to use these routines to read data by time or by offset in the NRT system.
- ICSS_NRT_ACTIVE determines whether the KPGS is running in the playback or NRT environment. If the KPGS is being designed to run unchanged in both environments, calling this routine is required and should be done early in the program. The NRT environment does not support CDF key parameter files; therefore, calls to the CDF routines must be branched around in the NRT environment. The KPGS must include logic to check the environment and perform the appropriate subroutine calls to output the key parameter data as IEEE 754 binary data.
- When in NRT mode, the KPGS program should check the telemetry mode indicator located in the header of each level-zero data record prior to processing the data. Transition mode indicates that the spacecraft is changing from one mode to the other and the data is not associated with any mode. Unknown mode indicates that the mode of the spacecraft associated with the data in this level-zero data record is not known. [Refer to Section 3 of the DFCD (Reference 5) for the location and the values associated with the telemetry mode indicator.] In either case, the KPGS program should probably discard the use of level-zero data associated with either of these telemetry modes.
- The KPGS is responsible for filling key parameters with “fill” data. On the CDHF, the CDF key parameter file processing handles the fill data. There are no CDF key parameter files on the NRT. The user should use the fill data as defined in the include files ICSS_INC:ICSS_KP_FILL_VALUES.INC and ICSS_INC:ICSS_KP_FILL_ VALUES.H. Copies of these include files can be found in the directory defined by the logical name KPGS_SUPPORT.
- The KPGS should generate key parameters for each major frame of level-zero data.
- Because the NRT can return a zero absolute time code (ATC) time, the KPGS should check for this condition and bypass the processing of level-zero data records containing an ATC time of zero.

- The KPGS should always call ICSS_WRITE_3DP_KP, ICSS_WRITE_MFI_KP, or ICSS_WRITE_SWE_KP with the second argument, END_OF_SESSION, set to 0, which indicates data. The KPGS should call the termination routine, ICSS_KPG_TERM, when no more data are indicated (end of pass).
- The KPGS calls ICSS_WRITE_3DP_KP, ICSS_WRITE_MFI_KP, or ICSS_WRITE_SWE_KP to effect the transfer of a key parameter record from the NRT to the RDAF. The size of a record must be limited to 512 bytes. The first word (4 bytes) contains the size of the record, including the first word. The next byte of the record is reserved for an end-of-pass status. The following common block names are reserved: NRT_KP_3DP_BUFFER, NRT_KP_MFI_BUFFER, and NRT_KP_SWE_BUFFER. Figure 5–1 shows a sample structure for a key parameter record.
- The key parameter structure element KP_SIZE in the argument KP_RECORD of the NRT write routines must accurately contain the number of bytes in the key parameter record. It is imperative that this value is assigned by the KPGS program before invoking the write routine ICSS_WRITE_3DP_KP, ICSS_WRITE_MFI_KP, or ICSS_WRITE_SWE_KP. KP_SIZE is defined by the common blocks listed above.
- The status values returned by the NRT versions of the support routines are denoted by logical symbols starting with “NRT_”. These logical symbols are not defined in the ICSS_MESSAGES.INC include file. To check for one of these status values, the KPGS program must declare the appropriate symbol as “EXTERNAL” (or as a global reference in C).
- The KPGS program should always call ICSS_RET_LZ to access the level-zero data file before calling ICSS_RET_HK to access the housekeeping data file.

5.4 NRT-Specific Support Routines

There are several routines required in the NRT environment that are not used in the playback environment. The ICSS_NRT_ACTIVE support routine should be called by all KPGS applications that run on both the NRT and CDHF systems. It informs the KPGS applications of the system on which they are currently running so they can take the appropriate action. Key parameters generated in the NRT system are not created in CDF; therefore, a set of special routines is provided for writing the key parameters in the NRT environment. There is a specific write routine for each instrument supported in the NRT environment. These routines are described in the subsections that follow.

```
C
C NAME:         NRT_MFI_KPS
C
C PURPOSE:    MFI Key Parameter Definitions
C
C UNIT TYPE:   Include File
C
C DEVELOPMENT HISTORY:
C                                RELEASE/                                DESCRIPTION OF
C       AUTHOR                   CHANGE ID              DATE             CHANGE
C       K. Developer        V1.0                    12/08/93        Initial Development
C

        STRUCTURE / nrt_mfi_kps /

C KPGS SR specific files:
                INTEGER*4               kp_size                 !Size of KP structure, in
bytes
                BYTE                    mfi_end_session_flg     !Reserved by NRT KPG

C NRT KP common parameters:
                REAL*8          Epoch
                INTEGER*4               Time_PB5(3)
                INTEGER*4               data_qual_flag
                INTEGER*4               post_gap_flag
                INTEGER*4               mode_flag
                REAL*4          xgse_pos
                REAL*4          ygse_pos
                REAL*4          zgse_pos
                REAL*4          rad_distance
                REAL*4          xgsm_pos
                REAL*4          zgsm_pos

C Instrument-specific data:
                INTEGER*4               num_pt_in_avg
                REAL*4          magnitude_avg
                REAL*4          avg_of_magn
                REAL*4          rms
                REAL*4          gsm_bx
                REAL*4          gsm_by
                REAL*4          gsm_bz
                REAL*4          gsm_field_lat
                REAL*4          gsm_field_long
                REAL*4          gse_bx
                REAL*4          gse_by
                REAL*4          gse_bz
                REAL*4          gse_field_lat
                REAL*4          gse_field_long
        END STRUCTURE
```

Figure 5–1. Sample Structure for Key Parameter Record

5.4.1 ICSS_NRT_ACTIVE

5.4.1.1 Purpose

ICSS_NRT_ACTIVE allows the KPGS to determine whether it is running on the NRT or CDHF system.

5.4.1.2 Description

ICSS_NRT_ACTIVE calls a utility to translate the ICSS_NRT_ACTIVE process logical. If the translated value indicates NRT, set the returned four-character string to “NRT”. If the translated value indicates CDHF, set the returned four-character string to “CDHF”.

5.4.1.3 Interfaces

Output from ICSS_NRT_ACTIVE to the KPGS:

NRT_OR_CDHF	“NRT” indicates NRT “CDHF” indicates CDHF “ ” indicates translation failure
RETURN_STATUS	Status of translation

5.4.1.4 Calling Sequence

For FORTRAN:

```
Call ICSS_NRT_ACTIVE (NRT_OR_CDHF, RETURN_STATUS)
    CHARACTER*4     NRT_OR_CDHF          !NRT or CDHF or failure
    INTEGER*4       RETURN_STATUS
```

For C:

```
ICSS_NRT_ACTIVE (&NRT_OR_CDHF_DESC, &RETURN_STATUS);
    char    NRT_OR_CDHF[5];
    auto    $DESCRIPTOR (NRT_OR_CDHF_DESC, NRT_OR_CDHF);
                                          /*NRT or CDHF or failure*/
    int     RETURN_STATUS;
```

5.4.1.5 Error Conditions

The following errors may be recorded in the system message log by ICSS_NRT_ACTIVE:

NRT_GN_TRANS_ERR	Error translating system logical
NRT_KP_INVALID_LOG	Invalid value found in ICSS_NRT_ACTIVE

All errors returned from this support routine are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

5.4.2 ICSS_WRITE_3DP_KP

5.4.2.1 Purpose

ICSS_WRITE_3DP_KP puts the 3-D Plasma key parameter data into shared memory for the transfer task.

5.4.2.2 Description

ICSS_WRITE_3DP_KP reads an event flag to determine whether the transfer task has moved the last 3-D Plasma data record out of the global section. If it has not, then an error message is logged. A new 3-D Plasma data record is moved into the global section and an event flag is set to let the transfer task know another data record is available. This routine checks whether previously generated data have been transferred. If they have not, an error message is logged before writing out new data.

5.4.2.3 Interfaces

Input to ICSS_WRITE_3DP_KP from the KPGS:

KP_RECORD 3-D Plasma key parameter data record
END_OF_SESSION End-of-session flag

Output from ICSS_WRITE_3DP_KP to the KPGS:

RETURN_STATUS The return status from this routine

Output from ICSS_WRITE_3DP_KP to the transfer task:

KP_3DP_RECORD 3-D Plasma key parameter data record

5.4.2.4 Calling Sequence

For FORTRAN:

```
Call ICSS_WRITE_3DP_KP (KP_RECORD, END_OF_SESSION, RETURN_STATUS)
    RECORD      KP_RECORD       !3-D Plasma KP data record
    INTEGER*2   END_OF_SESSION  !0 = Data (The only value for
    END_OF_SESSION
                                !that the KPGS should use is 0.)
                                !1 = No more data
    INTEGER*4   RETURN_STATUS   !Returned status, SS$_NORMAL = Successful
```

For C:

```
ICSS_WRITE_3DP_KP (&KP_RECORD, &END_OF_SESSION, &RETURN_STATUS);
    struct      KP RECORD;      /*3-D Plasma KP data record*/
    short       END_OF_SESSION; /*0 = Data (The only value for
    END_OF_SESSION
                                  that the KPGS should use is 0.)
                                  1 = No more data */
    int         RETURN_STATUS;  /*Returned status, SS&_NORMAL =
    Successful*/
```

5.4.2.5 Error Conditions

The following warning condition may be detected by this routine:

NRT_GN_DROPPED_DATA Transfer task has not cleared event flag, meaning that it did not “finish” reading the 3-D Plasma KP record from the global section (not returned to KPGS)

The following errors may be recorded in the system message log by ICSS_WRITE_3DP_KP:

NRT_GN_READ_EF_ERR Error reading event flag
NRT_GN_SET_EF_ERR Error setting event flag

All errors returned from this support routine, except NRT_GN_DROPPED_DATA, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

5.4.3 ICSS_WRITE_MFI_KP

5.4.3.1 Purpose

ICSS_WRITE_MFI_KP puts the MFI key parameter data into shared memory for the transfer task.

5.4.3.2 Description

ICSS_WRITE_MFI_KP reads an event flag to determine whether the transfer task has moved the last MFI data record out of the global section. If it has not, then an error message is logged. A new MFI data record is moved into the global section and an event flag is set to let the transfer task know another data record is available. This routine checks whether previously generated data have been transferred. If they have not, an error message is logged before writing out new data.

5.4.3.3 Interfaces

Input to ICSS_WRITE_MFI_KP from the KPGS:

KP_RECORD	MFI key parameter data record
END_OF_SESSION	End-of-session flag

Output from ICSS_WRITE_MFI_KP to the KPGS:

RETURN_STATUS	The return status from this routine

Output from ICSS_WRITE_MFI_KP to the transfer task:

KP_MFI_RECORD	MFI key parameter data record

5.4.3.4 Calling Sequence

For FORTRAN:

```
Call ICSS_WRITE_MFI_KP (KP_RECORD, END_OF_SESSION, RETURN_STATUS)
    RECORD        KP_RECORD          !MFI KP data record
    INTEGER*2     END_OF_SESSION     !0 = Data (The only value for
    END_OF_SESSION
                                     !that the KPGS should use is 0.)
                                     !1 = No more data
    INTEGER*4     RETURN_STATUS      !Returned status, SS$_NORMAL = Successful
```

For C:

```
ICSS_WRITE_MFI_KP (&KP_RECORD, &END_OF_SESSION, &RETURN_STATUS);
    struct        KP_RECORD;         /*MFI KP data record*/
    short         END_OF_SESSION;          /*0 = Data (The only value for
                                             END_OF_SESSION that the KPGS
                                             should use is 0.)
                                             1 = No more data */
    int           RETURN_STATUS;           /*Returned status, SS&_NORMAL =
                                             Successful*/
```

5.4.3.5 Error Conditions

The following warning condition may be detected by this routine:

```
    NRT_GN_DROPPED_DATA   Transfer task has not cleared event flag, meaning
                                that it did not “finish” reading the MFI KP
    record
                                from the global section (not returned to KPGS)
```

The following errors may be recorded in the system message log by ICSS_WRITE_MFI_KP:

```
    NRT_GN_READ_EF_ERR          Error reading event flag
    NRT_GN_SET_EF_ERR           Error setting event flag
```

All errors returned from this support routine, except NRT_GN_DROPPED_DATA, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

5.4.4 ICSS_WRITE_SWE_KP

5.4.4.1 Purpose

ICSS_WRITE_SWE_KP puts the SWE key parameter data into shared memory for the transfer task.

5.4.4.2 Description

ICSS_WRITE_SWE_KP reads an event flag to determine whether the transfer task has moved the last SWE data record out of the global section. If it has not, then an error message is logged. A new SWE data record is moved into the global section and an event flag is set to let the transfer task know another data record is available. This routine checks whether previously generated data have been transferred. If they have not, an error message is logged before writing out new data.

5.4.4.3 Interfaces

Input to ICSS_WRITE_SWE_KP from the KPGS:

```
    KP_RECORD             SWE key parameter data record
    END_OF_SESSION        End-of-session flag
```

Output from ICSS_WRITE_SWE_KP to the KPGS:

```
    RETURN_STATUS         The return status from this routine
```

Output from ICSS_WRITE_SWE_KP to the transfer task:

KP_SWE_RECORD SWE key parameter data record

5.4.4.4 Calling Sequence

For FORTRAN:

```
Call ICSS_WRITE_SWE_KP (KP_RECORD, END_OF_SESSION, RETURN_STATUS)
    RECORD      KP_RECORD          !SWE KP data record
    INTEGER*2   END_OF_SESSION     !0 = Data (The only value for
    END_OF_SESSION
                                   !that the KPGS should use is 0.)
                                   !1 = No more data
    INTEGER*4   RETURN_STATUS      !Returned status, SS$_NORMAL = Successful
```

For C:

```
ICSS_WRITE_SWE_KP (&KP_RECORD, &END_OF_SESSION, &RETURN_STATUS);
    struct      KP_RECORD;         /*SWE KP data record*/
    short       END_OF_SESSION;          /*0 = Data (The only value for
                                           END_OF_SESSION that the KPGS
                                           should use is 0.)
                                           1 = No more data */
    int         RETURN_STATUS;           /*Returned status, SS$_NORMAL =
                                           Successful*/
```

5.4.4.5 Error Conditions

The following warning condition may be detected by this routine:

NRT_GN_DROPPED_DATA Transfer task has not cleared event flag, meaning that it did not “finish” reading the SWE KP record from the global section (not returned to KPGS)

The following errors may be recorded in the system message log by ICSS_WRITE_SWE_KP:

NRT_GN_READ_EF_ERR	Error reading event flag
NRT_GN_SET_EF_ERR	Error setting event flag

All errors returned from this support routine, except NRT_GN_DROPPED_DATA, are fatal. The KPGS application must check the return status immediately after every support routine call. If the status indicates an error, the KPGS application must terminate processing.

5.4.5 ICSS_VAX_TO_IEEE

5.4.5.1 Purpose

ICSS_VAX_TO_IEEE performs data format conversion from VAX/virtual memory storage (VMS) representation to IEEE representation.

5.4.5.2 Description

This support routine invokes a CDF function to perform data format conversion. For the NRT support, the KPGS data need to be converted to IEEE format before they are written to the shared memory for the transfer task. In other words, ICSS_VAX_TO_IEEE is called before ICSS_WRITE_3DP_KP, ICSS_WRITE_MFI_KP, or ICSS_WRITE_SWE_KP.

Note that the KPGS program that uses the ICSS_VAX_TO_IEEE routine must be linked with the latest CDF library (i.e., CDF Version 2.3 or higher).

It should be further noted that ICSS_VAX_TO_IEEE should NOT be used to convert the NRT specific fields in the output buffer to the IEEE format. These fields include the field containing the size of the KP structure in bytes and the field containing the end-of-pass flag. The size of the KP structure is located in bytes 1-4. The end-of-pass flag is in byte 5.

5.4.5.3 Interfaces

Input to ICSS_VAX_TO_IEEE from the KPGS:

BUFFER	Array containing data to be converted
DATA_TYPE	CDF data type in buffer Note: CDF include file must be included in the KPGS program to access the CDF data type.
NUM_ELEM	Number of data elements in buffer

Output from ICSS_VAX_TO_IEEE to the KPGS:

BUFFER	Array containing converted data

5.4.5.4 Calling Sequence

This support routine is used for all CDF types of data conversions.

The following two examples demonstrate how this routine should be called.

For FORTRAN:

```
Call ICSS_VAX_TO_IEEE (BUFFER, %VAL(DATA_TYPE), %VAL(NUM_ELEM))
    REAL*4      BUFFER(NUMELEMS)        !Suppose VAX REAL*4 data
    conversion is
                                        !desired; NUMELEMS should be
    replaced
                                        !with number of data elements to be
                                        !converted
    INTEGER*4   DATA_TYPE               !Assign CDF_REAL4 to this input
    argument
                                        !in the KPGS program
    INTEGER*4   NUM_ELEM                !Number of data elements to be
    converted
```

For C:

```
ICSS_VAX_TO_IEEE (BUFFER, DATA_TYPE, NUM_ELEM);
    double  BUFFER[NUMELEMS];       /*Suppose VAX REAL*8 data conversion is
                                      desired; NUMELEMS should be replaced
    with
                                      number of data elements to be converted*/
    long    DATA_TYPE;          /*Assign CDF_REAL8 to this input argument in the
                                      KPGS program*/
    long    NUM_ELEM;               /*Number of data elements to be converted*/
```

5.4.5.5 Error Conditions

This support routine has no error path.

5.5 NRT KPGS Testing

The KPGS program that is designed to use the NRT capabilities may be tested using simulators provided for this purpose on the CDHF. These simulators replace the NRT decommutation processes that would transfer telemetry to the KPGS programs in the operational environment. They provide a source of telemetry whose volume and rate are controlled with user-specified parameters. Also provided is a simulator that accepts the key parameter output data from the KPGS program and writes these data to a log file where they may be inspected. Orbit, attitude, calibration, and PI parameter files are accessed in the same manner as with the KPGS program running in the CDHF environment.

The scenario for testing the KPGS program in the NRT environment is as follows:

1. Using one of the VAX editors, create NAMELIST files that control the data rate and volume of the instrument and housekeeping (optional) telemetry.
2. Create a command file that executes the CDHF simulators and the KPGS program.
3. Execute a CDHF-supplied command file to initialize the NRT environment.
4. Execute the command file created in step 2.
5. Inspect the key parameter and log files created during the test.

The following subsections explain each of these steps in detail.

5.5.1 Creating the NAMELIST Files

Instrument-specific simulators are used to supply the instrument and housekeeping telemetry to the KPGS program. Each uses a FORTRAN NAMELIST file to control the amount and rate of the telemetry. The name of the file that controls the instrument telemetry simulator is LZ_NAMELIST.DAT, and the name of the housekeeping telemetry simulator is HK_NAMELIST.DAT. Both must exist in the directory from which the test is to be run. The format of these two files are identical, each being an American Standard Code for Information Interchange (ASCII) text file containing three parameters: START_DELAY, FRAME_DELAY, and NUMBER_MAJ_FRMS. START_DELAY defines the number of seconds that the simulator waits before sending the first frame. FRAME_DELAY is the number of seconds that the simulator waits between frame

transmissions. NUMBER_MAJ_FRMS is the number of major frames that the simulator will transfer to the KPGS program before terminating. A sample of the NAMELIST file is as follows:

```
$CONTROL NUMBER_MAJ_FRMS = 10, START_DELAY = 10.0,
  FRAME_DELAY = 2.0
$END
```

This example would cause the simulator to transmit 10 major frames, waiting 10 seconds before the first frame, and 2 seconds between each subsequent pair of frames. The "$" which starts the file must be in column 2. Normally, the NAMELIST files for the instrument and housekeeping simulators would be identical because equal data rates and volumes would be normal. However, by varying these parameters, certain anomalous conditions can be simulated.

5.5.2 Creating the NRT Command File

The command file that executes the KPGS test sequence is a text file containing VMS commands that define the support files to be used during the test, that execute the simulators as detached processes, and that run the KPGS program. An example of this command file follows.

Note: A machine-readable version of this command file is available in KPGS_IT:[KPGSIT.NRT.COM] MFI_NRT_SAMPLE_RUN.COM.

```
$ DELETE *MFI*.out;*
$ DELETE MH.DAT;*
$ DELETE OPS.DAT;*
$ DELETE *MFI*.ERR;*
$ DELETE *MFI*.LOG;*
$
$ DEFINE NRT_ORB_FILE WIND_ORB:WI_OR_DEF_19920921_V02
$ DEFINE NRT_ATT_FILE WIND_ATT:WI_AT_DEF_19920921_V01
$ DEFINE C001  KPG_DATA:GE_CD_EPI_19910930_V01.DAT
$ DEFINE NRT_PARM_FILE       KPG_DATA:WI_PF_MFI_19020921.DAT
$ DEFINE USR_MSG_FILE                MFI_NRT_SAMPLE:LOG
$
$!       Run the simulated mailboxes
$ @CDHF_DEV:[sdev.com]sim_DEF
$ @CDHF_DEV:[sdev.com]sim_ops
$ @CDHF_DEV:[sdev.com]sim_mh
$
$        wait 00:00:02
$
$!       Run the detached process that sets up the NRT globals
$
$ RUN/DETACH EXE_DIR:CREATE_GLOBALS_WIND/-
PROCESS_NAME=NRT_WIND_GBLS/OUT=WIND.OUT/error=WIND.ERR
$
$!       Wait for the globals to be defined
$
$        wait 00:00:02
$
$!       Run the detached process that will provide level-zero data to the KPGS program
$
$ RUN/DETACH EXE_DIR:PUT_MFI_MF/PROCESS_NAME=PUT_HK/INPUT=WI_-
LZ_SCR_19920921_V01.DAT/OUT=WIND_MFI_MF.OUT/error=wind_MFI_HK.ERR
$
$!       Run the detached process that will accept the KPGS program output data
$
$ RUN/DETACH EXE_DIR:TRANSPORT_MFI/PROCESS_NAME=TRANSPORT_MFI/-
OUT=TRANSPORT_MFI.OUT/error=TRANSPORT_MFI.ERR
$! SET NOON
```

```
$! SET PROCESS /PRIVILEGE=NOALL
$! SET PROCESS /PRIVILEGE=TMPMBX
$
!       Run the KPGS program
$ RUN EXE_DIR:MFI_NRT_SAMPLE.EXE
$
$       Stop the detached processes
$ STOP NRT_WIND_GBLS
$ STOP PUT_MFI_MF
$ STOP PUT_MFI_HK
$ STOP TRANSPORT_MFI
$ STOP KPGSIT_1
$ STOP KPGSIT_2
$ SET ON
$EXIT
```

The first five commands define logical symbols which determine the names of the orbit, attitude, calibration, PI parameter, and user message files that will be used during the test. Each of these lines should be included, and any files that are not being used should be left blank. The command(s) that selects the calibration file(s) uses the same symbol(s) as used for the instrument component argument passed to the ICSS_GET_CD support routine (see Section 4.18 of this document) in the KPGS program. The filenames used in these commands are physical filenames and should include the file directory path and extension.

The next three commands run the simulated mailboxes that are used by all KPGS programs. Following these is the command to run the NRT process that defines the global areas used by the NRT processes. This process must be run prior to the execution of any of the NRT simulators. After this process is executed, the command file waits 2 seconds to allow the globals to be established before invoking the simulators.

The next command executes the level-zero telemetry simulator as a detached process. There are separate telemetry simulators provided for each instrument. The name of the simulator is PUT_<instrument>_LZ, where <instrument> is the three-character identifier (e.g., MFI, SWE, 3DP) for the instrument whose KPGS program is being tested. For instance, the simulator for the WIND MFI instrument used in this example is named PUT_MFI_LZ.

The INPUT Qualifier in this command must be assigned the name of the level-zero telemetry file that shall be used as a source of the telemetry. This can be any valid CDHF playback-mode telemetry file. The OUTPUT Qualifier is assigned the name of a log file to which the simulator writes any system messages indicating anomalies in its processing.

The next command executes the housekeeping telemetry simulator. This line can be omitted if the housekeeping data are not used by the KPGS program. Here again, the INPUT Qualifier is assigned the name of a valid playback telemetry file, this time one containing housekeeping telemetry. The OUTPUT Qualifier is again assigned the name of a log file for the simulator.

The next command executes the simulator that accepts the key parameter data from the KPGS program and logs them to a file. There are separate simulators provided for each instrument. The name of this simulator is TRANSPORT_<instrument>, where <instrument> is the three-character identifier for the instrument whose KPGS program is being tested. The simulator in the example is named TRANSPORT_MFI. The OUTPUT Qualifier in this command is assigned the name of the file to which the key parameter data are to be logged.

The next command executes the KPGS program, which will run until the end of the level-zero or housekeeping data. Following the execution of the KPGS program, the command file stops any simulator processes that might still be running. It then stops the simulated mailboxes. The commands to stop the mailboxes in the example must be changed to reflect the account under which the test is being run (see example).

5.5.3 Establishing the NRT Environment

Next a CDHF-supplied command file must be run to establish the NRT environment. This is done by typing @KPGS_IT : [KPGSIT.NRT.COM] nrt_def. This command file must be invoked once at each logon session. (KPGS_DEV should have been executed before the execution of this command file.)

5.5.4 Executing the KPGS Program

The KPGS command file may now be invoked to test the KPGS program. The program may be linked with or without the debugger. However, since the simulators send data at a fixed rate, it may not be possible to avoid loss of data messages when running the debugger.

The program should run until one of the telemetry simulators has exhausted its data. Once this occurs, the simulator should signal the KPGS program that the end of a pass has occurred. This is evidenced in the program as an end-of-file status from the ICSS_RET_LZ or ICSS_RET_HK support routine. The KPGS program upon detecting this condition should then perform a normal termination. This includes calling the ICSS_KPGS_TERM support routine.

5.5.5 Inspecting the Test Results

The user message and key parameter files can be inspected. The names of these files are defined in the command file. If any problems have occurred during the test, the log files for the simulators, as well as the simulated system operator log (name mh.dat), are available for inspection.

SLP file does not contain the body requested

SECTION 6—SPECIAL PROGRAMMING NOTES

6.1 B Field Major Frame Averages

Components required to determine the B field major frame averages are in the POLAR housekeeping data file. The four elements required to calculate the B field major frame averages for the Magnetic Field Explorer (MFE) magnetometer and their locations in the POLAR housekeeping data are described below.

- MNTSCALE (8-bit binary integer used for scaling the vectors) in byte 12, minor frame 63
- BX (16-bit, two's complement uncalibrated integer vector component) in bytes 11 and 12, minor frame 87
- BY (16-bit, two's complement uncalibrated integer vector component) in bytes 11 and 12, minor frame 99
- BZ (16-bit, two's complement uncalibrated integer vector component) in bytes 11 and 12, minor frame 111

To access the MFE magnetometer data from the POLAR spacecraft housekeeping file, a user would call the ICSS routine to read in the spacecraft housekeeping file. The user would then use his/her code for extracting the MFE magnetometer data. An example of such code is given in Figure 6–1.

The magnitude range of the raw BX, BY, BZ values could extend to ±32767 data numbers (dn) depending on the instrument setup (sensor and scale factor). Under normal operations, the goal is to maintain less than full-scale values (i.e., ±25000 dn).

The detail format of the POLAR housekeeping data file is shown in Section 3.3 of the *Data Format Control Document (DFCD) Between the International Solar-Terrestrial Physics (ISTP) Mission Operations and Systems Development Division (MOSDD) Ground Data Processing System and the ISTP Mission Investigators* (Reference 5).

To compute the B components in nanoteslas (nT), the following operations are performed using floating-point arithmetic:

- Bxcal = Bx av/(Cnts/nT)
- Bycal = By av/(Cnts/nT)
- Bzcal = Bz av/(Cnts/nT)

These major-frame-averaged (9.2 seconds) data are despun, corrected vector components in spacecraft coordinates.

6.2 Status of Instruments

The spacecraft housekeeping data files for both WIND and POLAR contain the status of all of the instruments. The location of this information can be obtained from the DFCD (Reference 5).

```
C
C       Example Code for Extracting MFE Data
C       From the POLAR SPACECRAFT Housekeeping Data
C
        subroutine get_polar_mag_data (major_frame, mntscale, bx, by, bz)
              byte major_frame(*)                    !Buffer containing major frame
              integer*4     mntscale          !Magnetometer scale factor
              integer*4     bx, by, bz              !Magnetometer vector

              integer*4     frame_no,               !Minor frame number
                            index,                  !Index into major frame buffer
                            frame_size/9/,          !Minor frame size
                            iword                   !Equivalenced variable used to transfer bytes
                                                    !to vector

              byte          bword (4)               !Equivalenced array

              equivalence (iword, bword(1))

C
C       Get the MNTSCALE value out of byte 8 (array index 9) of minor frame 63
C
              frame_no = 63
              index = 300 + (frame_no*frame_size) + 9

              bword(1) = major_frame (index)
              bword(2) = 0
              mntscale = word
C
C       Get the BX value out of bytes 7 and 8 of minor frame 87
C
              frame_no = 87
              index = 300 + (frame_no*frame_size) + 8

              bword(2) = major_frame (index)
              bword(1) = major_frame (index + 1)
              bx = iword
C
C       Get the BY value out of bytes 7 and 8 of minor frame 99
```

Figure 6–1. Code To Extract MFE Magnetometer Data From POLAR Housekeeping Data File (1 of 2)

```
C
                frame_no = 99
                index = 300 + (frame_no*frame_size) + 8

                bword(2) = major_frame (index)
                bword(1) = major_frame (index + 1)
                by = iword
C
C       Get the BZ value out of bytes 7 and 8 of minor frame 111
C
                frame_no = 111
                index = 300 + (frame_no*frame_size) + 8

                bword(2) = major_frame (index)
                bword(1) = major_frame (index + 1)
                bz = iword

                return
                end
```

Figure 6–1. Code To Extract MFE Magnetometer Data From POLAR Housekeeping Data File (2 of 2)

6.3 Magnetic Local Time

Magnetic time is essentially a longitude coordinate. Two magnetic local time systems are proposed for use within ISTP: the eccentric dipole magnetic local time (EDMLT) system and the corrected geomagnetic local time (CGMLT) system.

6.3.1 EDMLT

This section discusses the advantages and disadvantages of this system.

6.3.1.1 Advantages

The eccentric dipole model produces an orthogonal system. It is reversible, retains spatial information, can be used in mathematical modeling, and can form the basis for field line tracing. It can be based on any model field, e.g., International Geophysical Reference Field 1990, the adopted ISTP field model. It can be converted simply to geodetic coordinates and hence to any other system, e.g., centered dipole, Polar Anglo-American Conjugate Experiment, corrected geomagnetic. The conversion algorithm is easily portable. It has proven to be moderately useful for ordering ionospheric phenomena.

6.3.1.2 Disadvantages

Beyond the ionosphere, it becomes a mere label that does, however, provide a measure of the position relative to the Earth-Sun line. Its intended use is for

- High-latitude particle data
- Comparison with existing databases

- Studies where differentiability or integrability of the coordinate system is important
- Doing modeling

6.3.2 CGMLT

This section discusses the advantages and disadvantages of this system.

6.3.2.1 Advantages

This system currently is used by the Dual Auroral Radar Network (DARN) community and is the chosen coordinate system for SUPERDARN. It is suitable for studying low-altitude conjugate phenomena and has been proven to be quite useful for ordering and comparing ionospheric phenomena. It can be converted easily to geodetic coordinates and hence to any other system, e.g., attitude-adjusted corrected geomagnetic coordinate, eccentric dipole, centered dipole. The conversion algorithm is easily portable.

6.3.2.2 Disadvantages

CGMLT is not an orthogonal system. It requires several sets of constants for the range of altitudes to be covered and an interpolation scheme for intermediate altitudes. Extension to very high altitudes becomes dependent on the model used for field line tracing.
The information in these subsections is an abstract from the ISTP KPGS standards and convention document (Reference 1). More detail can be found in Appendix H, pages H-5 and
H-6, of Reference 1.

APPENDIX A—SAMPLE TEST PROGRAMS

There are currently nine sample KPGS programs. All sample programs reside on the CDHF in the SYS$PUBLIC:[KPGS] directory.
For each sample program, a command file will compile and link the program. These command files are named

"program_name"_compile.com

and

"program_name"_link.com

Note that on return from every support routine call, the KPGS must check the return or completion status for errors. Although support routines record errors in the ISTP CDHF system message log, the KPGS applications are responsible for maintaining processing control.
Table A–1 briefly describes the sample programs.

Table A–1. Sample Programs

Program Name	Language	Description
F_SAMPLE	FORTRAN	Sample FORTRAN KPGS that uses level-zero, housekeeping, orbit, attitude, and calibration data files
C_SAMPLE	C	Sample C KPGS that uses level-zero, housekeeping, orbit, attitude, and calibration data files
S_SAMPLE	C	Sample C KPGS that uses a SIRIUS data file
FS_SAMPLE	FORTRAN	Sample FORTRAN KPGS that uses a SIRIUS data file
S64_SAMPLE	C	Sample C KPGS that uses a SIRIUS data file via the ICSS_RET_SD64 support routine
MFI_NRT_SAMPLE	FORTRAN	Sample FORTRAN KPGS that uses the NRT system for the WIND MFI instrument
SWE_NRT_SAMPLE	C	Sample C KPGS that uses the NRT system for the WIND SWE instrument
SOHO_FOR_SAMPLE	FORTRAN	Sample FORTRAN KPGS program demonstrating the use of the SOHO support routines
SOHO_C_SAMPLE	C	Sample C KPGS program demonstrating the use of the SOHO support routines

APPENDIX B—SAMPLE CDF SKELETON TABLES

The following are sample skeleton tables for SOHO's Charge, Element, and Isotope Analysis System (CELIAS) and Energetic and Relativistic Nuclei and Electron (ERNE) Experiment key parameter files:

SO_K0_CELS_V01.SKELETON_TABLE

SO_K0_ERNE_V01.SKELETON_TABLE

These sample skeleton tables illustrate the use of the required CDF attributes, r- and z-variables, and optional attributes and variables.

All sample skeleton tables reside on the CDHF in the SYS$PUBLIC:[KPGS] directory.

NOTE: The variable epoch must not be a z-variable for the SOHO ERNE skeleton table.

APPENDIX C—ERROR MESSAGES

C.1 Severe Error Messages

The following are indicators of severe errors:

ICSS_ATT_BUF_INV	Attitude buffer not applicable
ICSS_ATT_FILE_INV	Attitude files do not contain request
ICSS_ATT_READ_ERR	Error reading attitude file
ICSS_BAD_DATA	Invalid data record
ICSS_CATALOG_KP_FILE_ERROR	Failure to catalog KP file
ICSS_CDF_INQ	Error on CDF inquire
ICSS_CLOSE_ERR_CFL	Error closing CONC_FILE_LIST
ICSS_CLOSE_ERR_TAEF	Error closing TAE data file
ICSS_CLOSE_ERROR_UDS_SEND_ID	Error closing UDS_SEND_ID data store
ICSS_CLOSE_KP_UPD_QUAL_ERR	Error on closing KP quality update request file
ICSS_CLOSE_OLA_ERR	Error closing assembled level-zero data file
ICSS_CLOSE_OLZ_ERR	Error closing online level-zero data file
ICSS_CMD_LUN_ERR	Error obtaining logical unit number for transfer command file
ICSS_CMD_OPEN_ERR	Error opening the transfer command file
ICSS_CMD_WRITE_ERR	Error writing the transfer command file
ICSS_COOR_SYS_INV	Coordinate system request invalid
ICSS_CR_DEF	Error creating deferred request
ICSS_DATA_OPEN_ERR	Error opening data specification file
ICSS_DATA_READ_ERR	Error reading the transfer filenames
ICSS_DATA_WRITE_ERR	Error writing to data specification file
ICSS_DATABASE_COMMIT	Error committing changes to the database
ICSS_DB_CONNECT_ERR	Error on attempt to connect to the ICSS database
ICSS_DB_DAILY_XFER_ERR	Error retrieving daily_xfer_history table
ICSS_DB_ERR	Error accessing database
ICSS_DB_GET_DECOM	Error getting information from decom_spec table
ICSS_DB_GET_DESCR_ERR	Error getting descriptor information from database
ICSS_DB_GET_MISS_ERR	Error getting mission information from database
ICSS_DB_GET_PHYS_ERR	Error obtaining !AS physical filename from the database for logical identifier !AS,/fao=2
ICSS_DB_GET_SEND_ID	Error getting next send id from database
ICSS_DB_GET_VERSION_ERR	Error obtaining latest version from database for !AS file,/fao=1
ICSS_DB_INSERT_ERR	Error inserting into daily_xfer_history table of database
ICSS_DB_LOGOFF_ERR	Error logging off the database
ICSS_DB_MISSING_ADI_NUM	ADI number not in database for !AS file,/fao=1
ICSS_DB_REF_FILE_ERR	Error getting reference files from database
ICSS_DB_RETRIEVE_ERR	Error retrieving UDS request from database
ICSS_DB_USER_INFO_ERR	Error retrieving from istp_user table
ICSS_DB_USER_PRIV_ERR	Error getting information from USER_ACCESS_PRIVILEGE table
ICSS_DECOM_CLOSE_ERR	Error closing the decom specification file
ICSS_DECOM_CLOSE_NEW	Error closing the modified decom specification file, no changes made
ICSS_DECOM_DB_DELETE	Error deleting an entry from the database
ICSS_DECOM_DB_INSERT	Error creating an entry into the database
ICSS_DECOM_DB_UPDATE	Error updating an entry in the database

ICSS_DECOM_DEL_N_CMT	Error deleting old decom specification file and committing changes to database
ICSS_DECOM_DELETE_ERR	Error deleting the decom specification file
ICSS_DECOM_DELETE_OLD	Error deleting old decom specification file
ICSS_DECOM_GETLUN_ERR	Error retrieving logical unit number
ICSS_DECOM_MISMATCH	Decom specification file does not match mission/instrument of level-zero file
ICSS_DECOM_OPEN_ERR	Error opening the decom specification file
ICSS_DECOM_OPEN_NEW	Error opening the decom specification file, no changes made
ICSS_DECOM_OPEN_OLD	Error opening the old decom specification file, no changes made
ICSS_DECOM_READ_ERR	Error reading from the decom specification file
ICSS_DECOM_RENAME_ERR	Error renaming the decom specification file, no changes made
ICSS_DECOM_WRITE_ERR	Error writing to the decom specification file
ICSS_DEL_ATTR_FILE	Error deleting attribute file
ICSS_DELETE_CFL	Error deleting Concatenate File List
ICSS_DELETE_JOB_ERROR	Error deleting transfer request from job queue
ICSS_EREAD_TIMCOEF_FILE	Error reading from the timing coefficient. file
ICSS_ERR_CON_EPOCH	Error converting EPOCH time to IDTF time
ICSS_ERR_CONVERTING	Error converting VMS time to IDTF time
ICSS_ERR_DELETING_MAIL_FILE	Error deleting get data from DDF mail file
ICSS_ERR_READ_CDF	Error reading from CDF
ICSS_ERR_READ_SLP	Error reading from the SLP file
ICSS_ERR_SPAWNING_DDF_ MAIL_CMD	Error spawning the get DDF data mail command file
ICSS_EX_NUM_TRIES	Number of attempts for this job exceeded
ICSS_FILENAME_TOO_LONG	Specified filename exceeds max length
ICSS_FILE_SIZE_ERR	Error obtaining file size
ICSS_FORM_FAIL	Error executing SQL form
ICSS_GET_HK_REC_SIZE	Error obtaining the record size of the housekeeping file
ICSS_GET_LUN_ERROR	Error getting logical unit number
ICSS_GET_LUN_ERROR_UDS_ SEND_ID	Error getting logical unit number
ICSS_GET_LZ_REC_SIZE	Error obtaining the record size of the level-zero file
ICSS_GET_SD_FRAMES	Error getting the number of frames from the SIRIUS file, !AS,/fao=1
ICSS_GETLUN_PR_HK	Error obtaining a logical unit number for the primary housekeeping file
ICSS_GETLUN_PR_LZ	Error obtaining a logical unit number for the primary level-zero file
ICSS_GETLUN_PR_MAG	Error obtaining a logical unit number for the primary magnetometer file
ICSS_GETLUN_SC_HK	Error obtaining a logical unit number for the secondary housekeeping file
ICSS_GETLUN_SC_LZ	Error obtaining a logical unit number for the secondary level-zero file
ICSS_GETLUN_SC_MAG	Error obtaining a logical unit number for the secondary magnetometer file
ICSS_GETLUN_SD	Error obtaining a logical unit number for the SIRIUS file, !AS,/fao=1
ICSS_INQUIRE_LZ_ERR_ALZ	Error opening assembled level-zero data file
ICSS_INQUIRE_LZ_ERR_OLZ	Error determining online level-zero data file record size
ICSS_ILVMSTI	Invalid VMS time specified

ICSS_INIT_FAIL_PUT_MSG	Failure to assign mailbox channel; unable to call ICSS_PUT_MSG
ICSS_INIT_FAIL	ICSS_INIT failure; unable to assign mailbox channel
ICSS_INV_HK_READCALL	Requested file to read does not exist—this condition should not happen
ICSS_INV_HK_REQTYPE	Invalid housekeeping request type—request must be BY OFFSET, BY TIME, or IN SEQUENCE
ICSS_INV_HK_REQTIME	Invalid requested time—the time is not in the range of the housekeeping files
ICSS_INV_LZ_READCALL	Requested file to read does not exist—this condition should not happen
ICSS_INV_LZ_REQTYPE	Invalid level-zero request type—request must be BY OFFSET, BY TIME, or IN SEQUENCE
ICSS_INV_LZ_REQTIME	Invalid requested time—the time is not in the range of the level-zero files
ICSS_INV_LZSTART_TIME	Invalid LZ start time
ICSS_INV_LZSTOP_TIME	Invalid LZ stop time
ICSS_INV_PACK_TYPE	Invalid package type for spin-stabilized data
ICSS_INV_SD_OFF	Invalid SIRIUS minor frame offset specified, must be greater than 0
ICSS_INV_SD_READCALL	Requested file to read does not exist—this condition should not happen
ICSS_INV_SD_REQTIME	Invalid requested time—the time is not in the range of the SIRIUS files
ICSS_INV_SD_REQTYPE	Invalid SIRIUS request type—request must be BY OFFSET, BY TIME, or IN SEQUENCE
ICSS_INV_SRC_SYS	Invalid source coordinate system
ICSS_INV_SRC_TARGET_SYS	Invalid source and target coordinate systems
ICSS_INV_STABIL_TYPE	Invalid stabilization type for ret attitude routine
ICSS_INV_START_TIME	Invalid or non-existent KP file epoch start time
ICSS_INV_STOP_TIME	Invalid KP file epoch stop time
ICSS_INV_TARGET_SYS	Invalid target coordinate system
ICSS_INV_TIME_INT	Invalid time interval specified
ICSS_INV_TIME_ERR	Error obtaining file times for !AS file,/fao=1
ICSS_INV_TIME_PR_ATT	Error obtaining the file times for the primary attitude file
ICSS_INV_TIME_PR_ORB	Error converting the file times for the primary orbit file
ICSS_INV_TIME_SC_ATT	Error obtaining the file times for the secondary attitude file
ICSS_INV_TIME_SC_ORB	Error converting the file times for the secondary orbit file
ICSS_INV_VEC_INP	Attitude is colinear with reference vector
ICSS_INV_VMS_TIME	Invalid VMS time specified
ICSS_INV_XFER_TIME	Invalid transfer time specified
ICSS_INVALID_BYTE_RANGE	Invalid byte range for decom specification
ICSS_INVALID_BYTE_SPEC	Invalid byte number for decom specification
ICSS_INVALID_DECOM_SPEC	Invalid decom specification
ICSS_INVALID_MF_DECOM_SPEC	Invalid minor frame for decom specification
ICSS_INVALID_MF_RANGE	Invalid minor frame range specified
ICSS_INVALID_MF_RANGE_ INCREMENT	Invalid increment specified for range
ICSS_INVALID_MF_RANGE_START	Invalid minor frame specified for start of range
ICSS_INVALID_MF_RANGE_STOP	Invalid minor frame specified for end of range
ICSS_INVIDTFYR	Invalid idtf year
ICSS_INVIDTFDAY	Invalid idtf day
ICSS_INVIDTFMSEC	Invalid idtf millisecond
ICSS_IO_ERROR	Error reading/writing from/to std$input
ICSS_JOB_EXECUTING	Job cannot be deleted or updated, it is currently executing

ICSS_KP_OUTFILE_OPEN_ERR	Error opening KP output file
ICSS_KP_OUTFILE_WRITE_ERR	Error writing KP output file
ICSS_KPGS_INFILE_OPEN_ERR	Error opening KPGS parameter input file
ICSS_KPGS_INFILE_READ_ERR	Error reading from KPGS parameter input file
ICSS_KPGS_INFILE_CLOSE_ERR	Error closing KPGS parameter input file
ICSS_KPGS_INFILE_INV_REC	Invalid records in the KPGS parameter input file
ICSS_LZ_OUT_OF_RANGE	Level-zero data file out of requested time range
ICSS_LUN_ERR	Error obtaining a logical unit number
ICSS_LUN_ERR_DATA_FILE_SPEC	Error opening logical unit number for data file specification
ICSS_LUN_ERR_MAIL_FILE	Error opening logical unit number for the mail file
ICSS_MAIL_OPEN_ERR	Error on attempt to open mail file
ICSS_MAIL_WRITE_ERR	Error on attempt to write to mail file
ICSS_MF_OUT_OF_RANGE	Minor frame out-of-range for decom specification
ICSS_MF_RANGE_PREV_SET	Attempt to reset previously defined minor frame range
ICSS_NEW_SPEC_FILE	Error creating new data transfer file specification
ICSS_NO_CD_FILE	The specified instrument calibration file not in catalog
ICSS_NO_CDHF_DPA	Logical file for CDHF DPA does not exist for date
ICSS_NO_DATA	No data records in file
ICSS_NO_FILE_ERR	!AS file does not exist, /fao=1
ICSS_NO_MAIL	Unable to send user mail
ICSS_NO_PF_FILE	Instrument parameter file not in catalog
ICSS_NO_PR_ATT	Primary attitude file does not exist
ICSS_NO_PR_HK	Primary housekeeping file does not exist
ICSS_NO_PR_LZ	Primary level-zero file does not exist
ICSS_NO_PR_MAG	Primary magnetometer file does not exist
ICSS_NO_PR_ORB	Primary orbit file does not exist
ICSS_NO_PR_SD	Primary SIRIUS files do not exist
ICSS_NOT_KP_UPD_QUAL_FILE	Not valid KP quality update request file
ICSS_NOT_KP_UPD_QUAL_USER	User not owner of KP quality update request file
ICSS_NSSDC_COPY_ERR	Error performing copy of files to NSSDC
ICSS_OPEN_ATTR_FILE	Error opening attribute file
ICSS_OPEN_ERR	Error opening the data specification file
ICSS_OPEN_FILE_ERR	Error opening !AS file,/fao=1
ICSS_OPEN_ERR_ALZ	Error opening assemble level-zero data file
ICSS_OPEN_ERR_DATA_FILE_SPEC	Error opening data file specification
ICSS_OPEN_ERR_MAIL_FILE	Error opening mail message file
ICSS_OPEN_ERR_OLZ	Error opening online level-zero data file
ICSS_OPEN_ERR_SLZ	Error opening the scratch decommutated level-zero data file
ICSS_OPEN_ERR_TAEF	Error opening TAE data file
ICSS_OPEN_ERROR_UDS_SEND_ID	Error opening UDS_SEND_ID data store
ICSS_OPEN_FILE_SD	Error opening the SIRIUS file, !AS,/fao=1
ICSS_OPEN_KP_UPD_QUAL_ERR	Error opening KP quality update request file
ICSS_OPEN_PR_ATT	Error opening the primary attitude file
ICSS_OPEN_PR_HK	Error opening the primary housekeeping file
ICSS_OPEN_PR_LZ	Error opening the primary level-zero file
ICSS_OPEN_PR_MAG	Error opening the primary magnetometer file
ICSS_OPEN_PR_ORB	Error opening the primary orbit file
ICSS_OPEN_SC_ATT	Error opening the secondary attitude file
ICSS_OPEN_SC_HK	Error opening the secondary housekeeping file
ICSS_OPEN_SC_LZ	Error opening the secondary level-zero file
ICSS_OPEN_SC_MAG	Error opening the secondary magnetometer file
ICSS_OPEN_SC_ORB	Error opening the secondary orbit file
ICSS_OPEN_SLP	Error opening the SLP file
ICSS_OPEN_SPEC_FILE	Error opening data transfer specification file

ICSS_OPEN_TIMCOEF	Error opening the timing coef. file
ICSS_ORB_BUF_INV	Orbit buffer not applicable
ICSS_ORB_FILE_INV	Orbit files do not contain request
ICSS_ORB_READ_ERR	Error reading orbit file
ICSS_OUTPUT_FILE_CLOSERR	Output file '!AS' cannot be closed, /fao=1
ICSS_PI_OPEN_ERR	Error opening PI account information file
ICSS_PI_READ_ERR	Error reading PI account information file
ICSS_PUT_SFDU_ERR	Error putting parameter to a SFDU header record
ICSS_READ_ATTR_FILE	Error reading attribute file
ICSS_READ_ERR_DATA_FILE_SPEC	Error reading from the data file specification
ICSS_READ_ERROR_LZ_HEADER	Error reading level-zero header record from disk
ICSS_READ_ERROR_LZ_RECORD	Error reading level-zero data record from disk
ICSS_READ_ERROR_SFDU	Error reading SFDU header record from disk
ICSS_READ_ERROR_UDS_SEND_ID	Error reading UDS_SEND_ID data store
ICSS_READ_KP_UPD_QUAL_ERR	Error reading KP quality update request file
ICSS_READ_PR_HK	Error reading the primary housekeeping file data record
ICSS_READ_PR_LZ	Error reading the primary level-zero file data record
ICSS_READ_SC_HK	Error reading the secondary housekeeping file data record
ICSS_READ_SC_LZ	Error reading the secondary level-zero file data record
ICSS_READ_SD	Error reading the data block from the SIRIUS file, !AS,/fao=1
ICSS_READ_SD_CBLK	Error reading the control block from the SIRIUS file, !AS,/fao=1
ICSS_READHDR_PR_HK	Error reading the header (file label record) in the primary housekeeping file
ICSS_READHDR_PR_LZ	Error reading the header (file label record) in the primary level-zero file
ICSS_READHDR_PR_MAG	Error reading the header (file label record) in the primary magnetometer file
ICSS_READHDR_SC_HK	Error reading the header (file label record) in the secondary housekeeping file
ICSS_READHDR_SC_LZ	Error reading the header (file label record) in the secondary level-zero file
ICSS_READHDR_SC_MAG	Error reading the header (file label record) in the secondary magnetometer file
ICSS_READMSGDATA_SD	Error reading the message data block from the SIRIUS file, !AS,/fao=1
ICSS_READTIMES_PR_ATT	Error reading the file times from the primary attitude file
ICSS_READTIMES_SC_ATT	Error reading the file times from the secondary attitude file
ICSS_READTIMES_PR_ORB	Error reading the file times from the primary orbit file
ICSS_READTIMES_SC_ORB	Error reading the file times from the secondary orbit file
ICSS_READTIMES_ERR	Error reading the file times from the !AS file,/fao=1
ICSS_REWIND_ERROR_UDS_ SEND_ID	Error rewinding UDS_SEND_ID data store
ICSS_SC_AFTER_PR_ATT	Invalid secondary attitude file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_HK	Invalid secondary housekeeping file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_LZ	Invalid secondary level-zero file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_MAG	Invalid secondary magnetometer file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_ORB	Invalid secondary orbit file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_PSP	Invalid secondary POLAR spin phase file, begins on or after start time of primary file

ICSS_SC_AFTER_PR_SA	Invalid secondary SOHO attitude file, begins on or after start time of primary file
ICSS_SC_AFTER_PR_SD	Invalid secondary SIRIUS files, begin on or after start time of primary SIRIUS files
ICSS_SD_CBLK_ERR	Error in control block from the file !AS,/fao=1
ICSS_SEND_REQ_PROCEED	Proceed with previous transfer request
ICSS_SFDU_HDR_CLOSE_ERR	Error closing SFDU header file '!AS',/fao=1
ICSS_SFDU_HDR_OPEN_ERR	Error opening SFDU header
ICSS_SFDU_HDR_WRITE_ERR	SFDU header cannot be written
ICSS_SFDU_READ_ERR	Error reading record from SFDU header
ICSS_SFDU_TERM_ERR	Error writing SFDU records to SFDU header file
ICSS_SLP_LUN	Error getting a logical unit number for the SLP file
ICSS_SLP_NO_BODY	SLP file does not contain the body requested
ICSS_SPAWN_ERR	Error spawning DCL command line
ICSS_SUBMIT_ERROR	Error submitting transfer request to the job queue
ICSS_TEMP_DISK_EXCEED	Temporary disk space allowed per transfer
ICSS_TIMCOEF_LUN	Error getting a logical unit number for the timing coefficient file
ICSS_UPDATE_DCM_LZ_HDR	Error updating the information in the decommutated level-zero header record
ICSS_WRITE_CFL	Error writing to the updated Concatenate File List
ICSS_WRITE_DCM_LZ	Error writing a decommutated level-zero record
ICSS_WRITE_DCM_SPEC	Error writing the decom specification record to the decom level-zero file
ICSS_WRITE_ERR_MAIL_FILE	Error writing to the mail message file
ICSS_WRITE_ERR_TAEF	Error writing TAE data file
ICSS_WRITE_ERROR_UDS_SEND_ID	Error writing to UDS_SEND_IDS data store
ICSS_WRITE_ERROR_SFDU	Error writing SFDU header record to disk
ICSS_WRITE_ERROR_LZ_HEADER	Error writing level-zero header record to disk
ICSS_WRITE_ERROR_LZ_RECORD	Error writing level-zero data record to disk
ICSS_XFER_DATA_AVAIL	Data is available for user transfer
ICSS_XFER_QUOTA_EXCEED	Transfer quota for 24 hours exceeded
[RTC_CONNECT] connecting to host	Connection to host in progress
[RTC_CONNECT] error issuing connect command	Error returned from connect call; implies a network or TCP/IP problem
[RTC_CONNECT] error issuing socket command	Error returned from socket call; implies a network or TCP/IP problem
[RTC_CONNECT] host unknown	Specified host is unknown
[RTC_CONNECT] unable to connect error	Unable to connect to the host; implies a network or TCP/IP problem
[RTC_LOGINUSER] error from socket_read	An error occurred on a socket read operation
[RTC_LOGINUSER] error from socket_write	An error occurred on a socket write operation
[RTC_LOGINUSER] error reading username command	An error occurred reading the username string
[RTC_LOGINUSER] error sending password command	An error occurred sending the password string to or receiving the password string from the server
[RTC_LOGINUSER] error sending username command	An error occurred sending the username string to or receiving the username string from the server
[RTC_LOGINUSER] error sending WAIT command	An error occurred sending the WAIT command
[RTC_LOGINUSER] error sending WTG ACK	An error occurred sending the WTG acknowledgment
SS$_BADPARAM	REQ_SYS is something other than 1 or 2
SS$_SSFAIL	An error occurred while trying to calculate the tilt angle

Accepted	Access to desired data is successful
Denied	Access to desired data is denied
Remote system not operational	Server system is not operational
Socket read error—socket failed	TCP/IP error

C.2 Warning Messages

The following messages warn the operator of potential problems:

ICSS_ATT_EOF	End of file error reading attitude file
ICSS_AVG_SPIN_RATE_ERR	Calculated average spin rate for !AS level-zero housekeeping data record is equal to zero, /fao_count=1
ICSS_CLOSE_ERR_DATA_FILE_ SPEC	Error closing the data file specification
ICSS_CLOSE_ERR_MAIL_FILE	Error closing the mail file
ICSS_EOF_SD	End of file reached in the SIRIUS files
ICSS_EOF_PR_HK	End of file reached reading the primary housekeeping file
ICSS_EOF_PR_LZ	End of file reached reading the primary level-zero file
ICSS_INPUT_FILE_CLOSERR	Error closing '!AS' input file, /fao=1
ICSS_INVALID_MODE	Invalid telemetry mode indicator !AS found in !AS level-zero housekeeping data record header, /fao_count=2
ICSS_KP_DATATYPE_ERR	Key parameter file contains variables having non-standard data type
ICSS_MAN_CLEANUP	Manual cleanup of the ICSS_UDS directory may be needed
ICSS_MSG_BLK_RET_SD	SIRIUS data file message block has been returned
ICSS_MF_DECOM_NOT_SET	Decom specification not set for indicated minor frame
ICSS_NOT_ENOUGH_DATA	Not enough data to perform interpolation
ICSS_ORB_EOF	End of file error reading orbit file
ICSS_REQ_TIME_ERR	User requested time not in !AS level-zero housekeeping data record, /fao_count=1
ICSS_SFDU_BUF_FULL	SFDU buffer is full, no comment written
ICSS_SPIN_RATE_ERR	Cannot compute accurate spin rate and/or standard deviation of spin rate for !AS level-zero housekeeping data record
ICSS_STANDING_RESUBMIT	Problem occurred before standing request query performed.
ICSS_TIME_OUTRANGE	Request time out of range of files

C.3 Success Messages

The following messages indicate that operations have proceeded without errors:

ICSS_SUCCESSFUL	Successful completion
ICSS_UNSUCCESSFUL	Unsuccessful completion
ICSS_QUEUE_UDS_SUCCESS	Queue UDS send jobs completed successfully
ICSS_QUEUE_UDS_FAIL	Queue UDS send jobs completed unsuccessfully
ICSS_NSSDC_XFER_FAIL	Transfer to NSSDC job completed unsuccessfully
ICSS_NSSDC_XFER_SUCCESS	Transfer to NSSDC job completed successfully

C.4 Informational Messages

The following messages inform the operator of system conditions:

ICSS_DEL_SEND_REQ	Outstanding user send data request to be deleted
ICSS_GAP_IN_FILE	A gap occurred since the last major frame read
ICSS_GET_DDF_RET_MES	User '!AS' requests the retrieval of '!AS',/fao=2
ICSS_GET_NEXT_FILE	Next file is needed, no interpolation performed
ICSS_NEW_SEND_REQ	New user send data request to be generated

ICSS_NEXT_RECORD	Next record needs to be read
ICSS_NO_INTERP	No interpolation necessary
ICSS_PREV_EQU_INV	Previous equation invalid for request
ICSS_PREV_BUFF_INV	Previous buffer invalid
ICSS_SRC_EQ_TARGET	Source and target coordinate systems are identical
ICSS_UPD_SEND_REQ	Outstanding user send data request to be updated

APPENDIX D—KPGS DELIVERY FORM

KPGS deliveries are made electronically. For a sample delivery form, go to the Sample_Delivery Form file in the SYS$PUBLIC:[KPGS] directory on the CDHF.

ABBREVIATIONS AND ACRONYMS

3-DP	3-Dimensional Plasma
ACF	attitude control reference frame
ANSI	American National Standards Institute
AP	A index value
ASCII	American Standard Code for Information Interchange
ATC	absolute time code
bpi	bits per inch
CCR	configuration change request
CDF	common data format
CDHF	Central Data Handling Facility
CELIAS	Charge, Element, and Isotope Analysis System
CGMLT	corrected geomagnetic local time
CM	configuration management
CPI	Comprehensive Plasma Composition
DARN	Dual Auroral Radar Network
DBA	database administrator
DBMS	database management system
DCL	Digital Command Language
DECnet	Digital Equipment Corporation’s network
DFCD	data format control document
dn	data number
DSN	Deep Space Network
EBCDIC	extended binary-coded decimal interchange code
ED	eccentric-dipole
EDMLT	eccentric-dipole magnetic local time
EFD	Electric Field Detector
EPIC	Energy Particle and Ion Composition
ERNE	Energetic and Relativistic Nuclei and Electron (Experiment)
FTP	File Transfer Protocol
GCI	Geocentric Celestial Inertial

GDCF	Generic Data Capture Facility
GEO	Geographic
GEOTAIL	Geomagnetic Tail Laboratory
GSE	Geocentric Solar Ecliptic
GSFC	Goddard Space Flight Center
GSM	Geocentric Solar Magnetospheric
IBM	International Business Machines, Inc.
ICSS	ISTP CDHF Software System
IDTF	internal day time format
IEEE	Institute of Electrical and Electronics Engineers
IMP	International Magnetosphere Physics
IMSL	International Mathematics and Statistics Library
IP	Internet Protocol
ISTP	International Solar-Terrestrial Physics
KDF	KPGS delivery form
KITT	KPGS Integration Test Team
km	kilometer
KP	K index value
KPGS	key parameter generation software
MFE	Magnetic Field Explorer
MFI	Magnetic Fields Investigation
MSPC	modified spin plan coordinate
MTC	modified topographic coordinate
NAG	Numerical Algorithms Group
NASA	National Aeronautics and Space Administration
NRT	near-real time
nT	nanoteslas
PI	principal investigator
POLAR	Polar Plasma Laboratory Mission
RDAF	remote data analysis facility
rpm	radians per minute
R-S	Reed-Solomon

SEAS	Systems, Engineering, and Analysis Support
sec	second
SFDU	standard formatted data unit
SIDS	simulated instrument data set
SIRIUS	Scientific Information Retrieval Integrated Utilization System
SLP	solar/lunar/planetary
SM	Solar Magnetic
SOHO	Solar and Heliospheric Observatory
SSDM	SEAS Systems Development Methodology
SWE	Solar Wind Experiment
SWIM	Solar Wind Interplanetary Mission
TAI	international atomic time
TBD	to be determined
TBS	to be supplied
TCP	Transmission Control Protocol
UTC	universal time coordinated
VAX	virtual address extension
VMS	virtual memory storage
WIND	Interplanetary Physics Laboratory Mission

REFERENCES

1. National Aeronautics and Space Administration (NASA), *International Solar-Terrestrial Physics (ISTP) Key Parameter Generation Software (KPGS) Standards and Conventions, Version 1.3*, March 1994

2. Computer Sciences Corporation (CSC), CSC/SD-92/6037R6.3, *International Solar-Terrestrial Physics (ISTP) Program Central Data Handling Facility (CDHF) Users Guide, Release 6.3*, May 1996

3. —, CSC/TM-91/6084, *International Solar-Terrestrial Physics (ISTP) Central Data Handling Facility (CDHF) Key Parameter Generation Software (KPGS) Integration Test Plan, Revision 1*, Review, March 1995

4. NASA, NSSDC/WDC-A-R&S 91-31, *NSSDC CDF User's Guide*, Version 2.4, January 1994

5. CSC, CSC/TR-91/6014, *Data Format Control Document Between the International Solar-Terrestrial Physics (ISTP) Program Mission Operations and Systems Development Division (MOSDD) Ground Data Processing System and the ISTP Mission Investigators, Revision 2*, May 1996

6. Numerical Algorithms Group Limited, *The NAG Fortran Library Manual, Mark 15*, First Edition, June 1991

7. International Mathematics and Statistics Library, *User's Manual, IMSL SFUN/LIBRARY, Version 2.1*, January 1989