CSC/SD-86/6704/UD6

UARS CDHF SOFTWARE SYSTEM (UCSS)

PROGRAMMER'S GUIDE

TO

PRODUCTION SOFTWARE SUPPORT SERVICES

Prepared for

GODDARD SPACE FLIGHT CENTER

By

COMPUTER SCIENCES CORPORATION

Under

Contract NAS 5-31000

February 1993

Prepared by:

B. Pedersen 2/25/93
B. Pedersen Date

Approved by:

K. D. Taylor 2/25/93
K. D. Taylor Date

S. Adamson
C. Agard
G. Blackwell
P. Goldstein
L. Lu
J. Martin

ABSTRACT

This document defines the interfaces to production software support services at the Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) and production testing services on Remote Analysis Computers (RACs). These services developed under the UARS CDHF Software System (UCSS) contract support access to all levels of instrument data files and other types of cataloged data including Level 0 engineering, quality, spacecraft, and onboard computer (OBC) data. In addition, the UCSS provides routines to initialize and terminate production programs and to perform error reporting.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this document is to define the production software support services provided under the Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) Software System (UCSS) contract. Calling sequences are provided for the services needed by the Principal Investigators (PIs) to develop their production processing software. The orbit and attitude services are addressed in the UARS Programmer's Guide for Orbit/Attitude Services (Reference 4).

The UCSS production software support services are divided into three areas. The production control routines are used to pass information to and from production programs. These routines aid in the control and monitoring of the production processing flow. The file access services provide access to UCSS-managed data files. Services are provided to access all levels of instrument data, calibration files, correlative files, user status files, and scratch files. The utility services provide functions including error reporting.

1.2 UARS PRODUCTION PROCESSING OVERVIEW

One of the primary activities performed at the UARS CDHF is the processing of the scientific data from Level 0 to Level 3B. Scientific data processing is performed at the CDHF for 9 of the 10 UARS instruments. The instrument investigators are responsible for developing the data processing software. The UCSS provides a collection of production software support services which are used by the PI-developed programs to access UCSS-managed data files and to control the processing flow.

The PIs will initially develop the data processing software on the Remote Analysis Computers (RACs). The UCSS provides a set of simulated software support services to aid the PIs in the testing of their software at the RACs. The calling sequences of the simulated

services are identical to those of the production services. As a result, production programs developed using the simulated services do not have to be modified in order to use the production services.

Eventually, after sufficient testing, the data processing software will be run in a production mode at the CDHF. The UCSS provides scheduling tools that are used by CDHF operations personnel to schedule and run production jobs. The information needed to schedule production processing is maintained under configuration control by operations personnel. Original (first time) processing is scheduled when the required input data becomes available and when there are sufficient computer resources to run the job. Reprocessing (subsequent times) is performed as requested with the approval of the UARS Project. Changes in software, calibration tables, or input files normally result in reprocessing of data.

The UCSS provides a capability to run data processing software in the production environment in test mode. The PIs can use this capability to perform final testing of their software or for testing after minor changes have been made to their programs.

1.3 DOCUMENT ORGANIZATION

This document is organized into five chapters. Chapter 1 provides an introduction to the types of support services provided for production programs. Chapter 2 presents an overview of production processing environment. Chapter 3 describes the detailed interfaces of the production software support services. Chapter 4 discusses the simulated services which are provided for use in testing production programs at the RACs. Chapter 5 describes the analysis services. Appendix A provides a detailed description of the UARS date and time format used in many of the calling sequences to the software support services. Appendix B provides an example to demonstrate the use of some of the services, and Appendix C presents the guidelines for Level 1 and 2 data processing. Appendix D presents a detailed description of the Level 0 file formats. Appendix E contains the description of the Level 3A file formats. Appendix F provides information about error handling. Appendix G gives the Level 0 SFDU File Description. Appendix H gives information needed for using the OBCDECODE routine. Appendix I is a glossary and Appendix J is a list of references.

CHAPTER 2

UCSS PRODUCTION PROCESSING ENVIRONMENT

One of the primary UCSS functions is the support of UARS scientific data processing. The instrument investigators are developing the software to process the data from Level 0 to Level 3A. The UCSS provides production software support services which are used by the PI-developed programs. These services provide access to UARS data files and aid in the control of the production processing flow. In addition, the UCSS provides scheduling tools which are used by CDHF operations personnel to schedule and run production jobs. The scheduling tools also support the capability of running jobs in a test mode.

The UCSS must provide a flexible production environment that accommodates a wide range of processing needs. In order to meet these diverse needs, the UCSS has established some guidelines for the production processing software. The purpose of this section is to define these guidelines. Section 2.1 provides an overview of the UARS Catalog. Section 2.2 describes in detail the elements of a production job. Section 2.3 discusses the UCSS approach to production scheduling and provides a description of the information the PIs must supply in order to schedule processing. Section 2.4 provides guidelines for recovery of production jobs. Section 2.5 describes the approach to testing production software in the production scheduling environment.

2.1 UARS CATALOG

The UARS Catalog is an index of the UCSS-managed files that are available to the UARS community. The files tracked in this catalog are the Levels 0, 1, 2, 3AT, 3AS, 3AL, and 3BS, correlative, calibration, orbit, attitude, Level 3 parameter files, and production program files. Files can be added to the Catalog during production processing or when the UCSS receives files from an external source (e.g. Level 0 files from the Data Capture Facility). The UARS Catalog is used to identify and locate the input files required for production processing.

The Catalog can be viewed as a collection of logical records describing important characteristics or attributes of each file. In production processing, some of the attributes of output files are supplied by the user program and some are supplied by the UCSS software. The user program provides the initial file attributes when assigning or opening a new file using the ASGCAT, OPENL3AT, OPENL3AL, OPENL3S, OPENL3TP, and OPENL3LP services (see Sections 3.2.2 and 3.2.7 through 3.2.11). Additional attributes can be provided when the user program requests that a file be cataloged using the CLOSELF or DASLID services (see Sections 3.2.24 and 3.2.25). For input cataloged files, the user program specifies the catalog attributes necessary to identify the required file when opening or assigning the files using the OPENL0, ASGCAT, ASGCAL, ASGCOR, OPENL3AT, OPENL3AL, OPENL3S, OPENL3TP, or OPENL3LP services (see Sections 3.2.1 through 3.2.4 and 3.2.7 through 3.2.11).

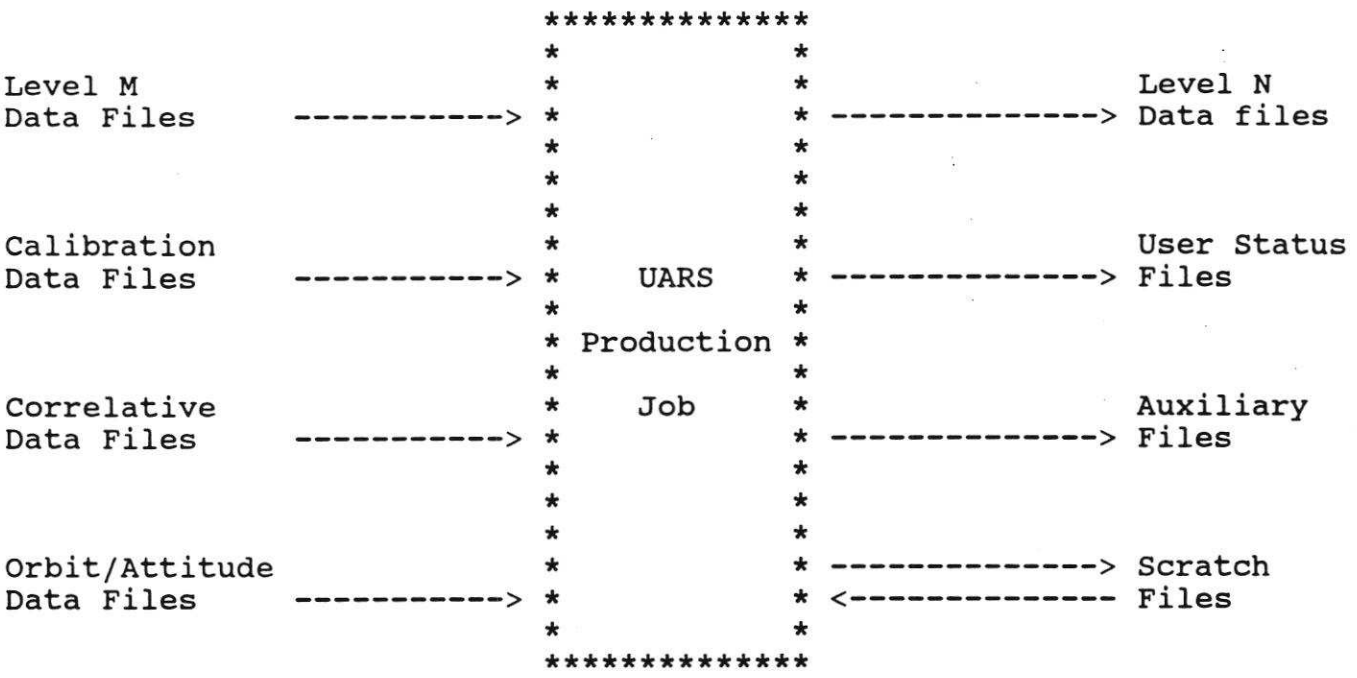
The UARS Catalog is maintained as a collection of relational tables managed by the INGRES data base management system. The UCSS Data Base Administrator (DBA) is responsible for defining the structure of these tables in the UARS Catalog. In order to accomplish this task, the DBA must have knowledge of the attributes applicable to each class of data and of the valid values of the attributes. For example, the DBA must know the valid Level 3AT data subtypes for each instrument. The UCSS is required to be able to support changes in the catalog structure as these changes are identified and approved. Attributes can be added or deleted from the catalog structure or their possible values can be changed with approval by the controlling authority.

Errors occur when a user program attempts to catalog a file with an unknown attribute or invalid attribute value.

2.2 PRODUCTION JOBS

A production job is a job requiring support services that is initiated by the UCSS scheduler. A production job nominally processes all data for an instrument from one level of data abstraction to the next higher level for a specified time period. Other types of production jobs can be supported such as single species or multilevel processing. A production job is run as a series of one or more production programs, executed in a specific order, using cataloged input data, and producing cataloged output files, user status files, and auxiliary files. Figure 2-1 depicts a sample production job. The UCSS supports the capability of conditional branching within the production job's runstream so that specific paths can be taken for data dependent processing (see Section 2.2.6).

Figure 2-1. Sample UARS Production Job



2.2.1 PRODUCTION PROGRAM

A production program is a load module that is used in the processing of UARS scientific data. Production programs are maintained under configuration control by CDHF operations personnel and are tracked in the UARS Catalog. A production program may be used in more than one production job. A production program processes data from a specific instrument for a given data level or levels. The input data time range is provided to the program by the UCSS PGINIT service (see Section 3.1.1) at run time. Nominally, the input data time range should correspond to the output data time range. All UCSS-managed files used by a production program, including both input and output files, are assigned dynamically using the UCSS subroutine interfaces (see Section 3.2) at run time.

2.2.2 PROCESSING TIME RANGE

The nominal time span for production processing jobs is one day. However, the UCSS provides the flexibility to support multiple day and partial day processing. These processing alternatives are addressed in the following sections.

2.2.2.1 Default Input File Time Span

A nominal Level 1 production processing job reads 24 hours worth of Level 0 data and produces a Level 1 file(s) spanning this same 24 hour period. The UCSS scheduler provides the actual time span to be processed. However, in order to handle events spanning day boundaries, the time span actually processed may not be exactly 24 hours. An event that crosses a day boundary can be processed into one output file. If the event is to be associated with the start day of the event, processing continues into the next day until the event is complete and the output data is stored in the file for the start day. Processing of the next day's data must ignore the partial event at the beginning of the day. If the event is to be associated with the stop day of the event, processing begins at the start of the event in the previous day and the output data is stored in the file for the stop day. Processing of the first day's data must ignore the partial event at the end of the day. Using either alternative, the output files do not correspond exactly to the nominal day boundaries, but the output files from successive days processing would not overlap. The catalog entries for these files indicate the actual time coverage.

The Level 1 file produced in this way is described in part by a day number (UARS day) that corresponds to the number of days from the launch date. The catalog entry for the file contains the UARS day number associated with the file, the file start and stop times, and other pertinent information. UARS day provides a means of identifying the production processing run for a set of data.

In general, each subsequent level of processing uses files with the same UARS day number as input and produces files with those same characteristics as output. The time range for a file cataloged with a given UARS day number must at least partially overlap the time range of the Level 0 file with the same UARS day number.

2.2.2.2 Multiple Day and Sub-Day Processing

Some production processing jobs require input data that spans several full days. The UCSS accommodates this type of processing job, but assumes that the output files produced by such jobs adhere to the UARS day conventions discussed above. For example, a production processing job requiring 3 days of input data would produce three output data files, each containing 1 day of data.

There are production processing job designs that would produce more than one output file of a given type for a day. For example, within a certain day, an instrument may have been cycled from a data-taking state to a standby state and back to a data-taking state. The developers of the production software may prefer to ignore standby periods, which would appear to result, in this case, in two separate output "files" of the same type for the given day. The UCSS assumes that the two "files" produced for the same day are concatenated to

form a single file that is identified by the UARS day.

2.2.3 PRODUCTION INPUT

2.2.3.1 Input Files

All input files to a production job must be cataloged. Cataloged files are read-only files that are assigned using catalog attributes of the file such as instrument, UARS day number, and level. Production programs can also use, as input, files created by a previous program in the same production job. These files can be files intended for subsequent cataloging or scratch files (see Section 2.2.5).

All input files must be assigned with the OPENL0, OPENL3AT, OPENL3S, OPENL3AL, OPENL3TP, OPENL3LP, ASGCAT, ASGCAL, ASGCOR, or ASGSCR services (see Section 3.2). For Level 0, 3AT, 3AS, or 3AL data and parameter files associated with Level 3AT or 3AL data, the UCSS treats the data as if it were a single virtual file. The production program does not need to be aware of how many physical files are to be accessed; it sees the data as one logical file. The UCSS provides read services for Level 0, 3AT, 3AS, and 3AL data and for parameter files associated with Level 3AT or 3AL data, which have nominal levels of 3TP and 3LP, respectively. The user is responsible for developing read services for Level 1, Level 2, calibration, correlative, and scratch files.

2.2.3.2 Program Parameters

Each production program can define and use up to 50 input parameters which are specific to the program. These parameters are passed from the UCSS scheduler to the production program by the PGINIT service. PGINIT also supplies the processing time range and the primary processing day to the program.

2.2.4 PRODUCTION OUTPUT

2.2.4.1 Cataloged Files

A production program creates files to be cataloged and can both write to and read from these files. Level 3AT files are opened using the OPENL3AT service (see Section 3.2.7). Level 3AL files are opened using the OPENL3AL service (see Section 3.2.8). Level 3AS or 3BS files are opened using the OPENL3S service (see Section 3.2.9). Parameter files are opened using the OPENL3TP or OPENL3LP service (see Sections 3.2.10 and 3.2.11). Level 1 and 2 files are assigned using the ASGCAT service (see Section 3.2.2). The UCSS provides write

services for all type of Level 3A files. The user is responsible for providing the write services for Level 1 and 2 files. The CLOSELF (see Section 3.2.24) and DASLID (see Section 3.2.25) services are used to actually request cataloging of the Level 3A and the Level 1 or 2 files, respectively. Once a file is requested to be cataloged, it cannot be modified by subsequent production programs.

The information used in creating catalog entries for data files comes from two sources. The production program supplies initial file attributes via the call to the OPENL3AT, OPENL3AL, OPENL3S, OPENL3TP, OPENL3LP, or ASGCAT services. Additional attributes are provided by the call to the CLOSELF or DASLID services. The UCSS supplies the other attributes including the file location.

These catalog entries are not actually finalized in the Catalog until the successful completion of the production job. If any of the programs in a production job fails or terminates abnormally, then catalog entries created by the programs in that production job are not inserted into the Catalog. The corresponding files remain online for further analysis.

2.2.4.2 User Status Files

User status files are temporary files that are maintained in the UCSS-managed disk space. There are separate user status file directories for each production job definition (e.g. HALOE Level 1 processing job). These files are maintained cyclically so that only an operationally controlled number of versions are saved on the disk. User status files are assigned using the ASGUSR service (see Section 3.2.6). The user is responsible for providing any I/O services required. User status files are deassigned using the DASLID service (see Section 3.2.25).

2.2.4.3 Auxiliary Files

Auxiliary files are output files that are created in the user-managed disk space of the instrument investigator responsible for the job. These files are not cataloged. Auxiliary files cannot be used as input to production jobs.

The instrument PI is responsible for insuring that there is sufficient quota and free space available in the auxiliary directory used by the production jobs. If sufficient disk space is not available, then the production job cannot generate the auxiliary files. In order to avoid job failure due to problems creating auxiliary files, the production software must have sufficient error detection logic to handle I/O errors encountered when processing auxiliary files. Auxiliary files are not primary production processing outputs. All primary production processing output files

should be cataloged.

The only UCSS support of auxiliary files is the definition of the logical name AUX_DIRECTORY in each production job's runstream. This logical name identifies the disk device and directory to be used to create auxiliary files. It must be used by the program to specify the device and directory when opening an auxiliary file. In addition, the Fortran logical unit numbers 100 to 119 are reserved for I/O to auxiliary files. Use of dedicated logical unit numbers is necessary in order to prevent collisions with assignments made internally within the UCSS services.

2.2.4.4 Program Summary Report

The UCSS produces a program summary report for each program executed during the job. This report provides information about the program including completion status, processing time range, input parameters, input files and output files. The format of this report is described in Figure 2-2. A wide discrepancy between an estimated output file size and the actual file allocation is marked in the Program Summary Report by an asterisk to the left of the output file name.

Figure 2-2. Program Summary Report

```
                    PROGRAM SUMMARY REPORT

UCSS JOB ID: . . .   JOB STEP NUMBER: . . .          PROGRAM ID: . . .

PROCESSING TIME RANGE: ... - ...      UARS PRIMARY PROCESSING DAY: ...

INPUT PARAMETERS:

    PARAMETER NAME            PARAMETER VALUE

        . . .                      . . .

CATALOGED INPUT FILES:

                                          UARS          CALIB SOURCE
  LOGICAL FILE ID     TYPE SUBTYPE LEVEL DAY  VERS CYC ID    ID

      . . .           ...  ...     ...   ...  ...  ... ...   ...

OUTPUT FILES:

                                          UARS          EST  ALLOC
  LOGICAL FILE ID     TYPE SUBTYPE LEVEL DAY  VERS CYC SIZE  SIZE  DISP

      . . .           ...  ...     ...   ...  ...  ... ...   ...   ...

SCRATCH FILES:

                  EST    ALLOCATED
  LOGICAL FILE ID SIZE   SIZE        DISP          SCRATCH FILE NAME

      . . .       ...    ...         ...                 ...

USER STATUS FILES:

  USER STATUS
  FILE NUMBER                 USER STATUS FILE NAME

    . . .                          . . .

ERROR MESSAGES:

  . . .

PROGRAM START TIME: . . .          PROGRAM STOP TIME: . . .
PROGRAM COMPLETION STATUS: . . .   PROGRAM CPU USAGE: . . .
DIRECT I/O COUNT: . . .            BUFFERED I/O COUNT: . . .

PROGRAM COMPLETION COMMENTS: . . .

* Asterisks mark wide discrepancies between allocation and estimate
```

2.2.4.5 Job Summary Report

The UCSS produces a job summary report at the end of each production job. This report provides information about the job including the job identifier, job completion status, job statistics, input files, and output files. The format of this report is described in Figure 2-3.

Figure 2-3. Job Summary Report

```
                    JOB SUMMARY REPORT

UCSS JOB ID: AAAAAAAAAAAAAAAAAAAA      CPU ID: AAAAA

UCSS VERSION: XXXXX                    UOAS VERSION: XXXXXX

JOB START TIME: DD-MMM-YYYY HH:MM:SS

JOB STOP TIME DD-MMM-YYYY HH:MM:SS

JOB COMPLETION STATUS: AAAA            JOB CPU USAGE: DDD HH:MM:SS.CC

DIRECT I/O COUNT: NNNNNNNNNN           BUFFERED I/O COUNT: NNNNNNNNNN

MAX WORKING SET SIZE: NNNNNNNNNN


JOB ERROR MESSAGES:

    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

2.2.4.6 Error Messages

A production program can use the UCSS service ERRCDE (see Section 3.3.1) to report and log any serious errors detected by the program. These error messages are written to a UCSS log file and are included on the program summary report.

2.2.5 SCRATCH FILES

Scratch files are maintained in the UCSS-managed disk space. They are created during a production job for the life of the job only. They can be used to pass information between programs in a job or as a scratch area. Scratch files can both be written and read by

production programs. The ASGSCR service (see Section 3.2.5) must be used to assign scratch files so that the UCSS can manage the disk allocation. The user is responsible for providing the I/O services to access scratch files. The DASLID service (see Section 3.2.25) is used to deassign scratch files.

All scratch files associated with a job are deleted at the successful completion of the job. Since the scratch files are not deleted when a job reports a failed condition or when a system failure occurs, they can aid in determining the cause of a failure and in recovering the job if the programs were written to take advantage of this capability.

2.2.6 CONDITIONAL PROCESSING

A production program exits with a condition code that can be tested by job control statements. This condition code is set using the PGTERM service (see Section 3.1.2). The results of these tests can be used to control further job execution (e.g., which program to execute next). The message number, associated message text, and mnemonic name for each message must be defined by each investigator supplying production software. The Virtual Address Extension (VAX) Virtual Memory System (VMS) message utility should be used to define the message number and mnemonic in order to generate a standard VMS condition code. See the VAX VMS Utility Reference Manual for further information.

2.3 PRODUCTION SCHEDULING

Production scheduling is the routine scheduling of scientific data processing jobs. The UCSS provides scheduling tools which are designed to aid the operations personnel in efficient and timely scheduling of production jobs. The UCSS provides automatic scheduling of production jobs over a specified time period. The scheduling tools also allow the operator to manually schedule individual production jobs. The primary functions of the scheduling software are to insure availability of the resources required by the production jobs and to submit the jobs for execution at the appropriate time.

The UCSS scheduler schedules production jobs based on information contained in production program catalog entries, production job definitions and scheduling requests. All of the scheduling information is maintained under configuration control. The UCSS scheduler uses this information to determine which jobs need to be run. These data structures are maintained by operations personnel, but rely heavily upon information supplied by the instrument PIs. The following sections describe each of these structures.

2.3.1 PRODUCTION PROGRAM CATALOG ENTRIES

The UCSS tracks all versions of the production programs in the UARS Catalog. A program catalog entry identifies the program name, program version, load module location, the memory and CPU resources required by the program, and other important information. New versions of production programs are cataloged upon approval by the UARS Project.

The following information must be supplied in order to create program catalog entries:

- Program identification, including program name, program version, and instrument identifier
- User status file information
- Resource usage information including estimated CPU usage, wall clock time, and working set size
- Auxiliary file flag indicating whether the program needs to access auxiliary files
- Required input file specifications (type, subtype, level, and relative time range for each required input file)
- Output file requirements (sizing estimates)
- Orbit and attitude data requirements
- Scratch file requirements (sizing estimates)

2.3.2 PRODUCTION JOB DEFINITIONS

A production job definition defines the basic structure of a production job. It identifies the production programs that are invoked by the job, the input data requirements, and the skeletal runstream including any special job control needed to test program exit status.

The following information must be supplied in order to define a production job to the scheduling software:

- Job identification, including job name and version
- Orbit and attitude data requirements (predicted, definitive, best, or none)

- Auxiliary file disk and directory
- DCL runstream defining the job
- Default program parameter values

2.3.3 SCHEDULING REQUESTS

A scheduling request is a request to run a specific production job for a given time period. It identifies the production job, the applicable time range, and parameters indicating execution frequency and times allowed. It also specifies the version rules to be used for the input and output cataloged files. Optionally, program parameters can be modified.

The following information must be supplied in order to schedule production processing:

- Job identification, including job name and version
- Start and end date/time of the processing period
- Input data file version information
- Modified program parameter values
- Auxiliary file output location
- Orbit and attitude data requirements (predicted, definitive, best, or none)

The UCSS scheduler uses the scheduling request to identify which jobs to run. The information in the scheduling request is used in conjunction with the production job definition to identify the input data required, the output files to be produced, the program parameters, and the time range for each production job to be scheduled. This information is used to create the expanded production job runstream for each job. The scheduler stages required input files to insure that the data is available on magnetic disk. The scheduler submits jobs for execution when the required resources are available and when the constraints on the job can be satisfied.

2.4 UCSS PRODUCTION RECOVERY GUIDELINES

It is expected that there will be times during the operation of the CDHF that the system will halt while a variety of activities are underway, including the processing of production jobs. The UCSS is

responsible for detecting and recovering from this kind of problem only for UCSS specific functions such as production scheduling and catalog management. The recovery of the production jobs is initiated by the UCSS but relies on PI-developed recovery software and instructions. PI-developed recovery systems must include the software and control language necessary to confirm the existence of all required files and data, eliminate questionable files, and reinitiate processing. Several options are available to the production processing software developers for this capability. These are discussed in the following sections.

2.4.1 PRODUCTION JOB RERUN

One option available to the production processing developers is to do nothing; recovery means simply deleting all partially completed files and rescheduling production processing from the beginning. This option is desirable for those production jobs that can run to completion without requiring massive CDHF resources and this avoids the expense of developing complicated recovery systems.

Another option is to restart the production job at a particular job step. In this case the state of the files generated by the job must be restored to the restart point and any files created at or beyond that point must be deleted.

Both options are available through the UCSS Job Recovery function.

2.4.2 OPERATING SYSTEM CHECKPOINT/RESTART

For those production jobs that may require a significant fraction of a day to complete, some means of periodically saving intermediate results is desirable. One option that may be available is the use of checkpoint/restart procedures provided with the operating system and utilities. Periodically taking file checkpoints during processing would cause the state of the production job to be saved to that point in processing. If a system failure were to occur, a UCSS invoked restart routine could reinitiate the processing of that job from the point of the last checkpoint. It should be noted that a proven vendor provided checkpoint/restart capability is not available with the DEC VAX system. Moreover, this type of capability can, when available, consume significant amounts of system resources. Given the expected probability of system failure, it may not be cost effective to use the vendor provided checkpoint/restart capability.

2.4.3 USER SUPPLIED RECOVERY

The remaining options for production job recovery rely exclusively on software and control language produced by the developers of the production job. Two alternatives to this option are apparent: multiple-program production jobs and periodic file closings. There are "gray" areas between these options as well. A UCSS production job may consist of a sequence of programs, where each program produces either scratch product files used in subsequent programs of the job, or a separate file for one of the components of the output product (e.g., specie concentrations). Production job recovery in this case could be implemented by determining which file had been partially completed among the sequence of files that should have been produced. That file would be eliminated and the corresponding job step reinitiated. Alternately, the production job developers may elect to use a single program that produces one or more files. An approach to recovery in this situation is to periodically execute a Fortran CLOSE on the currently open files. This saves the results produced to that point in the processing. The file(s) may subsequently be reopened and used for further processing. On recovery from system failure, the user developed software would need to OPEN the partially completed file(s), determine the point in processing at which the failure occurred, and reinitiate processing.

It should be reiterated that the last two recovery schemes described above rely heavily upon PI-developed software. The UCSS would detect the system failure condition, provide for recovery of UCSS functions, and invoke the appropriate PI-developed production processing recovery job. In general, production processing recovery jobs are required for each level of processing, and perhaps more than one may be required for a given level of processing.

2.5 PRODUCTION PROGRAM TESTING

The UCSS provides the capability of testing production programs while in the production processing environment. This test mode should be used after completion of initial testing of production programs using the UCSS simulated services (see Section 4).

The UCSS scheduling tools can be used to create catalog entries for test programs. Test programs do not have to be configuration-controlled. A test program can remain in the user directory so that it can be modified without updating the catalog entry each time. A job definition (see Section 2.3.2) that uses the test program must be created. To run the test job the user submits a scheduling request to CDHF operations.

Any job that includes a test program is defined as a test job. All of the output catalog products are identified as test files.

CHAPTER 3

UCSS PRODUCTION SOFTWARE SUPPORT ROUTINES

This section provides the detailed calling sequences for the UCSS production software support services. These services are designed for use by production programs run in the production environment on the CDHF.

The UCSS production software support services are divided into three areas. Section 3.1 describes the production control services which include production program initialization and termination routines. The file access services are discussed in Section 3.2. Section 3.3 describes the utility services. Appendix A documents the UARS date and time format (UDTF) that is used in many of the calling sequences. Appendixes B and C provide examples of the usage of the production software support services. Appendix F provides information about error handling.

3.1 PRODUCTION CONTROL ROUTINES

3.1.1 PROGRAM INITIALIZATION (PGINIT)

PGINIT provides the mechanism for passing input parameters to a production program. The parameters are supplied to the production job by the scheduling software. PGINIT also initializes the production environment for the program and updates the UCSS production accounting tables with the initial program statistics. PGINIT must be called at the start of each production program.

The processing start and stop times provided by PGINIT define the expected time range of the output data to be generated by the production program. For a nominal UARS production job, these times would specify a 24 hour period starting at 00 Greenwich Mean Time (GMT).

PGINIT supplies the user defined parameters to the program through an ASCII table. The dimension of this table is specified by the optional argument PARAM_TBL_SIZE. A maximum of 50 parameters can be specified. The default is 20 parameters if PARAM_TBL_SIZE is not

given. A table entry consists of two items, the name of the parameter and the parameter value. The parameter values are provided to the scheduler at job definition time or, optionally, at schedule request time (see Section 2.3). No specific order of the table entries should be assumed.

The calling sequence for PGINIT is as follows:

```
CALL PGINIT (PARAM_TABLE, STRT_DATTIM, STOP_DATTIM, UARS_DAY
             [, PARAM_TBL_SIZE] )
```

ARGUMENT	TYPE	I/O	DEFINITION
PARAM_TABLE	CHAR*20 (2,*)	O	A table used to pass parameters for control of processing. Each entry in the table consists of a pair, a parameter name and its corresponding value. Parameters are specific to a particular production program. The size of this table may be from 1 to 50 entries as specified by PARAM_TBL_SIZE.
STRT_DATTIM	I*4(2)	O	Start date and time of nominal processing range in UDTF
STOP_DATTIM	I*4(2)	O	Stop date and time of nominal processing range in UDTF
UARS_DAY	I*4	O	First UARS day (DDDD) for catalog output from this program
PARAM_TBL_SIZE	I*4	I	Specifies the size of PARAM_TABLE

The last argument, PARAM_TBL_SIZE, is optional. If it is not specified, the size of PARAM_TABLE is CHAR*20(2,20) by default. PARAM_TBL_SIZE may be from 1 to 50.

3.1.2 PROGRAM TERMINATION (PGTERM)

PGTERM terminates the production program. The production program is responsible for determining the success or failure of the processing and reports this determination to PGTERM. PGTERM updates the UCSS accounting statistics with program completion information and produces a standard format program summary report (see Figure 2-2) to a disk file. PGTERM must be called at the end of execution of each production program. Any program that does not call PGTERM is automatically marked with a failed status by the UCSS software. This precaution is necessary so that the UCSS can properly handle uncontrolled program aborts.

PGTERM sets the user program's exit condition code to the value supplied in COND_CODE. If the program fails, the condition code uniquely identifies the reason for the failure. In the case of a successful run, this parameter can be used to control the subsequent program flow via the use of conditional job control language.

The calling sequence for PGTERM is as follows:

```
CALL PGTERM (PASS_FAIL, COND_CODE, PROG_COMMENT)
```

ARGUMENT	TYPE	I/O	DEFINITION
PASS_FAIL	CHAR*4	I	Program completion status 'PASS' = successful completion 'FAIL' = unsuccessful completion
COND_CODE	I*4	I	A VMS condition code specifying additional status information about the program completion
PROG_COMMENT	CHAR*80	I	A character string supplied by the production program to indicate any additional information. This message will be displayed on the program summary report.

3.2 FILE ACCESS

This section describes the production software support services designed to provide access to UCSS-managed files. Services are provided to access all levels of instrument data, calibration files, UARS day oriented correlative files, user status files, and scratch files. Table 3-1 summarizes the use of the file access services by file type.

Table 3-1. Calling Routine Matrix

DATA TYPE	ASSIGN	FILE OPEN	READ	WRITE	FILE CLOSE	DE-ASSIGN
Level 0						
-Engineering	N/A	OPENL0	READL0	N/A	CLOSELF	N/A
-Instrument	N/A	OPENL0	READL0	N/A	CLOSELF	N/A
-Onboard Computer	N/A	OPENL0	READL0	N/A	CLOSELF	N/A
-Quality	N/A	OPENL0	READL0 or QUALRD	N/A	CLOSELF	N/A
-Spacecraft	N/A	OPENL0	READL0	N/A	CLOSELF	N/A
Level 1	ASGCAT	*	*	*	*	DASLID
Level 2	ASGCAT	*	*	*	*	DASLID
Level 3AT	N/A	OPENL3AT	READL3AT	WRITEL3AT	CLOSELF	N/A
Level 3AS	N/A	OPENL3S	READL3S	WRITEL3S	CLOSELF	N/A
Level 3BS	N/A	OPENL3S	READL3S	WRITEL3S	CLOSELF	N/A
Level 3AL	N/A	OPENL3AL	READL3AL	WRITEL3AL	CLOSELF	N/A
Level 3LP	N/A	OPENL3LP	READL3LP	WRITEL3LP	CLOSELF	N/A
Level 3TP	N/A	OPENL3TP	READL3TP	WRITEL3TP	CLOSELF	N/A
Calibration	ASGCAL	*	*	*	*	DASLID
Correlative	ASGCOR	*	*	*	*	DASLID
Scratch	ASGSCR	*	*	*	*	DASLID
User Status	ASGUSR	*	*	*	*	DASLID
Auxiliary	*	*	*	*	*	*

* = PI-SUPPLIED

The UCSS provides open, read, and close services for Level 0 data. The Level 0 read services provide a time range read capability so that the user does not have to be concerned with physical file boundaries. A special service is available to read quality data. Appendix D provides a description of the Level 0 file record formats.

The UCSS provides open, read, write, and close services for Level 3AT data. A time range read capability is provided for the Level 3AT data. The close service allows the user to inform the UCSS of the file's disposition and furnishes the capability to catalog new Level 3AT files. Appendix E provides a description of the Level 3AT data file formats.

The UCSS provides open, read, write, and close services for Level 3AS and Level 3BS solar data. A time range read capability is provided for the Level 3AS and Level 3BS data. The close service allows the user to inform the UCSS of the file's disposition and furnishes the capability to catalog new Level 3AS and Level 3BS files. Appendix E provides a description of the Level 3AS and Level 3BS data file formats.

The UCSS provides open, read, write, and close services for Level 3AL data. The read service provides the ability to retrieve data for a specified latitude band over a time range. The close service allows the user to inform the UCSS of the file's disposition and furnishes the capability to catalog new Level 3AL files. Appendix E provides a description of the Level 3AL data file formats.

The UCSS provides open, read, write and close services for Level 3TP data, i.e. parameter data associated with Level 3AT files. A time range read capability is provided for the Level 3TP data. The close service allows the user to inform the UCSS of the file's disposition and furnishes the capability to catalog new Level 3TP files. Appendix E provides a description of the Level 3TP data file formats.

The UCSS provides open, read, write and close services for Level 3LP data, i.e. parameter data associated with Level 3AL files. A time range read capability is provided for the Level 3LP data. The close service allows the user to inform the UCSS of the file's disposition and furnishes the capability to catalog new Level 3LP files. Appendix E provides a description of the Level 3LP data file formats.

The UCSS provides services to assign and deassign Level 1, Level 2, correlative, calibration, user status, scratch, and orbit/attitude files. For input cataloged files, the assign routines identify the file specified using the supplied attributes, insure that it is on magnetic disk, and associate the logical file identifier with the physical file name. For new output files, the UCSS assign routines reserve the requested file space on a UCSS-managed disk, generate a file name, and associate the full file specification with the logical file identifier. For existing output files, the assign services identify the physical file to be accessed. The user program issues the open/read/write/close calls for Level 1, Level 2, correlative, calibration, user status, and scratch files. The logical file identifier supplied at assign time must be used to open the file since the program does not know the physical location of the data. The logical unit number returned at assign time must also be used when

calling the Fortran I/O services to prevent conflict with any logical unit numbers used internally by the production service routines. The deassign service is called to release the file and, optionally, to catalog a file (Level 1 or Level 2 only). The user must specify the file's disposition.

The user's program is responsible for issuing the I/O service calls to access auxiliary files. The UCSS only provides the logical name that specifies the disk and directory name where the auxiliary files are to be created. Logical unit numbers 100 to 119 are reserved for use in accessing auxiliary files (see Section 2.2.4.3).

3.2.1 OPEN LEVEL 0 DATA (OPENL0)

The OPENL0 routine is used to initiate read access to Level 0 data. The production program supplies the data type and the time range of the Level 0 data required for Level 0 to 1 processing. The time range required should be calculated relative to the processing time range provided by PGINIT. OPENL0 identifies the physical Level 0 files containing the data covering the requested time range, insures that the files are on magnetic disk, and opens the files for read access in shared mode. The production program can subsequently use the logical file identifier (LID) to read any data in the time range specified by the open calling sequence parameters.

The calling sequence for OPENL0 is as follows:

```
CALL OPENL0 (DATA_TYPE, STRT_DATTIM, STOP_DATTIM, LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Level 0 data type = 'ACRIM' = 'CLAES' = 'HALOE' = 'HRDI' = 'ISAMS' = 'MLS' = 'PEM' = 'SOLSTICE' = 'SUSIMA' = 'SUSIMB' = 'WINDII' = 'ENGINEERING' = 'OBC' = 'QUALITY' = 'SPACECRAFT'
STRT_DATTIM	I*4(2)	I	Start of processing date and time range in UDTF

ARGUMENT	TYPE	I/O	DEFINITION
STOP_DATTIM	I*4(2)	I	Stop of the processing date and time range in UDTF
LID	CHAR*16	I	Logical file identifier associated with the virtual file
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_NODATARECS (RSS) - Physical file without data exists in user's processing range PFA_NOOLDFILE - No data found or held file does not exist PFA_NOOPTDATA - No optional data available PFA_OPTFILMISS - Missing one or more optional files in a multiday range PFA_OVRLPTIME - Two physical files have overlapping times PFA_SOMEFILSTGD - File was staged

3.2.2 ASSIGN CATALOGED FILE (ASGCAT)

ASGCAT assigns a logical file identifier (LID) to a physical cataloged file for input or to a Level 1 or 2 file for output from a production program. ASGCAT provides a logical unit number (LUN) that can be used to perform Fortran I/O.

This routine provides access to existing files which include cataloged files and files that have been created by a previous job step and that are to be cataloged subsequently. For cataloged files, ASGCAT identifies the file using the input parameters, stages the file to magnetic disk if necessary, and associates the file name with the specified logical file identifier. The production program must open the cataloged file for read-only access. To access a file that was created by a previous program in the same job and that has not been cataloged, the LID must be the same as the one used by the program that created the file. Files that have not yet been cataloged can be modified.

ASGCAT also provides access to new files. It reserves disk space on a UCSS-managed disk, generates a unique file name, and associates the logical file identifier with the physical file name. The production program is responsible for the actual creation of the file. The logical file identifier must be used to open the file. The logical unit number can be used to perform Fortran I/O.

The calling sequence for ASGCAT is as follows:

```
CALL ASGCAT (UARS_DAY, DATA_TYPE, LEVEL, SUBTYPE, OLD_NEW, SIZE, LID,
             LUN, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
UARS_DAY	I*4	I	UARS day number (DDDD)
DATA_TYPE	CHAR*12	I	Data type Instrument identifier for Level 1 or for Level 2 data: = 'CLAES' = 'HALOE' = 'HRDI' = 'ISAMS' = 'MLS' = 'PEM' = 'SOLSTICE' = 'SUSIM' = 'WINDII'
LEVEL	CHAR*3	I	Data level '0 ' = Level 0 '1 ' = Level 1 '2 ' = Level 2 '3AS' = Level 3AS '3AL' = Level 3AL '3AT' = Level 3AT '3BS' = Level 3BS '3B' = Level 3B '3LP' = Level 3LP '3TP' = Level 3TP ' ' = no level applicable
SUBTYPE	CHAR*12	I	Subtype of data (dependent on the DATA_TYPE and LEVEL). Supply blank string if no subtype.
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = new file 'OLD ' = existing file 'HELD' = held file
SIZE	I*4	I	Estimated size of data file in blocks. This argument is only required when creating a new file.
LID	CHAR*16	I	Logical file identifier
LUN	I*4	O	Logical unit number

ARGUMENT	TYPE	I/O	DEFINITION
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_NOOLDFILE - No data found PFA_NOOPTDATA (PDS) - Optional file not available PFA_SOMEFILSTGD - File was staged

3.2.3 ASSIGN CORRELATIVE FILE (ASGCOR)

ASGCOR provides Fortran-callable read access to UARS day oriented correlative data. It identifies the file using the input parameters, insures that it is on magnetic disk, and associates the logical file identifier with the physical file name. The unique logical unit number should be used to perform Fortran I/O and the logical file identifier must be used to open the file. Correlative files must be opened for read only access. The user's program is responsible for issuing the read.

The calling sequence for ASGCOR is as follows:

CALL ASGCOR (SOURCE, SUBTYPE, UARS_DAY, LID, LUN, STATUS)

ARGUMENT	TYPE	I/O	DEFINITION
SOURCE	CHAR*12	I	Source of correlative data
SUBTYPE	CHAR*12	I	Subtype of data. Supply blank string if no subtype
UARS_DAY	I*4	I	UARS day number assigned to identify the correlative file
LID	CHAR*16	I	Logical file identifier
LUN	I*4	O	Logical unit number
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_NOOLDFILE - No data found PFA_NOOPTDATA (PDS) - Optional file not available

3.2.4 ASSIGN CALIBRATION FILE (ASGCAL)

ASGCAL assigns a logical file identifier (LID) to a cataloged calibration file for input or to a calibration file for output from a production program. It returns a unique logical unit number (LUN) that can be used to perform FORTRAN I/O on the file. Calibration files are those user-generated, instrument-oriented files of data that

are brought into the CDHF, placed under configuration control, and made available for production processing.

ASGCAL provides access to existing calibration files, namely cataloged files and files that have been created by a previous job step and are to be cataloged subsequently. For cataloged calibration files, ASGCAL identifies the file using the input parameters, stages the file to magnetic disk, if necessary, and associates the file name with the specified LID. The production program must open the cataloged for read-only access. To access a calibration file that was created by a previous program in the same job, and that has not yet been cataloged, the LID must be the same as the one used by the program that created the file. Files that have not yet been cataloged can be modified. Since calibration tables are time-indexed but are not always generated on a daily basis, a parameter is provided that allows the user to select the calibration file closest (either before, after, or nearest) to the processing day.

ASGCAL also provides access to new calibration files. It reserves disk space on a UCSS-managed disk, generates a unique file name, and associates the LID with the physical file name. The production program is responsible for the actual creation of the file. The LID must be used to open the file.

The calling sequence for ASGCAL is as follows:

```
CALL ASGCAL (SUBTYPE, CALB_ID, LEVEL, UARS_DAY, DMATCH, LID, LUN,
             STATUS, SIZE)
```

ARGUMENT	TYPE	I/O	DEFINITION
SUBTYPE	CHAR*12	I	Instrument ID associated with calibration data = 'CLAES' = 'HALOE' = 'HRDI' = 'ISAMS' = 'MLS' = 'PEM' = 'SOLSTICE' = 'SUSIM' = 'WINDII'
CALB_ID	CHAR*12	I	Calibration table identifier
LEVEL	CHAR*3	I	Data level associated with the calibration table '0 ' = Level 0 '1 ' = Level 1 '2 ' = Level 2 '3AL' = Level 3AL '3AS' = Level 3AS '3BS' = Level 3BS

ARGUMENT	TYPE	I/O	DEFINITION
			'3AT' = Level 3AT ' ' = no level applicable
UARS_DAY	I*4	I/O	UARS day number (DDDD). Actual day returned for input file. 0 = UARS day not applicable
DMATCH	CHAR*4	I	Day match criteria if file is old (Not used if UARS_DAY is not applicable) 'EXCT' = Locate file for the specified day 'PREV' = Locate file for the specified day or for the closest day less than the specified day 'NEXT' = Locate file for the specified day or for the closest day greater than the specified day 'NEAR' = Locate file for the closest day to the specified day Old_new_flag if file is new or held 'NEW' = New file 'HELD' = Held file
LID	CHAR*16	I	Logical file identifier
LUN	I*4	O	Logical unit number
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_NOOLDFILE - No data found PFA_NOOPTDATA (PDS) - Optional file not available
SIZE	I*4	I	Estimated size of data in blocks. This argument is only required when creating a new file.

3.2.5 ASSIGN SCRATCH FILE (ASGSCR)

ASGSCR provides access to scratch files. It reserves disk space for the file on a UCSS-managed disk and associates the logical file identifier with the physical scratch file name. The production program must use the logical file identifier to open the scratch file. A unique logical unit number is provided that must be used to perform Fortran reads and writes. Scratch files exist only for the duration of the production job. Upon successful completion of the production job, all scratch files are deleted. Scratch files are not deleted when a production job fails so that the files can be used to determine the reason for the failure. All scratch files must be assigned using

the ASGSCR routine to allow proper management of the UCSS production processing storage space.

Scratch files can be used to pass information from one program to another in the same production job. To access a scratch file that was created by a previous program, the same LID must be used. For example, PROGRAM1 created a 'NEW' scratch file with LID 'XYZ'. If PROGRAM2 needs to read the same scratch file, the ASGSCR parameters must specify that the file is 'HELD' and that the LID is 'XYZ'. If the same LID is used to access more than one new scratch file in the same job, then no subsequent program can use the LID to access the older scratch file(s).

The calling sequence for ASGSCR is as follows:

```
CALL ASGSCR (SIZE, OLD_NEW, LID, LUN, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
SIZE	I*4	I	Estimated size of data file in blocks. This argument is required only when creating a new file.
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = new file 'HELD' = held file
LID	CHAR*16	I	Logical file identifier
LUN	I*4	O	Logical unit number
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_NOOLDFILE - File not found

3.2.6 ASSIGN USER STATUS FILE (ASGUSR)

ASGUSR assigns the user-supplied LID to a user status file so that the production program can write to it. User status files are maintained in a directory associated with a specific type of job. These files are maintained cyclically so that the oldest version is deleted when a new version is created. The production program must use the logical file identifier to open the user status file. A unique logical unit number is provided that must be used to perform Fortran writes. The user's program is responsible for issuing the actual writes.

The calling sequence for ASGUSR is as follows:

```
CALL ASGUSR (LID, FILE_NUM, LUN, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier
FILE_NUM	I*4	I	User status file number. A job may access a number of status files up to a maximum specified in the job definition.
LUN	I*4	O	Logical unit number
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_NOOLDFILE - File not found

3.2.7 OPEN LEVEL 3AT DATA (OPENL3AT)

The OPENL3AT routine is used to initiate access to Level 3AT data.

To open for reading, the production program supplies the file type and the time range of the Level 3AT data to be read. OPENL3AT identifies the physical Level 3AT files required, insures that the files are on magnetic disk, and opens the files for read access in shared mode. OPENL3AT returns the base index within the UARS standard data array and maximum number of points to indicate the lowest index and maximum number of points available for the time range of the data. The production program must subsequently use the logical file identifier to read any data in the time range specified by the open. Appendix E describes the Level 3AT file format.

For output files, OPENL3AT reserves the necessary UCSS-managed disk space, generates a unique file name, and opens the file. The base index and maximum number of points parameters are used to determine the record size. The production program must use the LID when writing the Level 3AT record.

The calling sequence for OPENL3AT is as follows:

```
CALL OPENL3AT (DATA_TYPE, SUBTYPE, STRT_DATTIM, STOP_DATTIM, UARS_DAY,
               OLD_NEW, SIZE, BASE_INDEX, MAX_POINTS, LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Instrument identifier: = 'CLAES' = 'HALOE'

ARGUMENT	TYPE	I/O	DEFINITION
			= 'HRDI' = 'ISAMS ' = 'MLS' = 'PEM' = 'WINDII'
SUBTYPE	CHAR*12	I	Type of data. The set for each instrument is defined by the investigator.
STRT_DATTIM	I*4(2)	I	Start date and time in UDTF. Required only when accessing cataloged Level 3AT data.
STOP_DATTIM	I*4(2)	I	Stop date and time in UDTF. Required only when accessing cataloged Level 3AT data.
UARS_DAY	I*4	I	UARS day number (DDDD). Required only when accessing a new or held Level 3AT file.
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = new file 'OLD ' = existing file 'HELD' = held file from previous job step
SIZE	I*4	I	Estimated size of data file in blocks. This argument is required only when creating a new file.
BASE_INDEX	I*4	I/O	Start index (lowest) into the standard data array to be included in the file Input when creating a new file. Output when accessing an existing file.
MAX_POINTS	I*4	I/O	Maximum number of data points reported in the data array. Input when creating a new file. Output when accessing an existing file.
LID	CHAR*16	I	Logical file identifier
STATUS	I*4	O	Open status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_NODATARECS - No data records in physical file in user's processing range PFA_NOOLDFILE - No data found or file does not exist

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_NOOPTDATA - No optional data available PFA_OPTFILMISS - One or more optional file(s) missing in the time range PFA_OVRLPTIME - Two physical files have overlapping data

3.2.8 OPEN LEVEL 3AL DATA (OPENL3AL)

The OPENL3AL routine is used to initiate access to Level 3AL data.

To open for reading, the production program supplies the file type and the subtype of the Level 3AL data to be read. The time range of the data to be retrieved is supplied in UDTF. OPENL3AL identifies the physical Level 3AL files required, insures that the files are on magnetic disk, and opens the first file for read access in shared mode. OPENL3AL returns the base index and maximum number of points to indicate the lowest index and maximum number of points available for the time range of the data. The minimum and maximum latitudes are also returned to identify the latitude range of the available data for the specified time range. The production program must subsequently use the logical file identifier to read any data in the time range specified by the open. Appendix E describes the Level 3AL file record formats.

For output files, OPENL3AL reserves the necessary UCSS-managed disk space, generates a unique file name, and opens the file. The base index and maximum number of points parameters are used to determine the Level 3AL record size. The production program must use the LID when writing the Level 3AL records.

The calling sequence for OPENL3AL is as follows:

```
CALL OPENL3AL (DATA_TYPE, SUBTYPE, STRT_DATTIM, STOP_DATTIM, UARS_DAY,
               OLD_NEW, SIZE, BASE_INDEX, MAX_POINTS, MAX_LAT,
               MIN_LAT, LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Instrument identifier: = 'CLAES' = 'HRDI' = 'ISAMS ' = 'MLS' = 'PEM' = 'WINDII'
SUBTYPE	CHAR*12	I	Type of data. The set for each instrument is determined by the investigator.

ARGUMENT	TYPE	I/O	DEFINITION
STRT_DATTIM	I*4(2)	I	Start date and time in UDTF. Required only when accessing cataloged Level 3AL data.
STOP_DATTIM	I*4(2)	I	Stop date and time in UDTF. Required only when accessing cataloged Level 3AL data.
UARS_DAY	I*4	I	UARS day number (DDDD). Required only when accessing an uncataloged Level 3AL file.
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = new file 'OLD ' = existing file 'HELD' = held file from previous job step
SIZE	I*4	I	Estimated size of data file in blocks. This argument is required only when creating a new file.
BASE_INDEX	I*4	I/O	Start index (lowest) into the standard data array to be included in the file. Input when creating a new file. Output when accessing an existing file.
MAX_POINTS	I*4	I/O	Maximum number of data points reported in the data array. Input when creating a new file. Output when accessing an existing file.
MAX_LAT	REAL*4	O	For existing files, the highest latitude value available for the physical files spanned by the requested time range (between -88. and 88.)
MIN_LAT	REAL*4	O	For existing files, the lowest latitude value available for the physical files spanned by the requested time range (between -88. and 88.)
LID	CHAR*16	I	Logical file identifier
STATUS	I*4	O	Open status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_NODATARECS - No data records in physical file in user's processing range PFA_NOOLDFILE - No data found or file does not exist

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_NOOPTDATA - No optional data available PFA_OPTFILMISS - One or more optional file(s) missing in the time range PFA_OVLPTIME - Two physical files have overlapping times

3.2.9 OPEN LEVEL 3S DATA (OPENL3S)

The OPENL3S routine is used to initiate access to Level 3AS and 3BS data.

To create a new Level 3 solar data file, the calling program supplies the instrument ID, data level, UARS day number, starting wave length, wave length units, and the number of wave length bins. The number of wave length bins is used to calculate the Level 3 solar data record size. OPENL3S reserves the necessary UCSS-managed disk space (as specified in SIZE), generates a unique file name and opens the file.

To open a cataloged Level 3 solar data, the calling program supplies the instrument ID, data level, and UARS day range. OPENL3S uses these attributes to identify the required Level 3 solar data files, opens the first physical file, and returns the base wave length in nanometers as well as the number of wave length bins available in the data.

The calling sequence for OPENL3S is as follow:

```
CALL OPENL3S (DATA_TYPE, LEVEL, START_DAY, STOP_DAY, UARS_DAY,
              OLD_NEW, SIZE, BASE_WVLNGTH, MAX_VALUES, WVLNGTH_UNITS,
              LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Instrument identifier: = 'SOLSTICE' = 'SUSIM'
LEVEL	CHAR*3	I	Data level: = '3AS' = '3BS'
START_DAY	I*4	I	The first UARS day of a range from which data may be subsequently read Required only when accessing cataloged Level 3AS or 3BS data
STOP_DAY	I*4	I	The last UARS day of a range from which data may be subsequently read

ARGUMENT	TYPE	I/O	DEFINITION
			Required only when accessing cataloged Level 3AS or 3BS data
UARS_DAY	I*4	I	UARS day number (DDDD). Required only when accessing a new or uncataloged file.
OLD_NEW	CHAR*4	I	File existence flag: 'NEW ' = new file 'OLD ' = old file 'HELD' = held file from previous job step
SIZE	I*4	I	Estimated size of data file in blocks. This argument is required only when creating a new file.
BASE_WVLNGTH	REAL*4	I/O	The wavelength associated with the first value to be retrieved or written Input when creating a new file. Output when accessing an existing file.
MAX_VALUES	I*4	I/O	Maximum number of data values to be written or retrieved. Input when creating a new file. Output when accessing an existing file.
WVLNGTH_UNITS	CHAR*8	I/O	Indicates the unit of BASE_WVLNGTH. On output, will only be 'NM'. Possible values are: 'NM' - For nanometers, the standard bin size 'STANDARD' - Equivalent to 'NM' 'A' - For angstroms, calculated as the standard wavelength values times 10 'MICRON' - Calculated as the standard wavelength value times 1.E-03 'CM' - For centimeters, calculated as the standard wavelength value times 1.E-07
LID	CHAR*16	I	Logical file identifier
STATUS	I*4	O	Open status condition code SS$_NORMAL - Success PFA_CLSEEROLD - Error closing file PFA_NODATARECS - No data records in file PFA_NOOLDFILE - No old fileor file does not exist PFA_NOOPTDATA - No optional data available

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_OPTFILMISS - One or more optional files missing PFA_OVLPTIME - Two physical files have overlapping times

3.2.10 OPEN LEVEL 3TP DATA (OPENL3TP)

The OPENL3TP routine is used to initiate access to Level 3AT parameter files, also known as Level 3TP files.

To open for reading, the production program supplies the file type and the time range of the Level 3TP parameter data to be read. OPENL3TP identifies the physical Level 3TP files required, insures that the files are on magnetic disk, and opens the files for read access in shared mode. OPENL3TP returns the maximum number of 32-bit words to be contained in a parameter file record. The production program must subsequently use the logical file identifier to read any parameter data in the time range specified by the open. Appendix E describes the Level 3TP file format.

For output files, OPENL3TP reserves the necessary UCSS-managed disk space, generates a unique file name, and opens the file. The maximum number of parameters is used to determine the record size. The production program must use the LID when writing the Level 3TP record.

The calling sequence for OPENL3TP is as follows:

```
CALL OPENL3TP (DATA_TYPE, SUBTYPE, START_DATTIM, STOP_DATTIM,
               UARS_DAY, OLD_NEW, SIZE, MAX_NP, LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Instrument identifier
SUBTYPE	CHAR*12	I	Type of data. The subtypes for each instrument are defined by the investigator
START_DATTIM	I*4(2)	I	Start date and time in UDTF. Required only when accessing cataloged data.
STOP_DATTIM	I*4(2)	I	Stop date and time in UDTF. Required only when accessing cataloged data.
UARS_DAY	I*4	I	UARS day number (DDDD). Required only when accessing a new or held Level 3 parameter file.

ARGUMENT	TYPE	I/O	DEFINITION
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = New file 'OLD ' = Existing file 'HELD' = Held file from previous job step
SIZE	I*4	I	Estimated size of the parameter file in blocks. This argument is required only when creating a new file.
MAX_NP	I*4	I/O	For a new file, the number of 32-bit words to be contained in a parameter file record. For an existing file, the maximum number of 32-bit words contained in a record.
LID	CHAR*16	I	Logical file identifier
STATUS	I*4	O	Open status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_NODATARECS - No data records in file PFA_NOOLDFILE - No data found or file does not exist PFA_NOOPTDATA - No optional data available PFA_OPTFILMISS - One or more optional files missing in time range PFA_OVRLPTIME - Two physical files have overlapping data

3.2.11 OPEN LEVEL 3LP DATA (OPENL3LP)

The OPENL3LP routine is used to initiate access to Level 3AL parameter files, also known as Level 3LP files.

To open for reading, the production program supplies the file type and the subtype of the Level 3LP data to be read. The time range of the data to be retrieved is supplied in UDTF. OPENL3LP identifies the physical Level 3LP files required, insures that the files are on magnetic disk, and opens the first file for read access in shared mode. OPENL3LP returns the maximum number of 32-bit words available from each parameter data record. The production program must subsequently use the logical file identifier to read any data in the time range specified by the open. Appendix E describes the Level 3LP file formats.

For output files, OPENL3LP reserves the necessary UCSS-managed disk space, generates a unique file name, and opens the file. The number of parameters is used to determine the Level 3LP record size.

The production program must use the LID when writing the Level 3LP records.

The calling sequence for OPENL3LP is as follows:

```
CALL OPENL3LP (DATA_TYPE, SUBTYPE, START_DATTIM, STOP_DATTIM,
               UARS_DAY, OLD_NEW, SIZE, MAX_NP, MAX_LAT, MIN_LAT, LID,
               STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Instrument identifier
SUBTYPE	CHAR*12	I	Type of data. The subtypes for each instrument are defined by the investigator
START_DATTIM	I*4(2)	I	Start date and time in UDTF. Required only when accessing cataloged data.
STOP_DATTIM	I*4(2)	I	Stop date and time in UDTF. Required only when accessing cataloged data.
UARS_DAY	I*4	I	UARS day number (DDDD). Required only when accessing a new or held Level 3 parameter file.
OLD_NEW	CHAR*4	I	File existence flag 'NEW ' = New file 'OLD ' = Existing file 'HELD' = Held file from previous job step
SIZE	I*4	I	Estimated size of the parameter file in blocks. This argument is required only when creating a new file.
MAX_NP	I*4	I/O	For a new file, the maximum number of 32-bit words to be contained in a parameter file record. For an existing file, the maximum number of 32-bit words contained in a record.
MAX_LAT	REAL*4	O	For existing files, the highest latitude value available (between -88 and 88)
MIN_LAT	REAL*4	O	For existing files, the lowest latitude value available (between -88 and 88)
LID	CHAR*16	I	Logical file identifier

ARGUMENT	TYPE	I/O	DEFINITION
STATUS	I*4	O	Open status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_NODATARECS - No data records in file PFA_NOOLDFILE - No data found or file does not exist PFA_NOOPTDATA - No optional data available PFA_OPTFILMISS - One or more optional files missing in time range PFA_OVRLPTIME - Two physical files have overlapping data

3.2.12 QUALITY READ (QUALRD)

QUALRD provides the Fortran-callable read service for the Level 0 quality data. Requests for data are time-referenced by Engineering Major Frame (EMAF). Each call returns the instrument data from one EMAF. If the requested time does not correspond to an actual record time, the closest EMAF with a time greater than the requested time is returned. The time of the EMAF is returned along with the time of the next available EMAF.

The calling sequence for QUALRD is as follows:

```
CALL QUALRD (LID, REQ_DATTIM, RET_DATTIM, PARITY, FILL, VERSION,
             STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL0 call
REQ_DATTIM	I*4(2)	I/O	On input, date and time of the requested EMAF in UDTF. On output, date and time of the next EMAF available. If the end of data has been reached, REQ_DATTIM will be zero. If requested time is beyond the file stop time, REQ_DATTIM will be the file stop time.
RET_DATTIM	I*4(2)	O	Date and time in UDTF of the EMAF returned. RET_DATTIM will be zero if the requested time is beyond the file stop time.
PARITY	BYTE(256)	O	An array of bytes, each bit corresponding to one of the 2048 Science Minor Frames (SMIFs) of the

ARGUMENT	TYPE	I/O	DEFINITION
			EMAF, indicating parity errors detected or presence of fill 0 = SMIF has good cyclical redundancy check (CRC) 1 = SMIF has bad CRC or fill
FILL	BYTE(256)	O	An array of bytes, each bit corresponding to one of the 2048 SMIFs of the EMAF, indicating whether the SMIF is filled 0 = SMIF contains data 1 = SMIF contains fill
VERSION	I*2(2)	O	CCB version and cycle number associated with Level 0 file read
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_ATCINCRMENT - ATC increment error PFA_CLSEERROLD - Error closing file PFA_EOF - Last record of file PFA_FILETMGAP - Time gap between two physical files exceeded normal gap PFA_REQTMPAST - Requested time is beyond file stop time PFA_RETTMPAST - Retrieved time is beyond processing stop time PFA_RETTMPREV - Retrieved time precedes processing start time

3.2.13 READ LEVEL 0 (READL0)

READL0 provides a Fortran-callable read service for all types of Level 0 data. Requests for data are time-referenced by EMAF. Each call returns the instrument data from one EMAF. If the requested time does not correspond to an actual record time, the closest EMAF with a time greater than the requested time is returned. The time of the EMAF is returned along with the time of the next available EMAF. For files with one record per EMAF, the data returned is in the format described in Appendix D. For files with two records per EMAF, the data returned consists of the data header from the first record followed by the data from both records.

When the last EMAF of a Level 0 file has been returned as part of a read, the returned status will be set to PFA_EOF to show that no more data is available for further sequential input from the file and the time of the next available EMAF will be set to zero.

The calling sequence for READL0 is as follows:

```
CALL READL0 (LID, REQ_DATTIM, RET_DATTIM, EMAF_REC, PARITY, FILL,
             GAP_FLAG, TIME_FLAG, EMAF_RATE, VERSION, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL0 call
REQ_DATTIM	I*4(2)	I/O	On input, date and time of the requested EMAF in UDTF. On output, date and time of the next EMAF available. If the end of data has been reached, REQ_DATTIM will be zero. If requested time is beyond the file stop time, REQ_DATTIM will be the file stop time.
RET_DATTIM	I*4(2)	O	Date and time in UDTF of the start of the EMAF returned. RET_DATTIM will be zero if the requested time is beyond the file stop time.
EMAF_REC	BYTE(*)	O	Level 0 telemetry record for the selected data type. See Appendix D for the specific format for the type of Level 0 data to be read. EMAF_REC contains one EMAF of data.
PARITY	BYTE(8)	O	A binary array of parity flags for the 64 Science Major Frames (SMAFs) in the EMAF. There is one bit flag for each SMAF. 0 = all SMIFs in SMAF have good CRC codes 1 = one or more SMIFs have CRC errors or contain fill data
FILL	BYTE(8)	O	A binary array of fill flags for the SMAFs in the EMAF. There is one bit flag for each SMAF. 0 = all SMIFs in the SMAF contain data 1 = one or more SMIFs contain fill
GAP_FLAG	I*2	O	Indicates whether or not the EMAF follows a gap 0 = no gap 1 = EMAF follows a gap

ARGUMENT	TYPE	I/O	DEFINITION
TIME_FLAG	I*2	O	ATC time increment flag 0 = normal ATC increment 1 = abnormal ATC increment
EMAF_RATE	I*4	O	EMAF rate (msec/EMAF)
VERSION	I*2(2)	O	CCB version and cycle number of the Level 0 file read
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_ATCINCRMENT - ATC increment error PFA_CLSEERROLD - Error closing file PFA_EOF - Last record of file PFA_FILETMGAP - Time gap between two physical files exceeded normal gap PFA_REQTMPAST - Requested time is beyond file stop time PFA_RETTMPAST - Retrieved time is beyond processing stop time PFA_RETTMPREV - Retrieved time precedes processing start time

3.2.14 READ LEVEL 3AT (READL3AT)

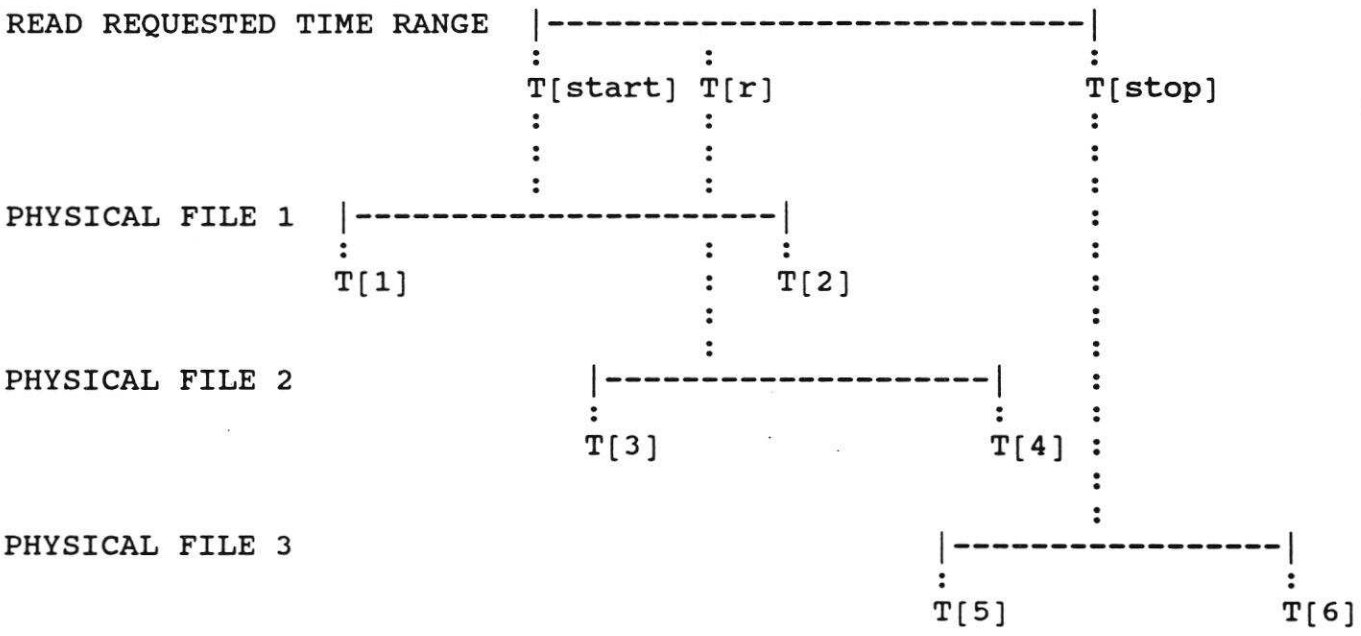
READL3AT provides a Fortran-callable read service for nonsolar, time-referenced Level 3AT data. Data is requested by time range, allowing the user to read multiple records of data at a time. START_INDEX and NUM_POINTS must overlap the range that was returned by the OPENL3AT routine via BASE_INDEX and MAX_POINTS. READL3AT retrieves the requested portions of all of the records within the specified time range, with their corresponding times. READL3AT returns the actual number of records read and the time of the next available record. A fill value of X'00008000' is used when data for a requested element of the UARS standard array is not available. This value was chosen because it is a reserved value and not a valid floating point number (special handling required). If the number of records in the time range exceeds the maximum dimension of the user array, READL3AT only reads MAX_DIM records and returns the appropriate status.

When the last record of a Level 3AT file has been returned as part of a read, the returned status will be set to PFA_EOF to show that no more data is available for further sequential input from the file and the time of the next available record will be set to zero.

The values of the local solar time and the solar zenith angle associated with each profile are also returned if requested in the call via the LST and SZA arguments.

The time ranges of the Level 3AT files are not expected to overlap. However, if there is time overlap of files in the virtual time range requested, READL3AT handles the situation. In the example shown in Figure 3-1, READL3AT retrieves records from File 1 starting at time T[start] through time T[2], continues reading records from File 2 with times after T[2] through T[4], and finishes by retrieving records from File 3 with times between T[4] and T[stop]. In the case of retrieving a single record with a time that lies within two physical files, the file from which the record is retrieved is dependent upon which file is the last one to have been read. For example, if T[r] lies between T[3] and T[2] and T[r] is the first record to be read or the last record read was from File 1, then the requested record is retrieved from File 1. Otherwise, the record is retrieved from File 2. Stated another way, records in the overlap time range are retrieved from the first file when reading sequentially in the forward direction, and are retrieved from the second file when reading backwards through the time range.

Figure 3-1. READL3AT Record Overlap Example



The calling sequence for READL3AT is as follows:

```
CALL READL3AT (LID, STRT_DATTIM, STOP_DATTIM, START_INDEX, NUM_POINTS,
               MAX_DIM, RET_DATTIM, NXT_DATTIM, NUM_REC, DATA3A, QUAL,
               LAT, LONG, VERSION, STATUS, LST, SZA)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3AT call
STRT_DATTIM	I*4(2)	I	Start date/time of Level 3AT data to be retrieved, in UDTF
STOP_DATTIM	I*4(2)	I	Stop date/time of Level 3AT data to be retrieved, in UDTF
START_INDEX	I*4	I	Index of first element in the UARS standard data array to be retrieved
NUM_POINTS	I*4	I	Number of elements in the UARS standard data array (NP) to be retrieved
MAX_DIM	I*4	I	Maximum number of records (NR) to be retrieved
RET_DATTIM	I*4 (2,NR)	O	Array containing the dates and times for the Level 3AT records retrieved, in UDTF
NXT_DATTIM	I*4(2)	O	Date/time of next available Level 3AT record in UDTF. Zero if end of data has been reached.
NUM_REC	I*4	O	Number of Level 3AT records retrieved
DATA3A	REAL*4 (NP,NR)	O	Two dimensional array containing the data type specified at OPENL3AT time. The first index, offset by START_INDEX is associated with the element number in the UARS standard data array. The second index is associated with time.
QUAL	REAL*4 (NP,NR)	O	Two dimensional array containing quality information associated with the data values returned in DATA3A. The indices are the same as for DATA3A.
LAT	REAL*4 (NR)	O	Array of geodetic latitudes corresponding to the Level 3AT records retrieved

ARGUMENT	TYPE	I/O	DEFINITION
LONG	REAL*4 (NR)	O	Array of geodetic longitudes corresponding to the Level 3AT records retrieved
VERSION	I*2 (2,NR)	O	Array containing the source file CCB version and cycle associated with each Level 3AT record retrieved
STATUS	I*4	O	Read status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing cataloged file PFA_EOF - Last record of file returned PFA_FILETMGAP - Time gap between two physical files exceeded normal gap PFA_NODATARECS - New or held file has no data PFA_NOOVRLAPTRNG - No overlap between requested time range and files time range PFA_NROVRMXDIM - More records in time range than can be retrieved at one time PFA_RETTMPAST - Retrieved time(s) are beyond processing stop time PFA_RETTMPREV - Retrieved time(s) precede processing start time
LST	REAL*4 (NR)	O	Array containing the local solar times associated with each Level 3AT record retrieved (optional)
SZA	REAL*4 (NR)	O	Array containing the solar zenith angles associated with each Level 3AT record retrieved (optional)

3.2.15 READ LEVEL 3S (READL3S)

READL3S provides a Fortran-callable read service for the Level 3AS and Level 3BS data. Requests are time-referenced by UARS day. A fill value of X'00008000' is used when data for a requested element is not available. The calling program specifies the UARS day range to be read, the starting wavelength bin, and the number of flux values to be retrieved. The program also provides the wavelength unit, the flux unit and the distance flag which are used to specify the units of the wavelengths and flux values returned and to indicate whether the flux values should be corrected or not. READL3S reads the data record from each Level 3 solar file in the specified day range or up to the number of days specified by MAX_DAYS if the range is too large. The wavelengths are returned in WVLNGTHS and are in the units

specified by WVLNGTH_UNITS. The UARS day number of the next available day is also returned.

When the last record of a Level 3S file has been returned as part of a read, the returned status will be set to PFA_EOF to show that no more data is available for further sequential input from the file and the time of the next available record will be set to zero.

The calling sequence for READL3S is as follows:

```
CALL READL3S (LID, START_DAY, STOP_DAY, MAX_DAYS, START_WVLNGTH,
              NUM_VALUES, FLUX_UNITS, WVLNGTH_UNITS, DISTANCE_FLAG,
              RET_DAY, NXT_DAY, NUM_RET_DAYS, WVLNGTHS, DATA3S,
              QUALITY, NUM_PARAMS, PARAMS, VERSION, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3S call
START_DAY	I*4	I	The first UARS day of Level 3S data to be retrieved
STOP_DAY	I*4	I	The last UARS day of Level 3S data to be retrieved
MAX_DAYS	I*4	I	Maximum number of days (ND) of data to be retrieved
START_WVLNGTH	REAL*4	I	The wavelength associated with the first value to be returned (in the units indicated by the value of WVLNGTH_UNITS as defined below)
NUM_VALUES	I*4	I	The number of data values (NV) to be returned, and correspondingly, the number of wavelength values returned in WVLNGTHS
FLUX_UNITS	CHAR*17	I	Indicates the units in which the DATA3S array will be returned. Possible values are: 'W/M^3' - The standard unit in which the data is stored (Watts per cubic meter) 'STANDARD' - Same as the above 'W/CM^3' - Watts per cubic centimeter (calculated by multiplying the standard values by 1.E-06) 'MW/M^2/NM' - Milliwatts per meter squared per nanometer (calculated by multiplying the standard value by 1.E-06)

ARGUMENT	TYPE	I/O	DEFINITION
			'ERGS/S/CM^2/A' - Ergs per second per centimeter squared per angstrom (calculated by multiplying the standard value by 1.E-07) 'PHOTONS/S/CM^2/NM' - Photons per second per centimeter squared per nanometer (calculated by multiplying the standard value by 503.438 times the wavelength in nanometers) 'PHOTONS/S/CM^2/A' - Photons per second per centimeter squared per angstrom (calculated by multiplying the standard value by 50.3438 times the corresponding wavelength value in nanometers)
WVLNGTH_UNITS	CHAR*8	I	Indicates the units of START_WVLNGTH. Possible values are: 'NM' - Nanometers, the standard bin size 'STANDARD' - Equivalent to 'NM' 'A' - Angstroms, calculated as the standard wavelength value times 10 'MICRON' - Calculated as the standard wavelength value times 1.E-03 'CM' - Centimeters, calculated as the standard wavelength value times 1.E-07
DISTANCE_FLAG	CHAR*11	I	Indicates whether the DATA3S array of solar fluxes should reported at 1 AU distance from the sun or reported at the actual point of measurement. Values are: '1_AU' - Reported at 1 AU distance (stored this way) 'UNCORRECTED' - Reported at the point of measurement adjusted by applying the inverse square law to the mean solar distance attribute stored with the data
RET_DAY	I*4 (ND)	O	Array containing the UARS day number for the Level 3 solar records retrieved
NXT_DAY	I*4	O	The UARS day of the next available Level 3 solar data record. Zero if end of data has been reached
NUM_RET_DAYS	I*4	O	Number of days of Level 3 solar data returned

ARGUMENT	TYPE	I/O	DEFINITION
WVLNGTHS	REAL*4 (NV)	O	Wavelength values in the units specified by WVLNGTH_UNITS corresponding to the solar flux array DATA3S. NV is the number of values.
DATA3S	REAL*4 (NV,ND)	O	Returned flux values in FLUX_UNITS. The first subscript (NV) is the wavelength bin index. The second subscript (ND) is the UARS day number index.
QUALITY	REAL*4 (NV,ND)	O	The quality values corresponding to the flux data
NUM_PARAMS	I*4 (ND)	O	The number of parameter name and value pairs provided in PARAMS
PARAMS	CHAR*20 (2,40,ND)	O	A table used to return parameters stored with the data. Each entry in the table consists of a pair, a parameter name and its corresponding value.
VERSION	I*2 (2,ND)	O	Array containing the CCB version and cycle associated with each Level 3 solar record retrieved
STATUS	I*4	O	READL3S status condition code: SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_EOF - Last record of file PFA_FILETMGAP - Time gap between two physical files exceeded normal gap PFA_NODATARECS - New or held file has no data PFA_NOOVRLAPTRNG - No overlap between requested time range and files time range PFA_NROVRMXDIM - Number of records requested exceeds MAX_DAYS PFA_RETTMPAST - Returned records beyond processing time range PFA_RETTMPREV - Returned records precedes processing time range

3.2.16 READ LEVEL 3AL (READL3AL)

READL3AL provides a Fortran-callable read service for Level 3AL data. Data is requested for a latitude band (at 4 degree intervals between -88. and 88.) by time range and profile range. The profile range as specified by START_INDEX and NUM_POINTS must fall within the

range that was returned by the OPENL3AL routine via BASE_INDEX and MAX_POINTS. READL3AL retrieves the requested portions of all of the records for the requested latitude band within the specified time range. The time, the longitude, the quality values, and the version numbers are also returned for each set of profiles. READL3AL returns the actual number of records read and the time of the next available record. A fill value of X'00008000' is used when data for a requested element of the UARS standard array is not available. This value was chosen because it is a reserved value and not a valid floating point number (special handling required). If the number of records in the time range exceeds the maximum dimension of the user array, READL3AL only reads MAX_DIM records and returns the appropriate status.

When the last record of a Level 3AL file has been returned as part of a read, the returned status will be set to PFA_EOD to show that no more data is available for further sequential input at the desired latitude and the time of the next available record will be set to zero.

The values of the local solar time and the solar zenith angle associated with each profile are also returned if requested in the call via the LST and SZA arguments.

The calling sequence for READL3AL is as follows:

```
CALL READL3AL (LID, LAT, STRT_DATTIM, STOP_DATTIM, START_INDEX,
               NUM_POINTS, MAX_DIM, RET_DATTIM, NXT_DATTIM, NUM_REC,
               DATA3A, QUAL, LONG, VERSION, STATUS, LST, SZA)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3AL call
LAT	REAL*4	I	Geodetic grid latitude of data to be retrieved (must be at 4 degree interval between -88. and 88 with an allowed tolerance of 0.5 degrees.)
STRT_DATTIM	I*4(2)	I	Start date/time of Level 3AL data to be retrieved, in UDTF
STOP_DATTIM	I*4(2)	I	Stop date/time of Level 3AL data to be retrieved, in UDTF
START_INDEX	I*4	I	Index of first element in the UARS standard data array to be retrieved
NUM_POINTS	I*4	I	Number of elements in the UARS standard data array (NP) to be retrieved
MAX_DIM	I*4	I	Maximum number of records (NR) to be retrieved

ARGUMENT	TYPE	I/O	DEFINITION
RET_DATTIM	I*4 (2,NR)	O	Array containing the dates and times for the Level 3AL records retrieved, in UDTF
NXT_DATTIM	I*4(2)	O	Date/time of next available Level 3AL record in UDTF. Zero, if end of data has been reached.
NUM_REC	I*4	O	Number of Level 3AL records retrieved
DATA3A	REAL*4 (NP,NR)	O	Two-dimensional array containing the data type specified at OPENL3AL time. The first index, offset by START_INDEX is associated with the element number in the UARS standard data array. The second index is associated with time.
QUAL	REAL*4 (NP,NR)	O	Two dimensional array containing quality information associated with the data values returned in DATA3A. The indices are the same as for DATA3A.
LONG	REAL*4 (NR)	O	Array of geodetic longitudes corresponding to the Level 3AL records retrieved
VERSION	I*2 (2,NR)	O	Array containing the source file CCB version and cycle associated with each Level 3AL record retrieved
STATUS	I*4	O	Read status code SS$_NORMAL - Normal return PFA_CLSEERROLD Error closing cataloged file PFA_EOD - Last record of file returned PFA_NODATAFND - No data in file for requested time range at requested latitude PFA_NODATARECS - New or held file has no data PFA_NOOVRLPTRNG - No overlap between requested time range and file's time range PFA_NROVRMXDIM - More records in time range than can be retrieved at one time PFA_REQLATOUT - No data for requested latitude PFA_RETTMPAST - Retrieved time(s) are beyond processing stop time PFA_RETTMPREV - Retrieved time(s) precede processing start time

ARGUMENT	TYPE	I/O	DEFINITION
LST	REAL*4 (NR)	O	Array containing the local solar times associated with each Level 3AL record retrieved (optional)
SZA	REAL*4 (NR)	O	Array containing the solar zenith angles associated with each Level 3AL record retrieved (optional)

3.2.17 READ LEVEL 3TP DATA (READL3TP)

READL3TP provides a Fortran-callable read service for non-solar time-referenced Level 3AT parameter files, also known as Level 3TP files. Parameter data is requested by time range, allowing the user to read multiple records of data at a time. The value of MAX_NP requested must not exceed the corresponding value returned by the OPENL3TP routine. READL3TP retrieves parameter data in the requested portions of all of the records that fall within the specified time range, with their corresponding times. READL3TP returns the actual number of records, the number of parameters, and the time of the next available record. If the number of records in the time range exceeds MAX_DIM, the maximum dimension of the user array, READL3TP only reads MAX_DIM records and returns the appropriate status.

Overlapping time ranges in Level 3TP files are handled in the same manner as for Level 3AT files (see Section 3.2.14).

The calling sequence for READL3TP is as follows:

```
CALL READL3TP (LID, START_DATTIM, STOP_DATTIM, MAX_NP, MAX_DIM,
               RET_DATTIM, NEXT_DATTIM, NUM_REC, NP, PARAMETERS, LAT,
               LONG, VERSION, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier specified in the OPENL3TP call
START_DATTIM	I*4(2)	I	Start date and time (in UDTF) of the Level 3 data records associated with the parameters to be retrieved
STOP_DATTIM	I*4(2)	I	Stop date and time (in UDTF) of the Level 3 data records associated with the parameters to be retrieved
MAX_NP	I*4	I	Maximum number of 32-bit words to be retrieved from the parameter file record

ARGUMENT	TYPE	I/O	DEFINITION
MAX_DIM	I*4	I	Maximum number of records (NR) to be retrieved. If number of records found exceeds this the first MAX_DIM records are returned.
RET_DATTIM	I*4 (2,NR)	O	Array containing the dates and times (in UDTF) of the Level 3 records associated the parameter records retrieved
NEXT_DATTIM	I*4(2)	O	Date and time (in UDTF) of the next available parameter record
NUM_REC	I*4	O	Number of parameter records returned
NP	I*4(NR)	O	Array containing the number of 32-bit words contained in each parameter file record
PARAMETERS	BYTE (4*NP,NR)	O	Array containing the parameter records retrieved. The array contains NUM_REC parameter records. The format and structure of each parameter record is the instrument investigator's responsibility.
LAT	REAL*4 (NR)	O	Array containing the latitudes of the Level 3 data records associated with each parameter record retrieved
LONG	REAL*4 (NR)	O	Array containing the longitudes of the Level 3 data records associated with each parameter record retrieved
VERSION	I*4 (2,NR)	O	Array containing the source file CCB versions and cycles associated with each parameter record retrieved
STATUS	I*4	O	Read status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_EOF - Last record of file returned PFA_FILETMGAP - Time gap between two physical files exceeded normal gap PFA_NODATARECS - New or held file has no data PFA_NOOVRLPTRNG - No overlap between requested time range and file's time range PFA_NROVRMXDIM - More records in time range than can be retrieved at one time

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_RETTMPAST - A retrieved time is beyond processing stop time PFA_RETTMPREV - A retrieved time precedes processing start time

3.2.18 READ LEVEL 3LP DATA (READL3LP)

READL3LP provides a Fortran-callable read service for Level 3AL parameter files, also known as Level 3LP files. Parameter data is requested for a latitude band (at 4 degree intervals between -88 and 88.) by time range and number of parameters. The number of parameters must not exceed the value of MAX_NP returned by the OPENL3TP routine. READL3AL retrieves parameters within the requested portions of all of the records that lie at the requested latitude band and within the specified time range. The time, longitude, version numbers and number of parameters are returned for each parameter record. READL3TP also returns the actual number of records retrieved and the time of the next available record. If the number of records available in the time range exceeds MAX_DIM, the maximum dimension of the user array, READL3TP only reads MAX_DIM records and returns the appropriate status.

The calling sequence for READL3LP is as follows:

```
CALL READL3LP (LID, LAT, START_DATTIM, STOP_DATTIM, MAX_NP, MAX_DIM,
               RET_DATTIM, NEXT_DATTIM, NUM_REC, NP, PARAMETERS,
               LONG, VERSION, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier specified in the OPENL3LP call
LAT	REAL*4	I	Latitude corresponding to the associated Level 3 data records
START_DATTIM	I*4(2)	I	Start date and time (in UDTF) of the Level 3 data records associated with the parameters to be retrieved
STOP_DATTIM	I*4(2)	I	Stop date and time (in UDTF) of the the Level 3 data records associated with the parameters to be retrieved
MAX_NP	I*4	I	Maximum number of 32-bit words to be retrieved from the parameter file record

ARGUMENT	TYPE	I/O	DEFINITION
MAX_DIM	I*4	I	Maximum number of records (NR) to be retrieved. If number of records found exceeds this, the first MAX_DIM records are returned.
RET_DATTIM	I*4 (2,NR)	O	Array containing the dates and times (in UDTF) of the Level 3 records associated with the parameter records retrieved
NEXT_DATTIM	I*4 (2)	O	Date and time (in UDTF) of the next available Level 3 record associated with a parameter record
NUM_REC	I*4	O	Number of parameter records returned
NP	I*4 (NR)	O	Array containing the number of 32-bit words contained in each parameter file record. NP may be greater than MAX_NP, but only MAX_NP 32-bit words will be returned.
PARAMETERS	BYTE (4*MAX_NP,NR)	O	Array containing the parameter records retrieved. The array contains NUM_REC parameter records. The format and structure of each parameter record is the instrument investigator's responsibility
LONG	REAL*4 (NR)	O	Array containing the longitudes of the Level 3 data records associated with each parameter record retrieved
VERSION	I*4 (2,NR)	O	Array containing the source file CCB versions and cycles associated with each parameter record retrieved
STATUS	I*4	O	Read status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing file PFA_EOD - Last record returned PFA_NODATFND - No data in file for requested time range at requested latitude PFA_NODATARECS - New or held file has no data PFA_NOOVRLPTRNG - No overlap between requested time range and file's time range PFA_NROVRMXDIM - More records in time range than can be retrieved at one time

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_REQLATOUT - No data in file for requested latitude PFA_RETTMPAST - A retrieved time is beyond processing stop time PFA_RETTMPREV - A retrieved time precedes processing start time

3.2.19 WRITE LEVEL 3AT (WRITEL3AT)

WRITEL3AT writes time-referenced Level 3AT data in the standard record format (see Appendix E). The Level 3AT file first must be created by calling the OPENL3AT routine. Level 3AT records are written on UARS minute boundaries. START_INDEX and NUM_POINTS specify the range of the data provided by the user. This range must fall within the range specified to OPENL3AT via the BASE_INDEX and MAX_POINTS parameters. If the user-provided data range is a subset of the file data range, WRITEL3AT inserts the fill value (X'00008000') for the remaining data elements. The user must provide the fill value for any missing elements in the middle of the user-provided data range. The user does not need to create fill records.

WRITEL3AT also calculates the local solar time and the solar zenith angle for the record to be written and stores their values in the record's header. These calculated values may then be retrieved when the record is read by specifying the LST and SZA arguments in the call to READL3AT.

The calling sequence for WRITEL3AT is as follows:

```
CALL WRITEL3AT (LID, DATTIM, START_INDEX, NUM_POINTS, DATA3A, QUAL,
                LAT, LONG, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3AT call
DATTIM	I*4(2)	I	Date and time of the Level 3AT record in UDTF
START_INDEX	I*4	I	Index of first element of the UARS standard data array provided
NUM_POINTS	I*4	I	Number of elements in the UARS standard data array provided
DATA3A	REAL*4 (NUM_POINTS)	I	One dimensional array containing the data type specified at OPENL3AT time. This array contains NUM_POINTS data values for consecutive elements in the

ARGUMENT	TYPE	I/O	DEFINITION
			UARS standard data array starting at element index, START_INDEX
QUAL	REAL*4 (NUM_POINTS)	I	Array containing the quality information associated with the data values in DATA3A
LAT	REAL*4	I	Geodetic latitude corresponding to the Level 3AT data record
LONG	REAL*4	I	Geodetic longitude corresponding to the Level 3AT data record (0-360)
STATUS	I*4	O	Write status code SS$_NORMAL - Normal return PFA_TIMAFTUARS - Record time beyond nominal UARS day PFA_TIMPREUARS - Record time before nominal UARS day

3.2.20 WRITE LEVEL 3S (WRITEL3S)

WRITEL3S writes a single record of Level 3AS or Level 3BS data. The Level 3AS or Level 3BS file must first be created by calling the OPENL3S routine. The calling program must use the same LID as specified to OPENL3S.

The Level 3 solar data is stored in a UARS standard solar data array, where each array element contains the integrated flux from a 1.0 nm wide wavelength bin centered on the 0.5 nm from 115.5 to 425.5 nm. The array can, therefore contain up to 311 solar flux values. The DATA3S array must contain the number of flux values specified by MAX_VALUES in the call to OPENL3S. The calling program must supply the same number of values in the QUALITY array. The units of the flux values must be watts per cubic meter.

Additional information that is stored in the solar data file with a solar spectrum includes the irradiance values for 4 coronal lines, Lyman Alpha, a Magnesium line and a Calcium line. Also, the mean solar distance value (MSD) which is needed to perform the 1 AU-to-actual distance irradiance correction in READL3S must be provided. This information is supplied by the calling program in the PARAMS array, which can hold up to 40 parameters. Each parameter in the array is specified by a pair of values, the first one containing the parameter's name and the second one, its value.

The calling sequence for WRITEL3S is as follows:

CALL WRITEL3S (LID, DATA3S, QUALITY, NUM_PARAMS, PARAMS, STATUS)

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3S call
DATA3S	REAL*4 (NP)	I	Level 3AS or 3BS data. The irradiance array is assumed to be in WATTS/M*3 units. NP is the value specified as MAX_VALUES in the OPENL3S call.
QUALITY	REAL*4 (NP)	I	Level 3AS or Level 3BS data quality. NP same as above.
NUM_PARAMS	I*4	I	The number of parameter name and value pairs provided in PARAMS
PARAMS	CHAR*20 (2,40)	I	A table used to pass parameters to be stored with the data for subsequent use. Each entry in the table consists of a pair of values, namely a parameter name and its corresponding value in ASCII. The Mean Solar Distance (MSD) parameter MUST be provided.
STATUS	I*4	O	Write status condition code SS$_NORMAL - Normal return PFA_PREVSOLDAT - Already wrote solar record to file

3.2.21 WRITE LEVEL 3AL (WRITEL3AL)

WRITEL3AL writes Level 3AL data in the standard record format (see Appendix E). The Level 3AL file must first be created by calling the OPENL3AL routine. Level 3AL records are written on UARS minute boundaries. START_INDEX and NUM_POINTS specify the range of the data provided by the user. This range must fall within the range specified to OPENL3AL via the BASE_INDEX and MAX_POINTS parameters. If the user-provided data range is a subset of the file data range, WRITEL3AL inserts the fill value (X'00008000') for the remaining data elements. The user must provide the fill value for any missing elements in the middle of the user-provided data range.

WRITEL3AL also calculates the local solar time and the solar zenith angle for the record to be written and stores their values in the record's header. These calculated values may then be retrieved when the record is read by specifying the LST and SZA arguments in the call to READL3AL.

The Level 3AL records are written at standard values of latitude, i.e. every 4 degrees of latitude from -88. to 88. For each data array provided, the user must provide the associated GMT date and time and longitude values.

The calling sequence for WRITEL3AL is as follows:

```
CALL WRITEL3AL (LID, DATTIM, START_INDEX, NUM_POINTS, DATA3A, QUAL,
                LAT, LONG, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier as specified in the OPENL3AL call
DATTIM	I*4(2)	I	Date and time of the Level 3AL record in UDTF
START_INDEX	I*4	I	Index of first element of the UARS standard data array provided
NUM_POINTS	I*4	I	Number of elements in the UARS standard data array provided
DATA3A	REAL*4 (NUM_POINTS)	I	One dimensional array containing the data type specified at OPENL3AL time. This array contains NUM_POINTS data values for consecutive elements in the UARS standard data array starting at element index, START_INDEX.
QUAL	REAL*4 (NUM_POINTS)	I	Array containing the quality information associated with the data values in DATA3A
LAT	REAL*4	I	Geodetic latitude grid value corresponding to the Level 3AL data record. A tolerance of 0.5 degrees is allowed in the specification of this value.
LONG	REAL*4	I	Geodetic longitude corresponding to the Level 3AL data record (0-360)
STATUS	I*4	O	Write status code SS$_NORMAL - Normal return PFA_TIMAFTUARS - Record time beyond nominal UARS day PFA_TIMPREUARS Record time before nominal UARS day

3.2.22 WRITE LEVEL 3TP DATA (WRITEL3TP)

WRITEL3TP writes time-referenced Level 3AT parameter files, also know as Level 3TP files, in the standard record format (see Appendix E). The Level 3TP file first must be created by calling the OPENL3TP routine. Level 3TP records, like the Level 3AT records, are written on UARS minute boundaries. NUM_PARAMS specifies the number of 32-bit words to be written to the parameter file. This number must not be greater than the maximum number of parameters specified to the OPENL3TP routine via MAX_NP. If the user-provided number of parameters is less than that value of MAX_NP, WRITEL3TP inserts zeros as fill data.

The calling sequence for WRITEL3TP is as follows:

```
CALL WRITEL3TP (LID, DATTIM, LAT, LONG, NUM_PARAMS, PARAMETERS,
                STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier specified in the OPENL3TP call
DATTIM	I*4(2)	I	Date and time of the associated Level 3 data record (in UDTF)
LAT	REAL*4	I	Latitude corresponding to the associated Level 3 data record
LONG	REAL*4	I	Longitude corresponding to the associated Level 3 data record (0-360)
NUM_PARAMS	I*4	I	Number of 32-bit words to be written to the parameter file
PARAMETERS	BYTE (4*MAX_NP)	I	Buffer containing the parameters to be associated with the Level 3 data record. The format and structure of this buffer is the instrument investigator's responsibility.
STATUS	I*4	O	Write status code SS$_NORMAL - Normal return PFA_TIMAFTUARS - Record time is beyond nominal UARS day PFA_TIMPREUARS - Record time precedes nominal UARS day

3.2.23 WRITE LEVEL 3LP DATA (WRITEL3LP)

WRITEL3LP writes Level 3AL parameter file, also known as Level 3LP files, in the standard record format (see Appendix E). The Level 3LP file must first be created by calling the OPENL3LP routine. Level 3LP records, like Level 3AL records, are written on UARS minute boundaries. NUM_PARAMS specifies the number of parameters provided by the user. This number must not be greater than the value of MAX_NP specified to the OPENL3LP routine. If the user-provided number of parameters is less than the value of MAX_NP, WRITEL3LP inserts zeros as fill data.

The Level 3LP records are written at standard values of latitude, i.e. every 4 degrees of latitude from -88 to 88. For each parameter array provided, the user must provide the associated GMT date and time and longitude values.

The calling sequence for WRITEL3LP is as follows:

```
CALL WRITEL3LP(LID, DATTIM, LAT, LONG, NUM_PARAMS, PARAMETERS, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier specified in the OPENL3LP call
DATTIM	I*4(2)	I	Date and time of the associated Level 3 data record (in UDTF)
LAT	REAL*4	I	Latitude corresponding to the associated Level 3 data record
LONG	REAL*4	I	Longitude corresponding to the associated Level 3 data record (0-360)
NUM_PARAMS	I*4	I	Number of 32-bit words to be written to the parameter file
PARAMETERS	BYTE (4*MAX_NP)	I	Buffer containing the parameters to be associated with the Level 3 data record. The format and structure of this buffer is the instrument investigator's responsibility.
STATUS	I*4	O	Write status code SS$_NORMAL - Normal return PFA_TIMAFTUARS - Record time is beyond nominal UARS day PFA_TIMPREUARS - Record time precedes nominal UARS day

3.2.24 CLOSE LOGICAL FILE (CLOSELF)

The CLOSELF routine is used to terminate file access activity for the Level 0, 3AT, 3AS, 3BS, 3AL, 3LP, and 3TP data. The production program specifies the logical file identifier associated with the virtual or physical file to be closed, and the file disposition (FDISP). Table 3-2 indicates the valid values for the DISP parameter. If the disposition specifies cataloging of a new Level 3 data file, CLOSELF closes the file and creates a catalog entry in the UARS Catalog. The production program provides the file attributes for the catalog entry (see Table 3-3). If the program asks for a new Level 3 file to be held for further use within the job, the file is closed and the hold status is entered into the UCSS accounting. For all other cases, the physical files are closed and the accounting is updated.

Table 3-2. File Disposition Usage

ASSIGN	ASGCAT/OPENL3/ASGCAL					
DISP	PRE-EXISTING CATALOGED FILE	POTENTIAL CATALOGED FILE	ASGCOR	ASGSCR	ASGUSR	OPENL0
CATALOG	N/A	1	N/A	N/A	N/A	N/A
FREE	2	3	2	3	N/A	2
HOLD	N/A	4	N/A	5	6	N/A

1 -- A REQUEST TO CATALOG THE FILE IS GENERATED AND THE CATALOG ENTRY WILL BE CREATED UPON SUCCESSFUL COMPLETION OF THE JOB

2 -- THE FILE IS NO LONGER NEEDED BY THE PROGRAM AND IS RELEASED

3 -- THE FILE IS NO LONGER NEEDED BY THE JOB AND IS DELETED FROM THE SYSTEM

4 -- THE FILE IS SAVED FOR USE BY A SUBSEQUENT PROGRAM IN THE SAME PRODUCTION JOB AND THE DECISION TO CATALOG IS DEFERRED

5 -- THE SCRATCH FILE IS SAVED FOR USE BY A SUBSEQUENT PROGRAM IN THE PRODUCTION JOB

6 -- THE USER STATUS FILE IS SAVED FOR USE BY A SUBSEQUENT PROGRAM IN THE SAME PRODUCTION JOB FOR POST-PRODUCTION ANALYSIS

Table 3-3. CLOSELF Catalog Attributes

ATTRIBUTE NAME	DESCRIPTION	REQUIRED OPTIONAL	FORMAT
DATA_GAP(N)*	start and stop times of data gap N	optional	two 23 character VMS times (DD-MMM-YYYY HH:MM:SS.CC) separated by a space
DATA_QUALITY_PI	user assigned quality value	optional	n.m
DATA_QUALITY_UARS	user assigned quality value	optional	n.m
COMMENTS	user comments	optional	up to 80 characters

* N = 1, 2, ... 100

The calling sequence for CLOSELF is as follows:

```
CALL CLOSELF (LID, DISP, NUM_ATTR, DATA_ATTR, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier associated with this data file. This LID must be the same logical file identifier specified in the corresponding open for this data file.
DISP	CHAR*4	I	File disposition 'FREE' = File no longer needed by program 'HOLD' = Hold file for use by subsequent program in job 'CAT ' = Catalog a new Level 3A file
NUM_ATTR	I*4	I	Number of user supplied catalog attributes. Required only when cataloging a file. A maximum of 100 attributes may be given.
DATA_ATTR	CHAR*80 (2,NUM_ATTR)	I	User supplied attributes for cataloging a created data file (see Table 3-3). The table of attributes is meaningful only when cataloging a new file. A

ARGUMENT	TYPE	I/O	DEFINITION
			dummy string is required when not cataloging the file.
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_CLSEERROLD - Error closing input file PFA_FILNOTFREE - Input file could not be freed PFA_NODATARECS - No data records in file PFA_UNKOPTSFDU - Unknown optimal SFDU descriptor id PFA_UNMTCHFDSP - Specified wrong file disposition for input file

3.2.25 DEASSIGN LOGICAL ID (DASLID)

DASLID terminates the logical connection between the production program and the data file assigned by ASGCAT, ASGCOR, ASGCAL, ASGUSR, or ASGSCR. The production program specifies the disposition of the file with the DISP parameter. For a file (Level 0, 1, 2, 3, or level-less) with a disposition of 'CAT', DASLID creates a catalog entry. The user provides the file attributes (see Table 3-4) for the catalog entry via the DATA_ATTR parameter. If an existing catalog file is freed, DASLID ignores the data attributes and updates the catalog entry only for accounting purposes. For all other types of files a disposition of 'FREE' results in deletion of the file at job end and no catalog access. The 'HOLD' option is used when the user wishes to keep track of a scratch file, user status file or an uncataloged file for use in a subsequent program in the job. Table 3-2 provides a description of the usage of the DISP parameter.

Table 3-4. DASLID Catalog Attributes

ATTRIBUTE	DESCRIPTION	REQUIRED OPTIONAL	FORMAT
START_TIME	file start time	required	23 character VMS time (DD-MMM-YYYY HH:MM:SS.CC)
STOP_TIME	file stop time	required	23 character VMS time
RECORD_SIZE	record size for files with fixed length records	optional	encoded integer
DATA_GAP(N)	start and stop times of data gap N	optional	two 23 character VMS times separated by a space
DATA_QUALITY_UARS	user assigned quality value	optional	n.m
DATA_QUALITY_PI	user assigned quality value	optional	n.m
COMMENTS	user comments	optional	up to 80 characters

* N = 1, 2, ... 100

The calling sequence for DASLID is as follows:

```
CALL DASLID (LID, DISP, NUM_ATTR, DATA_ATTR, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier
DISP	CHAR*4	I	File disposition 'FREE' - release file 'HOLD' - hold file for subsequent use 'CAT ' - catalog file
NUM_ATTR	I*4	I	Number of user supplied attributes in DATA_ATTR. Required only when cataloging a file. A maximum of 100 attributes may be given.

ARGUMENT	TYPE	I/O	DEFINITION
DATA_ATTR	CHAR*80 (2,N)	I	User supplied attributes for cataloging a created data file (see Table 3-4). The table of attributes is meaningful only when cataloging a new file. Otherwise, the attributes are ignored. A dummy string is required when not cataloging the file
STATUS	I*4	O	Status code SS$_NORMAL - Normal return PFA_FILNOTFREE - Input file could not be freed PFA_NODATARECS - File contains no data PFA_UNMTCHFDSP - Specified wrong file disposition for input file or user status file

3.3 UTILITY SERVICES

3.3.1 ERROR CODE REPORTING (ERRCDE)

ERRCDE allows the user to report error conditions encountered during a production program. An entry is made in the system error file each time the subroutine is called and the error is included on the program summary report.

The calling sequence for ERRCDE is as follows:

```
CALL ERRCDE (ERROR, COMMENTS)
```

ARGUMENT	TYPE	I/O	DEFINITION
ERROR	I*4	I	VMS message facility condition code
COMMENTS	CHAR*80	I	Comments about the error condition

3.3.2 UDTF TO VMS TIME CONVERSION (UTL_CON_UDTF_VMS)

UTL_CON_UDTF_VMS converts date/time in UDTF to the 23 character VMS time format, DD-MMM-YYYY HH:MM:SS.CC.

The calling sequence for UTL_CON_UDTF_VMS is as follows:

```
CALL UTL_CON_UDTF_VMS (UDTF_TIME, VMS_TIME, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
UDTF_TIME	I*4(2)	I	Date/time in UDTF
VMS_TIME	CHAR*23	O	Date/time in VMS character format
STATUS	I*4	O	Conversion status SS$_NORMAL - Normal return PFA_INVUDTFYR - Invalid year PFA_INVUDTFDAY - Invalid day of year PFA_INVUDTFMSEC - Invalid milliseconds of day

3.3.3 PRESSURE/ALTITUDE GRID UTILITY (VERT_DEF)

VERT_DEF provides a Fortran-callable support service to obtain UARS grid definitions. Instrument and level 3 subtypes are used to return the associated UARS grid which includes index values, units of grid, and valid pressure and altitude levels.

The calling program specifies the instrument and level 3 subtype. The VERT_DEF routine returns the base index, number of points, pressure and altitude levels, and units of grid.

When an unknown instrument or level 3 subtype in specified, the VERT_DEF routine returns a warning status in the status field.

Please note that the User's Guide also describes the grid utility function.

The calling sequence for VERT_DEF is as follows:

```
CALL VERT_DEF(INSTRUMENT_ID, SUBTYPE, BASE_INDEX, MAX_POINTS,
              PRESSURE, ALTITUDE, UNITS, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
INSTRUMENT_ID	C*12	I	Instrument Identifier
SUBTYPE	C*12	I	Type of data. The subtypes for each instrument are defined by the investigator.
BASE_INDEX	I*4	O	Start index (lowest) into the standard data array for which measurements can be taken for this data type

ARGUMENT	TYPE	I/O	DEFINITION
MAX_POINTS	I*4	O	The maximum number of pressure levels or altitudes for which measurements can be taken for this data type (N)
PRESSURE	R*4 (N)	O	An array of the pressure levels for this data type in millibars
ALTITUDE	R*4 (N)	O	An array of the geometric altitudes for this data type in kilometers
UNITS	C*12	O	The units in which the measurements for this data type are expressed
STATUS	I*4	O	Completion status SS$_NORMAL - Normal return PFA_INVINSTR - Unknown instrument PFA_INVDATAGRID - Unknown subtype

3.3.4 DECODE OBC EMAF INTO OBC REPORTS (OBCDECODE)

OBCDECODE extracts information contained in an OBC report from an OBC Level 0 record. The OBC reports and the OBC report variables are defined in PIR 1k21-UARS-403 Rev. B. The calling program supplies the OBC Level 0 record containing the desired OBC report, a report number identifying the type of report requested and the time of the report requested. If the requested time does not correspond to an actual report time, the time of the first report after the requested time is used. If more than one report exists for the requested time the first occurrence of the report with the best data quality is returned. The requested time must be greater than zero when calling OBCDECODE.

Only the two least significant digits of the report number are used to identify a report. Predefined OBC report elements are converted to VAX format and returned in the data arrays. A copy of the entire OBC report in telemetry format is returned as well. A FORTRAN include file OBC_REP_PARMS.INC is available to allow reports and report elements to be referenced using the G.E. mnemonics. OBC_REP_PARMS.INC contains parameter statements that equate the report name to report numbers and report item names to offsets in the returned VAX formatted data arrays. Appendix H lists the OBC report names and numbers and the OBC variables that are reformatted by OBCDECODE. An example using mnemonics to access report items appears in Appendix H.

The calling sequence for OBCDECODE is as follows:

```
CALL OBCDECODE(EMAF_REC, OBC_RPT_NUM, REQ_DATTIM, RET_DATTIM,
               OBC_QUALITY, OBC_REAL,OBC_INTEGER, OBC_BYTE,
               OBC_REC, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
EMAF_REC	BYTE(14400)	I	Level 0 OBC telemetry record containing one EMAF of data
OBC_RPT_NUM	I*4	I	The OBC report number of the report requested. The include file OBC_REP_PARMS.INC contains the parameter statements to associate mnemonics for the OBC reports with the OBC report numbers.
REQ_DATTIM	I*4(2)	I/O	On input, date and time in UDTF format of the generation time of the requested OBC report. On output the generation time of the next available OBC report of the requested type available. If no more reports are in the EMAF, REQ_DATTIM will be set to zero. On input the value of REQ_DATTIM must be greater than zero.
RET_DATTIM	I*4(2)	O	Date and time in UDTF format of the returned OBC report generation
OBC_QUALITY	BYTE(1)	O	Indicates parity or fill data for returned report 0 = good data 1 = parity error 3 = fill data 5 = no data returned
OBC_REAL	R*8(*)	O	Floating point values for report. Use mnemonics defined in the include file to reference returned values.
OBC_INTEGER	I*4(*)	O	Integer values for report. Use mnemonics defined in the include file to reference returned values.
OBC_BYTE	BYTE(*)	O	Integer byte and unpacked bit values for report. Use mnemonics defined in the include file to reference returned values.
OBC_REC	BYTE(28)	O	Returned copy of the specified report unformatted. The first byte is the

ARGUMENT	TYPE	I/O	DEFINITION
			report number followed by the report data. In terms of the G.E. documentation this buffer contains words 0 through 27.
STATUS	I*4	O	Status Code SS$_NORMAL - Normal return PFA_BADEPOCHYR - Bad ASC09 Epoch year (UFL reports only) PFA_BADOBCEMAF - Bad EMAF record header PFA_INVUDTFDAY - Bad UDTF day requested PFA_INVUDTFMSEC - Bad UDTF msec requested PFA_INVUDTFYR - Bad UDTF year requested PFA_OBCDATATIM - No data for time specified PFA_UNKOBCRPT - Unknown report

* Indicates that the minimum size needed varies by OBC report. The Maximum dimension for OBC_REAL is 12, for OBC_INTEGER is 11, and for OBC_BYTE is 52.

3.3.5 COMPARE TIMES (UTL_COMPARE_TIME)

UTL_COMPARE_TIME is a function that compares two times expressed in 8-byte format and returns a 2-byte integer result. The value of the result is 1 if the first time is later than the second, zero if the times match, and -1 if the first time is earlier than the second.

The calling sequence for UTL_COMPARE_TIME is as follows:

```
Result = UTL_COMPARE_TIME (FIRST_TIME, SECOND_TIME)
```

ARGUMENT	TYPE	I/O	DEFINITION
FIRST_TIME	I*4(2)	I	First time to be compared
SECOND_TIME	I*4(2)	I	Second time to be compared
UTL_COMPARE_TIME	I*2	O	Result: 1 iff FIRST_TIME > SECOND_TIME 0 iff FIRST_TIME = SECOND_TIME -1 iff FIRST_TIME < SECOND_TIME

3.3.6 COMPUTE SECONDS BETWEEN UDTF TIMES (UTL_SEC_TIME_DIF)

UTL_SEC_TIME_DIF is a function that returns a real*8 result containlning the number of seconds between two UDTF times. The difference is positive when the first time exceeds the second.

The calling sequence for UTL_SEC_TIME_DIF is as follows:

```
Result = UTL_SEC_TIME_DIF (FIRST_UTDF_TIME, SECOND_UTDF_TIME)
```

ARGUMENT	TYPE	I/O	DEFINITION
FIRST_UTDF_TIME	I*4(2)	I	First UTDF time
SECOND_UTDF_TIME	I*4(2)	I	Second UTDF time
UTL_SEC_TIME_DIF	R*8	O	Result: Difference in seconds (first_time - second_time)
FIRST_TIME	I*4(2)	I	First time to be compared

3.3.7 CONVERT UARS DAY TO UDTF FORMAT (UTL_UARS_TO_UDTF)

UTL_UARS_TO_UDTF converts a UARS day into a two-word time array in UDTF format.

The calling sequence for UTL_UARS_TO_UDTF is as follows:

```
CALL UTL_UARS_TO_UDTF (UARS_DAY, UDTF_TIME)
```

ARGUMENT	TYPE	I/O	DEFINITION
UARS_DAY	I*4	I	UARS processing day
UDTF_TIME	I*4(2)	O	Time in UDTF format

3.3.8 CONVERT UDTF FORMAT TO UARS DAY (UTL_UDTF_TO_UARS)

UTL_UDTF_TO_UARS converts a date in UTDF format to a date in UARS format.

The calling sequence for UTL_UDTF_TO_UARS is as follows:

```
CALL UTL_UDTF_TO_UARS (UDTF_TIME, UARS_DAY)
```

ARGUMENT	TYPE	I/O	DEFINITION
UDTF_TIME	I*4(2)	O	Time in UDTF format
UARS_DAY	I*4	I	UARS processing day

CHAPTER 4

RAC SIMULATED SERVICES

The UCSS provides a collection of services developed to simulate the production software support services described in Section 3. These services are designed to facilitate the testing of production processing software outside of the production environment. Programs using the simulated services can run at the RACs or in user directories on the CDHF.

In addition to providing a test capability, the simulated services can be used by analysis programs that are run on the RACs or on the CDHF. For example, quick-look files can be transferred to a RAC and the program to analyze the data can use the OPENL0, READL0, QUALRD, and CLOSELF routines to access the quick-look files.

The calling sequences for the simulated services are the same as for the production services. Table 4-1 lists the simulated services, identifies the section defining the calling sequence, and indicates the differences in the services.

Table 4-1. Simulated Services

SUBROUTINE NAME	INTERFACE DEFINITION	DIFFERENCES BETWEEN THE SIMULATED SERVICES AND THE PRODUCTION SERVICES
PGINIT	3.1.1	1. Uses user-supplied PROGRAM_PARAMS namelist to supply program parameters 2. Uses user-supplied FILE_PARAMS namelist to provide file information 3. Uses user-supplied DEFAULT_PARAMS namelist to supply default values for file parameters 4. Creates file parameter table to simulate the catalog access
PGTERM	3.1.2	1. Program summary report is sent to SYS$OUTPUT
OPENL0 OPENL3AT OPENL3AL OPENL3S OPENL3TP OPENL3LP	3.2.1 3.2.7 3.2.8 3.2.9 3.2.10 3.2.11	1. Uses file parameter table to identify file(s) 2. Provides access to either a single pseudo-virtual file created via RAC data transfer or a pool of day files 3. A pool of virtual input day files is specified in the FILE_PARAMS Namelist via DATA_FILE_NAME and VIRTUAL_UARS_DAY 4. REQUIRED_FLAG in FILE_PARAMS Namelist is used to indicate if all files in the user's processing range are required to be present
ASGCAT	3.2.2	1. Uses file parameter table to identify file 2. Output file location provided in file parameter table
ASGCOR	3.2.3	1. Uses file parameter table to identify file
ASGCAL	3.2.4	1. Uses file parameter table to identify file
ASGSCR	3.2.5	1. Uses file parameter table to identify file 2. Output file location provided in file parameter table
ASGUSR	3.2.6	1. Uses file parameter table to identify file

Table 4-1. Simulated Services (Continued)

SUBROUTINE INTERFACE		DIFFERENCES BETWEEN SIMULATED AND PRODUCTION
QUALRD	3.2.12	1. For virtual input, files in the user's processing range are selected from the pool of files specified via FILE_PARAMS
READL0	3.2.13	
READL3AT	3.2.14	
READL3S	3.2.15	
READL3AL	3.2.16	
READL3TP	3.2.17	
READL3LP	3.2.18	
WRITEL3AT	3.2.19	1. Requires Ephemeris file to be specified as cataloged input via FILE_PARAMS namelist
WRITEL3AL	3.2.21	
WRITEL3S	3.2.20	No differences
WRITEL3TP	3.2.22	
WRITEL3LP	3.2.23	
CLOSELF	3.2.24	1. No access to the catalog 2. Catalog attributes are output to SYS$OUTPUT
DASLID	3.2.25	1. No access to the catalog 2. Catalog attributes are output to SYS$OUTPUT
ERRCDE	3.3.1	1. Error is output to SYS$ERROR 2. Error is not logged to log file
UTL_CON _UDTF_VMS	3.3.2	Same routine (UTL_CON_UDTF_VMS)

4.1 PROGRAM CONTROL SERVICES

4.1.1 JOB INITIALIZATION (RSS_JOB_INIT)

The first program executed in a job run in the simulated environment is the UCSS job initialization program, RSS_JOB_INIT. It generates the first portion of the job summary report, the initialization statistics. The job initialization program is optional in the runstream, but is provided to be consistent with the production services.

4.1.2 PROGRAM INITIALIZATION (PGINIT)

The PGINIT subroutine provides the mechanism for passing input parameters to a user program run in the simulated environment. It returns the processing time range, the UARS day number, and any user-defined parameters specific to the program. In the production environment these parameters are supplied to PGINIT by the scheduler, but in the simulated environment they are provided by the user in the job's runstream. The user must provide the required parameters via the PROGRAM_PARAMS namelist. The PROGRAM_PARAMS namelist is described in Table 4-2.

Table 4-2. PROGRAM_PARAMS Namelist

NAMELIST PARAMETER	DESCRIPTION	FORMAT
PROG_NAME	program name	C20
PROCESSING_START_TIME	processing start time - 'DD-MMM-YYYY HH:MM:SS.CC'	C23
PROCESSING_STOP_TIME	processing stop time 'DD-MMM-YYYY HH:MM:SS.CC'	C23
UARS_PROCESSING_DAY	primary UARS processing day	I
LAUNCH_DATE	UARS launch date used as the epoch date for UARS day number - 'DD-MMM-YYYY HH:MM:SS.CC'	C23
DEF_EXISTS	flag for specifying presence of DEFAULT_PARAMS namelist (T or F)	C1
PARAMS(n) *	user parameter name n	C20
VALUES(n) *	user parameter value n	C20

* n = 1 to 50

In the RAC simulated services environment, the user supplies information about files to be accessed by the program via the FILE_PARAMS namelists. The FILE_PARAMS namelist identifies the primary catalog attributes and the fully qualified file(s) specification. In the case of a virtual input file, one FILE_PARAMS namelist is used to identify a pool of physical files in which all share the same file attributes. The physical file names and their associated UARS day are specified via the DATA_FILE_NAME and VIRTUAL_UARS_DAY parameters in the FILE_PARAMS namelist. During the open, the pool matching the user's file attributes is selected, files which exist and which contain data and which are in the user's specified processing range are then selected.

PGINIT creates a file parameter table which is used to simulate the UARS Catalog using FILE_PARAMS namelist data from the runstream. This namelist is described in Table 4-3. The namelist parameters required for a file are determined by the type of file and its usage. Table 4-4 identifies the required parameters by file service.

Table 4-3. FILE_PARAMS Namelist

NAMELIST PARAMETER	DESCRIPTION	FORMAT	VALUES
CALIBRATION_ID	calibration table table identifier	C12	
CALIBRATION_MATCH	calibration day match criteria	C4	'EXCT' or 'PREV' or 'NEXT' or 'NEAR'
DATA_FILE_NAME	list of one or more VMS file specifications	C80(*)	See Notes 1,2
VIRTUAL_UARS_DAY	list of UARS days in virtual input pool	I(*)	See Note 2
DATA_LEVEL	data level	C3	1st char '0', '1','2','3', or blank
DATA_TYPE	data type	C12	
ESTIMATED_FILE_SIZE	estimated file size in blocks	I	
FILE_VERSION_NUMBER(1)	CCB file version number	I	
FILE_VERSION_NUMBER(2)	file cycle number	I	
LOGICAL_FILE_ID	logical file identifier	C16	
OLD_NEW	file status flag	C4	'OLD' or 'NEW' or 'HELD'
PRE_NXT_UARS_DAY	actual UARS day	I	
SOURCE	correlative data source	C12	
SUBTYPE	data subtype	C12	
UARS_DAY	UARS day number	I	
USER_STATUS_FILE_NUMBER	user status file file number	I	

Table 4-3. FILE_PARAMS Namelist (Continued)

NAMELIST PARAMETER	DESCRIPTION	FORMAT	VALUES
REQUIRED_FLAG	optional parameter which, for input, indicates if all files in UARS processing range are required to be present	C1	'T' or 'F'

Notes:

1. Only one file name may be specified via DATA_FILE_NAME for all file types except for virtual input files. For virtual input files up to 250 files may be specified.

2. The VIRTUAL_UARS_DAY parameter is required for virtual input files containing more than one physical file. It supplies, in one-to-one correspondence, the nominal UARS days associated which each physical file specified by the DATA_FILE_NAME parameter.

Table 4-4. Required FILE_PARAMS Parameters and Defaults

	PARAMETER														
FILE SERVICE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OPENL0			X	OD	D										X2
ASGCAT															
OLD			X	OD	D				D			X	OD		
NEW			X	ND	D				D			X	ND		
HELD			X	X	D	X		X	X			X	X		
OPENL 3AT, 3AL,3TP,3LP															
OLD			X	OD	D				D			X			X2
NEW			X	ND	D		X		D			X	ND		
HELD			X	X	D	X		X	X			X	X		
OPENL3S															
OLD			X	OD	D				D						X2
NEW			X	ND	D		X		D				ND		
HELD			X	X	D	X		X	X				X		
*ASGCAL															
OLD	X	D1	X	OD						X1		D	OD		
NEW	X		X	ND					X			D	ND		
HELD	X		X	X		X		X	X			D	X		
ASGCOR			X								X	X	OD		
ASGSCR															
NEW			X					X	SD						
HELD			X			X		X	X						
ASGUSR			X											X	

Legend:

1 CALIBRATION_ID	6 ESTIMATED_FILE_SIZE	11 SOURCE
2 CALIBRATION_MATCH	7 FILE_VERSION_NUMBER	12 SUBTYPE
3 DATA_FILE_NAME	8 LOGICAL_FILE_ID	13 UARS_DAY
4 DATA_LEVEL	9 OLD_NEW	14 USER_STATUS_FILE_NUMBER
5 DATA_TYPE	10 PRE_NXT_UARS_DAY	15 VIRTUAL_UARS_DAY

Table 4-4. Required FILE_PARAMS Parameters and Defaults (Continued)

* For dayless calibration, UARS_DAY is set to zero and CALIBRATION_MATCH and PRE_NXT_UARS_DAY are omitted

X Parameter always required, no default exists
X1 Parameter required only for calibration by day with CALIBRATION_MATCH not equal to 'EXCT'
X2 Parameter required only when more than one physical file has been specified by DATA_FILE_NAME
D Parameter required, if not given, use DEF_<file-parameter-name>
D1 Parameter required only for calibration by day, if not given, use DEF_CALIBRATION_MATCH
ND Parameter required, if not given, use DEF_NEW_<file-parameter-name>
OD Parameter required, if not given, use DEF_OLD_<file-parameter-name>
SD Parameter required, if not given, use default value 'NEW'

The user has the ability to supply default file parameters via the DEFAULT_PARAMS namelist. Default file parameters are used when required file parameters are not supplied in the FILE_PARAMS namelist and a default parameter is applicable and has been specified. The DEFAULT_PARAMS namelist is described in Table 4-5. Table 4-4 shows which default parameters are applicable based on file access type. The DEFAULT_PARAMS namelist is entered in the job runstream after the PROGRAM_PARAMS namelist and before the first FILE_PARAMS namelist. If default parameters are supplied, the DEF_EXISTS parameter in the PROGRAM_PARAMS namelist must be set to 'T'. If DEF_EXISTS is not specified in the PROGRAM_PARAMS namelist then defaults will not be applied and values specified via the DEFAULT_PARAMS namelist will be ignored.

Table 4-5. DEFAULT_PARAMS Namelist

NAMELIST PARAMETER	DESCRIPTION	FORMAT	VALUES
DEF_OLD_NEW	default for old or new files requiring OLD_NEW (see Note 1.)	C4	'OLD', or 'NEW'
DEF_OLD_DATA_LEVEL	default for old files requiring DATA_LEVEL (see Note 1.)	C3	1st char '0','1', '2','3', or blanks
DEF_NEW_DATA_LEVEL	default for new files requiring DATA_LEVEL (see Note 1.)	C3	1st char '1','2', '3', or blanks
DEF_OLD_UARS_DAY	default for old files requiring UARS_DAY (see Note 1.)	I	
DEF_NEW_UARS_DAY	default for new files requiring UARS_DAY (see Note 1.)	I	
DEF_DATA_TYPE	default for all files requiring DATA_TYPE	C12	
DEF_CALIBRATION_MATCH	Default for Calibration files with nonzero UARS_DAY specified (see Note 2.)	C4	'EXCT', 'PREV', 'NEXT', or 'NEAR'
DEF_SUBTYPE	Default for Calibration files requiring subtype	C12	

Notes:

1. There is no default for DATA_LEVEL, UARS_DAY, and OLD_NEW for held files.

2. If CALIBRATION_MATCH is omitted for a calibration file then DEF_CALIBRATION_MATCH will be used if and only if UARS_DAY is non-zero.

4.1.3 PROGRAM TERMINATION (PGTERM)

PGTERM terminates a program run in the simulated environment. The user's program is responsible for determining the success or failure of the processing and reports the result via PGTERM. PGTERM updates the accounting information and produces a program summary report which is sent to the SYS$OUTPUT device. PGTERM must be called at the end of each program.

4.1.4 JOB TERMINATION (RSS_JOB_TERM)

The last program executed in a job run in the simulated environment is the job termination program, RSS_JOB_TERM. It generates the second portion of the job summary report, the job completion statistics. The job termination program is optional in the runstream, but is provided to be consistent with the production services.

4.2 FILE ACCESS

This section describes the software support services designed to provide access to user-managed files in the simulated environment. Services are provided to access all levels of instrument files, calibration files, correlative files, user status files, and scratch files.

The UCSS provides the OPENL0, READL0, QUALRD, and CLOSELF services to access Level 0 data in the simulated environment. The access to the UARS Catalog required to identify the requested file is simulated using the file parameter table created by PGINIT. The simulated environment allows access to both day files and pseudo-virtual files generated via RAC data transfer. However, when a pseudo_virtual file is specified, no other physical file may be listed as part of the DATA_FILE_NAME parameter.

The UCSS provides the OPENL3AT, OPENL3AL, OPENL3S, OPENL3TP, OPENL3LP, READL3AT, READL3S, READL3AL, READL3TP, READL3LP, WRITEL3AT, WRITEL3S, WRITEL3AL, WRITEL3TP, WRITEL3LP, and CLOSELF services to access Level 3 data and Level 3 data parameters at processing levels 3AT, 3AL, 3AS, and 3BS in the simulated environment. The access to the UARS Catalog required to identify the requested input file is simulated using the file parameter table created by PGINIT. The file parameter table is also used to identify output file locations. The simulated environment allows access to both day files and pseudo-virtual files in the same manner as described above for Level 0 data.

There are no differences in the functions of the Level 3 write services from the production versions. Write services for Level 3AT and Level 3AL files require the use of an ephemeris file as input.

This file provides information that is used to calculate the values of local solar time (LST) and solar zenith angle (SZA) that are stored in each data record. The CLOSELF service simulates the cataloging function by writing the catalog attributes to SYS$OUTPUT when the user program requests cataloging of a Level 3 file.

The UCSS provides the ASGCAT, ASGCOR, ASGCAL, ASGUSR, and ASGSCR services to assign Level 0, Level 1, Level 2, Level 3, correlative, calibration, user status, and scratch files in the simulated environment. The access to the UARS Catalog required to identify the requested cataloged files is simulated using the file parameter table created by PGINIT. The file parameter table is also used to identify the locations of output files. The UCSS also provides the DASLID service to record the user supplied file disposition and to simulate the cataloging function.

The user is responsible for providing the I/O services to access auxiliary files. If the user program generates auxiliary files, the user must define the AUX_DIRECTORY logical name in the runstream to identify the disk and directory where the files are to be created.

4.3 UTILITY SERVICES

The UCSS provides the utility services (see Section 3.3) in the simulated environment. Table 4-1 shows the functional differences in these services between the simulated and production environments.

4.4 JOB RUNSTREAM FOR THE SIMULATED ENVIRONMENT

Figures 4-1 and 4-2 present sample runstreams for jobs that use the simulated services. The Level 1 processing job in the first example consists of two program steps. The job uses Level 0, calibration, and correlative data as input, generates an intermediate scratch file to pass information between programs, and produces a Level 1 file. The second example illustrates a job that produces a Level 3AT file using a Level 2 data file and an ephemeris file as input. The ephemeris file is needed for the solar zenith angle (SZA) and local solar time (LST) stored with each record in the Level 3AT file. The following notes pertain to the annotated runstreams in Figure 4-1 and Figure 4-2:

1. The AUX_DIRECTORY logical name, defined for the job, identifies the disk and directory to be used for auxiliary files. AUX_DIRECTORY must be defined for any job that creates auxiliary files.

2. The UCSS_JOB_ID logical name is a 21 character identifier for the job. It is not required for simulated runstreams, but the job identifier is included on the job summary reports if

it is provided.

3. The RSS_JOB_INIT program is the UCSS job initialization program for the simulated environment. It is the first program run in the job. RSS_EXE is the logical name identifying the disk and directory location of the UCSS executable code. The RSS_JOB_INIT can be omitted from the simulated runstream.

4. The UARS_PASS_FLAG is used to indicate the success or failure of each job step. The UARS_PASS_FLAG must be tested after each job step to prevent further processing in the event of job failure. The UARS_PASS_FLAG is controlled by the UCSS software.

5. The start of each job step can be labeled to accommodate user-supplied conditional tests.

6. The JOB_STEP logical name identifies the job step number. Each job step is numbered sequentially. The job step number appears on the program summary report.

7. This run command causes execution of the user-supplied program. MLSEXE is the logical name identifying the disk and directory location of the MLS executable code.

8. The PROGRAM_PARAMS namelist provides the input parameters (see Table 4-2) to the program.

9. The PROGRAM_PARAMS namelist parameters PROCESSING_START_TIME, PROCESSING_STOP_TIME, and LAUNCH_DATE may be specified in either VAX/VMS 23 character date and time format or UARS standard Date and Time (UDTF) format. When UDTF time/date format is used the two integers must be separated by one or more blanks. UDTF format is described in Appendix A.

10. The DEFAULT_PARAMS namelist provides default values for required file parameters. Default values are used if required parameters are not specified in a FILE_PARAMS namelist. Table 4-4 shows the required file parameters and applicable defaults by file access type. Table 4-5 describes each DEFAULT_PARAMS parameter.

11. A FILE_PARAMS namelist must be provided for each file accessed by the program. Table 4-3 identifies the namelist parameters and Table 4-4 identifies which parameters are required for each type of file access.

12. In the case of a virtual input Level 0 or Level 3 file, the FILE_PARAMS namelist is used to set up a pool of physical files each sharing the same general file attributes. The DATA_FILE_NAME parameter specifies each physical file and the VIRTUAL_UARS_DAY parameter lists the nominal UARS day

associated with each physical file specified. During the open, the pool with attributes matching those specified by the user is identified and files within the user's processing range are selected.

13. The VIRTUAL_UARS_DAY parameter is only required for a virtual input file containing two or more physical files.

14. The job termination or exit step must be labeled. This label is required even when RSS_JOB_TERM is not used.

15. The RSS_JOB_TERM program is the UCSS job termination program for the simulated environment. It is the last program run in the job. RSS_EXE is the logical name identifying the disk and directory location of the UCSS executable code. RSS_JOB_TERM can be omitted from the simulated runstream.

16. An SFDU file containing appropriate information for generating SFDU headers should be provided whenever new Level 3 data files are to be generated. (See Appendix G for a description of the format and content of this file.)

17. An appropriate ephemeris file must be specified whenever a new Level 3AL, Level 3AT, Level 3AS, or Level 3S file is to be generated.

Figure 4-1. First Sample Simulated Environment Job Runstream

Figure 4-1. First Sample Simulated Environment Job Runstream

```
$ !
$ ON ERROR THEN GOTO JOBTERM
$ set default DISK4:[MLSSCRATCH]
$ define/process AUX_DIRECTORY DISK3:[AUXFILES]                    1
$ define/process UCSS_JOB_ID MLS100100100010002000                2
$ !
$ !  First program is UCSS Job Initialization
$ !
$ run RSS_EXE:RSS_JOB_INIT                                         3
$ if (UARS_PASS_FLAG .EQS. "FAIL") then goto JOBTERM               4
$ !
$ !               STEP 1
$ !
$ JOB_STEP_1:                                                      5
$     define/process JOB_STEP 1                                    6
$     run MLSEXE:MLSL1CAL                                          7
 $PROGRAM_PARAMS                                                   8
   PROG_NAME='MLSL1CAL'
   PROCESSING_START_TIME='92089  0'
   PROCESSING_STOP_TIME='92089  86399990'                          9
   UARS_PROCESSING_DAY=119
   LAUNCH_DATE='91335  0'
   DEF_EXISTS='T'
   PARAMS(1)='CALIBRATION_FLAG'
   VALUES(1)='1'
 $END
 $DEFAULT_PARAMS                                                   10
   DEF_DATA_TYPE= 'MLS'
   DEF_OLD_UARS_DAY= 119
 $END
 $FILE_PARAMS                                                      11
   DATA_FILE_NAME='DISK3:[MLSLEVEL0]MLSL0D119.DAT',                12
                  'DISK3:[MLSLEVEL0]MLSL0D120.DAT',
                  'DISK3:[MLSLEVEL0]MLSL0D121.DAT',
                  'DISK3:[MLSLEVEL0]MLSL0D122.DAT',
   VIRTUAL_UARS_DAY=119,120,121,122                                13
   DATA_LEVEL='0'
 $END
 $FILE_PARAMS
   CALIBRATION_ID='CAL_PARAMS'
   CALIBRATION_MATCH='EXCT'
   DATA_FILE_NAME='DISK3:[MLSCAL]MLSL1CAL_PARAMSL1.DAT'
   DATA_LEVEL='1'
   SUBTYPE='MLS'
 $END
```

Figure 4-1. First Sample Simulated Environment Job Runstream

```
 $FILE_PARAMS
   CALIBRATION_ID='CAL_PARAMS'
   DATA_FILE_NAME='DISK3:[MLSCAL]MLSL1CAL_PARAMSL1.NEW'
   DATA_LEVEL='1'
   OLD_NEW='NEW'
   SUBTYPE='MLS'
   UARS_DAY=120
 $END
 $FILE_PARAMS
   DATA_FILE_NAME='MLSL1SCRATCH.DAT'
   LOGICAL_FILE_ID='SCRATCH_LID'
 $END
$ if (UARS_PASS_FLAG .EQS. "FAIL") then goto JOBTERM
$ !
$ !             STEP 2
$ !
$ JOB_STEP_2:
$     define/process JOB_STEP 2
$     run MLSEXE:MLSL1OUT
 $PROGRAM_PARAMS
   PROG_NAME='MLSL1OUT'
   PROCESSING_START_TIME='29-MAR-1992 00:00:00.00'
   PROCESSING_STOP_TIME='29-MAR-1992 23:59:59.99'
   UARS_PROCESSING_DAY=119
   LAUNCH_DATE='01-DEC-1991 00:00:00.00'
   DEF_EXISTS='T'
 $END
 $DEFAULT_PARAMS
   DEF_NEW_DATA_LEVEL='1'
   DEF_NEW_UARS_DAY= 119
   DEF_OLD_UARS_DAY= 119
   DEF_DATA_TYPE= 'MLS'
 $END
 $FILE_PARAMS
   DATA_FILE_NAME='MLSL1SCRATCH.DAT'
   LOGICAL_FILE_ID='SCRATCH_LID'
   OLD_NEW='HELD'
   ESTIMATED_FILE_SIZE=500
 $END
 $FILE_PARAMS
   DATA_FILE_NAME='DISK3:[MLSLEVEL1]MLSL1D119.DAT'
   OLD_NEW='NEW'
   SUBTYPE='NONE'
 $END
```

Figure 4-1. First Sample Simulated Environment Job Runstream

```
 $FILE_PARAMS
   DATA_FILE_NAME='DISK1:[CORREL]NMCD119.DAT'
   SOURCE='NMC'
   SUBTYPE= 'NMC_DATA'
 $END
$ !
$ !      Last Program is UCSS Job Termination
$ !
$JOBTERM:                                                    14
$ run RSS_EXE:RSS_JOB_TERM                                   15
$ exit
```

Figure 4-2. Second Sample Simulated Environment Job Runstream

Figure 4-2. Second Sample Simulated Environment Job Runstream

```
$ !
$ !
$ !  THIS JOB PRODUCES A LEVEL 3AT FILE USING A LEVEL-2
$ !  DATA FILE AND AN EPHEMERIS FILE AS INPUT.
$ !
$ ON ERROR THEN GOTO JOBTERM
$ set default DISK4:[MLSSCRATCH]
$ define/process UCSS_JOB_ID HRDI_L3AT_ITTEST00001
$ define/process UARS_SFDU_FILE sfdu_dir:UARS_SFDU_FILE.DATA          16
$ !
$ !   First program is UCSS Job Initiation
$ !
$ run  RSS_EXE:RSS_JOB_INIT
$ if (UARS_PASS_FLAG .EQS. "FAIL") then goto JOBTERM
$ !
$ !               STEP 1
$ JOB_STEP_1:
$     define/process JOB_STEP 1
$     run   HRDIEXE:L2_TO_L3AT.exe
 $PROGRAM_PARAMS
   PROG_NAME='l0_to_l3a_held2',
   PROCESSING_START_TIME='02-FEb-1992 00:00:00.00',
   PROCESSING_STOP_TIME ='02-fEB-1992 23:59:59.99',
   UARS_PROCESSING_DAY=125,
   Launch_Date='01-OcT-1991 00:00:00.00'
   DEF_EXISTS = 'T'
   PARAMS(1)='L3A_EST_FSIZE',
   VALUES(1)='400'
   PARAMS(2)='START_INDEX',
   VALUES(2)='1'
   PARAMS(3)='NUM_POINTS',
   VALUES(3) = '20'
   PARAMS(4)='MAX_REC_COUNT'
   VALUES(4)='1000'
 $END
 $DEFAULT_PARAMS
   DEF_DATA_TYPE = 'HRDI'
   DEF_OLD_DATA_LEVEL = '2'
   DEF_NEW_DATA_LEVEL = '3AT'
   DEF_OLD_UARS_DAY = 125
   DEF_NEW_UARS_DAY = 125
 $END
```

Figure 4-2. Second Sample Simulated Environment Job Runstream

```
  !
  ! EPHEMERIS file
  !
 $FILE_PARAMS
   DATA_FILE_NAME='IPD$DISK:[UOAS]SLPEPHEM_D0001.V0001_C01_PROD',    17
   DATA_LEVEL=' ',
   DATA_TYPE='SLPephem',
   SUBTYPE=' ',
   OLD_NEW='OLD',
   UARS_DAY=1
 $END
  !
  ! HRDI level 2 input file
  !
 $FILE_PARAMS
   DATA_FILE_NAME='hrdi_data:hrd_l2_day_0125.dat'
   SUBTYPE = 'TEMPERATURE'
   OLD_NEW = 'OLD'
 $END
  !
  !  hrdi Level 3at output data
  !
 $FILE_PARAMS
   DATA_FILE_NAME='hrdi_data:hrd_l3at_day_125.dat',
   FILE_VERSION_NUMBER(1)=2,
   FILE_VERSION_NUMBER(2)=2,
   OLD_NEW='NEw',
   SUBTYPE='teMPErature',
 $END
$ !
$ !
$ !      Last Program is UCSS Job Termination
$ !
$JOBTERM:
$
$ run  RSS_EXE:RSS_JOB_TERM
$ EXIT
```

CHAPTER 5

UCSS ANALYSIS SERVICES

5.1 ANALYSIS SERVICES

The UCSS provides a collection of services that allows a user program read access to cataloged files in the UCSS-managed system space and that can stage the cataloged data from the MSS, if the data is offline.

The calling sequences for these services is compatible with the corresponding services available in the production environment. Table 5-1 lists the analysis service, identifies the section defining the calling sequences, and indicates the difference between the analysis service and its production counterpart. The major difference for all the services is that all errors are returned to the caller in the analysis environment.

Table 5-1. Analysis Services

SUBROUTINE NAME	INTERFACE DEFINITION	DIFFERENCES BETWEEN THE ANALYSIS SERVICES AND THE PRODUCTION SERVICES
PGINIT	5.2.1	1. Initializes global tables 2. No production accounting
PGTERM	5.2.2	1. Cleans up files 2. No production accounting or summary report
OPENL0	3.2.1	1. No production accounting
ASGCAT	3.2.2	1. Existing cataloged files only 2. No production accounting 3. No output files
ASGCOR	3.2.3	1. No production accounting
ASGCAL	3.2.4	1. No production accounting
ASGQL	5.3.2.1	1. Not available as a production service
GENASG	5.3.2.2	1. Not available as a production service
OPENL3AT	3.2.7	1. Access (Read) to existing cataloged files only 2. No production accounting 3. No output files
OPENL3AL	3.2.8	1. Access (Read) to existing cataloged files only 2. No production accounting 3. No output files
OPENL3S	3.2.9	1. Access (Read) to existing cataloged files only 2. No production accounting 3. No output files
OPENL3TP	3.2.10	1. Access (Read) to existing cataloged files only 2. No production accounting 3. No output files
OPENL3LP	3.2.11	1. Access (Read) to existing cataloged files only 2. No production accounting 3. No output files
OPENQL	5.3.3	1. Not available as production service
QUALRD	3.2.12	1. No production accounting

Table 5-1. Analysis Services (Continued)

SUBROUTINE INTERFACE		DIFFERENCES BETWEEN ANALYSIS AND PRODUCTION
QUALQL	5.3.5	1. Not available as production service
READL0	3.2.13	1. No production accounting
READL3AT	3.2.14	1. No production accounting
READL3S	3.2.15	1. No production accounting
READL3AL	3.2.16	1. No production accounting
READL3TP	3.2.17	1. No production accounting
READL3LP	3.2.18	1. No production accounting
READQL	5.3.4	1. Not available as production service
CLOSELF	3.2.22	1. No production accounting 2. No output files
DASLID	3.2.23	1. No production accounting 2. No output files
SETVERCY	5.4.1	1. Not available as production service
GETVERCY	5.4.2	1. Not available as production service

5.2 PROGRAM CONTROL SERVICES

5.2.1 PROGRAM INITIALIZATION (PGINIT)

The PGINIT service establishes an exit handler and initializes the global tables and variables used by the analysis services. In particular, it initializes the file version parameters in the file version table to default values, generates and stores the current process job identification in the program status table and obtains a virtual memory zone from the system for any dynamic memory the services may need. PGINIT should be called at the beginning of each analysis program.

The calling sequence for PGINIT is as follows:

```
CALL PGINIT (STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return Other codes - Error (See Table F-1)

5.2.2 PROGRAM TERMINATION (PGTERM)

The PGTERM service initiates program termination by invoking its exit handler, which in turn closes and releases any cataloged files left open and releases the virtual memory zone assigned the current process. PGTERM should be called at the end of each analysis program.

The calling sequence for PGTERM is as follows:

```
CALL PGTERM (PASS_FAIL, COND_CODE, PROG_COMMENT)
```

ARGUMENT	TYPE	I/O	DEFINITION
PASS_FAIL	CHAR*4	I	Program completion status 'PASS' Successful completion 'FAIL' Unsuccessful completion
COND_CODE	I*4	I	A VMS condition code specifying additional status information about program completion
PROG_COMMENT	CHAR*80	I	Dummy string provided to make interface consistent with production and RAC simulated services

5.3 FILE ACCESS

This section describes the software support services designed to provide access to any cataloged file by user programs in the analysis environment. Most of these services have their counterpart in the production and RAC simulated environments and will be described only briefly here insofar as they differ from the corresponding services in the production environment. The others, namely those that access cataloged quick-look data files, will be described in more detail.

The UCSS provides the OPENL0, READL0, QUALRD, and CLOSELF services to access cataloged Level 0 data from programs in the analysis environment. The mode of access is the same as in the production environment but, unlike the production versions, the analysis versions provide no production accounting and no summary report. The appropriate sequences in which these routines are to be called is shown in Table 3-1.

By default, the Analysis Services access production files. To access test files, define the logical name UCSS_TEST_DATA_FLAG to TRUE, causing the UCSS software to disregard the test/prod file attribute.

5.3.1 LEVEL 3 FILE SERVICES

The UCSS provides the OPENL3AT, OPENL3AL, OPENL3S, OPENL3TP, OPENL3LP, READL3AT, READL3AL, READL3S, READL3TP, READL3LP, and CLOSELF services to access cataloged Level 3 data as well as Level 3 parameters from programs in the analysis environment. The mode of access is the same as in the production environment but, unlike the production versions, the analysis versions provide no production accounting and no summary report and do not support the generation of output files. The appropriate sequence in which these routines are to be called is shown in Table 3-1.

5.3.2 ASSIGN / DEASSIGN SERVICES

In the analysis enviroment, the UCSS provides most file assign/deassign services available in the production enviroment. They include:

ASGCAL - assign instrument-oriented cataloged file
ASGCOR - assign UARS day oriented correlative file
ASGCAL - assign user-generated, instrument-oriented calibration file
DASLID - terminate logical connection between analysis program and assigned data file

The mode of access is the same as in the production enviroment.

However, unlike the production versions, the analysis version does not support production accounting, summary report generation, nor generation of output files. For detail description, refer to Sections 3.2.2 - 3.2.4 and Section 3.2.25.

The UCSS also provides two additional services which are available only to the analysis enviroment. ASGQL assigns logical unit number to a QUICKLOOK data file, and GENASG, a generic file assignment service, assigns any types of cataloged data file based on user provided file attributes.

5.3.2.1 ASSIGN QUICKLOOK DATA FILE (ASGQL)

ASGQL assigns a logical file identifier (LID) to a cataloged QUICKLOOK data file (e.g. instrument, engineering, OBC, spacecraft, quality and attitude) for read-only access. It returns a unique logical unit number (LUN) that can be used to perform Fortran I/O on the file.

ASGQL identifies the file using the input parameters (see calling sequence below), stages the file to magnetic disk, if necessary, and associates the file name with the given LID. The analysis program must open the file using the returned LUN for read-only access in shared mode.

The calling sequence for ASGQL is as follows:

```
CALL ASGQL (DATA_TYPE, QL_PASS, QL_UARS_DAY, LID, LUN, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA_TYPE	CHAR*12	I	Type of QUICKLOOK data to be accessed, namely 'CLAES' 'HALOE' 'HRDI' 'ISAMS' 'MLS' 'PEM' 'SOLSTICE' 'SUSIMA' 'SUSIMB' 'WINDII' 'ENGINEERING' 'OBC' 'QUALITY' 'SPACECRAFT' 'EXTRSC' 'SSPPGIMBALS'

ARGUMENT	TYPE	I/O	DEFINITION
QL_PASS	I*4	I/O	QUICKLOOK pass number (n) on input, if > 0, pass n of day QL_UARS_DAY if = 0, the latest pass if < 0, the nth previous pass on output, the actual pass number selected
QL_UARS_DAY	I*4	I/O	UARS day number on input, required, if QL_PASS > 0 ignored, otherwise on output, the UARS day number of the pass selected
LID	CHAR*12	I	Logical file identifier to be associated with the requested QUICKLOOK data file
LUN	I*4	O	Logical unit number assigned to the LID
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return other codes - Error (See Table F-1)

5.3.2.2 GENERIC FILE ASSIGNMENT SERVICE (GENASG)

GENASG provides a generic manner of assigning a cataloged file. Files that can be assigned via this service include all levels of instrument data files, all types of calibration data files, all types of user-generated, instrument-oriented correlative data files, and all types of quick-look data files. GENASG identifies the requested file using the input attributes, ensures that the file is on magnetic disk, and associates the input LID with the name of the identified file.

It returns a unique logical unit number (LUN) that can be used to perform FORTRAN I/O on the file. The analysis program must open the file for READONLY access.

The calling sequence for GENASG is as follows:

```
CALL GENASG (LID, NUM_ATTRS, ATTR_NAMES, ATTR_VALUES, LUN, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier to be associated with the requested data file

ARGUMENT	TYPE	I/O	DEFINITION
NUM_ATTRS	I*4	I	Number of attributes (N)
ATTR_NAMES	CHAR*(*)(N)	I	Names of attributes defining the requested file (See Notes below)
ATTR_VALUES	CHAR*(*)(N)	I	Associated Values of attributes in ATTR_NAMES
LUN	I*4	O	Logical unit number assigned to the LID
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return other codes - Error (See Table F-1)

Notes:

Required attributes per data type:

INSTRUMENT	CORRELATIVE	CALIBRATION	QUICKLOOK
TYPE (=instr. id)	TYPE (='CORRELATIVE')	TYPE (='CALIBRATION')	TYPE (='QUICKLOOK')
SUBTYPE (Note 1)	SUBTYPE	SUBTYPE	SUBTYPE
LEVEL	SOURCE	LEVEL (Note 2)	QUICKLOOK_ID
DAY	DAY	DAY (Note 3)	DAY
		CALIBRATION_ID	

Notes 1. Not applicable for Level 0 data
2. Not applicable for levelless file
3. Not applicable for dayless file

5.3.3 OPEN QUICK-LOOK FILE (OPENQL)

The OPENQL service opens a quick-look data file (e.g. instrument, engineering, OBC, spacecraft or quality) for read access by a program in the analysis environment. OPENQL identifies the file, ensures that it is on magnetic disk and opens the file for read access in shared mode. The analysis program can then use the logical file identifier (LID) to read data from the quick-look file.

The calling sequence for OPENQL is as follows:

```
CALL OPENQL (DATA_TYPE, QL_PASS, UARS_DAY, LID, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
DATA TYPE	CHAR*12	I	Mnemonic for type of quick-look data to be accessed, namely 'CLAES' 'HALOE' 'HRDI' 'ISAMS' 'MLS' 'PEM' 'SOLSTICE' 'SUSIMA' 'SUSIMB' 'WINDII' 'ENGINEERING' 'OBC' 'QUALITY' 'SPACECRAFT'
QL_PASS	I*4	I/O	Quick-look pass number (n) On input, if > 0, pass n of day UARS_DAY if = 0, the latest pass if < 0, the nth previous pass On output, the actual pass number
UARS_DAY	I*4	I/O	UARS day number. Required on input if QL_PASS > 0; ignored otherwise. On output, the UARS day number of the pass selected.
LID	CHAR*16	I	Logical file identifier
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return Other codes - Error (See Table F-1)

5.3.4 READ QUICK-LOOK FILE (READQL)

READQL provides a read service for cataloged quick-look data from a program in the analysis environment. OPENQL must be called to select the quick-look file before READQL can be used. Requests for data are time-referenced by Engineering Major Frame (EMAF). Each call returns the instrument data from one EMAF. If the specified time does not match the time associated with any EMAF, the first EMAF of the quick-look pass after the specified time is returned. Therefore, the first EMAF in a file can be read by specifying a zero date and time in

the REQ_DATTIM argument field. On return the REQ_DATTIM field contains the date and time of the next available EMAF. RET_DATTIM provides the actual date and time of the returned EMAF. If a time after the last EMAF in the pass is specified, a 'no-data-read' status is returned.

When the last EMAF of a quicklook data file has been returned as part of a read, the returned status will be set to PFA_EOF to show that no more data is available for further sequential input from the file and the time of the next available EMAF will be set to zero.

The calling sequence for READQL is as follows:

```
CALL READQL (LID, REQ_DATTIM, RET_DATTIM, EMAF_REC, PARITY, FILL,
             GAP_FLAG, TIME_FLAG, EMAF_RATE, VERSION, STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier
REQ_DATTIM	I*4(2)	I/O	On input, date and time of the requested EMAF in UDTF format. On output, date and time of the next EMAF available.
RET_DATTIM	I*4(2)	O	Date and time in UDTF format of the returned EMAF, namely EMAF_REC
EMAF_REC	BYTE(*)	O	Quick-look telemetry record for the selected instrument. See Appendix D for the specific format based on data type. The record contains one EMAF of data.
PARITY	BYTE(8)	O	A binary array of parity flags for the 64 Science Major Frames (SMAFs) in EMAF_REC. There is one bit flag per SMAF. 0 All SMIFs in the SMAF have good CRC codes 1 One or more SMIFs have CRC errors or contain fill data
FILL	BYTE(8)	O	A binary array of fill flags for the 64 SMAFs in EMAF_REC. There is one bit flag per SMAF. 0 All SMIFs in the SMAF contain data 1 One or more SMIFs contain fill
GAP_FLAG	I*2	O	Indicates whether or not EMAF_REC follows a gap 0 No gap 1 EMAF follows a gap

ARGUMENT	TYPE	I/O	DEFINITION
TIME_FLAG	I*2	O	Indicates a questionable absolute time code (ATC) time in EMAF_REC 0 Normal ATC increment 1 Abnormal ATC increment
EMAF_RATE	I*4	O	Time interval between EMAFs in msec
VERSION	I*2(2)	O	CCB version and cycle number of the quick-look file read
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return Other codes - Error (See Table F-1)

5.3.5 READ QUICK-LOOK DATA QUALITY FILE (QUALQL)

QUALQL provides a read service for cataloged quick-look quality data from a program in the analysis environment. OPENQL must be called to select the quick-look file before QUALQL can be used. Requests for data are time-referenced by Engineering Major Frame (EMAF). Each call returns one quality record associated with a particular EMAF. If the specified time does not match the time associated with any EMAF, the first record of the quality file after the specified time is returned. On return the REQ_DATTIM field contains the date and time of the next available record. RET_DATTIM provides the actual date and time of the returned record. If a time after the last EMAF in the pass is specified, a 'no-data-read' status is returned.

When the last EMAF of a quicklook quality file has been returned as part of a read, the returned status will be set to PFA_EOF to show that no more data is available for further sequential input from the file and the time of the next available EMAF will be set to zero.

The calling sequence for QUALQL is as follows:

```
CALL QUALQL (LID, REQ_DATTIM, RET_DATTIM, PARITY, FILL, VERSION,
             STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	CHAR*16	I	Logical file identifier
REQ_DATTIM	I*4(2)	I/O	On input, date and time of the requested EMAF in UDTF format. On output, date and time of the next EMAF available.
RET_DATTIM	I*4(2)	O	Date and time in UDTF format of the start of the EMAF returned

ARGUMENT	TYPE	I/O	DEFINITION
PARITY	BYTE(256)	O	A binary array of flags for the 2048 Science Minor Frames (SMIFs) in an EMAF indicating parity errors detected 0 SMIF has good CRC code 1 SMIF has bad CRC code or fill data
FILL	BYTE(256)	O	A binary array of flags for the 2048 SMIFs in an EMAF indicating whether the SMIFs are filled or not 0 SMIF contains data 1 SMIF contains fill
VERSION	I*2(2)	O	CCB version and cycle number of the quick-look quality file read
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return Other codes - Error (See Table F-1)

5.4 OTHER SERVICES

5.4.1 SET VERSION/CYCLE PARAMETERS (SETVERCY)

The SETVERCY service provides a means by which the file version and/or cycle information for a cataloged file may be specified before a file is actually opened or assigned via the analysis services.

SETVERCY allows the version/cycle information to be input in two different forms. The first form provides a version and/or cycle range, a corresponding selection rule that applies to that range and a threshold time used to exclude files created after that time. It is particularly applicable to virtual files spanning more than one day. The second form provides individual values, of version and cycle for specific days in a file's processing time range. Both forms can be specified at the same time. When they are, the second form takes precedence over the first for any days where the specifications overlap.

The calling sequence for SETVERCY is as follows:

```
CALL SETVERCY (LID, START_VERSION, STOP_VERSION, VERSION_RULE,
               START_CYCLE, STOP_CYCLE, CYCLE_RULE, THRESHOLD_TIME,
               FILE_REQUIRED_FLAG, NUM_ENTRIES, DAY, VERSION, CYCLE,
               STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	C*16	I	Logical file identifier of file to be opened or assigned
START_VERSION	I*2	I	Lower bound of version range over which

ARGUMENT	TYPE	I/O	DEFINITION
			version rule will apply for file with given LID
STOP_VERSION	I*2	I	Upper bound of version range over which version rule will apply for file with given LID
VERSION_RULE	I*2	I	Rule to be used in selecting the file version over specified version range 1 HIGHEST in range 2 HIGHEST in common range 9 Do not stage
START_CYCLE	I*2	I	Lower bound of cycle range over which cycle rule applies
STOP_CYCLE	I*2	I	Upper bound of cycle range over which cycle rule applies
CYCLE_RULE	I*2	I	Rule to be used in selecting file cycle over specified cycle range. Only meaningful if version range is not specified. 1 Highest in range 2 Highest common in range (default = 1)
THRESHOLD_TIME	C*23	I	Time in VAX ASCII format used as a threshold in the selection of files by version rule. Files with values of creation time exceeding THRESHOLD_TIME will not be selected (If field is left blank current time will be used)
FILE_REQUIRED_FLAG	L*1	I	Flag indicating requirement for all files for all days within virtual time range
NUM_ENTRIES	I*4	I	Number of entries (NE) in DAY, VERSION and CYCLE arrays
DAY	I*4 (NE)	I	UARS days for which specific values of file version and cycle are to be used
VERSION	I*2 (NE)	I	Values of file version to be used for each UARS day in DAY array
CYCLE	I*2 (NE)	I	Values of file cycle to be used in conjunction with values of version in VERSION array and UARS day in DAY array
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return

ARGUMENT	TYPE	I/O	DEFINITION
			PFA_FVPARALRUSD - Specified LID already in use PFA_FVPARALRSET - Parameters already set for current or other LID Other codes - Error (See Table F-1)

5.4.2 GET VERSION/CYCLE PARAMETERS (GETVERCY)

The GETVERCY service provides a means for querying an opened or assigned file for its version and cycle information.

The calling sequence for GETVERCY is as follows:

```
CALL GETVERCY (LID, MAX_NUM_ENTRIES, NUM_ENTRIES, DAY, VERSION, CYCLE,
               STATUS)
```

ARGUMENT	TYPE	I/O	DEFINITION
LID	C*16	I	Logical file identifier of opened or assigned file
MAX_NUM_ENTRIES	I*4	I	Maximum number of entries (NE) allowed for in day, version, and cycle arrays
NUM_ENTRIES	I*4	O	Number of physical files accessible as part of current file
DAY	I*4 (NE)	O	UARS day associated with each accessible physical file
VERSION	I*2 (NE)	O	Version number associated with each accessible physical file
CYCLE	I*2 (NE)	O	Cycle number associated with each accessible physical file
STATUS	I*4	O	Status code returned SS$_NORMAL - Normal return PFA_UNKNOWNLID - Specified LID not associated with any files Other codes - Error (See Table F-1)

APPENDIX A

UARS DATE AND TIME FORMAT

The UARS standard format for specifying a date and time (UDTF) is a two-word array. The date is in the first word in the form of YYYDDD, specifying the year (e.g., 90) and the day of year (e.g., January 1 = 001) requested.

(YEAR - 1900) * 1000 + DAY OF YEAR.

The time is the second word of the array and indicates the time in milliseconds of day.

APPENDIX B

UCSS PRODUCTION SERVICE FORTRAN EXAMPLE

This appendix provides an example of the usage of the UCSS services. This example uses Level 0 MLS instrument and quality data as input, generates a scratch file, and produces a Level 1 file.

```
      PROGRAM LEVEL0
            .
            .
            .
      BYTE MLS_EMAF(10304)                    !MLS L0 EMAF
      BYTE PARITY(8)                          !SMAF parity flags
      BYTE FILL(8)                            !SMAF fill flags
      BYTE QUAL_PARITY(256)                   !SMIF parity flags
      BYTE QUAL_FILL(256)                     !SMIF fill flags
      BYTE OZREC(5000)                        !L1 ozone record
      BYTE SCREC(512)                         !Scratch file record
C
      CHARACTER*3 LEVEL1/'1'/                 !Level 1 indicator
      CHARACTER*4 NEW_FILE/'NEW'/             !New file flag
      CHARACTER*4 PASS_FAIL_FLAG              !Pass/fail flag
      CHARACTER*12 INST_ID/'MLS'/             !Instrument ID
      CHARACTER*12 QUALITY/'QUALITY'/         !Quality data type
      CHARACTER*12 OZONE_DATA/'OZONE'/        !Ozone data type
      CHARACTER*16 MLID/'MLS_LEVEL0_LID'/     !MLS L0 LID
      CHARACTER*16 QLID/'QUALITY_LID'/        !Quality LID
      CHARACTER*16 OZLID/'L1_OZONE_LID'/      !L1 ozone LID
      CHARACTER*16 SCLID/'SCRATCH_LID'/       !Scratch file LID
      CHARACTER*20 PARAMS(2,20)               !Program parameters
      CHARACTER*80 OZONE_ATTR(2,22)           !Ozone catalog attribute
      CHARACTER*80 COMMENTS/' '/              !Program comments
      CHARACTER*80 DUMMY_ATTR/' '/            !Dummy AAA
C
      INTEGER*2 VERSION(2)                    !Version and cycle
      INTEGER*2 GAP_FLAG                      !Missing EMAFs flag
      INTEGER*2 TIME_FLAG                     !Questionable time flag
C
      INTEGER*4 STRTIME(2)                    !Proc. start time
      INTEGER*4 STPTIME(2)                    !Proc. stop time
      INTEGER*4 L1_START_TIME(2)              !L1 file start time
```

```
      INTEGER*4 L1_STOP_TIME(2)                !L1 file stop time
      INTEGER*4 MLS_REQ_TIME(2)                !MLS request time
      INTEGER*4 MLS_ACT_TIME(2)                !MLS record time
      INTEGER*4 QUAL_REQ_TIME(2)               !QUAL request time
      INTEGER*4 QUAL_ACT_TIME(2)               !QUAL record time
      INTEGER*4 UARS_DAY                       !UARS processing day
      INTEGER*4 EMAF_RATE                      !EMAF rate
      INTEGER*4 OZLUN,SCRLUN                   !Logical unit numbers
      INTEGER*4 OZONE_SIZE/14000/              !Ozone file size
      INTEGER*4 SCRATCH_SIZE/1400/             !Scratch file size
      INTEGER*4 COND_CODE                      !Program condition code
      INTEGER*4 STATUS                         !Service status
      INTEGER*4 IOERR                          !I/O error status
      INTEGER*4 NUM_OZONE_ATTR                 !No. ozone attributes
C
      EXTERNAL SS$_NORMAL                      !Normal status code
              .
              .
              .
C
C       INITIALIZE PASS/FAIL FLAG AND PROGRAM CONDITION CODE
C
      COND_CODE = %LOC(SS$_NORMAL
      PASS_FAIL_FLAG = 'PASS'
C
C       CALL UCSS PROGRAM INITIALIZATION SERVICE
C
      CALL PGINIT(PARAMS, STRTIME, STPTIME, UARS_DAY)
              .
              .
              .
C
C       OPEN MLS LEVEL 0 AND QUALITY FILES
C
      CALL OPENL0(INST_ID, STRTIME, STPTIME, MLID, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL) THEN
              .
              .
              .
      CALL OPENL0(QUALITY, STRTIME, STPTIME, QLID, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL) THEN
              .
              .
              .
```

```
C
C       ASSIGN AND OPEN LEVEL 1 OUTPUT FILE FOR OZONE
C
      CALL ASGCAT(UARS_DAY, INST_ID, LEVEL1, OZONE_DATA,
     1            NEW_FILE, OZONE_SIZE, OZLID, OZLUN, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL) THEN
              .
              .
              .
      OPEN(UNIT=OZLUN,FILE=OZLID,ACCESS='SEQUENTIAL', RECL=1250,
     1     INITIALSIZE=OZONE_SIZE,FORM='UNFORMATTED',
     2     STATUS='NEW',IOSTAT=IOERR)
              .
              .
              .
C
C       ASSIGN AND OPEN A SCRATCH FILE
C
      CALL ASGSCR(SCRATCH_SIZE, NEW_FILE, SCLID, SCRLUN, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL) THEN
              .
              .
              .
      OPEN(UNIT=SCRLUN, FILE=SCLID, ACCESS='SEQUENTIAL',
     1     RECL=128, INITIALSIZE=SCRATCH_SIZE,
     2     FORM='UNFORMATTED', STATUS='NEW', IOSTAT=IOERR)
              .
              .
              .
C
C       SET INITIAL TIME TO START TIME OF PROCESSING
C
      MLS_REQ_TIME = STRTIME
      QUAL_REQ_TIME = STRTIME
              .
              .
              .
C
C       READ THE MLS LEVEL 0 EMAF FOR THE TIME
C       SPECIFIED IN MLS_REQ_TIME
C
      CALL READL0(MLID, MLS_REQ_TIME, MLS_ACT_TIME, MLS_EMAF,
     1            PARITY, FILL, GAP_FLAG, TIME_FLAG,
     2            EMAF_RATE, VERSION, STATUS)
              .
              .
              .
C
C       READ THE QUALITY DATA FOR THE EMAF WITH THE
C       TIME SPECIFIED IN QUAL_REQ_TIME
C
      CALL QUALRD(QLID, QUAL_REQ_TIME, QUAL_ACT_TIME,
     1            QUAL_PARITY, QUAL_FILL, VERSION, STATUS)
```

```
                .
                .
                .
C
C       WRITE DATA TO OUTPUT FILES
C
      WRITE(OZLUN,IOSTAT=IOERR) OZREC
                .
                .
                .
      WRITE(SCRLUN,IOSTAT=IOERR) SCREC
                .
                .
                .
C
C       CLOSE MLS LEVEL 0, ENGINEERING AND QUALITY FILES
C
      CALL CLOSELF(MLID, 'FREE',, DUMMY_ATTR, STATUS)
      CALL CLOSELF(QLID, 'FREE',, DUMMY_ATTR, STATUS)
                .
                .
                .
C
C       CLOSE AND CATALOG MLS LEVEL 1 OZONE FILE
C
      CLOSE(OZLUN,IOSTAT=IOERR)
      NUM_OZONE_ATTR = 3
      OZONE_ATTR(1,1) = 'START_TIME'
      CALL UTL_CON_UDTF_VMS(L1_START_TIME, OZONE_ATTR(2,1),
     1                      STATUS)
      OZONE_ATTR(1,2) = 'STOP_TIME'
      CALL UTL_CON_UDTF_VMS(L1_STOP_TIME, OZONE_ATTR(2,2),
     1                      STATUS)
      OZONE_ATTR(1,3) = 'RECORD_SIZE'
      OZONE_ATTR(2,3) = '14000'
      CALL DASLID(OZLID, 'CAT ', NUM_OZONE_ATTR, OZONE_ATTR, STATUS)
                .
                .
                .
C
C       CLOSE SCRATCH FILE
C
      CLOSE(SCRLUN,IOSTAT=IOERR)
      CALL DASLID(SCLID, 'FREE',, DUMMY_ATTR, STATUS)
                .
                .
                .
C
C       CALL PGTERM TO WRAP UP PROGRAM PROCESSING
C
      CALL PGTERM(PASS_FAIL_FLAG, COND_CODE, COMMENTS)
      STOP
      END
```

APPENDIX C

LEVEL 1 AND LEVEL 2 DATA PROCESSING GUIDELINES

This appendix provides guidelines for the processing of Level 1 and Level 2 data by production programs. The UCSS provides the basic interface tools for assigning and deassigning the Level 1 and Level 2 data. The programmer is responsible for supplying the read and write services.

The UCSS provides the ASGCAT, ASGSCR, ASGCOR, and ASGCAL routines to assign data sets to the production programs. Following assignment of the file using the appropriate service, the programmer can use standard Fortran OPEN, READ, WRITE, and CLOSE services to perform the actual I/O on the file. When opening an existing cataloged file, the programmer must specify READONLY on the Fortran OPEN statement. DASLID must be called upon completion of all I/O (after the CLOSE) to the file. If a new cataloged file was created, DASLID generates the catalog entry for the file. For existing cataloged files, DASLID updates the accounting data in the Catalog.

Following are examples of the UCSS services available to the programmer with sample calls for assigning Level 1 and Level 2 data sets. In addition an example is presented of how the scratch file capability can be used to pass data from one program to another within a processing run.

```
      PROGRAM LVL1A
              .
              .
              .
      BYTE L1REC(2400)                        !Level 1 record
      BYTE SCREC(1200)                        !Scratch record
C
      CHARACTER*3 LEVEL1/'1 '/                !Level 1 indicator
      CHARACTER*4 PASS_FAIL_FLAG              !Pass/fail flag
      CHARACTER*4 NEW_FLAG/'NEW'/             !New file flag
      CHARACTER*4 OLD_FLAG/'OLD'/             !Existing file flag
      CHARACTER*12 INST_ID/'HALOE       '/    !Instrument ID
      CHARACTER*12 LEVEL1_TYPE/'NONE        '/ !Level 1 file subtype
      CHARACTER*16 L1LID/'L1_INPUT_LID    '  !Level 1 file LID
      CHARACTER*16 SCLID/'SCRATCH_LID     '/ !Scratch file LID
```

```
      CHARACTER*20 PARAMS(2,20)                !Program parameters
      CHARACTER*80 COMMENTS                    !Program comment
      CHARACTER*80 DUMMY_ATTR/' '/             !Dummy attribute
C
      INTEGER*4 COND_CODE                      !Program condition code
      INTEGER*4 IOERR                          !I/O error status
      INTEGER*4 L1LUN,SCRLUN                   !Logical unit numbers
      INTEGER*4 RECNO                          !Scratch record number
      INTEGER*4 SCRATCH_SIZE/1500/             !Scratch file size
      INTEGER*4 STATUS                         !Service status
      INTEGER*4 STRTIM(2)                      !Processing start time
      INTEGER*4 STPTIM(2)                      !Processing stop time
      INTEGER*4 UARS_DAY                       !UARS processing day
C
      EXTERNAL  SS$_NORMAL                     !Normal return code
              .
              .
              .
      PASS_FAIL_FLAG = 'PASS'
      COND_CODE = %LOC(SS$_NORMAL)
C
C        CALL PGINIT TO RETRIEVE PROGRAM PARAMETERS
C
      CALL PGINIT(PARAMS, STRTIM, STPTIM, UARS_DAY)
              .
              .
              .
C
C        ASSIGN LEVEL 1 INPUT FILE FOR SPECIFIED DAY
C
      CALL ASGCAT(UARS_DAY, INST_ID, LEVEL1, LEVEL1_TYPE,
     1            OLD_FLAG,, L1LID, L1LUN, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL)) THEN
              .
              .
              .
C
C        ASSIGN A SCRATCH FILE
C
      CALL ASGSCR(SCRATCH_SIZE, NEW_FLAG, SCLID, SCRLUN, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL)) THEN
              .
              .
              .
C
C        OPEN THE LEVEL 1 INPUT FILE FOR READ ACCESS IN SHARED
C        MODE.  FORMAT IS USER SPECIFIED.
C
      OPEN(UNIT=L1LUN, FILE=L1LID, READONLY, SHARED,
     1     FORM='UNFORMATTED', ACCESS='SEQUENTIAL', STATUS='OLD',
     2     IOSTAT=IOERR)
```

```
C
C         (CHECK OPEN STATUS)
C
                .
                .
                .
C
C         OPEN THE SCRATCH FILE (FORMAT IS USER SPECIFIED)
C       OPEN(UNIT=SCRLUN, FILE=SCLID, INITIALSIZE=SCRATCH_SIZE,
     1      ACCESS='DIRECT', FORM='UNFORMATTED', STATUS='NEW',
     2      RECL=300, IOSTAT=IOERR)
                .
                .
                .
C
C         READ RECORD FROM LEVEL 1 INPUT FILE
C
       READ(L1LUN, IOSTAT=IOERR) L1REC
                .
                .
                .
C
C         WRITE A RECORD TO THE SCRATCH FILE
C
       WRITE(SCRLUN,RECNO, IOSTAT=IOERR) SCREC
                .
                .
                .
C
C         CLOSE THE LEVEL 1 INPUT FILE
C
       CLOSE(L1LUN, IOSTAT=IOERR)
C
C         CLOSE THE SCRATCH FILE
C
       CLOSE(SCRLUN, IOSTAT=IOERR)
C
C         DEASSIGN THE LEVEL 1 INPUT FILE
C
       CALL DASLID(L1LID, 'FREE', , DUMMY_ATTR, STATUS)
C
C         DEASSIGN AND HOLD THE SCRATCH FILE
C
       CALL DASLID(SCLID, 'HOLD',, DUMMY_ATTR, STATUS)
C
C         CALL PGTERM TO WRAP UP PROGRAM PROCESSING
C
       CALL PGTERM('PASS', COND_CODE, COMMENTS)
       STOP
       END
```

```
      PROGRAM LVL1B
              .
              .
              .
      BYTE L2REC(1000)                          !Level 2 record
      BYTE SCREC(1200)                          !Scratch record
C
      CHARACTER*3 LEVEL2/'2'/                   !Level 2 indicator
      CHARACTER*4 PASS_FAIL_FLAG                !Pass/fail flag
      CHARACTER*4 NEW_FLAG/'NEW'/               !New file flag
      CHARACTER*4 HELD_FLAG/'HELD'/             !Existing file flag
      CHARACTER*12 INST_ID/'HALOE'/             !Instrument ID
      CHARACTER*12 LEVEL2_TYPE/'NONE'/          !Level 2 file subtype
      CHARACTER*16 L2LID/'L2_OUTPUT_LID'/       !Level 2 file LID
      CHARACTER*16 SCLID/'SCRATCH_LID'/         !Scratch file LID
      CHARACTER*20 PARAMS(2,20)                 !Program parameters
      CHARACTER*80 COMMENTS                     !Program comment
      CHARACTER*80 DATA_ATTR(2,10)              !Catalog attributes
C
      INTEGER*4 COND_CODE                       !Program condition code
      INTEGER*4 IOERR                           !I/O error status
      INTEGER*4 L2_START_TIME(2)                !L2 file start time
      INTEGER*4 L2_STOP_TIME(2)                 !L2 file stop time
      INTEGER*4 L2LUN,SCRLUN                    !Logical unit numbers
      INTEGER*4 LEVEL2_SIZE/2800/               !Level 2 file size
      INTEGER*4 NUM_ATTR                        !No. catalog attributes
      INTEGER*4 SCRNUM                          !Scratch record number
      INTEGER*4 STATUS                          !Service status
      INTEGER*4 STRTIM(2)                       !Processing start time
      INTEGER*4 STPTIM(2)                       !Processing stop time
      INTEGER*4 UARS_DAY                        !UARS processing day
C
      EXTERNAL  SS$_NORMAL                      !Normal return code
              .
              .
              .
C
C        INITIALIZE PASS/FAIL FLAG AND PROGRAM CONDITION CODE
C     PASS_FAIL_FLAG = 'PASS'
      COND_CODE = %LOC(SS$_NORMAL)
C
C        CALL PGINIT TO RETRIEVE PROGRAM PARAMETERS
C     CALL PGINIT(PARAMS, STRTIM, STPTIM, UARS_DAY)
              .
              .
              .
C
C        ASSIGN LEVEL 2 OUTPUT FILE FOR SPECIFIED DAY
C
      CALL ASGCAT(UARS_DAY, INST_ID, LEVEL2, LEVEL2_TYPE,
     1             NEW_FLAG, LEVEL2_SIZE, L2LID, L2LUN,STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL) THEN
```

```
                .
                .
                .
C
C          ASSIGN AN EXISTING SCRATCH FILE
C
      CALL ASGSCR(, HELD_FLAG, SCLID, SCRLUN, STATUS)
      IF (STATUS .EQ. %LOC(SS$_NORMAL)) THEN
                .
                .
                .
C
C          OPEN THE LEVEL 2 OUTPUT FILE (FORMAT IS USER
C          SPECIFIED)
C
      OPEN(UNIT=L2LUN, FILE=L2LID, FORM='UNFORMATTED',
     1     ACCESS='DIRECT', STATUS='NEW', INITIALSIZE=LEVEL2_SIZE,
     2     RECL=250, IOSTAT=IOERR)
                .
                .
                .
C
C          OPEN THE EXISTING SCRATCH FILE (FORMAT IS USER SPECIFIED)
C
      OPEN(UNIT=SCRLUN,FILE=SCLID,ACCESS='DIRECT',
     1     FORM='UNFORMATTED',STATUS='OLD',IOSTAT=IOERR)
                .
                .
                .
C
C          READ RECORD FROM SCRATCH FILE
C
      READ(SCRLUN,SCRNUM,IOSTAT=IOERR) SCREC
                .
                .
                .
C
C          WRITE A RECORD TO THE LEVEL 2 FILE
C
      WRITE(L2LUN,L2NUM,IOSTAT=IOERR) L2REC
                .
                .
                .
C
C          CLOSE THE LEVEL 2 OUTPUT FILE
C
      CLOSE(L2LUN, IOSTAT=IOERR)
C
C          CLOSE THE SCRATCH FILE
C
      CLOSE(SCRLUN, IOSTAT=IOERR)
```

```
C
C        SET UP LEVEL 2 CATALOG ATTRIBUTE ARRAY TO SPECIFY FILE
C        START TIME, FILE STOP TIME, AND RECORD SIZE
C
      NUM_ATTR = 3
      DATA_ATTR(1,1) = 'START_TIME'
      CALL UTL_CON_UDTF_VMS(L2_START_TIME, DATA_ATTR(2,1),
     1                      STATUS)
      DATA_ATTR(1,2) = 'STOP_TIME'
      CALL UTL_CON_UDTF_VMS(L2_STOP_TIME, DATA_ATTR(2,2), STATUS)
      DATA_ATTR(1,3) = 'RECORD_SIZE'
      DATA_ATTR(2,3) = '2800'
C
C        DEASSIGN THE LEVEL 2 OUTPUT FILE
C
      CALL DASLID(L2LID, 'CAT ', NUM_ATTR, DATA_ATTR, STATUS)
C
C        DEASSIGN AND RELEASE THE SCRATCH FILE
C
      CALL DASLID(SCLID, 'FREE',,DUMMY_ATTR, STATUS)
C
C        CALL PGTERM TO WRAP UP PROGRAM PROCESSING
C
      CALL PGTERM(PASS_FAIL_FLAG, COND_CODE, COMMENTS)
      STOP
      END
```

APPENDIX D

LEVEL 0 FILE FORMATS

This appendix defines the formats for the Level 0 files.

D.1 SCIENCE TELEMETRY FORMATS AND DECOMMUTATION

The formats for the UARS science telemetry and the engineering telemetry are being defined by the UARS spacecraft development and integration contractor, General Electric (GE), Space Systems Division, in conjunction with the UARS Project. The science telemetry formats are defined by the GE Program Information Release (PIR) U-1K21-UARS-700, Reference 8. A copy of the current science minor frame format is shown in Table D-1.

Table D-1. Science Minor Frame Format

WORD	FUNCTION	WORD	FUNCTION	WORD	FUNCTION	WORD	FUNCTION
0	SYNC 'D7'	1	SYNC '99'	2	SYNC '07'	3	CDCUSTAT
4	CDFRMCNT	5	CDFRMCNT	6	CDCMDCNT	7	AAIAUXARY
8	ENG_DATA	9	ENG_DATA	10	ENG_DATA	11	ENG_DATA
12	OBC	13	OBC	14	OBC	15	OBC
16	OBC	17	OBC	18	OBC	19	PWIIMLO
20	PSIPULSEA	21	PSIPULSEB	22	SSPPAPOSA	23	SSPPBPOS
24	ES1PITCHF	25	ES1ROLLF	26	ES2PITCHF	27	ES2ROLLF
28	ACRIM II	29	ACRIM II	30	SC_SPARE	31	PWIBAT1HI
32	CLAES	33	CLAES	34	CLAES	35	CLAES
36	CLAES	37	CLAES	38	CLAES	39	CLAES
40	CLAES	41	CLAES	42	CLAES	43	CLAES
44	HALOE	44	HALOE	44	HALOE	44	HALOE
48	HALOE	49	HALOE	50	HALOE	51	HALOE
52	HALOE	53	HALOE	54	HALOE	55	HALOE
56	HALOE	57	HALOE	58	HALOE	59	HALOE
60	HRDI	61	HRDI	62	HRDI	63	HRDI
64	HRDI	65	HRDI	66	HRDI	67	HRDI
68	HRDI	69	HRDI	70	HRDI	71	HRDI
72	HRDI	73	HRDI	74	HRDI	75	HRDI
76	HRDI	77	HRDI	78	HRDI	79	PWIBAT2HI
80	ISAMS	81	ISAMS	82	ISAMS	83	ISAMS
84	MLS	85	MLS	86	MLS	87	MLS
88	MLS	89	SSPPAPOS	90	SSPPBPOS	91	PWIACS
92	PEM	93	PEM	94	PEM	95	PEM
96	PEM	97	PEM	98	PEM	99	PEM
100	PEM	101	PEM	102	PEM	103	PEM
104	PEM	105	PEM	106	SOLSTICE	107	PWICDH
108	SUSIM	109	SUSIM	110	SUSIM	111	SUSIM
112	SUSIM	113	SUSIM	114	SUSIM	115	SUSIM
116	WINDII	117	WINDII	118	WINDII	119	WINDII
120	WINDII	121	WINDII	122	WINDII	123	WINDII
124	PWIBAT3HI	125	PWISCCU	126	PARITY	127	PARITY

As described in PIR 700, there are two science formats. One format, referred to as SCI-1, is the nominal format. The second format, SCI-2, is appropriate to periods of propulsion module activity. In both SCI-1 and SCI-2 formats, the Science Minor Frame (SMIF) is 128 words in length, where a word in this context is 8 bits. In both formats, the word allocations and assignments are constant. The difference between the two formats is that the interpretation changes for 6 of the 17 words of spacecraft telemetry.

D.2 DECOMMUTATED FILE FORMATS

D.2.1 GENERAL COMMENTS

The UARS telemetry data is decommutated into 5 Level 0 files. The first record of each Level 0 file is a file label record which identifies the type of file and the file contents. The file label record is followed by data records, where the number of data records is dependent on the type of file and the time span of telemetry contained by the file. Each data record contains a standard 64 byte record header followed by the telemetry words. The record header contains information describing the record contents.

There is one physical record per EMAF for files with less than 32 Kbytes of telemetry data per EMAF (i.e., 15 or fewer telemetry words per SMIF). There are two physical records per EMAF for those files with more than 32 Kbytes of telemetry data per EMAF (i.e., 16 or more words per SMIF). For these files, the first record contains the telemetry data for the first 32 SMAFs of the EMAF and the remaining 32 SMAFs are in the second data record.

Because the number of telemetry words varies by file type, the record length is dependent on the type. If the data record length is greater than the file header length, the file label record is filled so that it is the same length. For data records smaller than file label records, the data records are filled out to the length of the label record, 2532 bytes.

The Level 0 files are stored on the CDHF as flat files without any index structure.

This appendix describes the format of the Level 0 data files stored on the CDHF, the quick-look files, and the "virtual" Level 0 files produced by the UCSS data transfer software. The format of the virtual Level 0 files is discussed further in Section D.2.3.

D.2.2 FILE LABEL RECORD FORMAT

The file label record format is presented in Table D-2. All file label record fields are ASCII fields.

Table D-2. Level 0 File Label Record

ITEM NO.	BYTES	FIELD NAME	COMMENTS
1	1 - 4	satellite id	'UARS'
2	5 - 8	data set #	see Table D-3
3	9 - 12	data set id	see Table D-3
4	13 - 16	format version #	see Note 1
5	17 - 20	physical record count (=' 1')	Rec. # in file
6	21 - 24	# phys. records/EMAF	e.g., ' 1'
7	25 - 28	# physical records in file	see Note 2
8	29 - 36	CCB version number assigned to file	
9	37 - 40	file cycle (transferred files only)	
10	41 - 44	spare	
11	45 - 48	ATC epoch year - begin. of first EMAF	'1989'
12	49 - 64	ATC - .5 msec - " " first "	
13	65 - 68	ATC epoch year - " " last "	
14	69 - 84	ATC - .5 msec - " " last "	
15	85 - 88	JATC:year - begin. of first EMAF	year
16	89 - 92	JATC:day - " " first "	day
17	93 -100	JATC:msec - " " first "	millisecond
18	101 -104	JATC:usec - " " first "	microsecond
19	105 -108	JATC:year - " " last "	year
20	109 -112	JATC:day - " " last "	day
21	113 -120	JATC:msec - " " last "	millisecond
22	121 -124	JATC:usec - " " last "	microsecond
23	125 -132	# SMIFs expected	
24	133 -140	# SMIFs in file (excluding fill)	
25	141 -148	# SMIFs fill	
26	149 -156	# SMIFs with CRC error	
27	157 -162	# SMAFs expected	
28	163 -168	# SMAFs in file (excluding all fill SMAFs)	
29	169 -174	# SMAFs of total fill	
30	175 -180	# SMAFs with partial fill or CRC errors	
31	181 -184	# EMAFs expected	
32	185 -188	# EMAFs present	
33	189 -192	# EMAFs with fill or CRC errors	
34	193 -196	# EMAFs missing from coverage	
35	197 -200	# EMAF level gaps in coverage	
36	201 -204	type of data time period	see Note 3
37	205 -208	UARS day number	" " 4
38	209 -212	spare	
39	213 -220	decommutation program version	
40	221 -236	decommutation run date/time	
41	237 -260	merge file name	
42	261 -264	merge rerun #	
43	265 -272	merge program version	
44	273 -288	merge run date/time	
45	289 -292	# edit files	<= 40

Table D-2. Level 0 File Label Record (Continued)

ITEM	BYTES	FIELD NAME	COMMENTS
ITEMS 46 TO 50 REPEAT 40 TIMES. THE NUMBER OF REPS CONTAINING NONFILL DATA IS GIVEN BY ITEM #45; THE REMAINING REPS ARE FILL. THE ACTUAL BYTE OFFSET FROM THE BEGINNING OF THE RECORD OF AN ITEM IN THE "NTH" REP IS DETERMINED BY ADDING 292+(N-1)*56 TO THE BYTE VALUE LISTED BELOW FOR THE ITEM.			
46	1 - 24	edit file N - filename	
47	25 - 28	edit file N - edit rerun #	
48	29 - 36	edit file N - edit program version	
49	37 - 52	edit file N - edit run date/time	
50	53 - 56	edit file N - data type	"R/T" or "P/B"
51	2533 - X	fill characters	see Note 5

Length of nonrepeating fields (bytes)	292
Length of nonrepeating fields & 40 edit files (bytes)	2532

NOTES:

1. Identifies version number of the Level 0 format.

2. For virtual files (see Note 3) the number of physical records in the file is contained in the continuation file label record

3. Identifies which of the four time period types supported under this format are contained by the files, as follows:
   " QL" = quick look data, approximately 92 EMAFs
   "24HR" = 24 hours of data, approximately 1319 EMAFs
   "VIRT" = data covering a virtual time range
   "NRT" = near real-time data, approximately 15 EMAFs

4. Contains the UARS day number of the day in which the first EMAF of the file occurs.

5. When fixed in a SOLSTICE or QUALITY file, the file label record is not filled out at all. When fixed in any other type of UARS Level 0 file, the file label record is filled out to the length of the data record for that file type, as specified in Tables D-6 to D-20.

Table D-3. Level 0 Data Set Information

FILE TYPE	DATA SET ID	DATA SET #
CLAES	CLS	1
HALOE	HAL	2
HRDI	HRD	3
ISAMS	ISM	4
MLS	MLS	5
PEM	PEM	6
SOLSTICE	SOL	7
SUSIM "A"	SMA	8
SUSIM "B"	SMB	9
WINDII	WIN	10
ACRIM	ACR	11
ENGINEERING	ENG	12
SPACECRAFT	SCT	13
OBC	OBC	14
QUALITY	QAL	15

The file label record is intended to carry information that is of interest to the operations personnel of the GTDM DCF and the CDHF. The following paragraphs are provided to clarify the meaning of the less obvious fields of the file label record.

- Item 4: format version number - Over the life of the UARS mission, several Level 0 file formats may be necessary. This document will define those formats and the format version number field will distinguish between them.

- Item 5,6 and 7: physical records - As described earlier, a physical record is intended to correspond to one EMAF, but in certain cases an EMAF may be split into two physical records. Item 5 identifies the file label record as the first physical record of the file, Item 6 identifies the number of physical

records per EMAF and Item 7 identifies the number of physical records in the file.

- Item 8: CCB version number - The configuration controlled version number assigned to the Level 0 file when cataloged on the CDHF.

- Item 9: The cycle number associated with the cataloged file. This field has meaning only for a file that has been created via the UCSS data transfer services as described in the UCSS User's Guide (Reference 9).

- Items 11 to 14: ATC - These fields correspond to the first ATC value occurring in the first and last EMAF of the file. The ATC will only be processed to remove obvious spike errors. In the event that these values are not available in the telemetry, the DCF will compute the value expected.

- Items 15 to 22: JATC - These fields contain the Julian format Absolute Time Code (JATC) values corresponding to the smoothed ATC values of items 11 to 14. The values are obtained by converting the ATC values to Greenwich Mean Time (GMT) and reformatting to Julian format. These fields correspond to the first value occurring in the first and last EMAF of the file. In the event that these values are not available in the telemetry, the DCF will compute the value expected.

- Item 36: type of data time period - The Level 0 files are intended to contain 24 hours of telemetry, one quick-look or near real-time pass, or a virtual time range of data. This field distinguishes between the four possibilities.

- Item 37: UARS day number - Each 24 hour time period (0 to 400 hours GMT) will be numbered, beginning with 1 and incremented by 1 where 1 is the time period (day) in which the UARS launch occurs. This field will contain that value. Time period ID values from 9000 to 9999 are reserved to indicate test data sets.

- Items 39 & 40: decommutation run description - These fields contain information produced by the DCF for quality control and traceability of the decommutation processing performed to produce the associated file.

- Items 41 to 44: merge run description - These fields contain information produced by the DCF for quality control and traceability of the merge processing performed to produce the associated file.

- Items 45 to 50: edit file description - These fields contain information produced by the DCF for quality control and traceability of the edit files and the edit processing

performed for each of the edit files for up to 40 edit files.

D.2.3 LEVEL 0 VIRTUAL FILES

A Level 0 virtual file is a file containing Level 0 data covering a user-specified time range. The virtual Level 0 data records are copied from the Level 0 file(s) that contain data for the requested time range. The virtual Level 0 data files are created by the UCSS data transfer services as described in the UCSS User's Guide (Reference 9). The following paragraphs describe how the Level 0 format defined in this appendix accommodates the virtual Level 0 files.

A virtual Level 0 file is distinguished from normal Level 0 files by the value "VIRT" in item 35 of the file label record format (see Table D-2). The following additional comments apply to the file label record:

- Item 4: format version - Virtual Level 0 files can only be constructed using Level 0 files with the same format version number.
- Item 7: # of physical records in file - This field will be blank and the corresponding information will be contained in the continuation file label record (see Table D-4).
- Item 8: CCB version number - The value is the version number of the Level 0 file used as the source for the first data record in the virtual Level 0 file.
- Item 9: cycle number - The value is the cycle number of the Level 0 file used as the source for the first data record in the virtual Level 0 file.
- Items 11 to 14: ATC - These fields should be ignored for virtual files.
- Items 15 to 2: JATC - These fields will contain the JATC times corresponding to the first and last EMAFs in the virtual file.
- Items 23 to 5 and 37 to 0: These fields should be ignored for virtual Level 0 files.

Table D-4. Level 0 Continuation File Label Record

ITEM NO.	BYTE	FIELD NAME	COMMENTS
1	1 - 4	satellite id	'UARS'
2	5 - 8	data set #	see Table D-3
3	9 - 12	data set id	see Table D-3
4	13 - 16	physical record count	
5	17 - 24	number of physical records in file	
6	25 - 28	number of time/version entries.....	see Note
7	29 + (I-1)*36 - 32 + (I-1)*36	JATC: year - start time for version entry	
8	33 + (I-1)*36 - 36 + (I-1)*36	JATC: day - start time for version entry	
9	37 + (I-1)*36 - 44 + (I-1)*36	JATC: milliseconds - start time of version entry	
10	45 + (I-1)*36 - 48 + (I-1)*36	JATC: microseconds - start time of version entry	
11	49 + (I-1)*36 - 56 + (I-1)*36	CCB version number for version entry	
12	57 + (I-1)*36 - 64 + (I-1)*36	cycle number for version entry	

REPEAT ITEMS 7 - 12 FOR I = 1 to number time/version entries (Item 6)

Note: If there are no changes in version/cycle for the virtual file, this number will be zero and no time/version entries will follow.

D.2.3.1 Continuation File Label Record Format

A continuation file label record is present only when the type of data time period field in the file label record (see Table D-2) indicates that the file covers a virtual time period. Table D-4 describes the format of the continuation file label record. The UCSS data transfer software creates this continuation record in order to identify the CCB version and cycle numbers of the source files from which the Level 0 file was generated. The number of version entries in the record is determined by the number of changes in the CCB version and cycle numbers of the source files. Each version entry in this record defines the version number for a specific time range. The time in the version entry specifies the start time of the range and the time of the next version entry specifies the start time of the next range.

D.2.4 DATA RECORD HEADER INFORMATION

The data record header format is presented in Table D-5. This information is contained in the first 64 bytes of the record. Of these 64 bytes, 4 bytes are spare. The record header information pertains to the EMAF from which the associated data words were extracted.

Table D-5. Level 0 Data Record Header

ITEM NO.	BYTES	NAME	TYPE	COMMENTS
1	1 - 2	instrument data set #	I	see Table D-1
2	3 - 4	record type	I	see Note 1
3	5 - 8	physical record count	I	
4	9 - 10	16-bit SMIF Count - begin EMAF	I	
5	11 - 12	ATC - Epoch year	I	
6	13 - 20	ATC - 0.5 msec count	I	
7	21 - 22	JATC:year - begin EMAF	I	year
8	23 - 24	JATC:day - begin EMAF	I	day
9	25 - 28	JATC:msec - begin EMAF	I	millisecond
10	29 - 30	JATC:usec - begin EMAF	I	microsecond
11	31 - 32	# of SMIFs of fill	I	
12	33 - 34	# of SMIFs with bad sync	I	
13	35 - 36	# of SMIFs with CRC error	I	
14	37 - 38	FLAG - EMAF gap	I	see Note 2
15	39 - 40	FLAG - abnormal ATC increment	I	see Note 3
16	41 - 44	EMAF rate (msec/EMAF)	I	
17	45 - 48	spare		
18	49 - 56	SMAF fill flags	bit	see Note 4
19	57 - 64	SMAF parity flags	bit	see Note 5

NOTES:

1. Identifies record type as follows:
   1 = data record, SMAFs 0 to 31
   2 = data record, SMAFs 32 to 63
   3 = data record, SMAFs 0 to 63

2. Interpret as follows:
   0 = "no gap"
   1 = "current EMAF follows a gap"

3. Interpret as follows:
   0 = "normal ATC increment from last EMAF"
   1 = "abnormal ATC increment"

4. 1 bit for each SMAF in the EMAF. Interpret as follows:
   0 = "all SMIFs in SMAF contain data"
   1 = "1 or more SMIFs contain fill"

5. 1 bit for each SMAF in the EMAF. Interpret as follows:
   0 = "all SMIFs in SMAF have good CRC"
   1 = "1 or more SMIFs have CRC errors or contain fill"

The comments made in Section D.2.2 on the SMIF counter, ATC, and JATC are applicable to Items 4 to 10 of the data record header. The ATC and the JATC are the first of the EMAF; the 16 bit SMIF counter is taken from the first SMIF of the EMAF.

D.2.5 DATA RECORD BODY

The body of the data record contains the telemetry from one of the instruments, the subcommutated engineering telemetry, the OBC telemetry, the spacecraft telemetry, or the detailed quality information for one EMAF. The detailed data record formats for each of these types of files are presented in Table D-6 to D-20.

Table D-6. CLAES Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #32,SMIF=I,SMAF=J	1	0 + 64 + 12 * (I + 32 * J)
3	WORD #33,SMIF=I,SMAF=J	1	1 + 64 + 12 * (I + 32 * J)
4	WORD #34,SMIF=I,SMAF=J	1	2 + 64 + 12 * (I + 32 * J)
5	WORD #35,SMIF=I,SMAF=J	1	3 + 64 + 12 * (I + 32 * J)
6	WORD #36,SMIF=I,SMAF=J	1	4 + 64 + 12 * (I + 32 * J)
7	WORD #37,SMIF=I,SMAF=J	1	5 + 64 + 12 * (I + 32 * J)
8	WORD #38,SMIF=I,SMAF=J	1	6 + 64 + 12 * (I + 32 * J)
9	WORD #39,SMIF=I,SMAF=J	1	7 + 64 + 12 * (I + 32 * J)
10	WORD #40,SMIF=I,SMAF=J	1	8 + 64 + 12 * (I + 32 * J)
11	WORD #41,SMIF=I,SMAF=J	1	9 + 64 + 12 * (I + 32 * J)
12	WORD #42,SMIF=I,SMAF=J	1	10 + 64 + 12 * (I + 32 * J)
13	WORD #43,SMIF=I,SMAF=J	1	11 + 64 + 12 * (I + 32 * J)

REPEAT ITEM NO.'S 2-13 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 24640

Table D-7. HALOE Level 0 Data Record

HALOE RECORD TYPE #1

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #44,SMIF=I,SMAF=J	1	0 + 64 + 16 * (I + 32 * J)
3	WORD #45,SMIF=I,SMAF=J	1	1 + 64 + 16 * (I + 32 * J)
4	WORD #46,SMIF=I,SMAF=J	1	2 + 64 + 16 * (I + 32 * J)
5	WORD #47,SMIF=I,SMAF=J	1	3 + 64 + 16 * (I + 32 * J)
6	WORD #48,SMIF=I,SMAF=J	1	4 + 64 + 16 * (I + 32 * J)
7	WORD #49,SMIF=I,SMAF=J	1	5 + 64 + 16 * (I + 32 * J)
8	WORD #50,SMIF=I,SMAF=J	1	6 + 64 + 16 * (I + 32 * J)
9	WORD #51,SMIF=I,SMAF=J	1	7 + 64 + 16 * (I + 32 * J)
10	WORD #52,SMIF=I,SMAF=J	1	8 + 64 + 16 * (I + 32 * J)
11	WORD #53,SMIF=I,SMAF=J	1	9 + 64 + 16 * (I + 32 * J)
12	WORD #54,SMIF=I,SMAF=J	1	10 + 64 + 16 * (I + 32 * J)
13	WORD #55,SMIF=I,SMAF=J	1	11 + 64 + 16 * (I + 32 * J)
14	WORD #56,SMIF=I,SMAF=J	1	12 + 64 + 16 * (I + 32 * J)
15	WORD #57,SMIF=I,SMAF=J	1	13 + 64 + 16 * (I + 32 * J)
16	WORD #58,SMIF=I,SMAF=J	1	14 + 64 + 16 * (I + 32 * J)
17	WORD #59,SMIF=I,SMAF=J	1	15 + 64 + 16 * (I + 32 * J)

REPEAT ITEM NO.'S 2-17 FOR J := 0 TO 31
AND FOR I := 0 TO 31

Total Record Length (bytes): 16448

HALOE RECORD TYPE #2

HALOE RECORD TYPE #2 IS IDENTICAL TO HALOE RECORD TYPE #1 WITH THE FOLLOWING EXCEPTIONS:

- THE RECORD HEADER CONTENT CHANGES AS FOLLOWS:
  - -- VALUE FOR "RECORD TYPE" CHANGES FROM 1 TO 2
  - -- THE PHYSICAL RECORD COUNT INCREMENTS
- THE RANGE OF THE LOOP ON J BECOMES "32 TO 63"

Table D-8. HRDI Level 0 Data Record

HRDI RECORD TYPE #1

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #60,SMIF=I,SMAF=J	1	0 + 64 + 19 * (I + 32 * J)
3	WORD #61,SMIF=I,SMAF=J	1	1 + 64 + 19 * (I + 32 * J)
4	WORD #62,SMIF=I,SMAF=J	1	2 + 64 + 19 * (I + 32 * J)
5	WORD #63,SMIF=I,SMAF=J	1	3 + 64 + 19 * (I + 32 * J)
6	WORD #64,SMIF=I,SMAF=J	1	4 + 64 + 19 * (I + 32 * J)
7	WORD #65,SMIF=I,SMAF=J	1	5 + 64 + 19 * (I + 32 * J)
8	WORD #66,SMIF=I,SMAF=J	1	6 + 64 + 19 * (I + 32 * J)
9	WORD #67,SMIF=I,SMAF=J	1	7 + 64 + 19 * (I + 32 * J)
10	WORD #68,SMIF=I,SMAF=J	1	8 + 64 + 19 * (I + 32 * J)
11	WORD #69,SMIF=I,SMAF=J	1	9 + 64 + 19 * (I + 32 * J)
12	WORD #70,SMIF=I,SMAF=J	1	10 + 64 + 19 * (I + 32 * J)
13	WORD #71,SMIF=I,SMAF=J	1	11 + 64 + 19 * (I + 32 * J)
14	WORD #72,SMIF=I,SMAF=J	1	12 + 64 + 19 * (I + 32 * J)
15	WORD #73,SMIF=I,SMAF=J	1	13 + 64 + 19 * (I + 32 * J)
16	WORD #74,SMIF=I,SMAF=J	1	14 + 64 + 19 * (I + 32 * J)
17	WORD #75,SMIF=I,SMAF=J	1	15 + 64 + 19 * (I + 32 * J)
18	WORD #76,SMIF=I,SMAF=J	1	16 + 64 + 19 * (I + 32 * J)
19	WORD #77,SMIF=I,SMAF=J	1	17 + 64 + 19 * (I + 32 * J)
20	WORD #78,SMIF=I,SMAF=J	1	18 + 64 + 19 * (I + 32 * J)

REPEAT ITEM NO.'S 2-20 FOR J := 0 TO 31
AND FOR I := 0 TO 31

Total Record Length (bytes): 19520

HRDI RECORD TYPE #2

HRDI RECORD TYPE #2 IS IDENTICAL TO HRDI RECORD TYPE #1 WITH THE FOLLOWING EXCEPTIONS:

- THE RECORD HEADER CONTENT CHANGES AS FOLLOWS:
  - -- VALUE FOR "RECORD TYPE" CHANGES FROM 1 TO 2
  - -- THE PHYSICAL RECORD COUNT INCREMENTS
- THE RANGE OF THE LOOP ON J BECOMES "32 TO 63"

Table D-9. ISAMS Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #80,SMIF=I,SMAF=J	1	0 + 64 + 4 * (I + 32 * J)
3	WORD #81,SMIF=I,SMAF=J	1	1 + 64 + 4 * (I + 32 * J)
4	WORD #82,SMIF=I,SMAF=J	1	2 + 64 + 4 * (I + 32 * J)
5	WORD #83,SMIF=I,SMAF=J	1	3 + 64 + 4 * (I + 32 * J)

REPEAT ITEM NO.'S 2-5 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 8256

Table D-10. MLS Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #84,SMIF=I,SMAF=J	1	0 + 64 + 5 * (I + 32 * J)
3	WORD #85,SMIF=I,SMAF=J	1	1 + 64 + 5 * (I + 32 * J)
4	WORD #86,SMIF=I,SMAF=J	1	2 + 64 + 5 * (I + 32 * J)
5	WORD #87,SMIF=I,SMAF=J	1	3 + 64 + 5 * (I + 32 * J)
6	WORD #88,SMIF=I,SMAF=J	1	4 + 64 + 5 * (I + 32 * J)

REPEAT ITEM NO.'S 2-6 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 10304

Table D-11. PEM Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #92,SMIF=I,SMAF=J	1	0 + 64 + 14 * (I + 32 * J)
3	WORD #93,SMIF=I,SMAF=J	1	1 + 64 + 14 * (I + 32 * J)
4	WORD #94,SMIF=I,SMAF=J	1	2 + 64 + 14 * (I + 32 * J)
5	WORD #95,SMIF=I,SMAF=J	1	3 + 64 + 14 * (I + 32 * J)
6	WORD #96,SMIF=I,SMAF=J	1	4 + 64 + 14 * (I + 32 * J)
7	WORD #97,SMIF=I,SMAF=J	1	5 + 64 + 14 * (I + 32 * J)
8	WORD #98,SMIF=I,SMAF=J	1	6 + 64 + 14 * (I + 32 * J)
9	WORD #99,SMIF=I,SMAF=J	1	7 + 64 + 14 * (I + 32 * J)
10	WORD #100,SMIF=I,SMAF=J	1	8 + 64 + 14 * (I + 32 * J)
11	WORD #101,SMIF=I,SMAF=J	1	9 + 64 + 14 * (I + 32 * J)
12	WORD #102,SMIF=I,SMAF=J	1	10 + 64 + 14 * (I + 32 * J)
13	WORD #103,SMIF=I,SMAF=J	1	11 + 64 + 14 * (I + 32 * J)
14	WORD #104,SMIF=I,SMAF=J	1	12 + 64 + 14 * (I + 32 * J)
15	WORD #105,SMIF=I,SMAF=J	1	13 + 64 + 14 * (I + 32 * J)

REPEAT ITEM NO.'S 2-15 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 28736

Table D-12. SOLSTICE Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #106,SMIF=I,SMAF=J	1	0 + 64 + 1 * (I + 32 * J)

REPEAT ITEM NO. 2 FOR J := 0 TO 63
AND FOR I := 0 TO 31

3	FILL	420	2112

Total Record Length (bytes): 2532

Table D-13. SUSIM "A" Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #108,SMIF=I,SMAF=J	1	0 + 64 + 4 * (I + 32 * J)
3	WORD #110,SMIF=I,SMAF=J	1	1 + 64 + 4 * (I + 32 * J)
4	WORD #112,SMIF=I,SMAF=J	1	2 + 64 + 4 * (I + 32 * J)
5	WORD #114,SMIF=I,SMAF=J	1	3 + 64 + 4 * (I + 32 * J)

REPEAT ITEM NO.'S 2-5 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 8256

Table D-14. SUSIM "B" Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #109,SMIF=I,SMAF=J	1	0 + 64 + 4 * (I + 32 * J)
3	WORD #111,SMIF=I,SMAF=J	1	1 + 64 + 4 * (I + 32 * J)
4	WORD #113,SMIF=I,SMAF=J	1	2 + 64 + 4 * (I + 32 * J)
5	WORD #115,SMIF=I,SMAF=J	1	3 + 64 + 4 * (I + 32 * J)

REPEAT ITEM NO.'S 2-5 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 8256

Table D-15. WINDII Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #116,SMIF=I,SMAF=J	1	0 + 64 + 8 * (I + 32 * J)
3	WORD #117,SMIF=I,SMAF=J	1	1 + 64 + 8 * (I + 32 * J)
4	WORD #118,SMIF=I,SMAF=J	1	2 + 64 + 8 * (I + 32 * J)
5	WORD #119,SMIF=I,SMAF=J	1	3 + 64 + 8 * (I + 32 * J)
6	WORD #120,SMIF=I,SMAF=J	1	4 + 64 + 8 * (I + 32 * J)
7	WORD #121,SMIF=I,SMAF=J	1	5 + 64 + 8 * (I + 32 * J)
8	WORD #122,SMIF=I,SMAF=J	1	6 + 64 + 8 * (I + 32 * J)
9	WORD #123,SMIF=I,SMAF=J	1	7 + 64 + 8 * (I + 32 * J)

REPEAT ITEM NO.'S 2-9 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 16448

Table D-16. ACRIM Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #28,SMIF=I,SMAF=J	1	0 + 64 + 2 * (I + 32 * J)
3	WORD #29,SMIF=I,SMAF=J	1	1 + 64 + 2 * (I + 32 * J)

REPEAT ITEM NO.'S 2 & 3 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 4160

Table D-17. Engineering Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #8,SMIF=I,SMAF=J	1	0 + 64 + 4 * (I + 32 * J)
3	WORD #9,SMIF=I,SMAF=J	1	1 + 64 + 4 * (I + 32 * J)
4	WORD #10,SMIF=I,SMAF=J	1	2 + 64 + 4 * (I + 32 * J)
5	WORD #11,SMIF=I,SMAF=J	1	3 + 64 + 4 * (I + 32 * J)

REPEAT ITEM NO.'S 2-5 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 8256

Table D-18. Spacecraft Level 0 Data Record

SPACECRAFT RECORD TYPE #1

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #3,SMIF=I,SMAF=J	1	0 + 64 + 21 * (I + 32 * J)
3	WORD #6,SMIF=I,SMAF=J	1	1 + 64 + 21 * (I + 32 * J)
4	WORD #7,SMIF=I,SMAF=J	1	2 + 64 + 21 * (I + 32 * J)
5	WORD #19,SMIF=I,SMAF=J	1	3 + 64 + 21 * (I + 32 * J)
6	WORD #20,SMIF=I,SMAF=J	1	4 + 64 + 21 * (I + 32 * J)
7	WORD #21,SMIF=I,SMAF=J	1	5 + 64 + 21 * (I + 32 * J)
8	WORD #22,SMIF=I,SMAF=J	1	6 + 64 + 21 * (I + 32 * J)
9	WORD #23,SMIF=I,SMAF=J	1	7 + 64 + 21 * (I + 32 * J)
10	WORD #24,SMIF=I,SMAF=J	1	8 + 64 + 21 * (I + 32 * J)
11	WORD #25,SMIF=I,SMAF=J	1	9 + 64 + 21 * (I + 32 * J)
12	WORD #26,SMIF=I,SMAF=J	1	10 + 64 + 21 * (I + 32 * J)
13	WORD #27,SMIF=I,SMAF=J	1	11 + 64 + 21 * (I + 32 * J)
14	WORD #30,SMIF=I,SMAF=J	1	12 + 64 + 21 * (I + 32 * J)
15	WORD #31,SMIF=I,SMAF=J	1	13 + 64 + 21 * (I + 32 * J)
16	WORD #79,SMIF=I,SMAF=J	1	14 + 64 + 21 * (I + 32 * J)
17	WORD #89,SMIF=I,SMAF=J	1	15 + 64 + 21 * (I + 32 * J)
18	WORD #90,SMIF=I,SMAF=J	1	16 + 64 + 21 * (I + 32 * J)
19	WORD #91,SMIF=I,SMAF=J	1	17 + 64 + 21 * (I + 32 * J)
20	WORD #107,SMIF=I,SMAF=J	1	18 + 64 + 21 * (I + 32 * J)
21	WORD #124,SMIF=I,SMAF=J	1	19 + 64 + 21 * (I + 32 * J)
22	WORD #125,SMIF=I,SMAF=J	1	20 + 64 + 21 * (I + 32 * J)

REPEAT ITEM NO.'S 2-20 FOR J := 0 TO 31
AND FOR I := 0 TO 31

Total Record Length (bytes): 21568

SPACECRAFT RECORD TYPE #2

SPACECRAFT RECORD TYPE #2 IS IDENTICAL TO SPACECRAFT RECORD TYPE #1 WITH THE FOLLOWING EXCEPTIONS:

- THE RECORD HEADER CONTENT CHANGES AS FOLLOWS:
  - -- VALUE FOR "RECORD TYPE" CHANGES FROM 1 TO 2
  - -- THE PHYSICAL RECORD COUNT INCREMENTS
- THE RANGE OF THE LOOP ON J BECOMES "32 TO 63"

Table D-19. OBC Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	WORD #12,SMIF=I,SMAF=J	1	0 + 64 + 7 * (I + 32 * J)
3	WORD #13,SMIF=I,SMAF=J	1	1 + 64 + 7 * (I + 32 * J)
4	WORD #14,SMIF=I,SMAF=J	1	2 + 64 + 7 * (I + 32 * J)
5	WORD #15,SMIF=I,SMAF=J	1	3 + 64 + 7 * (I + 32 * J)
6	WORD #16,SMIF=I,SMAF=J	1	4 + 64 + 7 * (I + 32 * J)
7	WORD #17,SMIF=I,SMAF=J	1	5 + 64 + 7 * (I + 32 * J)
8	WORD #18,SMIF=I,SMAF=J	1	6 + 64 + 7 * (I + 32 * J)

REPEAT ITEM NO.'S 2-8 FOR J := 0 TO 63
AND FOR I := 0 TO 31

Total Record Length (bytes): 14400

Table D-20. Quality Level 0 Data Record

ITEM NO.	DESCRIPTION	LENGTH	OFFSET
1	RECORD HEADER	64	0
2	SMIF FILL SMAF=J,SMIF=0 TO 7	1	0 + 64 + 4 * J
3	SMIF FILL SMAF=J,SMIF=8 TO 15	1	1 + 64 + 4 * J
4	SMIF FILL SMAF=J,SMIF=16 TO 23	1	2 + 64 + 4 * J
5	SMIF FILL SMAF=J,SMIF=24 TO 31	1	3 + 64 + 4 * J
	REPEAT ITEM NO.'S 2-5	FOR J := 0 TO	63
6	SMIF CRC SMAF=J,SMIF=0 TO 7	1	0 + 64 + 256 + 4 * J
7	SMIF CRC SMAF=J,SMIF=8 TO 15	1	1 + 64 + 256 + 4 * J
8	SMIF CRC SMAF=J,SMIF=16 TO 23	1	2 + 64 + 256 + 4 * J
9	SMIF CRC SMAF=J,SMIF=24 TO 31	1	3 + 64 + 256 + 4 * J
	REPEAT ITEM NO.'S 6-9	FOR J := 0 TO	63
10	FILL	1956	576
	Total Record Length (bytes):		2532

NOTE:

- Each bit of a SMIF Fill byte corresponds to a SMIF as described above and is interpreted as follows:
  0 = "the corresponding SMIF contains data"
  1 = "the corresponding SMIF is all fill data"

- Each bit of a SMIF CRC byte corresponds to a SMIF as described above and is interpreted as follows:
  0 = "the corresponding SMIF has a good CRC"
  1 = "the corresponding SMIF has a bad CRC or is all fill data"

The first item of the data record formats as shown in Tables D-6 to D-20 is the 64 byte record header starting at byte 0 of the record. Each subsequent item in the tables account for one of the telemetry words assigned to that instrument, engineering, OBC, quality, or spacecraft data. The location of the telemetry word in the record is given in terms of an offset and a length.

For example, a given instrument may be assigned 12 words of telemetry per SMIF. One of the words of telemetry contained in SMIF i of SMAF j is stored in a one byte location in the record, with an offset from the beginning of the record specified by the "offset" field for the word. The offset value accounts for the number of words

preceding the desired word in the SMIF, the 64 byte record header, and the product of the number of SMIFs preceding SMIF i of SMAF j with the number of words per SMIF, 12 words in this case.

D.2.6 MULTIPART RECORDS

All logical records are intended to contain one EMAF of data, each of a specific type. As mentioned above, certain logical record types (HALOE, HRDI and Spacecraft) consist of two physical records. These record types are indicated below as record type 1 or record type 2, the first record type carrying the first 32 SMAFs of the EMAF, and the second record type carrying the last 32. In these cases, the type 1 and type 2 records are interleaved, record type 1 occurring first followed by record type 2.

D.3 ABSOLUTE TIME CODE (ATC) JUMPS AND SPLIT EMAFS

The time that appears in the EMAF header is based on the Absolute Time Code (ATC) that appears 16 times in each EMAF. It is corrected such that the first bit of the EMAF has as its timetag the EMAF header time.

The ATCs within the EMAF increment throughout the EMAF and, nominally, there is a 65536 millisecond difference between two successive EMAF header times. ATC drift management appears as an occurrence of a difference of 65536.5 milliseconds rather than the nominal difference of 65536 milliseconds between successive EMAF header times. If the caller does not examine the microsecond of ATC field in the EMAF header, then differences of 65537 are seen interspersed within groups of the nominal 65536 differences. A clock jump is an unanticipated change in the value of the ATC as it varies through the EMAF

The DCF handles ATC (or clock) jumps as follows:

The EMAF in which the jump occurs is split into two EMAFs. The first EMAF contains the timetag (the EMAF header time) associated with the original ATC stream. The second EMAF contains a timetag associated with the changed ATC stream. The former EMAF contains data up to the point of the time jump. The latter contains data beginning at that point until the end of the EMAF.

If the jump is forward; i.e., the ATC value increments more than expected between two adjacently reported times, then the timetag of the first EMAF has a value less than that of the second EMAF. If the jump is backward, then the timetag of the first EMAF is greater than the timetag of the second EMAF. Reading sequentially, the EMAF times are out of order in this 'backward jump' condition.

APPENDIX E

LEVEL 3 FILE FORMATS

It is intended that all UARS scientific instrument data be stored in one or more of the common file formats at Level 3. These Level 3 file formats are referred to as 3AT (time referenced), 3AL (latitude referenced), or 3AS/3BS (solar data). The access to files in these common formats is achieved by use of certain of the UCSS services.

E.1 GENERAL COMMENTS

As with all UARS scientific instrument data, Level 3 data is maintained in files containing data from one instrument for one UARS day. In addition, at Level 3, a file contains data for only one parameter or species.

E.1.1 LEVEL 3AT DATA

A Level 3AT file consists of a time-ordered collection of data records. Each record contains a single array of data values of one parameter or species type for a specific time. The data array is organized according to the rules of the UARS standard data array (see Section E.2). The reference time values at which Level 3AT records are created are common across all Level 3AT files from all instruments. The Level 3AT data record time is the time associated with SMIF 0 of SMAF 32 of the EMAF at Level 0.

The Level 3AT files are stored as flat files without any index structure. All records of a given file are of the same length.

The actual record length is dependent upon the maximum number of data points that can be stored in the data records.

If the file is a virtual file, the label record may be followed by one or more continuation file label records. The remaining records in the file are data records.

Level 3AT files are generated by the HALOE, MLS, ISAMS, CLAES, HRDI, PEM, and WINDII instrument investigations.

E.1.2 LEVEL 3AL DATA

A Level 3AL data file consists of a collection of profiles of atmospheric data that have been indexed by the latitude and time values associated with the profiles. Each record of the Level 3AL file contains a single array of data values for one parameter or species type for a specific time. The data array is organized according to the rules of the UARS standard data array (see Section E.2). The index key for the record is based on the concatenation of the latitude and time values associated with the profile. The standard latitude values at which Level 3AL records may be written are from -88.0 degrees to +88.0 degrees latitude in 4.0 degree increments. There is no standard time rule that applies to the Level 3AL profiles.

All records of a given file are of the same length. The actual record length is dependent upon the maximum number of data points that can be stored in the data records.

If the file is a virtual file, the label record may be followed by one or more continuation file label records. The remaining records in the file are data records.

Level 3AL files are generated by the MLS, ISAMS, CLAES, HRDI, PEM, and WINDII instrument investigations.

E.1.3 LEVEL 3AS/3BS DATA

A Level 3AS/3BS file contains a single data record for each UARS day. Each data record contains a single array representing a daily mean solar spectrum.

Additional information will also be stored in the record via a parameter array. Included in this information will be the irradiance values for 4 coronal lines, Lyman Alpha, a magnesium line, a calcium line, and the mean solar distance.

The Level 3S/3BS files are stored as flat files without any index structure. All records of a given file are of the same length. The actual record length is dependent upon the maximum number of data points that can be stored in the data record.

Level 3AS/3BS files are generated by the SUSIM and SOLSTICE instrument investigations.

E.1.4 LEVEL 3A PARAMETER FILES

A Level 3A Parameter File provides a means of associating parameters with Level 3 data. The parameters are defined by each Principal Investigator (PI) for his/her own Level 3 data. Level 3 Parameter Files will contain information describing the context of the Level 3 data with each Level 3 data record associated with a corresponding parameter file record.

Level 3A Parameter Files are identified by their own distinct level. The levels used to identify Level 3A parameter files are Level 3TP which refers to time ordered parameter files, and Level 3LP which refers to parameter files indexed by both latitude and time value. Level 3TP files have the same organization as the Level 3AT files (see Section E.1.1). Level 3LP files have the same organization as the Level 3AL file (see Section E.1.2).

E.2 UARS STANDARD DATA ARRAY

The UARS standard data array is the common data structure used for storing UARS data so that it can be accessed and interpreted properly by the entire UARS community. Since the UARS instruments are not all performing the same type of measurements, the interpretation of this standard data array is instrument dependent. The position of a data value within the standard array for a given instrument has a fixed meaning.

It should be noted that when a Level 3AT, 3TP, 3AL, 3LP, 3AS, or 3BS file is created, the full size of the UARS standard data array may not be required. In this case, only the values required are stored and the starting index for the first stored data point relative to the full UARS standard data array is stored with it.

E.2.1 PRESSURE REFERENCED ARRAY

The index into the data array may correspond to standard pressure levels. These standard pressure level values in millibars are given by:

$$P(i) = 1000.0 * (10^{**}(-i/6)) , \qquad i = 0,1, \ldots 35.$$

The CLAES, HRDI, ISAMS, MLS, HALOE and WINDII instrument investigations are expected to use pressure referenced data arrays.

E.2.2 ALTITUDE REFERENCED ARRAY

The index into the data array may correspond to standard altitude levels. These standard altitude level values in kilometers are given by:

```
Z(i) = 5 * i ,                      i <= 12
Z(i) = 60 + (i - 12) * 3 ,          13 <= i <= 32
Z(i) = 120 + (i - 32) * 5 ,         33 <= i <= 88.
```

The HRDI, PEM, and WINDII instrument investigations are expected to use altitude referenced data arrays. The HRDI and WINDII instrument investigations are expected to produce both pressure and altitude referenced data arrays for both Level 3AT and 3AL data. To distinguish between the pressure referenced and altitude referenced data for the same species at the same data level, it will be necessary to include additional descriptive information with the SUBTYPE name for the data file. For example, a pressure referenced temperature profile may have the SUBTYPE name of "TEMP_P", and an altitude referenced wind component profile may have the SUBTYPE name of "ZONWIN1_Z".

E.2.3 WAVELENGTH REFERENCED ARRAY

The index into the standard data array may correspond to standard wavelength values. Each element of the array is associated with a 1.0 nanometer (nm) interval centered on the half nm from 115 nm to 425 nm. Each element of the data array contains the averaged set of observations for the wavelength bin associated with it.

The SUSIM and SOLSTICE instrument investigations use the wavelength referenced data array.

E.3 LEVEL 3 FILE FORMAT

The following sections provide a description of the file format for the Level 3 files.

E.3.1 SFDU STANDARD INFORMATION

The Level 3AT, 3TP, 3AS, and 3BS files are constructed so as to adhere to the Standard Formatted Data Unit (SFDU) structure and construction rules (Reference 10). Level 3AT/3TP and 3AS/3BS data are stored in this format at the level of a single file. That is, the descriptor records that make these files consistent with the SFDU standard are analogous to an envelope; the "letter" contained within the envelope is a file. Other SFDU construction schemes are possible,

but this is the approach selected by the UARS Science Team.

The following paragraphs define the descriptor record ("envelope") that is required by the SFDU construction rules. This is followed by the definition of the UARS specific records ("letter") that make up the Level 3AT, 3TP, 3AS, or 3BS files.

The SFDU standards for UARS Level 3 data specify that the first 40 bytes of the file should contain header information that identifies the file as an SFDU-formatted file and that "points" to detailed file and record structure documentation. For Level 3AT, 3TP, 3AS, and 3BS files this information appears as the first 40 bytes in the first record of the file. However, for Level 3AL or 3TP files, because the files are indexed, the required 40 bytes of SFDU information will appear in one record with 20 bytes of record index data preceding it.

It should be noted that as long as the UCSS Level 3AT, 3AL, 3TP, 3LP, 3AS, or 3BS read and write routines are used, either in production processing or using simulated services at the RAC, the user need not concern himself with the SFDU header information.

E.3.2 SFDU DESCRIPTOR FORMATS FOR LEVEL 3AT/3TP AND 3AS/3BS FILES

The SFDU construction rules require that at least two Type, Length, Value (TLV) objects be used to construct a file. In general, the Type or T field contains information that can be used to properly interpret the contents of the V field; the L field is the length of the V field in bytes. The first TLV is referred to as a type Z object, the T[Z] field identifying the file as SFDU compliant; L[Z] is the length of V[Z] in bytes. In the case of Level 3 data, the V[Z] field is the second TLV object and is referred to as a type I TLV object. The T[I] field identifies the file as a product of the UARS Program; L[I] is the length of V[I] in bytes. The V[I] field is the "letter" containing the UARS specific Level 3 file information.

The first record in a Level 3AT, 3TP, 3AS, or 3BS file contains 20 bytes of T[Z] and L[Z] information followed by 20 bytes of T[I] and L[I] information. The format of these fields is described in Tables E-1 and Table E-2.

Table E-1. SFDU T[Z] and L[Z] Format for Level 3AT/3TP or Level 3AS/3BS Files

ITEM NO.	FIELD NAME	BYTE	SUBFIELD NAME	COMMENTS
1	TYPE	0-3	control authority identifier	"CCSD"
2	TYPE	4	version identifier	"1"
3	TYPE	5	class identifier	"Z"
4	TYPE	6-7	spare	"00"
5	TYPE	8-11	data descriptive record identifier	"0001"
6	LENGTH	12-19	length	see Note

Note: The length field will contain a number in ASCII format binary number specifying the length in bytes of the corresponding VALUE field. The VALUE fields includes the T[I] and L[I] fields as well as the V[I] field which is the UARS Level 3 file.

Table E-2. SFDU T[I] and L[I] Format for Level 3AT/3TP or Level 3AS/3BS Files

ITEM NO.	FIELD NAME	BYTE	SUBFIELD NAME	COMMENTS
1	TYPE	0-3	control authority identifier	see Note 1
2	TYPE	4	version identifier	"1"
3	TYPE	5	class identifier	"I"
4	TYPE	6-7	spare	"00"
5	TYPE	8-11	data descriptive record identifier	see Note 2
6	LENGTH	12-19	length	see Note 3

Notes:

1 The control authority for the UARS data is 'ZURS'.

2 The data description record for UARS is TBD.

3 The length field will contain a number in ASCII format specifying the length of the V[I]field, which is the UARS Level 3 file.

E.3.3 FILE LABEL RECORD FOR LEVEL 3AT/3TP FILES

The file label record format for Level 3AT and 3TP files is presented in Table E-3. All file label record fields are ASCII fields. The file label record is padded with zero fill when the data record size exceeds the file label record size.

Table E-3 Label Record Format for Level 3AT/3TP Files (1 of 2)

ITEM NO.	BYTE OFFSET	FIELD NAME	COMMENTS
1	0	satellite identifier	'UARS'
2	4	record type	' 1'
3	6	instrument identifier	
4	18	data subtype or species	
5	30	format version number	' 1'
6	34	physical record count	' 1'
7	42	number of continuation records for file label	
8	46	number of physical records in file	
9	54	file creation time in VAX VMS ASCII format	
10	77	year(3 digits) for first data record	
11	80	day of year for first data record	
12	83	milliseconds of day for first data record	
13	91	year(3 digits) for last data record	
14	94	day of year for last data record	
15	97	milliseconds of day for last data record	
16	105	data level	
17	108	UARS day number	
18	112	number of data points per record (3AT) number of 32-bit words (3TP)	
19	116	base index of data point values	see Note 1
20	120	record length in bytes	see Note 2
21	125	CCB version number	
22	134	file cycle number	see Note 3
23	139	virtual file flag	see Note 4
24	140	total number of time/version entries in file	see Note 5

Table E-3 Label Record Format for Level 3AT/3TP Files (Cont.)

ITEM	OFFSET	FIELD NAME	COMMENTS
25	144	number of time/version entries in record	
26	148	year for first version entry	
27	151	day of year for first version entry	
28	154	milliseconds of day for first version entry	
29	162	version number of first version entry	
30	171	cycle number of first version entry	
		.	
		.	
	B	year for nth version entry	
	B+3	day of year for nth version entry	
	B+6	milliseconds of day for nth version entry	
	B+14	version number of nth version entry	
	B+23	cycle number of nth version entry	
	B+28	.	

Legend: B = 148 + 28*(n - 1) n = 1, 2, 3, ...

Notes:

1 Not applicable for Level 3TP records.

2 Minimum record size is 148 bytes.

3 Supplied only during file creation via RAC data transfer.

4 ' ' = physical file
'V' = virtual file created via RAC data transfer

5 There is a time/version entry for each consecutive change in the version number of the source files used to produce this file. Only used for virtual files created via RAC data transfer.

E.3.4 CONTINUATION LABEL RECORD FOR LEVEL 3AT/3TP AND 3AS/3BS FILES

The continuation label record format is presented in Table E-4. All fields in this record are in ASCII format. This record is present only when the file label record indicates that the file is a virtual file created via the RAC transfer services and there is insufficient space in the file label record for all the time/version entries needed.

Table E-4. Continuation Label Record Format for Level 3AT/3TP or Level 3AS/3BS Files

ITEM NO.	BYTE OFFSET	FIELD NAME	COMMENTS
1	0	satellite identifier	'UARS'
2	4	record type	' 2'
3	6	instrument identifier	
4	18	data subtype or species	
5	30	format version number	' 1'
6	34	physical record count	' 1'
7	42	number of time/version entries in record	
8	46	spare	
9	48	start year for first version entry	
10	51	start day of year for first version entry	
11	54	start msec of day for first version entry	
12	62	version number of first version entry	
13	71	cycle number of first version entry	
		.	
		.	
	B	start year for nth version entry	
	B+3	start day of year for nth version entry	
	B+6	start msec of day for nth version entry	
	B+14	version number of nth version	
	B+23	cycle number of nth version	

Legend: B = 48 + 28*(n -1) n = 1, 2, ...

E.3.5 DATA RECORD FOR LEVEL 3AT FILES

The data record format for a Level 3AT file is presented in Table E-5. The data record contains data values in the UARS standard data array (see Section E.2) for the time range specified in the

file's label record. A fill value is used to indicate missing data points within a record. This fill value, X'00008000', is a reserved value that is not a valid floating point number. The data record is padded when the file label record size exceeds the data record size.

Table E-5. Data Record Format for Level 3AT or Level 3AS/3BS Files

ITEM NO.	BYTE OFFSET	FIELD NAME	FORMAT	COMMENTS
1	0	satellite identifier	C	'UARS'
2	4	record type	C	' 3'
3	6	instrument identifier	C	
4	18	physical record count	C	
5	26	spare	C	
6	28	total number of points in the record	I	
7	32	number of actual points (np)	I	
8	36	starting index of first actual point	I	
9	40	record time in UDTF format	T	see Note 1
10	48	latitude	R	see Note 2
11	52	longitude	R	see Note 2
12	56	local solar time (LST)	R	see Note 2
13	60	solar zenith angle (SZA)	R	see Note 2
14	64	data value for first point in record	R	
		.		
		.		
		data value for last point in record	R	
	B	quality for first point in record	R	
		.		
		.		
		quality for last point in record	R	

Legend: C = character I = integer R = real
T = time in UDTF format B = 64 + 4*total number of points

Notes:

1 For solar instruments (Level 3AS/3BS file) the milliseconds portion of the UDTF time is 0.

2 Not applicable for solar instruments (Level 3AS/3BS file)

E.3.6 DATA RECORD FOR LEVEL 3TP FILES

The data record format for a Level 3TP file is presented in Table E-6. The data record contains parameter values associated with the corresponding Level 3AT record of the appropriate Level 3AT data file, for the time specified in the Level 3AT record's header.

Table E-6. Data Record Format for a Level 3TP File

ITEM NO.	BYTE OFFSET	FIELD NAME	FORMAT	COMMENTS
1	0	satellite identifier	C	'UARS'
2	4	record type	C	' 3'
3	6	instrument identifier	C	
4	18	physical record count	C	
5	26	spare	C	
6	28	maximum number of 32-bit words in the record	I	
7	32	not used		
8	36	not used		
9	40	record time in UDTF format	T	
10	48	latitude	R	
11	52	longitude	R	
12	56	spare	C	
13	64	number of 32-bit parameter words	I	
14	68	first parameter word	C	
	B	last parameter word	C	

Legend: C = character I = integer R = real
T = time in UDTF format B = 64 + 4*number of parameter words

E.3.7 SFDU DESCRIPTOR FORMATS FOR LEVEL 3AL/3LP FILES

The SFDU descriptor records for Level 3AL or 3LP are constructed in the same manner as for Level 3AT, 3TP, 3AS, or 3BS files with the exception that the SFDU information in the records is preceded by the record index key field.

The first record in a Level 3AL/3LP file contains the record 20 byte key followed by 20 bytes of T[Z] and L[Z] information and 20 bytes of T[I] and L[I] information. The format of these fields is described in Table E7 and Table E8.

Table E-7. SFDU T[Z] and L[Z] Format for Level 3AL/3LP Files

ITEM NO.	FIELD NAME	BYTE	SUBFIELD NAME	COMMENTS
1	KEY	0-19	record key	see Note 1
2	TYPE	20-23	control authority identifier	"CCSD"
3	TYPE	24	version identifier	"1"
4	TYPE	25	class identifier	"Z"
5	TYPE	26-27	spare	"00"
6	TYPE	28-31	data descriptive record identifier	"0001"
7	LENGTH	32-39	length	see Note 2

Notes:

1 The record key has the following structure:

chars	1-4	"1001"
	5-10	blank
	11-12	"0:"
	13-19	blank
	20	"0"

2 The length field will contain a number in ASCII format specifying the length in bytes of the corresponding VALUE field. The VALUE field includes the T[I] and L[I]) fields as well as the V[I] field which is the UARS Level 3AL or 3LP file.

Table E-8. SFDU T[I] and L[I] Format for Level 3AL/3LP Files

ITEM NO.	FIELD NAME	BYTE	SUBFIELD NAME	COMMENTS
1	TYPE	40-43	control authority identifier	see Note 1
2	TYPE	44	version identifier	"2"
3	TYPE	45	class identifier	"I"
4	TYPE	46-47	spare	"00"
5	TYPE	48-51	data descriptive record identifier	see Note 2
6	LENGTH	52-59	length	see Note 3

Notes:

1 The control authority for the UARS data is 'ZURS'.

2 The data description record for UARS is TBD.

3 The length field will contain a binary number specifying the length of the V[I] field, which is the UARS Level 3AL or 3LP file.

E.3.8 FILE LABEL RECORD FOR LEVEL 3AL/3LP FILES

The file label record format is presented in Table E-9. All file label record fields are ASCII fields. The file label record is padded with zero fill when the data record size exceeds the file label record size.

Table E-9. Label Record Format for Level 3AL/3LP Files (1 of 2)

ITEM NO.	BYTE OFFSET	FIELD NAME	COMMENTS
1	0	record key	see Note 1
2	20	satellite identifier	'UARS'
3	24	record type	' 1'
4	26	instrument identifier	
5	38	data subtype or species	
6	50	format version number	
7	54	physical record count	' 1'
8	62	number of continuation records for file label	
9	66	number of physical records in file	
10	74	file creation time in VAX VMS ASCII format	
11	97	year(3 digits) for earliest data record	
12	100	day of year for earliest data record	
13	103	milliseconds of day for earliest data record	
14	111	year(3 digits) for latest data record	
15	114	day of year for latest data record	
16	117	milliseconds of day for latest data record	
17	125	data level	
18	128	UARS day number	
19	132	max. number of data points per record (3AL) max. number of 32-bit words per record (3LP)	
20	136	base index of data point values	see Note 2
21	140	record length in bytes	see Note 3
22	145	minimum latitude for records in file	
23	148	maximum latitude for records in file	
24	151	CCB version number	

Table E-9. Label Record Format for Level 3AL/3LP Files (Cont.)

ITEM	OFFSET	FIELD NAME	COMMENTS
25	160	file cycle number	see Note 4
26	165	virtual file flag	see Note 5
27	166	total number of time/version entries in file	see Note 6
28	170	number of time/version entries in record	
29	174	year for first version entry	
30	177	day of year for first version entry	
31	180	milliseconds of day for first version entry	
32	188	version number of first version entry	
33	197	cycle number of first version entry	
		.	
		.	
	B	year for nth version entry	
	B+3	day of year for nth version entry	
	B+6	milliseconds of day for nth version entry	
	B+14	version number of nth version entry	
	B+23	cycle number of nth version entry	
	B+28	.	

Legend: B = 174 + 28*(n - 1) n = 1, 2, 3, ...

Table E-9. Label Record Format for Level 3AL/3LP Files (Cont.)

Notes:

1 The record key has the following structure:
chars 1-4 '1002'
5-10 blank
11-12 '0:'
13-19 blank
20 '0'

2 Not applicable for Level 3LP files

3 Minimum record size is 174 bytes

4 Supplied only during file creation via RAC data transfer.

5 ' ' = physical file
'V' = virtual file created via RAC data transfer

6 There is a time/version entry for each consecutive change in the version number of the source files used to produce this file. Only used for virtual files created via RAC data transfer.

E.3.9 CONTINUATION LABEL RECORD FOR LEVEL 3AL/3LP FILES

The continuation label record format is presented in Table E-10. All fields in this record are in ASCII format. This record is present only when the file label record indicates that the file is a virtual file and there is insufficient space in the file label record for all the time/version entries needed.

Table E-10. Continuation Label Record Format for Level 3AL/3LP Files

ITEM	OFFSET	FIELD NAME	COMMENTS
1	0	record key	see Note
2	20	satellite identifier	'UARS'
3	24	record type	' 2'
4	26	instrument identifier	
5	38	data subtype or species	
6	50	format version number	
7	54	physical record count	
8	62	number of time/version entries in record	
9	66	spare	
10	68	start year for first version entry	
11	71	start day of year for first version entry	
12	74	start msec of day for first version entry	
13	82	version number of first version entry	
14	91	cycle number of first version entry	
		.	
		.	
	B	start year for nth version entry	
	B+3	start day of year for nth version entry	
	B+6	start msec of day for nth version entry	
	B+14	version number of nth version	
	B+23	cycle number of nth version	

Legend: B = 68 + 28*(n - 1) n = 1, 2, ...

Note: The record key has the following structure:

chars	1-4	(1000 + record number) in ASCII
	5-10	blank
	11-12	':'
	13-19	blank
	20	'0'

E.3.10 DATA RECORD FOR LEVEL 3AL FILES

The data record format is presented in Table E-11. The data record contains data values in the UARS standard data array (see Section E.2) for the latitude and time ranges specified in the file's label record. A fill value is used to indicate missing data points within a record. This fill value, X'00008000', is a reserved value that is not a valid floating point number. The data record is padded when the file label record size exceeds the data record size.

Table E-11. Data Record Format for a Level 3AL File

ITEM NO.	BYTE OFFSET	FIELD NAME	FORMAT	COMMENTS
1	0	record key	C	see Note
2	20	satellite identifier	C	'UARS'
3	24	record type	C	' 3'
4	26	instrument identifier	C	
5	38	physical record count	C	
6	46	spare	C	
7	48	total number of points in the record	I	
8	52	number of actual points (np)	I	
9	56	starting index of first actual point	I	
10	60	record time in UDTF format	T	
11	68	latitude	R	
12	72	longitude	R	
13	76	local solar time (LST)	R	
14	80	solar zenith angle (SZA)	R	
15	84	data value for first point in record	R	
		.		
		.		
		data value for last point in record	R	
	B	quality for first point in record	R	
		.		
		.		
		quality for last point in record	R	

Legend: C = character
I = integer
R = real
T = time in UDTF format
B = 84 + 4*(total number of points)

Note: The record key has the following structure:

chars	
1-4	(1000 + 90 + latitude + 1 + number of records in label) in ASCII
5	blank
6-11	date portion of UDTF record-time in ASCII
12	':'
13-20	millisecond portion of UDTF record-time in ASCII

E.3.11 DATA RECORD FOR LEVEL 3LP FILES

The data record format for a Level 3LP file is presented in Table E-12. The data record contains parameter values associated with the corresponding Level 3AL record of the appropriate Level 3AL data file, for the time specified in the file label record. A fill value of '0' is used where there are no parameter values.

Table E-12. Data Record Format for a Level 3LP File

ITEM NO.	BYTE OFFSET	FIELD NAME	FORMAT	COMMENTS
1	0	record key	C	see Note
2	20	satellite id	C	'UARS'
3	24	record type	C	' 3'
4	26	instrument identifier	C	
5	38	physical record count	C	
6	46	spare	C	
7	48	maximum number of 32-bit words	I	
8	52	not used		
9	56	not used		
10	60	record time in UDTF format	T	
11	68	latitude	R	
12	72	longitude	R	
13	76	spare	C	
14	84	number of 32-bit parameter words	I	
15	88	first parameter word	C	
	B	last parameter word		

Legend: C = character I = integer R = real
T = time in UDTF format B = 84 + 4*number of parameter words

Note: The record key has the following structure:
chars 1-4 (1000 + 90 + latitude + 1 + number of records in label) in ASCII
5 blank
6-11 date portion of UDTF record-time in ASCII
12 ':'
13-20 millisecond portion of UDTF record-time in ASCII

E.3.12 FILE LABEL RECORD FOR LEVEL 3AS/3BS FILES

The file label record format for a Level 3AS or 3BS file is presented in Table E-13. All file label record fields are ASCII fields the file label record is padded with zero fill when the data record size exceeds the file label record size.

Table E-13 Label Record Format for Level 3AS/3BS File (1 of 2)

ITEM NO.	BYTE OFFSET	FIELD NAME	COMMENTS
1	0	satellite identifier	'UARS'
2	4	record type	' 1'
3	6	instrument identifier	
4	18	data subtype or species	
5	30	format version number	
6	34	physical record count	' 1' (7 blanks)
7	42	number of continuation records for file label	
8	46	number of physical records in file	
9	54	file creation time in VAX VMS ASCII format	
10	77	year(3 digits) for first data record	
11	80	day of year for first data record	
12	83	milliseconds of day for first data record	
13	91	year(3 digits) for last data record	
14	94	day of year for last data record	
15	97	milliseconds of day for last data record	
16	105	data level	
17	108	UARS day number	
18	112	number of data points per record	
19	116	base wavelength of data point values	
20	122	record length in bytes	see Note 1
21	127	CCB version number	
22	136	file cycle number	see Note 2
23	141	virtual file flag	see Note 3
24	142	total number of time/version entries in file	see Note 4

Table E-13. Label Record Format for Level 3AS/3BS File (Cont.)

ITEM	OFFSET	FIELD NAME	COMMENTS
25	146	number of time/version entries in record	
26	150	year for first version entry	
27	153	day of year for first version entry	
28	156	milliseconds of day for first version entry	
29	164	version number of first version entry	
30	173	cycle number of first version entry	
		.	
		.	
	B	year for nth version entry	
	B+3	day of year for nth version entry	
	B+6	milliseconds of day for nth version entry	
	B+14	version number of nth version entry	
	B+23 B+28	cycle number of nth version entry	

Legend: B = 150 + 28*(n - 1) n = 1, 2, 3, ...

Notes:

1 Minimum record size is 150 bytes.

2 Supplied only during file creation via RAC data transfer.

3 ' ' = physical file
'V' = virtual file created via RAC data transfer

4 There is a time/version entry for each consecutive change in the version number of the source files used to produce this file. Only used for virtual files created via RAC data transfer.

E.3.13 CONTINUATION LABEL RECORD FOR LEVEL 3AS/3BS FILES

The continuation label record format for Level 3AS and 3BS files is as described in Section E.3.4

E.3.14 DATA RECORD FOR LEVEL 3AS/3BS FILES

The data record format for a Level 3AS or 3BS file is presented in Table E-14. The data record contains data values in the UARS standard data array (see Section E.2) for a specific UARS day. It also contains a parameter array consisting of pairs of parameter names and their corresponding values. The Mean Solar Distance (MSD) parameter MUST be presented in the parameter array for each record. A fill value is used to indicate missing data points within a record. This fill values, X'00008000', is a reserved value that is not a valid floating point number.

Table E-14. Data Record Format for a 3AS/3BS File (1 of 2)

ITEM NO.	BYTE OFFSET	FIELD NAME	FORMAT	COMMENTS
1	0	satellite identifier	C	'UARS'
2	4	record type	C	' 3'
3	6	instrument identifier	C	
4	18	physical record count	C	
5	26	spare	C	
6	28	total number of points in the record	I	
7	32	number of actual points (np)	I	
8	36	starting wavelength of first actual point	R	
9	40	record time in UDTF format	T	see Note
10	48	spare	C	
11	52	spare	C	
12	56	spare	C	
13	64	data value for first point in record	R	
		. . .		
		data value for last point in record	R	
	B	quality for first point in record	R	
		. . .		
		quality for last point in record	R	
	N	Number of parameter pairs	I	
	P	First parameter name	C	
	P+20	First parameter value	C	
	.	.		
	.	.		
	.	.		
	P+1560	Last parameter name	C	
	P+1580	Last parameter value	C	

Table E-14. Data Record Format for a 3AS/3BS File (Continued)

Legend:
C = character
I = integer
R = real
P = N + 4
T = time in UDTF format
B = 64 + 4*total number of points
N = B + 4* total number of points

Note: For solar instruments (Level 3AS file) the milliseconds portion of the UDTF time is 0.

APPENDIX F

ERROR HANDLING

F.1 STATUS CODES

A 32-bit status code is returned to the user's program to indicate the completion status for each of the UCSS software support services. These status codes are defined to the system using the VAX Message Utility. The Message Utility defines a symbolic name for each of the conditions and the user's program can use the symbolic name to check for a particular status.

In the RAC simulated and production environments, the UCSS software support services only return nonfatal status to the user's program. The standard SS$_NORMAL condition code is used to indicate a normal status for all of these services. The status codes applicable to each interface are identified in Section 3. In the analysis environment, all status codes are returned to the user.

The user's program should check the status code after calling a UCSS service. These status codes are normally warning indicators, but they may have a significant meaning to the program. For example, a status code from a read service might indicate an end of data condition and the user's program should not attempt reads beyond the requested time. The symbolic name for the status condition is used to check for a specific condition. Figure F-1 provides an example of status checking. Note that the user's program must specify all status symbolic names that are explicitly tested as external references.

Figure F-1. UCSS Software Support Service Status Checking Example

```
      PROGRAM LEVEL0
              .
              .
              .
      EXTERNAL PFA_NOOLDFILE
              .
              .
              .
      PASS_FAIL = 'PASS'
      CALL PGINIT(PARAMS, STRTIME, STPTIME, UARS_DAY)
              .
              .
              .
      CALL OPENL0(INST_ID, STRTIME, STPTIME, LID, STATUS)
C
C         CHECK OPEN STATUS FOR NO DATA CONDITION
C
      IF (STATUS .EQ. %LOC(PFA_NOOLDFILE)) THEN
          PASS_FAIL = 'FAIL'
          COMMENTS = 'FAILED FIRST LEVEL 0 OPEN'
      ELSE
              .
              .
              .
      ENDIF
      CALL PGTERM(PASS_FAIL, STATUS, COMMENTS)
      END
```

F.2 FATAL CONDITIONS

The UCSS software support services may also detect error conditions that prevent further processing. When the UCSS services detect a fatal condition, processing is terminated and the program is marked as failed. The fatal error condition appears on the program summary report with any appropriate error comments.

Most of the fatal conditions detected by the UCSS services relate to problems in calling sequence arguments. Conditions detected include:

- Missing required argument
- Argument is of wrong type or size
- Invalid values

- Inconsistent arguments

In addition, the services detect problems in the ordering of some calls (e.g. calling READL0 before OPENL0) or missing required calls (e.g. no PGINIT call). Table F-1 provides a list of the fatal error conditions detected by the UCSS services. Some fatal conditions are detected within VMS services and the user should refer to the appropriate VMS documentation.

Table F-1. UCSS Software Support Services Fatal Errors

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_ATTRCNTNEG	Attribute count is negative	Bad number of attributes supplied as an argument in CLOSELF or DASLID
PFA_ATTROMITTED	Required user supplied attribute not provided	Required user supplied attribute not provided to DASLID(see Table 3-4)
PFA_BADEPOCHYR	No valid ASC09 base epoch year found	Probable telemetry data error
PFA_DBRECERR	Unable to record processing error in data base	Data base access error.
PFA_DUPVIRDAY	Duplicate virtual UARS day specified	Error in FILE_PARAMS
PFA_EARLYEOF	Unexpected end of file encountered when positioning to or reading a data record	Probable error in data file format
PFA_FILALRDDEASG	File already deassigned	Two calls to DASLID to deassign the same LID without corresponding assign call
PFA_FILALRDYCLS	File already closed	Two calls to CLOSELF to close the same LID without corresponding open call
PFA_FILENOTOPEN	File has not been opened	
PFA_FILSTOPEN	Deassigned file is still open	Called DASLID before closing file
PFA_GENUNREC	General unrecoverable error	UCSS software error. Should not be reported to user.
PFA_ILUDTF	Invalid UDTF time provided	Possible error in UDTF time specification in PROGRAM_PARAMS namelist
PFA_ILVMSTI	Invalid VMS time provided	Possible error in VMS time specification in PROGRAM_PARAMS namelist

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INAPSOLRDAY	Requested date does not match UARS day of the file	Inconsistency between UARS day specified in FILE_PARAMS and day in file.
PFA_INCFILUSE	Inconsistent file usage specified by OLD_NEW	Attempted to open or assign a held file as new or old.
PFA_INCOMPEMAF	Incomplete Level 0 EMAF	Missing one of the two part EMAF records. Problem in data file format
PFA_INCONNUMREC	File record count does not exceed number of label records	Data error in file label record
PFA_INCONRECLEN	Inconsistent record length	Data problem. Record length for the file in catalog does not match actual record length of file.
PFA_INCONRECTYP	Level 0 record type field is invalid	Level 0 data problem. Record type for one record EMAFs is not 3. Record type for two record EMAFs is not 1 or 2.
PFA_INCORNUMARG	Service called with incorrect number of arguments	Missing or extra arguments in subroutine call
PFA_INVACCESSMD	Invalid access mode for file type	Attempted to write to a read only file by calling a Level 3A write service for a cataloged file.
PFA_INVALDOY	Invalid day of year	Day of year not within range of 1 to 366
PFA_INVALIDMSD	Invalid mean solar distance	Data problem. Mean solar distance value retrieved from solar data record by READL3S is negative or zero.

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVALMO	Invalid month	Month not within range of 1 to 12
PFA_INVARGDATTP	Invalid argument data type	Error in subroutine call
PFA_INVARGSUB	Internal error in arguments subtype	UCSS software problem. Contact UCSS software maintenance.
PFA_INVARGTYP	Internal error in argument type	UCSS software problem. Contact UCSS software maintenance.
PFA_INVBASNDX	Invalid base index in Level 3A file label record	Base index is not between 0 and 100
PFA_INVBASWVLEN	Invalid base wavelength in level 3 solar file label record	Base wavelength is not between 115.5 and 425.5 nm.
PFA_INVCALDAY	Invalid day of month	Day of month not within range of 1 to 31
PFA_INVCALMAT	Invalid CALIBRATION_MATCH namelist parameter	CALIBRATION_MATCH must be 'PREV', 'NEXT', 'EXCT', or 'NEAR'
PFA_INVCMATCHV	Invalid calibration match rule specified	Invalid DMATCH argument to ASGCAL
PFA_INVCONVDAY	Invalid UARS_DAY obtained by conversion from a UDTF time	Inappropriate launch date used for conversion
PFA_INVCYCARG	File cycle argument is not between 1 and 31 inclusive	Error in call to SETVERCY
PFA_INVDATALEV	Wrong UCSS service called for the data level	Called the wrong service for the data level associated with the LID. Examples: 1. Called CLOSELF for a file that is not a Level 0, 3AT, 3AL, 3AS, or 3BS file instead of calling DASLID 2. Called Level 0 service to access Level 3A data or vice versa

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVDATARNG	Requested data range does not overlap virtual file data range	Problem with START_INDEX START_WVLNGTH or NUM_POINTS in read
PFA_INVDAYARG	Invalid UARS day argument	UARS_DAY is negative
PFA_INVDEFCMATCH	Invalid CALIBRATION_MATCH in DEFAULT_PARAMS namelist	CALIBRATION_MATCH must be 'PREV','NEXT','EXCT', or 'NEAR'
PFA_INVDEFNDLEV	Invalid NEW_DATA_LEVEL in DEFAULT_PARAMS namelist	First character of NEW_DATA_LEVEL must be '1', '2', '3', or field must be blank
PFA_INVDEFODLEV	Invalid OLD_DATA_LEVEL in DEFAULT_PARAMS namelist	First character of OLD_DATA_LEVEL must be '0', '1', '2', '3', or field must be blank
PFA_INVDEFOLDNEW	Invalid OLD_NEW parameter in DEFAULT_PARAMS namelist	OLD_NEW parameter must be 'OLD' or 'NEW'
PFA_INVDISTARG	Invalid distance argument	Distance flag is not 'I=AU' or 'UNCORRECTED' in call to READL3S
PFA_INVDLEARG	Invalid data level argument	Data level argument is not one of the defined data levels
PFA_INVESIZEARG	Invalid estimated file size argument	SIZE argument is zero
PFA_INVFDISP	File disposition with type with type of file accessed	FDISP parameter is not valid for the type of file accessed and the UCSS is unable to determine requested position. Called DASLID with 'CAT' dispositions for a scratch file.
PFA_INVFDISPARG	Invalid file disposition argument	Invalid FDISP in CLOSELF or DASLID call (not 'CAT', 'FREE', or 'HOLD')
PFA_INVFILETYP	Invalid file type specified for usage of file	UCSS software problem Contact UCSS software maintenance.

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVFILUTIN	Invalid file utilization indicator in UCSS internal table	UCSS software problem. Contact UCSS software maintenance.
PFA_INVFLXUARG	Invalid flux unit argument	Flux unit specified in call to READL3S is invalid
PFA_INVHDRDASET	Data set in L0 file label does not match expected value	Data-type is not consistent with data set id. Wrong Level 0 file specified or bad data in file
PFA_INVHDRDATLV	Invalid data level in Level 3A file label	Data problem
PFA_INVHDRDATTP	Instrument id in file label does not match expected value	Wrong Level 3 file specified or bad data in file
PFA_INVHDRDAY	UARS day in file label does not match expected value	Wrong file specified or bad data in file
PFA_INVHDRLAT	Invalid latitude range field in label record of Level 3AL data file	Data problem
PFA_INVHDRSUBTP	Data subtype in file label does not match expected value	Wrong Level 3 file specified or bad data in file
PFA_INVHDRTMRNG	Invalid time range in file label record	Data problem
PFA_INVINDEXARG	Invalid index argument	Index argument is not between 0 and 100
PFA_INVLATGRID	Invalid latitude grid value	Invalid latitude value in WRITEL3AL, READL3AL, WRITEL3LP, or READL3LP
PFA_INVLATLONG	Invalid latitude or longitude	Invalid latitude or longitude value in WRITEL3AT, OPENL3AL, WRITEL3TP, or OPENL3LP
PFA_INVLSTSZA	Invalid local solar time and/or solar angle calculated	UCSS software problem. Contact UCSS software maintenance

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVMAXPMS	Specified number of params is greater than max params in file	Used a number of parameters value greater than number of parameters returned from the open in reading a parameter file
PFA_INVMAXPTS	Invalid maximum number of data points	1. Invalid MAX_POINTS argument to OPENL3AT or OPENL3AL when creating a new file 2. Invalid maximum points field in Level 3A file label record
PFA_INVNEGDYARG	Correlative UARS day arg. is not between -99999 and 9999	Invalid UARS day in call to ASGCOR
PFA_INVNMLDLEV	Invalid DATA_LEVEL parameter in FILE_PARMS namelist	First character of DATA_LEVEL must be '0', '1', '2', '3', or field must be blank
PFA_INVNMLPARM	Invalid combination of parameters in FILE_PARMS namelist	Wrong combination of parameters specified for file
PFA_INVNUMPRMS	Invalid number of parameters specified for a parameter file	The number of parameters specified for file in READL3TP and READL3LP exceeds the maximum value allowed for the file
PFA_INVNUMPTS	Invalid number of points arguments	1. Invalid NUM_POINTS argument to READL3AT or READL3AL. Inconsistent with START_INDEX and OPENL3AT or OPEN3AL MAX_POINTS value. 2. Invalid NUM_POINTS argument to WRITEL3AT or WRITEL3AL. Inconsistent with START_INDEX and MAX_POINTS supplied to OPENL3AT or OPENL3AL.

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVNUMRECS	Physical record count in file label record is invalid	Data problem
PFA_INVODNWHLD	Invalid OLD_NEW namelist parameters	OLD_NEW must be 'OLD', 'NEW', or 'HELD'
PFA_INVOLDNWARG	Invalid OLD_NEW argument	OLD_NEW argument to open or assign call is not 'OLD', 'NEW', or 'HELD'
PFA_INVPGCSARG	Invalid program completion status argument	PASS_FAIL argument to PGTERM is not 'PASS' or 'FAIL'
PFA_INVPRGPMSIZ	Invaild program parameter table size argument	PARAM_TBL_SIZE argument to PGINIT is not between 1 and 50
PFA_INVPSEUD	Invalid use of pseudo-virtual file	Pseudo-virtual file specified as held or in multi-file virtual input file
PFA_INVQLCODARG	Quicklook code argument is not between -100 and 30	Bad Quicklook pass code specified in call to OPENQL for Analysis Services
PFA_INVRECPEMAF	Invalid number of records per EMAF field in file label record	Level 0 data problem
PFA_INVRECRNG	Invalid record time range specification	STRT_DATTIM exceeds STOP_DATTIM in READL3AT, READL3AL, READL3TP, or READL3LP
PFA_INVRECSARG	Number of records argument does not exceed zero	Bad value of MAX_DIM or MAX_DAYS specified in READL3AL, READL3AT, READL3S, READL3LP, or READL3TP
PFA_INVRECTYP	Unexpected record type value	Data problem. Level 0 data record type is not 1, 2, or 3.

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVRULEARG	Version/cycle rule argument is not between 0 and 9	Bad version or cycle specified in call to SETVERCY for Analysis Services
PFA_INVSTRINDX	Start index less than base index of Level 3A file	START_INDEX in READL3AT is less than the BASE_INDEX in OPENL3AT. START_INDEX in READL3AL less than the BASE_INDEX in OPENL3AL.
PFA_INVSTRLEN	Incorrect character string length	Character string improperly sized
PFA_INVSTRWVLN	Start wavelength is outside allowed range	1. START_WVLNGTH in READL3S is less than BASE_WVLNGTH in OPENL3S. 2. START_WVLNGTH exceeds BASE_WVLNGTH + MAX_NUM_VALUES - NUM_VALUES. START_WVLNGTH and NUM_VALUES are supplied in the call to READL3S. BASE_WVLNGTH and MAX_NUM_VALUES are supplied in the call to OPENL3S.
PFA_INVSVC	Wrong service called for given file type	Used QUALRD or QUALQL to read non-QUALITY data or used OPENL3AT to read Level 3AS/BS solar data
PFA_INVTIMPRD	Invalid time period type in file label record	Data problem. The type of data time period field in the Level 0 file header is invalid (not ' QL', '24HR', 'VIRT' or 'NEAR')

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVTMERNG	Invalid time range parameters	1. STRT_DATTIM exceeds STOP_DATTIM in PGINIT. In simulated environment, a problem in the PROGRAM_PARAMS namelist. 2. STRT_DATTIM exceeds STOP_DATTIM in OPENL0, OPENL3AT, or OPENL3AL
PFA_INVTMVERS	Inconsistent time fields in version entries of the Level 3A label record	Data problem
PFA_INVUDAYRNGE	Invalid UARS day range	UCSS Software error. Contact UCSS Software maintenance
PFA_INVUDTFARG	Invalid UDTF time	UDTF time argument not a valid time
PFA_INVUDTFDAY	Invalid day of year in UDTF time	UDTF day of year not between 1 and 366
PFA_INVUDTFMSEC	Invalid milliseconds of day in UDTF time	UDTF milliseconds of day not between 0 and 86399999
PFA_INVUDTFYR	Invalid UDTF year	No year on UDTF time
PFA_INVVERSARG	CCB version argument is not between 0 and 9999 inclusive	Bad version specified in call to SETVERCY for Analysis Services
PFA_INVVERTIM	Inconsistent time in time/ version entries	Data problem. Times in the time version entries in the label record(s) are not increasing.
PFA_INVVIRSPEC	Invalid virtual file specification	More than one physical file specified for a non-virtual input file
PFA_INVVFLAG	Invalid virtual flag in Level 3A file label record	Data problem

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_INVWVLUARG	Invalid wavelength unit argument	Wavelength unit specified in OPENL3S or READL3S is not 'NM', 'STANDARD', 'A', 'MICRON', or 'CM'
PFA_JOBALRDYRUN	Current job has already been run	UCSS Software error. Contact UCSS software maintenance
PFA_LIDINUSE	Specified LID in use	Reused LID without calling DASLID or CLOSELF
PFA_LIDNOREUSE	Attempted to reuse the LID that is assigned held file or a newly cataloged file	Called ASGCAT, OPENL3AT, OPENL3AL, OPENL3LP, OPENL3TP, or OPENL3S with a LID associated with a file that was held or cataloged
PFA_LIDNOTOPEN	File corresponding to LID is not open	Called read or write service without calling the open service first
PFA_LIDOTHERUSE	Specified LID is reserved for other use	Attempted to reuse LID assigned to newly cataloged file
PFA_MISINITPARM	Missing required parameter in FILE_PARAMS namelist	Missing namelist parameter
PFA_MISSARG	Missing a required argument	UCSS service called without all required arguments
PFA_MISSMSD	Missing mean solar distance	Mean solar distance not supplied as parameter in call to WRITEL3S
PFA_MSDCONVERR	Mean solar distance conversion error from OTS$CVTTD	Mean solar distance specified as a solar parameter is negative or zero
PFA_NOCLSNEW	Failed to close a new Level 3A file	Missing CLOSELF call for a new Level 3A file
PFA_NODASGNEW	Failed to deassign an new Level 1 or 2 file	Missing DASLID call for a new Level 1 or 2 file

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_NOFILE	File does not exist	Possibly specified nonexistent file name in FILE_PARAMS namelist
PFA_NOFILECRE	New file was not created	Assigned file was not opened before call to DASLID
PFA_NOFIPARENT	No matching entry in file parameter table for requested file	No FILE_PARAMS namelist corresponding to the requested file in the runstream
PFA_NOFSTAVAIL	Exceeded number of entries in file status table	Contact UCSS software maintenance
PFA_NOHELDFILE	Held file not found	1. Failed to specify "HOLD" on call to DASLID or CLOSELF 2. Did not specify same LID
PFA_NOMATVIRPMS	VIRTUAL_UARS_DAY and DATA_FILE_NAME list sizes not equal	Error in FILE_PARAMS namelist
PFA_NOMORLUNS	No more logical unit numbers available	Attempting to access too many files at one time
PFA_NOOVRLAPTM	File time range and requested time range do not overlap	The time range specified in the open call does not overlap the file time range. In the simulated environment, probable inconsistency between the processing time range and the file time range.
PFA_NOPGINTCAL	PGINIT was not called	Missing PGINIT call before calling UCSS services
PFA_NOPGTRMCAL	PGTERM was not called	Program terminated without calling PGTERM
PFA_NOREQDATA	Required data not available	File specified as required input by the scheduler is not available

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_NOREQRECS	Virtual file contains no data records	All physical files have no data
PFA_NOUSFREQT	Attempted to assign user status file when no user status files are defined for the job	Scheduler does not know of the use of user status files in this job
PFA_NOVERTIMRNG	Version time range not found in time version array	UCSS software error. Contact UCSS software maintenance.
PFA_NOVIRFILID	No virtual file table entry for the logical file identifier	UCSS software error. Contact UCSS software maintenance.
PFA_NOVIRTAVAIL	No room in virtual file table	Contact UCSS software maintenance
PFA_PGINTPREV	PGINIT already called	Two calls to PGINIT in same program
PFA_RECLENERR	Expected record length does not match actual record length	Data error. Data record length is incorrect.
PFA_REQATTNOSUP	Required catalog attribute not provided	Required catalog attribute(s) not provided to DASLID (see Table 3-4)
PFA_REQFILMISS	Missing one or more required physical files for a virtual read	One or more Level 0 or 3A files needed for a virtual file are indicated as required by the scheduler, but are unavailable
PFA_SEQTIMERR	Current record time is not later than previous record time	Data times are not increasing
PFA_TOMANYFILE	Exceeded maximum number of FILE_PARAMS namelists supported by UCSS	User provided more FILE_PARAMS namelists than supported by the UCSS software. Contact UCSS software maintenance.

Table F-1. UCSS Software Support Services Fatal Errors (Continued)

SYMBOLIC NAME	DESCRIPTION	POSSIBLE CAUSES
PFA_UNEXBLKARG	Unexpected blank argument	1. Logical file id is blank in call to any of the services 2. Old-new-flag is blank in call to OPENL3AT, OPENL3AL, OPENL3S, or ASGCAT 3. Flux units, wavelength units, or distance flag is blank in call to READL3S
PFA_UNKNOWNLID	Attempt to close or deassign an unknown LID	1. Called CLOSELF or DASLID with incorrect LID 2. Called CLOSELF or DASLID without corresponding open or assign
PFA_UNKREQSFDU	Required description id is not available for current file	All portions of the current file are required and: 1. UARS SFDU file missing or unassigned, or, 2. Error in reading UARS SFDU file, or, 3. SFDU descriptor id with attributes that are subset of current file's attributes is not present in UARS SFDU file
PFA_USFNUMGTMAX	User status file number greater than maximum defined for job	The use of this user status file number has not been defined to the scheduler

APPENDIX G

LEVEL 0 SFDU FILE DESCRIPTION

The information used to build the SFDU record for Level 3A data is obtained from the SFDU file, an example of which is shown in Figure G-1. Note that the file is known to the UCSS Software Services by its logical name, UARS_SFDU_FILE, which must therefore be linked to the actual file's name before a job that is to generate a new Level 3A file with the desired SFDU record is run.

Figure G-1. Sample UCSS SFDU File

```
$SFDU_GEN_PARAMS
  CONTROL_AUTHORITY_ID    = 'ZURS'
  DEFAULT_DESCRIPTION_ID = 'ZERO'
$END

$DESCRIPTION_ID_PARAMS
  DESCRIPTION_ID    =    'HR75'
  ATTRIBUTE NAMES   =    'TYPE', 'SUBTYPE', 'LEVEL'
  ATTRIBUTE VALUES =    'HRDI', 'Z_WIND',  '3AL'
$END

$DESCRIPTION_ID_PARAMS
  DESCRIPTION_ID    =    'HR12'
  ATTRIBUTE NAMES   =    'TYPE', 'SUBTYPE', 'LEVEL'
  ATTRIBUTE VALUES =    'HRDI', 'WINDS',   '2'
$END

$DESCRIPTION_ID_PARAMS
  DESCRIPTION_ID    =    'HR06'
  ATTRIBUTE NAMES   =    'TYPE', 'LEVEL'
  ATTRIBUTE VALUES =    'HRDI'  '3AL'
$END

$DESCRIPTION_ID_PARAMS
  DESCRIPTION_ID    =    'HR01'
  ATTRIBUTE NAMES   =    'TYPE'
  ATTRIBUTE VALUES =    'HRDI'
$END
```

The file is composed of two different types of namelists whose structures are described in Tables G-1 and G-2. The first namelist contains general parameters required for constructing the SFDU record and occurs only once in the file. The other namelist contains a specific data descriptive record identifier (DDRI) and the attributes of the data for which it is defined. It occurs once for each defined DDRI supported by the UCSS.

Table G-1. Structure of SFDU_GEN_PARAMS Namelist

NAMELIST PARAMETER	DESCRIPTION	FORMAT	VALUES
CONTROL_AUTHORITY_ID	control authority identifier for UARS data as described in Tables E-2 and E-7	C*4	"ZURS"
DEFAULT_DESCRIPTION_ID	data descriptive record identifier to be used if attribute matching is unsuccessful	C*4	"ZERO"

Table G-2. Structure of DESCRIPTION_ID_PARAMS Namelist

NAMELIST PARAMETER	DESCRIPTION	FORMAT	VALUES
DESCRIPTOR_ID	data descriptive record identifier for UARS data at a particular documentation level	C*4	Note 1
ATTRIBUTE_NAMES	array of attribute names (up to 20 allowed)	C*20	Note 2
ATTRIBUTE_VALUES	array of attribute values (up to 20 allowed)	C*20	Note 3

Notes:

1 The first two characters are associated with the instrument identifier and the last two are numeric digits (See Reference 1).

2 The attribute names must belong to the set described in Table G-3.

3 The allowed values for the specified attributes are the same as those with which the pertinent files can be cataloged.

The attributes that can be used to define the sets of data associated with a particular DDRI are those that normally characterize a science data file in the UCSS environment. Their names and possible values are shown in Table G-3. One or more of these attributes can be used in the definition. See Reference 1 for a more detailed description of the manner in which DDRIs are defined and maintained. Note however that because of the way attribute matching is done in the UCSS Software Services, if different DDRIs are to be assigned to

different levels in the document hierarchy, e.g. 'TYPE' at one level and 'TYPE' and 'SUBTYPE' at another level, then the DDRI assigned to the lower level, e.g. the latter in the current example, should precede the one assigned at the higher level, e.g. the former, in the SFDU file. Otherwise, matching will complete before the desired DDRI is found. Moreover, if no DDRIs with matching attributes are found at any level of documentation, the default value in the GEN_SFDU_PARMS namelist will be used instead, or, if that value is missing, the default value assumed by Software Services when the SFDU file is not accessible or nonexistent, namely 'ZNON'.

Table G-3. Allowed attributes for DESCRIPTION_ID_PARAMS Namelist

ATTRIBUTE NAME	DESCRIPTION	POSSIBLE VALUES
TYPE	instrument identifier	See Note 1
SUBTYPE	data species or measurement type	See Note 2
LEVEL	processing level of data	'3AL','3AS','3AT', '3LP','3TP','3BS'
DAY	UARS day number	'1' to '9999'

Notes

1 Identifier for one of the UARS instruments, namely 'CLAES', 'HALOE', 'HRDI', 'ISAMS', 'MLS', 'PEM', 'SOLSTICE', 'SUSIM' and 'WINDII'.

See Reference 1, Item 4 for the range of data descriptive record identifiers presently allocated to each UARS instrument.

2 Dependent on UARS instrument.

APPENDIX H

LEVEL 0 OBC REPORT NAMES

H.1 OBC REPORT NAMES AND NUMBERS

Table H-1 shows the OBC report names and numbers that are decoded by the OBCDECODE routine.

Table H-1. OBC Report Names Decoded by OBCDECODE

REPORT NBR	VARIABLE	OFFSET	DECODE	SUBSCRIPT	
ACS%01 01					
	IRSLEW	00.6	1 Bit	OBC_BYTE	1
	IYSLEW	00.7	1 Bit	OBC_BYTE	2
	ICAL	03.3	2 Bits	OBC_INTEGER	1
	MODE	03.5	3 Bits	OBC_INTEGER	2
	EYSLEW3	08.0	3 Bytes	OBC_REAL	1
ACS%04 04					
	EX	00.0	2 Bytes	OBC_REAL	1
	EY	02.0	2 Bytes	OBC_REAL	2
	EZ	04.0	2 Bytes	OBC_REAL	3
ACS%09 09					
	TF	00.0	5 Bytes	OBC_INTEGER	1 _2 (UDTF)
	TFYEAR	05.0	1 Byte	OBC_INTEGER	3
	TUPDATE	17.0	5 Bytes	OBC_INTEGER	4 _5 (UDTF)
GYR%01 12					
	CNGX	0.0	1 Byte	OBC_BYTE	1
	CNGY	1.0	1 Byte	OBC_BYTE	2
	CNGZ	2.0	1 Byte	OBC_BYTE	3
	CNGX1	3.0	1 Byte	OBC_BYTE	4
	CNGY1	4.0	1 Byte	OBC_BYTE	5
	CNGZ1	5.0	1 Byte	OBC_BYTE	6
	CNGX2	6.0	1 Byte	OBC_BYTE	7
	CNGY2	7.0	1 Byte	OBC_BYTE	8
	CNGZ2	8.0	1 Byte	OBC_BYTE	9
	CNGX3	9.0	1 Byte	OBC_BYTE	10
	CNGY3	10.0	1 Byte	OBC_BYTE	11
	CNGZ3	11.0	1 Byte	OBC_BYTE	12
	CNGX4	12.0	1 Byte	OBC_BYTE	13
	CNGY4	13.0	1 Byte	OBC_BYTE	14
	CNGZ4	14.0	1 Byte	OBC_BYTE	15
	CNGX5	15.0	1 Byte	OBC_BYTE	16
	CNGY5	16.0	1 Byte	OBC_BYTE	17
	CNGZ5	17.0	1 Byte	OBC_BYTE	18
	CNGX6	18.0	1 Byte	OBC_BYTE	19
	CNGY6	19.0	1 Byte	OBC_BYTE	20
	CNGZ6	20.0	1 Byte	OBC_BYTE	21
	CNGX7	21.0	1 Byte	OBC_BYTE	22
	CNGY7	22.0	1 Byte	OBC_BYTE	23
	CNGZ7	23.0	1 Byte	OBC_BYTE	24
EPH%01 13					
	EOGBRF1	0.0	4 Bytes	OBC_REAL	1
	EOGBRF2	4.0	4 Bytes	OBC_REAL	2
	EOGBRF3	08.0	4 Bytes	OBC_REAL	3

Table H-1. OBC Report Names Decoded by OBCDECODE (Continued)

REPORT NBR	VARIABLE	OFFSET	DECODE	SUBSCRIPT	
	EOGBVF1	12.0	4 Bytes	OBC_REAL	4
	EOGBVF2	16.0	4 Bytes	OBC_REAL	5
	EOGBVF3	20.0	4 Bytes	OBC_REAL	6
EPH%02 15					
	EOGVFAL	08.0	2 Bytes	OBC_INTEGER	1
HGA%01 21					
	HGTAFLGA	00.0	1 BIT	OBC_BYTE	1
	HGTAFLGB	00.1	1 BIT	OBC_BYTE	2
	HGLRFLGA	00.2	1 BIT	OBC_BYTE	3
	HGLRFLGB	00.3	1 BIT	OBC_BYTE	4
	HGMODCUR	01.0	3 BIT	OBC_INTEGER	1
	HGTRGCUR	02.3	1 BIT	OBC_BYTE	5
	HGGIMCA	07.0	1 BYTE	OBC_REAL	1
	HGGIMCB	08.0	1 BYTE	OBC_REAL	2
UFL%01 24					
	S1	00.0	2 Byte	OBC_REAL	1
	S2	02.0	2 Byte	OBC_REAL	2
	S3	04.0	2 Byte	OBC_REAL	3
	PM111	12.0	4 Byte	OBC_REAL	4
	PM112	16.0	4 Byte	OBC_REAL	5
	PM113	20.0	4 Byte	OBC_REAL	6
	SCP11	24.0	1 Byte	OBC_REAL	7
	SCP12	25.0	1 Byte	OBC_REAL	8
	SCP22	26.0	1 Byte	OBC_REAL	9
UFL%02 25					
	PM115	00.0	4 Byte	OBC_REAL	1
	PM116	04.0	4 Byte	OBC_REAL	2
	PM119	08.0	4 Byte	OBC_REAL	3
UFL%09 32					
	TUS	14.0	5 Bytes	(RETURNED IN RET_DATTIM)	
SEP%01 43					
	BETA1	22.0	2 Bytes	OBC_REAL	1
PMO%01 53					
	TDAY	20.0	1Byte	OBC_INTEGER	1
SPP%01 54					
	PFTRGFLG	00.0	1 Bit	OBC_BYTE	1
	PFRATEFL	00.1	1 Bit	OBC_BYTE	2
	PFOFSETF	00.2	1 Bit	OBC_BYTE	3
	PFOCFLAG	00.3	1 Bit	OBC_BYTE	4
	PFPRFLAG	00.4	1 Bit	OBC_BYTE	5
	PFEPFLAG	00.5	1 Bit	OBC_BYTE	6
	PFAUTOFL	00.6	1 Bit	OBC_BYTE	7
	PFSTAT5	01.3	1 Bit	OBC_BYTE	8
	PFSTAT4	01.4	1 Bit	OBC_BYTE	9

Table H-1. OBC Report Names Decoded by OBCDECODE (Continued)

REPORT NBR	VARIABLE	OFFSET	DECODE	SUBSCRIPT	
	PFSTAT3	01.5	1 Bit	OBC_BYTE	10
	PFSTAT2	01.6	1 Bit	OBC_BYTE	11
	PFSTAT1	01.7	1 Bit	OBC_BYTE	12
	PFMODCUR	02	1 Byte	OBC_INTEGER	1
	PFACQST8	03.0	1 Bit	OBC_BYTE	13
	PFACQST7	03.1	1 Bit	OBC_BYTE	14
	PFACQST6	03.2	1 Bit	OBC_BYTE	15
	PFACQST5	03.3	1 Bit	OBC_BYTE	16
	PFACQST4	03.4	1 Bit	OBC_BYTE	17
	PFACQST3	03.5	1 Bit	OBC_BYTE	18
	PFACQST2	03.6	1 Bit	OBC_BYTE	19
	PFACQST1	03.7	1 Bit	OBC_BYTE	20
	PFFDCST8	04.0	1 Bit	OBC_BYTE	21
	PFFDCST7	04.1	1 Bit	OBC_BYTE	22
	PFFDCST6	04.2	1 Bit	OBC_BYTE	23
	PFFDCST5	04.3	1 Bit	OBC_BYTE	24
	PFFDCST4	04.4	1 Bit	OBC_BYTE	25
	PFFDCST3	04.5	1 Bit	OBC_BYTE	26
	PFFDCST2	04.6	1 Bit	OBC_BYTE	27
	PFFDCST1	04.7	1 Bit	OBC_BYTE	28
	PFGOLST8	05.0	1 Bit	OBC_BYTE	29
	PFGOLST7	05.1	1 Bit	OBC_BYTE	30
	PFGOLST6	05.2	1 Bit	OBC_BYTE	31
	PFGOLST5	05.3	1 Bit	OBC_BYTE	32
	PFGOLST4	05.4	1 Bit	OBC_BYTE	33
	PFGOLST3	05.5	1 Bit	OBC_BYTE	34
	PFGOLST2	05.6	1 Bit	OBC_BYTE	35
	PFGOLST1	05.7	1 Bit	OBC_BYTE	36
	PFTRGST8	06.0	1 Bit	OBC_BYTE	37
	PFTRGST7	06.1	1 Bit	OBC_BYTE	38
	PFTRGST6	06.2	1 Bit	OBC_BYTE	39
	PFTRGST5	06.3	1 Bit	OBC_BYTE	40
	PFTRGST4	06.4	1 Bit	OBC_BYTE	41
	PFTRGST3	06.5	1 Bit	OBC_BYTE	42
	PFTRGST2	06.6	1 Bit	OBC_BYTE	43
	PFTRGST1	06.7	1 Bit	OBC_BYTE	44
	PFSUNST8	07.0	1 Bit	OBC_BYTE	45
	PFSUNST7	07.1	1 Bit	OBC_BYTE	46
	PFSUNST6	07.2	1 Bit	OBC_BYTE	47
	PFSUNST5	07.3	1 Bit	OBC_BYTE	48
	PFSUNST4	07.4	1 Bit	OBC_BYTE	49
	PFSUNST3	07.5	1 Bit	OBC_BYTE	50
	PFSUNST2	07.6	1 Bit	OBC_BYTE	51
	PFSUNST1	07.7	1 Bit	OBC_BYTE	52
	PFTRGCUR	08	1 Byte	OBC_INTEGER	2
	PFTRGPRM	09	1 Byte	OBC_INTEGER	3
	PFTRGSEC	10	1 Byte	OBC_INTEGER	4
	PFACQCNT	11	1 Byte	OBC_INTEGER	5
	PFACQTHR	12	1 Byte	OBC_INTEGER	6
	PFOCSWCT	13	1 Byte	OBC_INTEGER	7
	PFTIMER	14	2 Bytes	OBC_INTEGER	8
	PFTIMESL	16	2 Bytes	OBC_INTEGER	9

Table H-1. OBC Report Names Decoded by OBCDECODE (Continued)

REPORT NBR	VARIABLE	OFFSET	DECODE	SUBSCRIPT
	PFGIMCUA	18	3 Bytes	OBC_REAL 1
	PFGIMCUB	21	3 Bytes	OBC_REAL 2
	PFGMCMDA	24	1 Byte	OBC_INTEGER 10
	PFGMCMDB	25	1 Byte	OBC_INTEGER 11
SPP%02 55				
	PFTARGA	00	3 Bytes	OBC_REAL 1
	PFTARGB	03	3 Bytes	OBC_REAL 2
	PFT1MAX	06	2 Bytes	OBC_REAL 3
	PFT1MIN	08	2 Bytes	OBC_REAL 4
	PFT2MAX	10	2 Bytes	OBC_REAL 5
	PFT2MIN	12	2 Bytes	OBC_REAL 6
	PFGOALA	14	3 Bytes	OBC_REAL 7
	PFGOALB	17	3 Bytes	OBC_REAL 8
	PFOFSETA	20	2 Bytes	OBC_REAL 9
	PFOFSETB	22	2 Bytes	OBC_REAL 10
	PFRTMAXA	24	1 Byte	OBC_INTEGER 1
	PFRTMAXB	25	1 Byte	OBC_INTEGER 2
SPP%03 56				
	PFPSCMDA	00	2 Bytes	OBC_REAL 1
	PFPSCMDB	02	2 Bytes	OBC_REAL 2
	PFSLRATA	04	2 Bytes	OBC_REAL 3
	PFSLRATB	06	2 Bytes	OBC_REAL 4
	PFSSERRA	08	2 Bytes	OBC_REAL 5
	PFSSERRB	10	2 Bytes	OBC_REAL 6
	PFSTCUA1	12	2 Bytes	OBC_REAL 7
	PFSTCUA2	14	2 Bytes	OBC_REAL 8
	PFSTCUA3	16	2 Bytes	OBC_REAL 9
	PFSTCUB1	18	2 Bytes	OBC_REAL 10
	PFSTCUB2	20	2 Bytes	OBC_REAL 11
	PFSTCUB3	22	2 Bytes	OBC_REAL 12

H.2 OBC REPORT MNEMONICS

The following sample code shows how to use the mnemonics defined in UCSS_INCDIR:OBC_REP_PARMS.INC to refer to OBC report variables.

```
                              .
                              .
                              .
      INCLUDE/LIST  'UCSS_INCDIR:OBC_REP_PARMS.INC'
                              .
                              .
                              .
      INTEGER*4  ACS04, OBC_EX, OBC_EY, OBC_EZ
                              .
                              .
                              .
      PARAMETER(    ACS04  =04)
         PARAMETER  (OBC_EX       =1)    ! OBC_REAL    s= 02 D
         PARAMETER  (OBC_EY       =2)    ! OBC_REAL    s= 02 D
         PARAMETER  (OBC_EZ       =3)    ! OBC_REAL    s= 02 D
                              .
                              .
                              .
C
C   END OF INCLUDE FILE
C
                              .
                              .
                              .
      CALL READL0 (LID, REQ_TIME, RET_TIME, OBC_FRM,
     1              PARITY, FILL, GAP_FLAG, TIME_FLAG, EMAF_RATE,
     1              VERSION, STATUS)
                              .
                              .
                              .
      CALL OBCDECODE(OBC_FRM,ACS04,REQ_TIME,RET_TIME,QUALITY,
     1                 OBC_REAL,INT_VAR,BYTE_VAR,OBC_REC,
     2                 STATUS)
                              .
                              .
                              .
C
C  USE THE MNEMONICS CONTAINED IN THE INCLUDE FILE TO REFERENCE THE
C  VALUES FOR EX, EY, AND EZ
C
      EX = OBC_REAL(OBC_EX)
      EY = OBC_REAL(OBC_EY)
      EZ = OBC_REAL(OBC_EZ)
                              .
                              .
                              .
      END
```

APPENDIX I

GLOSSARY

ATC	absolute time code
CCB	Configuration Control Board
CDHF	Central Data Handling Facility
CPU	central processing unit
CRC	cyclical redundancy check
DCF	Data Capture Facility
DCL	Digital Command Language
DEC	Digital Equipment Corporation
EMAF	engineering major frame
GE	General Electric
GMT	Greenwich Mean Time
GSFC	Goddard Space Flight Center
I/O	input/output
JATC	Julian format ATC
LID	logical file identifier
NASA	National Aeronautics and Space Administration
OBC	onboard computer
PI	Principal Investigator
RAC	Remote Analysis Computer
SFDU	standard formatted data unit
SMAF	science major frame
SMIF	science minor frame
UARS	Upper Atmosphere Research Satellite
UCSS	UARS CDHF Software System
UDTF	UARS date and time format
VAX	Virtual Address Extension
VMS	Virtual Memory System

APPENDIX J

REFERENCES

1. Goddard Space Flight Center (GSFC), Contractual Specification for the UARS CDHF Software System (UCSS), NAS 5-29250.

2. --, Statement of Work (SOW) for the UARS CDHF Software System (UCSS), October 10, 1985, attached to GSFC Contract NAS 5-29250.

3. --, UARS Ground Data Processing Capability and Requirements Document, GSFC Document No. 430-1401-00, February 1985.

4. --, UARS Programmer's Guide to Orbit and Attitude Services, August 1987 (preliminary).

5. Computer Sciences Corporation, CSC/SD-86/6705, Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) Software System (UCSS) Requirements Analysis Document, July 1986.

6. --, CSC/SD-87/6724, Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) Software System (UCSS) Critical Design Specification, October 1987.

7. --, CSC/SD-87/6729, Interface Control Document Between the Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) and the Generic Time Division Multiplexed (GTDM) Data Capture Facility (DCF), June 1987.

8. General Electric Astro Space Division, Program Information Release (PIR) U-1K21-UARS-7000, Preliminary Science Format Definition - Including S/C Contribution, February 8, 1987.

9. Computer Sciences Corporation, CSC/SD-87/6725, Upper Atmosphere Research Satellite (UARS) Central Data Handling Facility (CDHF) Software System (UCSS) User's Guide, October 1987.

10. Consultative Committee for Space Data Systems, CCSDS 620.0-B-1, Standard Formatted Data Units -- Structure and Construction Rules (draft), November 1987.