TRW

47467-H011-UX-00

CR-171972

RELBET 4.0

PROGRAMMER'S MANUAL

23 DECEMBER 1986

PREPARED FOR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

CONTRACT NAS9-17554

(NASA-CR-171972) RELBET 4.0 PROGRAMMER'S MANUAL (TRW Defense Systems Group) 671 p CSCL 09B

N87-18332

Unclas
G3/61 43517

PREPARED BY

B. P. HUYSMAN
P. S. KWONG
L. A. PIENIAZEK

SYSTEM DEVELOPMENT DIVISION
DEFENSE SYSTEMS GROUP
HOUSTON, TEXAS

47467-H011-UX-00

RELBET 4.0

PROGRAMMER'S MANUAL

23 DECEMBER 1986

PREPARED FOR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

CONTRACT NAS9-17554

PREPARED BY

B. P. HUYSMAN
P. S. KWONG
L. A. PIENIAZEK

SYSTEM DEVELOPMENT DIVISION
DEFENSE SYSTEMS GROUP
HOUSTON, TEXAS

TRW

47467-H011-UX-00

RELBET 4.0
PROGRAMMERS MANUAL

December 1986

Prepared for

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Contract NAS9-17554

Approved by _____
L. A. Pieniazek, Head
Navigation Analysis Section

Approved by _____
D. K. Phillips, Manager
Systems Engineering and
Analysis Department

System Development Division
TRW Defense Systems Group
Houston, Texas

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1.0 INTRODUCTION

This manual describes the RELBET 4.0 System as implemented on the Hewlett Packard model 9000 computer system. It is directed toward programmers and system maintenance personnel. It is intended to serve both as a reference and as a introductory guide to the software. The body of the manual provides an overview of major features and indicates where to look for further information. Full details are left to Appendices. It is recommended that you first read the body of this manual to familiarize yourself with terminology and system organization.

This manual is divided into the following sections:

1. **INTRODUCTION:** Identifies the RELBET System and provides an overview of the organization and use of the manual. Tells you what to expect.
2. **REFERENCES:** Provides references to associated documentation. Tells you where to find additional information.
3. **FUNCTIONAL OVERVIEW:** Provides a basic functional overview of the operation of the RELBET System. Explains what programs are supposed to do.
4. **DESIGN DESCRIPTION:** Provides an overview of key features of the design. Explains how things fit together.
5. **DIRECTORY STRUCTURE:** Describes the directory structure of the delivery version. Explains where to look for something.
6. **INSTALLATION AND MAINTAINCE:** Describes procedures for installing the delivery version on an HP9000 and provides an overview of maintenance facilities. Explains how to change something.

The following appendices provide extensive details.

I. **FILE FORMATS:** Provides details on the formats of the files used by the System.

II. **CONTEXT FILES:** Provides information on the contents of common block, default values, parameter declarations, type declarations, and so on. These files are generally associated with the "include" directive.

III. **SUBROUTINE MANUALS:** Provides a quick reference to subroutine function descriptions and associated calling arguments used.

IV. **SOURCE LISTINGS:** Provides listing of subroutine code suitably indented to reflect the processing structure.

1.1 IDENTIFICATION

The RELBET System is an integrated collection of computer programs that support the analysis and post-flight reconstruction of vehicle to vehicle relative trajectories of two on-orbit free-flying vehicles: the Space Shuttle Orbiter and some other free-flyer. The UNIVAC 1100 version of the system, RELBET 2.0, realizes the full production and analysis capability. The HP9000 version, RELBET 4.0, provides a basic post-flight data production capability and is a partial implementation of the full analysis version. RELBET 4.0 was created by carefully tailoring the RELBET analysis software to fit the production problem and reflects a streamlined production-oriented version that supports the post-flight reconstruction of relative trajectories and the generation of standard data products.

In particular the RELBET 4.0 System accepts Orbiter downlist telemetry input in the form of Computer Compatible Tapes (CCT's) and produces the following outputs:

- o RELBET Ancillary Data Product Tape
- o RELBET Ancillary Data Fiche Tape
- o SENSOR Program Input Data Tape
- o SENSOR Program Environmental Data Tape
- o Tables for Reports

It incorporates the following features not available in RELBET 2.0:

- o Organization of standard runstreams into shell scripts
- o Enhanced user input scheme via linput
- o Increased data QA programs
- o A text file QA data base

It lacks the following features available in RELBET 2.0:

- o Data and trajectory simulation capabilities
- o General purpose automatic editor
- o Least Squares Filter
- o Residual computation programs
- o Binary Data Base Editor
- o General interactive control of display processors
- o Miscellaneous display processors

The delivered version of RELBET 4.0 is available in magnetic tape media and consists of the following elements.

- o Source code (configured in SCCS format)
- o Relocatable subroutine code
- o Executable programs
- o Program creation directives (makefile's)
- o Maintenance tools
- o Program and subroutine documentation (for nroff formatting)

Tapes containing sample input and output are also available.

1.2 REQUIRED BACKGROUND

This manual assumes a basic familiarity with UNIX, C, and FORTRAN as implemented via BOX and ratfor. In addition particular areas may require additional expertise. This varies depending upon the application and is summarized in Table 1.2. Parenthetical numbers indicate the applicable section of standard UNIX manuals. For information on these areas, consult the references in section 2.0 or a standard UNIX reference.

Table 1.2. Background Summary

Area	Applicable Background
General source code	FORTRAN, BOX, ratfor, C, Bourne Shell
linput source	C, lex(1), yacc(1)
Graphics display	DISSPLA
Configuration Control	sccs(1)
Creating programs	make(1)
Common blocks	mtf utilities
Input blocks	mtf utilities
Documentation	nroff(1), man(7), cstrip, txman, blist, clist, mtf utilities

1.3 HARDWARE/SOFTWARE ENVIRONMENT

RELBET 4.0 assumes the following hardware configuration:

- o Hewlett Packard 9000 Computer model 540
- o Hewlett Packard 7935 440 Megabyte disk drives
- o Hewlett Packard 150 computers as terminals
- o Hewlett Packard 9872B plotter for plots
- o Hewlett Packard 7970 9 track 1600 bpi tape drive.

RELBET 4.0 assumes the following software environment:

- o RELBET 4.0
- o HPUX operating system version 5.1
- o DISSPLA Graphics Library.

1.4 TERMINOLOGY AND CONVENTIONS

Terms are generally used in the sense of UNIX. In the following, **program** refers to an executable object. When a distinction between an interpreted shell program and a compiled and linked program is needed, the terms **shell program** or **binary program** are used respectively. The term **processor** is also used to mean a binary program. The term **routine** means a C function or a

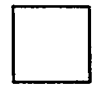
FORTRAN routine. Routines correspond to source and relocatable code. They are separately compiled but not linked. Thus there is a distinction between driver **routines** and executable **programs** that results from linking. The terms **directory** and **file** have the same meaning as in UNIX. The term **module** is used in a descriptive sense to indicate a set of related routines. The term **package** is also used descriptively. It indicates a set of related routines, data structures, or even programs. The term **context** is used to indicate information related to variables or subroutines defined elsewhere, as well as sizing, format, and defined type information.

The following lexical conventions are used:

Program and File Names: **Bold font** denotes program and file name within text discussion.

Terminal Display: Terminal screen print examples are always indented and separated from the discussion. **Bold font** denotes user input and regular font denotes program output. Two lines consisting of a single indented colon indicate an omission of several lines of text. The elipsis (...) indicates a omission of text on a given line.

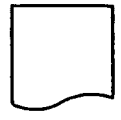
Flow charts use the following symbols:



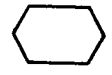
Processor Activity

Binary File

Text File



Display, e.g., Printer Plot



User Knowledge

2.0 REFERENCES

The following documentation is applicable to the RELBET 4.0. and the RELBET System in general.

1. "Space Transportation System Post-flight On-orbit Relative Trajectory and Ancillary Data Products," TRW Report 39107-H011-UX-00, B. P. Huysman, L. A. Pieniazek, 1983.
2. "RELBET Product Description (Update/May 1985)", TRW Memo 85:W482.1-57, B. P. Huysman, L. A. Pieniazek, 15 May 1985.
3. "Programming Standards for FORTRAN Deliverables," TRW Memo 82:W582.1-16, D. K. Phillips, 10 February 1982.
4. "RELBET Software Level A Requirements," TRW Report 39107-H003-UX-00, L. A. Pieniazek, 18 May 1982.
5. "RELBET Software Level B Requirements," TRW Report 39107-H005-UX-00, B. P. Huysman, L. A. Pieniazek, 8 November 1982.
6. "Rendezvous BET Program, LRBET3," JSC-18638, W. M. Lear, December 1982.
7. "STAR Version 1.6 User's Guide," TRW Memo 83:W482.8-13, S. W. Strom, 3 May 1984.
8. "Preliminary RELBET User's Manual," TRW Memo 83:W482.8-24, L. A. Pieniazek, 21 July 1983.
9. "RELBET Engineering Manual," TRW Memo 84:W482.8-125, L. D. Erdman, 15 October 1984.
10. "RELBET Programmer's Manual," TRW Memo 84:482.8-130, L. Morris, 26 October 1984.
11. "RELBET User's Manual Update," TRW Memo 85:W482.8-24, L. A. Pieniazek, 13 March 1985.
12. "HP General File Format (GFF) User's Guide," TRW Report 39107-H021-UX-00, D. G.Campbell, 14 December 1984.
13. "RELBET User's Manual Change Pages," TRW Memo 85:W482.8-106, P. S. Kwong, 12 November 1985.
14. "RELBET Programmer's Manual Change Pages," TRW Memo 85:W482.8-107, P. S. Kwong, 12 November 1985.
15. "BOX System User's Guide," TRW Report 47467-H002-UX-00, W. Pace and D. Poritz, February 1986.

16. "Preliminary HP RELBET Streamlining Level A Description," TRW Memo 86:W482.8-19, L. A. Pieniazek, 21 May 1986.
17. "SENSOR Tape Production Manual," TRW Memo 85.W482.1-60, J. Knoedler, 3 June 1986.
18. "RELBET 4.0 User's Guide," TRW Report 47467-H010-UX-00, B. P. Huysman et al., December 1986.
19. "RELBET 4.0 Programmer's Manual," TRW Report 47467-H011-UX-00, P. S. Kwong and L. A. Pieniazek, December 1986.
20. "HP-UX Reference," Hewlett-Packard Company, 1985.
21. "HP-UX Concepts and Tutorials," Hewlett-Packard Company, 1985.
22. "The C Programming Language," B. W. Kernighan and D. M. Ritchie, Prentice-Hall, Inc., 1978.
23. "Automated Code Generator for C Header Files," TRW Memo 86:W482.8-25, L. A. Pieniazek, 8 October 1986.
24. "Automated Software Documentation Utility," TRW Memo 86:W482.8-28, L. A. Pieniazek, 27 October 1986.
25. "Interface Control Document for ORDC Computer Compatible Tapes," CSC-1921, G. E. Habacek/CSC, February 1984.
26. "HP General File Format (GFF) User's Guide," TRW 39107-H021-UX-00, D. Campbell, 14 December 1984.
27. "DISSPLA User's Manual," Integrated Software Systems Corporation, 1981.

3.0 FUNCTIONAL DESCRIPTION

This section presents a functional description of the execution of RELBET 4.0. The description is at the level of major processing functions and will identify the various programs and "shell scripts" or run streams which one executes to accomplish each function. The overall process is first described and then the major steps examined in more detail. Section 5.0 of the User's Manual provides a detailed example of the process. The following discussion makes parenthetical references to the appropriate subsection of Section 5.0.

3.1 GENERAL OVERVIEW

The RELBET production procedure consists of five basics steps:

- o Data Collection
- o Input Data Formatting
- o Input Data Preparation
- o Trajectory Estimation
- o Generation and QA of Output Products.

Figure 3.1.1 summarizes this activity.

The activity begins with the collection of pertinent mission requirements and parameters. These are used to request post-flight telemetry products and to update a data base of standard program inputs. This preparation activity is primarily manual and does not utilize any of the RELBET programs.

The first actual processing phase begins with the receipt of telemetry data. The downlist telemetry comes in the form of Computer Compatible Tapes (CCT's). Various RELBET programs are used to reformat the CCT into files accessible to the various RELBET processors. These files contain information such as vehicle attitude data, relative observation data, and sensed velocity data. This step also transforms observation data into the RELBET reference frames and edits data according to the onboard data good flags.

Various specialized quality assurance (QA) programs support the next processing phase, data preparation. During this activity one checks the

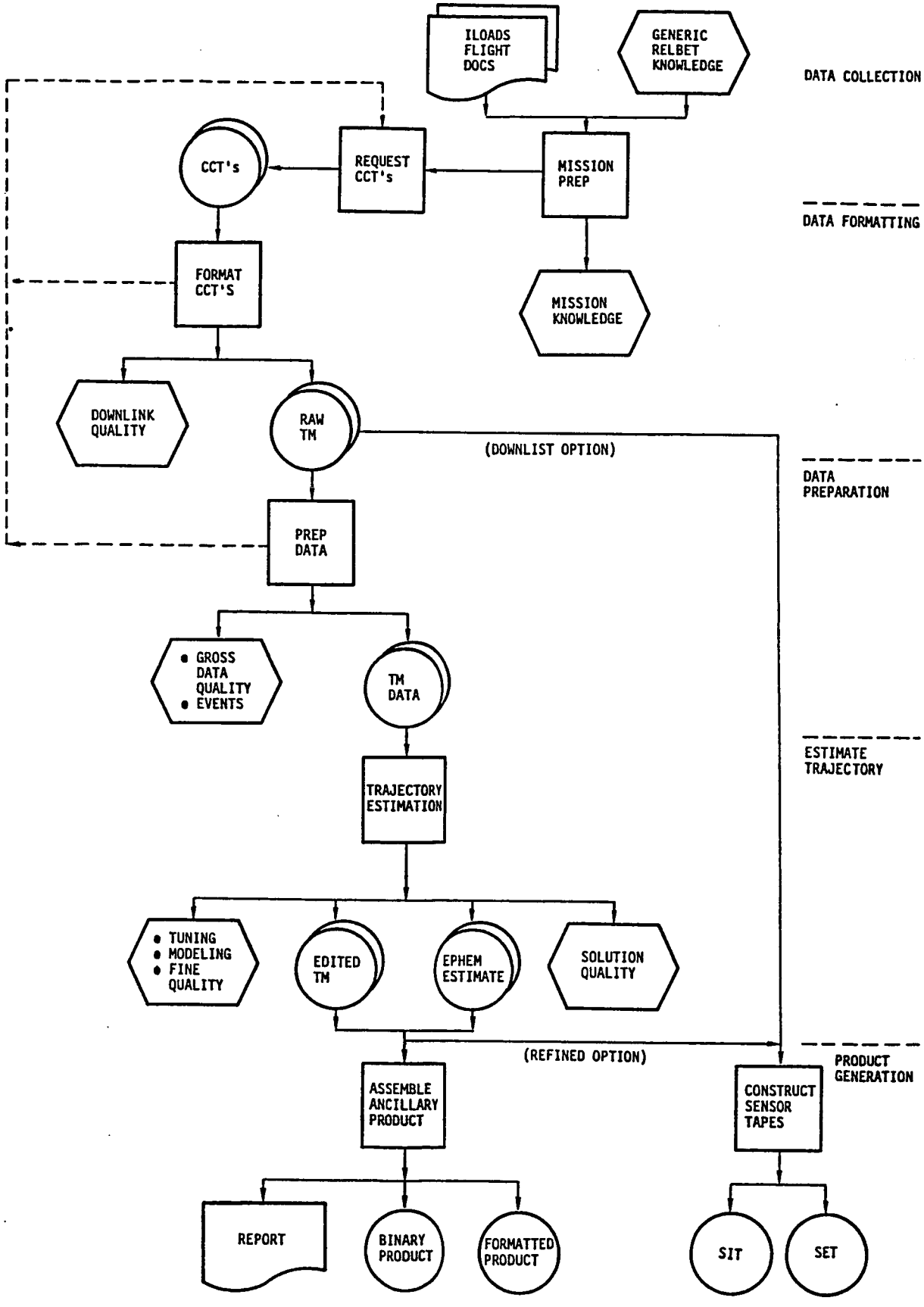


Figure 3.1.1. RELBET Production Overview

overall quality of the downlist for basic telemetry difficulties such as dropouts or invalid data. If the number of difficulties are excessive, a request for replacement CCT's may be made. This phase further examines the "raw" telemetry data and edits gross outliers. Precise times for pertinent events such as attitude maneuvers or tracking intervals are also identified. Such information is useful in resolving anomalies that may occur in subsequent processing. At the completion of this phase, one has determined that the input data are usable and and has generated information for later review and analysis.

The next phase deals with estimating the relative trajectory. One also gives the quality and consistancy of the data greater scrutiny during this phase. During this process a Kalman filter processes the telemetry. The output of the filter is reviewed to detect various anomalies. These anomalies are then either corrected or accounted for by correlating them with specific events that are known to cause problems. When satisfactory performance with the Kalman filter is obtained, its solution estimates are processed with a "smoother" program to obtain a "Best Estimate of the Relative Trajectory" (RELBET).

During the final processing phase the required output products are assembled from the various refined estimates and downlist telemetry. This activity is supported by a variety of computational, formatting and quality assurance programs.

3.2 CCT DATA FORMATTING

Before it can be used, the input telemetry data must be checked and reformatted into a format accessible by the RELBET processors.

The major source of input data for the RELBET process is the downlist telemetry contained in the CG011X CCT (Computer Compatible Tape). (CG011X is a particular shuttle data product generated for each mission.) In general the periods of interest are long enough that several CCT's are required. These must be reformatted and merged into files with the format used by the RELBET processors. Figure 3.2.1 depicts this process. The **dwnfmt** program strips

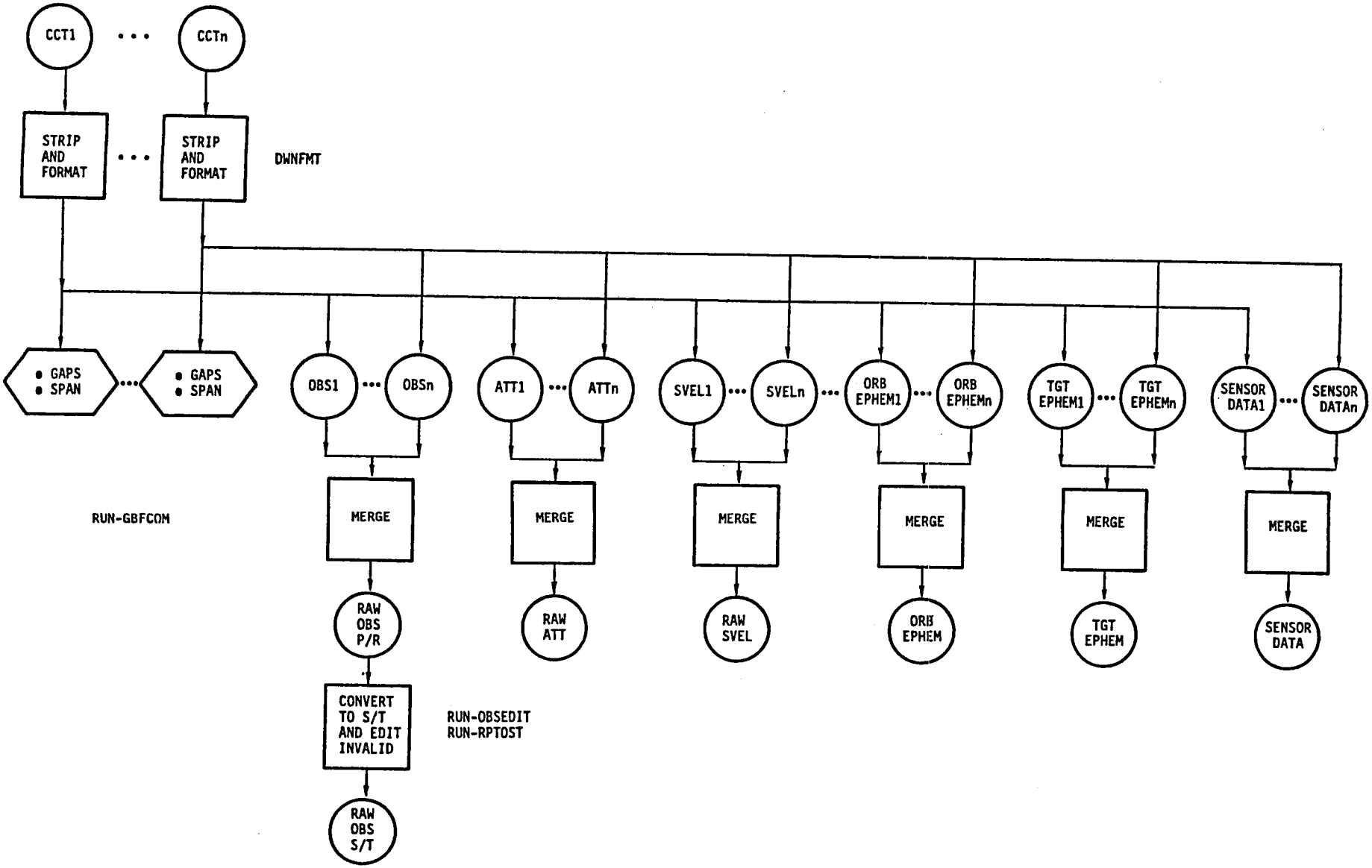


Figure 3.2.1. Format CCT's

required parameters from the CCT, checks them for validity, and generates time ordered binary files containing properly scaled parameters (5.2.1). In doing this it identifies and extracts the homogeneous data buffer created by the onboard rendezvous navigation program. This buffer is generated at the navigation cycle frequency (about a quarter of a Hertz) and downlisted asynchronously at a frequency of about one Hertz. Note that several files may be produced from single CCT, each file with its particular contents. The following files are required for later processing:

- o Attitude file: contains selected quaternions and is used to determine orbiter attitude. The following discussion uses the abbreviation "ATT" for this file.
- o Observation file: contains relative observations. The following discussion uses the abbreviation "OBS" for this file.
- o Sensed velocity file: contains selected sensed velocities for use in propagating the orbiter through burns. The following discussion uses the abbreviation "SVEL" for this file.
- o Vehicle ephemeris files: contain on-board state estimates that are used primarily for analysis and to obtain initial estimates. The following discussion uses the abbreviations "OEPH" and "TEPH" respectively for the orbiter and targer ephemerides.
- o SENSOR Information File: contains miscellaneous data required for generating the SENSOR tapes.

Several CCT's are usually needed to encompass an entire mission segment and **dwnfmt** handles only one CCT at a time. Thus the telemetry for a mission segment usually involves multiple telemetry files for each of the above types. Subsequent processors expect only one file of each type. Thus it is neccessary to merge the various files together. The shell program **run_gbfcom** will merge any number of files together (5.2.2).

The onboard systems associate a data good flag with each observation. If this flag indicates that the current observation is bad then the onboard filter does not process the datum. The shell program **run_obsedit** edits the **OBS** file for such data (5.2.3). This prevents the RELBET filters from processing bad data. The radar angles as downlisted are in the roll-pitch frame used for

crew display. The shell program **run_rptost** converts the roll-pitch angles to shaft-trunnion angles (5.2.4). The latter are preffered for filtering because they theorectically exhibit statistically independent biases.

At the conclusion of this processing phase the CCT's have been formatted into files required by the subsequent processors.

3.3 DATA PREPARATION

The input data formatting process gets the telemetry contained in the CCT's into the RELBET system. Before processing the raw telemetry with the filters or generating output products further processing is neccessary. Figure 3.3.1 depicts this process. It involves assessing the data coverage, editing bad data points, and identifying events and anomalies that are not detected while stripping the data. One then saves all this information into the event data base.

Several shell programs help in this process. The shell script **run_qacover** summarizes data coverage of all the data types on the downlisted observation, attitude, and sensed velocity files (5.3.1). These files are analyzed to determine the quality of the CCT's. If coverage is not complete, the Input Data Formatting must be repeated with a newly requested CCT. If it is complete, all problems associated with the downlink of the CCT's have been identified and handled.

The execution of the shell script **run_qadata** identifies events in the data to help in assessing filter performance later in the processing steps by providing text files and some data files in gff format which can be analyzed or plotted (5.3.2). The shell program **run_noise** runs the noise analysis procedure. It computes noise values for each observation type and enters this value into the proper slot of the observation file (5.3.3). The Kalman filter needs the noise value for its processing. The program **run_noise** also generates text files that summarize the noise information.

One then uses the text files output from **run_qadata** to edit the attitude, sensed velocity, and observation files for bad data. Several shell programs facillitate this editing process. The UNIX **sort** program merges the text files

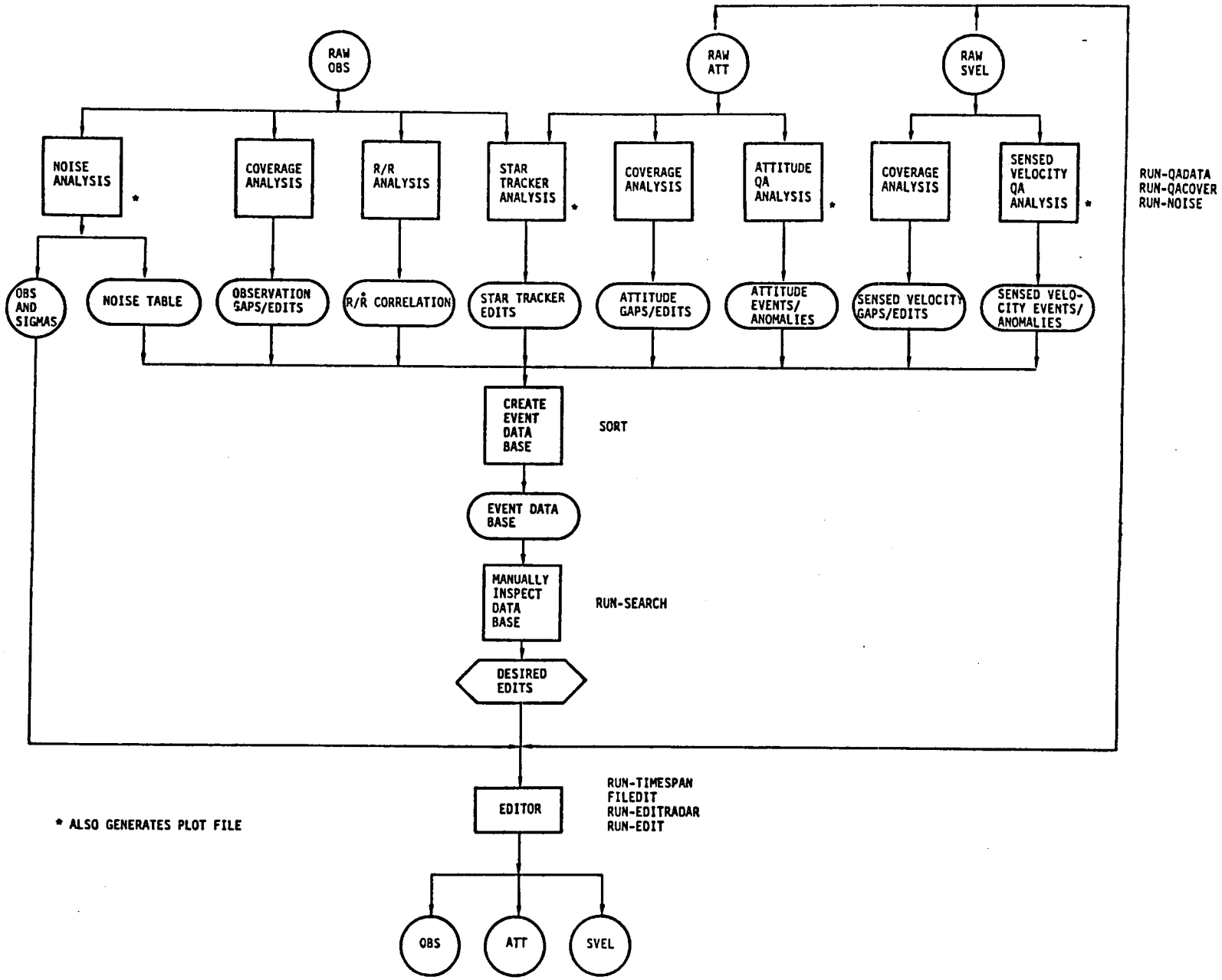


Figure 3.3.1. Data Preparation

output by these programs into the event data base. From the event data base, the shell program **run_search** (5.3.5) extracts information relating to possible difficulties. A knowledgeable engineer can review this information and identify time intervals during which files should be edited. The shell program **run_timespan** aids in preparing the time intervals (5.3.6). One uses the program **filedit** to edit the desired files. Two shell programs, **run editradar** and **run_editst** help in editing the radar and star tracker data.

At the conclusion of the data preparation phase, one has edited the input data and created a data base which will be useful for correlating problems during a particular time period. The various gff files may be plotted for more detailed analysis and review (5.3.7).

Table 3.3.1 summarizes the various QA programs.

Table 3.3.1. QA Programs

Name	Text File	Gff File
qaatt	identifies attitude manuevers and bad points on the attitude file	angular momenta and magnitudes between consecutive attitudes
qaranjmp	identifies inconsistencies in range and range rate to detect sudden biases or equipment reconfiguration	
qastar	identifies false targets that the star tracker may have locked onto	star file containing azimuth, elevation, and angle for each star tracker observation
qasv	identifies burns as jumps in the magnitudes of the sensed acceleration computed from the sensed velocity and bad data	sensed acceleration file
qanois	documents the noise for each observation type	noise files for each observation

3.4 TRAJECTORY ESTIMATION

Up until now the different input items have been looked at more or less individually. During the next processing phase they are combined together to obtain a best Kalman estimate. Figure 3.4.1 depicts this process.

The program **sfilt** uses a Kalman filter to process the various observation information and the quality of the estimate is assessed (5.4.3). The programs **sln2rl, cmp2sg,** and **rlvsrl** aid this assessment. The display programs **qplot, xqdisp,** and **gdisp** help to investigate the contents of gff files generated by the Kalman filter. In general there will be various spikes and violations of different success criteria (e.g., 3-sigma residuals, excessive Kalman edits, etc.). These are manually correlated with possible causes and corrective adjustments are effected when possible. This filter-evaluate-adjust process continues until the user decides that no further refinements are merited.

At the conclusion of this phase, the best Kalman estimate of relative trajectory has been obtained.

Kalman estimates tend to contain unrealistic spikes and also reflect uncertainties due to looking at only one side of the data arc. A smoother algorithm is used to remove the spikes and improve estimates by considering both sides of the data arc. The smoother program **smooth** processes the results of the best Kalman estimate. The results of this process are compared with the Kalman estimate with the aid of the programs **sln2rl, cmp2sg** , and **rlvsrl** and the various display programs. If there is a large discrepancy between the smoothed and Kalman estimates further adjustments may be neccessary. When a satisfactory smoothed estimate is obtained the program **sln2rl** extracts the relative trajectory information from the smoothed solution.

At the conclusion of this processing phase the best estimate of relative trajectory has been obtained.

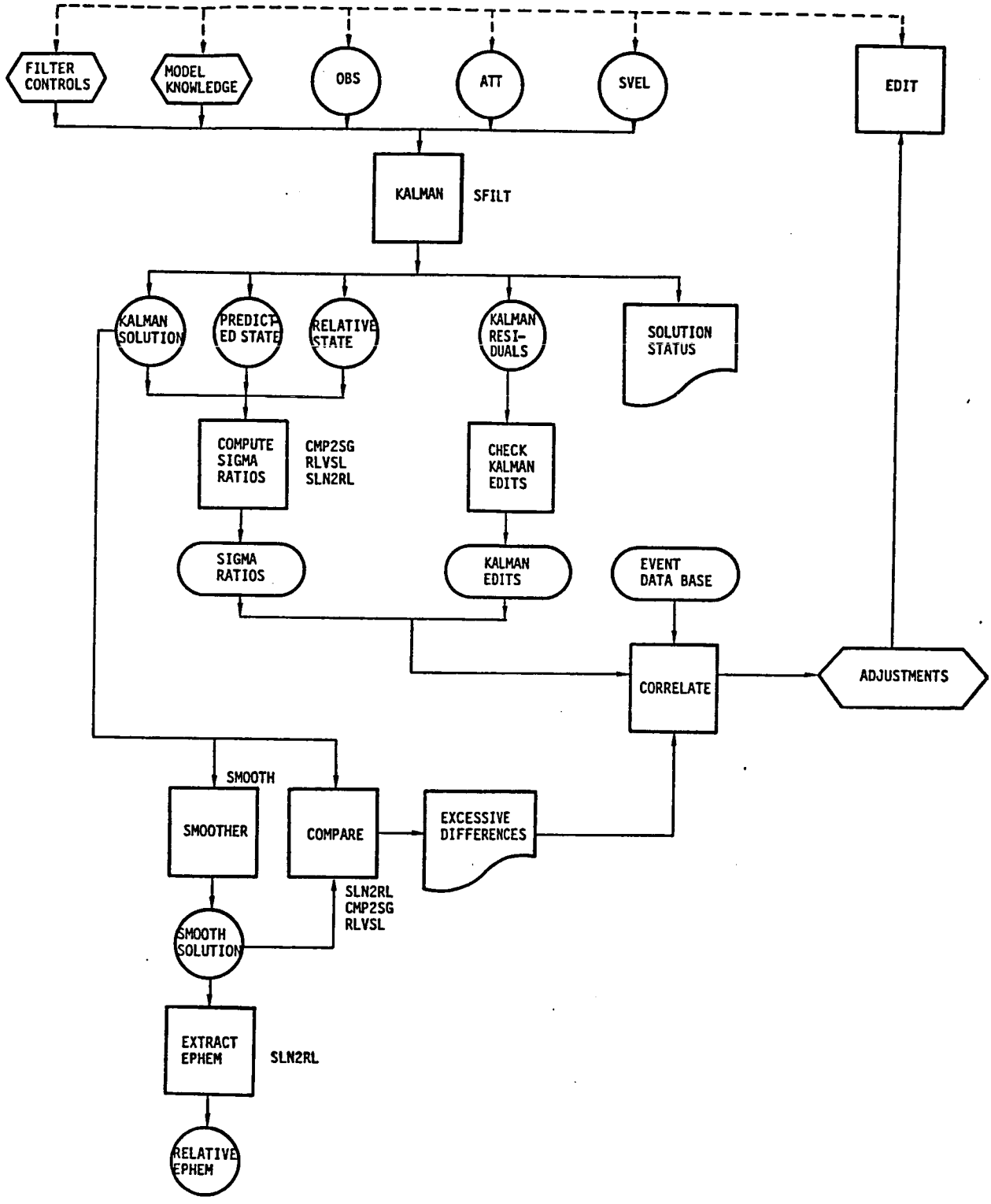


Figure 3.4.1. Trajectory Estimation

3.5 PRODUCTS GENERATION

When suitable quality telemetry data and trajectory estimates have been obtained, the output ancillary products are generated. Figure 3.5.1 depicts the process for the Ancillary Data Products. The program **prodx** is used to compute the required parameters (5.5.2). The program **qatape** is used to read the product tape and perform a check of its contents (5.5.3).

Other formatting programs such as **stop, fiche,** and **mktape** are available for generating other products such as the SENSOR tapes and microfiche print. Figure 3.5.2 depicts the process for creating the SENSOR tapes: the SIT (SENSOR Input Tape) and the SET (SENSOR Environment Tape). The final report is obtained by manually assembling various plots and tables obtained in previous processing (5.5.5).

RELBET processing is done when all identified products have been delivered.

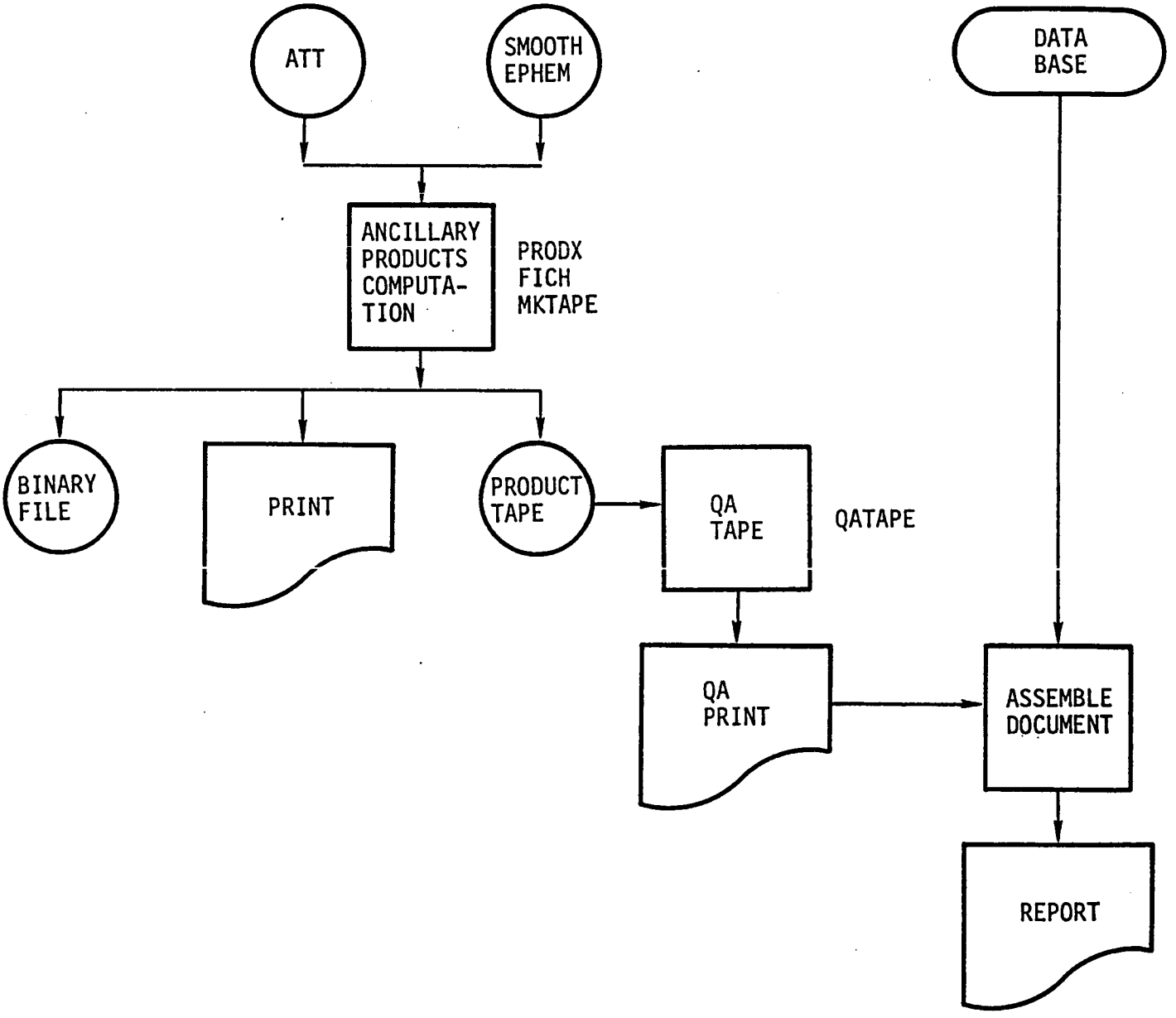


Figure 3.5.1. Ancillary Product Assembly

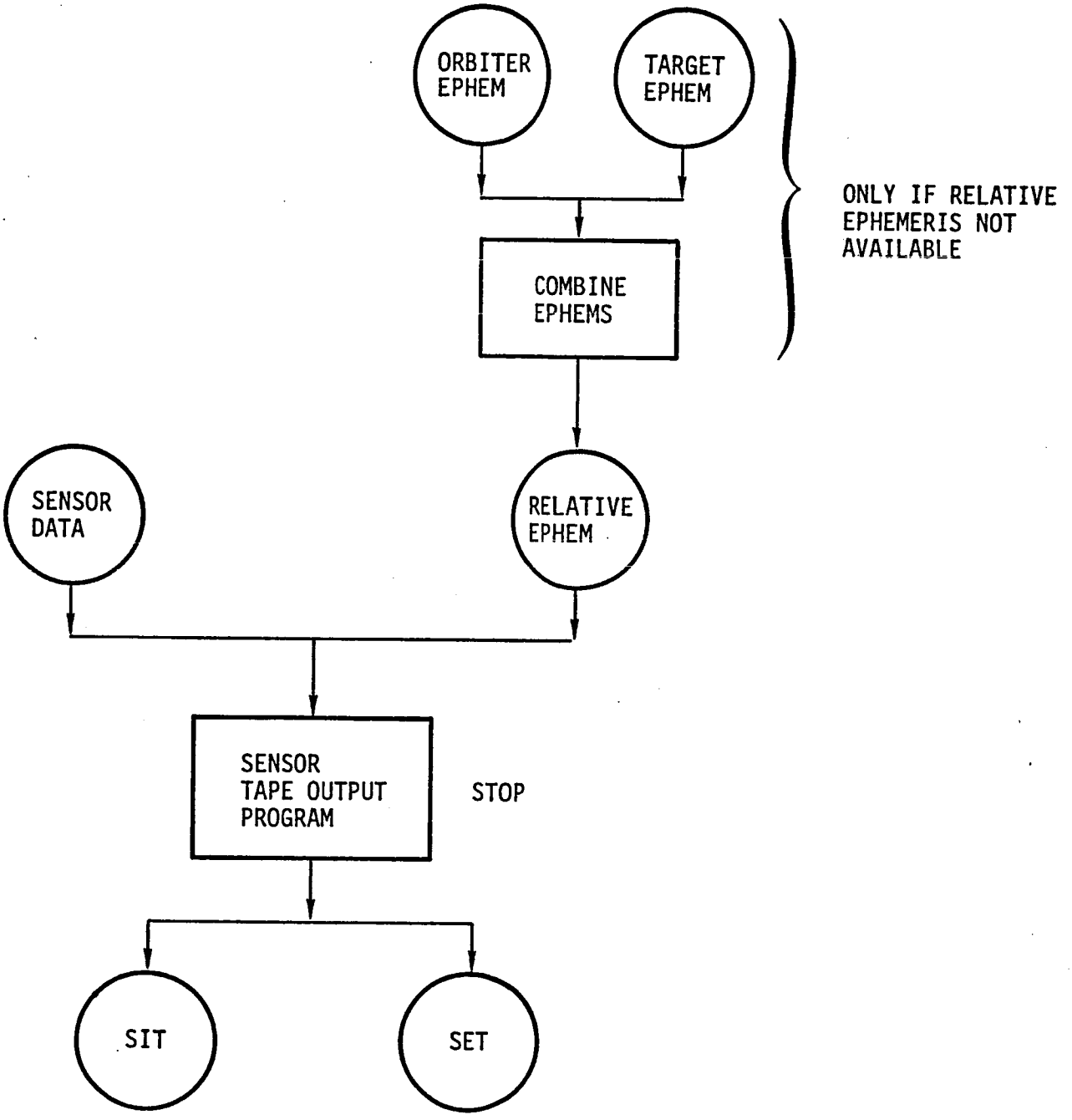


Figure 3.5.2. SENSOR Tape Construction

4.0 DESIGN OVERVIEW

The RELBET system was envisioned as a set of general purpose analysis tools that would be flexible, easily integrated, and easily maintained. The result is a design that stresses modularity, functionality, and commonality. Some key feaures of the design and implementation are:

- o Isolation of logic and data to well defined routines or blocks
- o Data structuring via blocks of functionally related variables
- o Extensive use of include directives to provide consistant context
- o A common library of utility routines organized in functional modules
- o A standardized binary file format that facillitates interprogram interfaces
- o A standardized input scheme (linput) that reflects data structuring.

In the following we expand the above and point out other general features that might not be immediately evident from reading the source code or the appendices.

4.1 BASIC PROGRAM STRUCTURE

The RELBET System includes both binary and shell programs. The main criterion for allocating whether logic occurs in a shell or binary program is that binary programs should be both simple and functionally specific. Shell scripts are used to tie together programs for more complicated applications. Thus instead of providing a single program with multiple user options, say different data type QA options, separate programs for each option are provided. This approach has two major benefits. First the logic of a single program is usually substantially reduced since it need only deal with a particular case. Secondly the user is not restricted to the logic in an executive when combining the various options together.

Thus one may identify two levels on which the programs are put together, i.e., integrated: that of shell programs and that of binary programs. The shell programs integrate binary programs together and control specific applications such as data stripping or product generation. The shell programs deal with

binary programs and particular files. At this level one may view the files as information objects and the binary programs as functions that change or filter the files. Input text files provide the detailed user interface.

The majority of the compiled software for RELBET is written in HP FORTRAN 77 using BOX. The remaining is written in C or in FORTRAN using ratfor. These programs make use of a library of utility routine modules. This means that the programs share many routines and exhibit a large commonality in techniques. Furthermore logic is generally isolated to a unique location rather that duplicated in many locations. Data structuring and access is with a standard set of blocks of functionally related variables and parameters. These blocks are maintained in their separate files and are accessed with the compiler include directive.

The binary programs generally consist of a driver and some specific routines that access the standard utility routine library. To a large extent, common blocks and context files pass information between the utility routines. To use a metaphor, the utility modules provide interchangeable components, the drivers provide a framework, and the common and context files provide the glue. Note that a given program generally does not use all modules although some modules are heavily used, e.g., **Files** and **Input.** Also note that some modules invoke routines from other modules so that there is a hierarchal dependency among the modules.

4.2 RELBET STANDARD FILE FORMATS

The main method of passing large amounts of data between RELBET processors is through the use of binary files with a standardized formats that are generically referred to as the RELBET format. This provides a standard consistant mechanism for interprogram communication. There are actually two basic formats: the **gff** (general format file), and the **gb** (general binary) format. Both consist of a fixed size header record followed by data records. They are more or less equivalent with the major differences envolving the use of certain header parameters and the size of allowable records. Appendix I discusses the file formats in detail. Routines in the **Files** directory provide access to them. The files in the **Parameters** directory and the C header file

Gbfiles.h provide information on file related data structures. Refer to Appendix I for additional information on specific **gff** file formats and to Appendix III for information on the low level routines.

WARNING: In order to maintain consistancy and simplify maintenance, a programmer should not access files directly.

NOTE: RELBET processors are not sensitive to the dictionary of frame variables found on the file headers nor do they necessarily place the GFF id in the header as do some versions of gff routines.

4.3 INCLUDE FILES

A standard set of files provide context, sizing, and design dependent parameters such as type declarations, data flag values, and file formats. The directory **Parameters** contains FORTRAN parameter statement blocks, and the directory **Linklib** provides C header or ".h" files. The source uses "include" statements to access them. Appendix II provides details.

The main vehicle for passing information between the various subroutines making up FORTRAN and ratfor programs is through FORTRAN common. The common blocks are contained in the directory **Mtfcommon** and include dimension, type and commented definitions for each variable in the named common. Note that the MTF utilities are available for their maintenance.

Many of the common blocks have defaults for some of the variables. These are provided using FORTRAN data statements associated with each named common. These data blocks appear in the block data subroutines associated with the various programs and are in the directory **Mtfdefault.**

Many of the common blocks have user input available for some of the variables. The common inputs are accessed by the user though a set of C language routines in the directory **Linput.** The interface calls between the FORTRAN and linput routines are done on a variable by variable basis, and the includes associated with the interface are under the directory **Mtfinput.**

The directory **Parameters** contains various "parameter" declarations. These deal primarily with FORTRAN and ratfor file I/O routines. These versions are basically copies of one another with appropriate changes.

The C language routines pass information to each other by the use of the data structures and typedefs. The definitions of these objects are normally found in header files (.h files) which are included in the C routines and are listed under **Linklib.**

All the include files are contained in Appendix II.

4.4 SUBROUTINE MODULES

Modules group together routines with related functions such as coordinate transformations or matrix operations. Routines unique to a given program receive their own module. In general, a module is shared by many programs. However, a particular piece of logic is isolated to a single routine. The directories reflect the modular organization. Section 5.0 provides an overview of the directories and the modules. Appendix III provides a synopsis of the contents of the modules and the routines. It also provides a cross reference between identifiers and modules. Appendix IV provides source code.

4.5 LINPUT INPUT SCHEME

The linput input scheme provides a standard user interface. The mtf utility **mtfuser** generates blocks of input parameters in the **Mtfinput** directory from an associated mtf text file. The "xx" routines provide the interface between the input text and the common blocks.

5.0 DIRECTORY STRUCTURE

The RELBET directory hierarchy structure is quite simple: a single master directory with about 40 subdirectories. Each of these subdirectories contains a group of closely related files. The basic benefit of this approach is that it greatly simplifies relative pathnames for includes. For the purpose of discussion, however, it is convenient to group directories as to their contents or function. The directories accordingly fall into the following major groups:

- o Control Directories that contain executables and input data
- o Program Directories that contain program unique code
- o Context Directories that contain data structure information such as common blocks and parameters
- o Utility Directories that contain general purpose utility routines

5.1 CONTROL DIRECTORIES

These directories contain the executable binary programs, shell scripts, and program maintenance directives and utilities.

Control	Executable and production shell scripts
Make	Make directives used to recreate executables
Test	Sample inputs and shell scripts
Tools	Maintenance tools
Manual	Program user manuals
Pmanual	Routine Manuals

5.2 PROGRAM DIRECTORIES

Table 5.1 lists those directories associated with particular programs or executable objects. In general a program is matched with the directory containing its driver and other routines specific to that application. Such directories are referred to as program directories.

Table 5.1. Program Directories

Directory	Program Name	Description
Downfor	dwnfmt	Downlist Formatter
	rptost	Roll/pitch to shaft trunnion observation convention
Filter	sfilt	Kalman filer
	smooth	Smoothing Filter
Fich	fiche	Create microfiche
Fman	cmp2sg	Compare state difference to sigma
	eph2rel	Convert ephemerides to relative trajectory
	filedit	File editor
	gbfcom	Combine gff files
	mkinit	Make the init input block
	obsnois	Place noise data on obs file
	qaatt	QA the attitude file
	qacover	QA the data coverage
	qanois	QA the data noise
	qaranjmp	QA the range observations
	qastar	QA the star tracker obs
	qasv	QA the sensed velocity
	rdwt	Read file then write file
	read_set	Read SET file and print
	read_sit	Read SIT file and print
	rlvsrl	Compare relative trajectory files
	search	Extract data from event data base
	sln2rl	Strip relative trajectory from solution file
	stop	Create SENSOR output tapes
Mktape	mktape	Create output UNIVAC tape
Noisanal	ddnois	Compute obs data noise
Numdis	gdisp	General gff display
	xcmpar	Compare two ephermeris files
	xqdisp	State/Attitude Parameter Display
Plot	ascale	Auto scale
	plotx	Graphic display of gff files
Product	prodx	Special product generation
Qatape	qatape	QA special product tape

5.3 CONTEXT DIRECTORIES

These directories contain text files which are "included" in FORTRAN or C language routine by the compiler to provide context information and data interfaces.

Mtf	MTF master textfiles
Mtfcommon	Common declarations
Mtfdefault	Data statements for common blocks
Mtfinput	linput interface calls for user inputs into common blocks
Parameters	FORTRAN and ratfor parameter declarations
Linklib	Header (.h) files used by C routines

5.4 UTILITY DIRECTORIES

These directories comprise a library of general purpose utility routines. They are used by various programs as opposed to the Program Directories whose routines are not required by any other program. These directories generally form packages of related routines and are accordingly synonomous with modules.

5.4.1 File Routines

Two sets of gff file routines are provided as well as a set of FORTRAN interfaces to the gff routines are used to actually invoke the gff routines.

Gff	FORTRAN routines modeled after the orignal gff routines
Gbfile	C routines allowing more general access to creating and processing of binary files used by **Downfor** and **Fman** processors
Files	The gff interface routines used by the programs **Kalman, Noisanal, Numdis, Plot, Product,** and **Smooth**

5.4.2 Input Manipulation Routines

These directories are involved in the process of reading and preparing user input for use by the processors.

Linput	The C routines which form the linput input scheme and the interfaces with FORTRAN
Lists	The C routines which manipulate the data structures used by linput
Input	FORTRAN routines for preliminary processing and display of inputs received through linput
Prompts	FORTRAN routines related to menu generation

5.4.3 Math Routines

These directories provide general mathematical tools.

Coordinate	Coordinate transformation routines
Interpolate	Interpolation routines
Math	Trig, matrix, vector, and quaternion routines

5.4.4 Propagation Routines

These directories contain routines for propagating vehicle trajectories.

Celestial	Compute positions of celestial objects
Force	Compute forces on an orbiting body
Propagate	Routines to initialize and drive the propagation process

5.4.5 Data Structure Manipulations

The routines in these directories manipulate various types of data or data structures. They include C and FORTRAN routines.

Charutil	Manipulate FORTRAN character strings
Dsputil	Gff file header frame data displays
Message	Provide general display of character strings related to warnings and other status messages
Stacks	Routines to manipulate stack data structures
Time	Routines for conversion and manipulation of time data

5.4.6 Observation Computation

This directory contains routines related to observation processing. The Kalman and Smoother filter programs need these routines.

Obs	Observation computation routines

6.0 INSTALLATION AND MAINTENANCE

The delivery tape is a tape archival of a configured RELBET directory installed on the TRW Houston System Services HP9000 system configuration. This installation reflects the system environment described in section 1.3, Hardware/Software Environment, of this manual. The installation is geared to maintaining configuration control and reflects a single user owner of all files. The maintenance is supported by a variety of standard UNIX and TRW utilities.

6.1 INSTALLATION

To install the delivery tape onto your system, you must read the delivery tape into a directory with a proper user environment. Depending on your needs and your system environment you may also need to reconfigure the user environment and device access.

6.1.1 Creating a RELBET Master Directory

All files on the tape are owned by user Relbet belonging to group Relbet. The examples in the User's Manual and all the shell programs assume that the directory **/users/Relbet/Master/Control** is included in the environment variable PATH. To use the maintenance tool include **/users/Relbet/Master/Tools/Abs** in PATH. The login shell should be sh.

6.1.2 Reading the Delivery Tape

The delivery tape is a 1600 bpi 9 track tape created from a configured Master RELBET directory using the UNIX tape archival utility **tar**. It was created with the command

```
tar cvf /dev/tape0 *
```

while in the master RELBET directory. You can use a procedure similar to the following to to read the tape into your master own RELBET directory.

```
cd /users/Relbet/Master
#assign tape
mt -t /dev/tape0 rew          #rewind
tar xvf /dev/tape0            #read tape
#release tape
```

6.1.3 Tailoring the System to Your Environment

The delivered version may not reflect your system environment so that it may be neccessary to make a few alterations before proceeding. The most likely candidates are display devices and maintenance tools.

6.1.3.1 Graphic Display Device Configuration

The delivery version of the graphics display program **plotx** uses the DISSPLA graphics library with devices configured for the HP150 terminals and the HP9872B plotter. The make directives assume the DISSPLA library is contained in the directory **/usr/lib/Disspla.**

6.1.3.2 Tools Configuration

The directory **Tools/Abs** contains links to executables for various code generation and documentation utilities. The source for these is found under the directories in **Tools.** Note that this directory uses multiple links to executables and documentation. You may need to reconfigure the line and page length controls for the text formatters depending upon your line printer. These controls are mostly found as arguements in shell scripts such as the **fmt** programs.

6.2 MAINTENANCE

The RELBET System was designed to be built and maintained using a variety of UNIX and TRW developed tools. The following provides an overview of their application. Look in **Tools/Manuals** for further information.

6.2.1 Configuration Control

The system is delivered with all source under **sccs** cntrol with a base version of 4.1. The tools under the **Tools** directory are also under **sccs** control, but with a base version of 1.1. As delivered only the system owner (user Relbet) can alter these files with programs such as **delta**. This scheme is designed to support a strict configuration control.

6.2.2 Recreating Programs

The directory **Make** contains directives for the UNIX **make** program. To create a new executable run **make** with the appropriate directives.

6.2.3 Modifying Common, Input, Default, etc., Blocks

The system was designed to use the MTF (Master Text File) utilities to create and maintain information associated with blocks of related variables. To alter one of these objects, modify the associated MTF file, found in **Mtf**, and run the appropriate reformatting program:

mtf_blk	generates FORTRAN common code
mtf_npt	generates linput invocations for input
mtf_data	generates FORTRAN data statements
mtf_user	generates user manual entries

Refer to the documentation section below for how to generate cross references.

6.2.4 Documentation

Listings may be obtained with the shell program **srclist**. Note that this shell uses the program **bxp** rather than the standard **blist** program which expands all includes.

Manual entries are generally available in the **txman** format for reformatting into the **nroff/man** format. Program manuals must be manually produced. Routine manuals may be automatically generated using the programs bcmanuals, **boxstrip** and **cstrip**. Cross references may generated with procedures like those outlined in the shell scripts **mtfcmntoc** and **mtfnpttoc**.

APPENDIX I

FILE FORMATS

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

1.0 INTRODUCTION

The RELBET System uses a variety of files for interprogram communication. The files may be grouped into four categories:

- o External data files that provide system input or output
- o Internal files used by the System to communicate between programs
- o Display files generated by the System
- o Data base files.

2.0 EXTERNAL FILES

These are files with specific formats and are treated as foreign by the RELBET System. They include input and product data files.

INPUT	Downlist CCTs
OUTPUT	RELBET Product Tape RELBET Microfiche Text SENSOR Input Tape (SIT) SENSOR Environment Tape (SET)

2.1 DOWNLIST CCT

The Downlist File contains on-orbit observation data and time-tags associated with various on-board sensors. By collecting and reformatting these data RELBET constructs files for sensed velocities, attitudes and observations as well as ephemerides for the Orbiter and target vehicle. The format of this tape is described in the CCT Interface Control Document, reference 1 at the end of the Appendix.

2.2 RELBET PRODUCT FILE

The program **prodx** generates the RELBET Special Products File. It utilizes trajectory and attitude information from input standard format files to produce data product files containing various parameters over a specified time interval. Output includes print describing the input and processing status. Options include tape, binary, and formatted data product files.

2.2.1 Binary Data Products File

The user has the option to generate a binary data products file corresponding to the RELBET Ancillary Data Products described in Reference 2 at the end of this Appendix. This file is tape or mass storage according to the user assignment. This multi-record file has the following general format:

First Record:	Identifier record
Second Record:	Dictionary record
Succeeding Records:	UNIVAC binary data records
END OF FILE	

The first record has a fixed length of 26 UNIVAC single precision words. The dictionary and data records all have the same length; however, this length varies depending on the particular parameters desired. Table 2.2.1 depicts the overall file format.

The file identifier record provides the user with information to identify the file contents. The format is shown in Table 2.2.2. The first twelve (12) words constitute a generic identifier message. The thirteenth word is the alphabetic "SPEC" which identifies the data as a special BET product. The next twelve integer words specify the time period in Greenwich Mean Time (GMT) covered by the data records. The last word is an integer defining the number of parameters in the dictionary.

Table 2.2.1. File Format

RECORD	INFORMATION
1	Tape Identifier
2	Parameter Dictionary
3	Data
.	.
.	.
.	Monotonically
.	Increasing
.	Time
.	.
.	.
L	Data

Table 2.2.2. File Identifier Record Format

SINGLE WORD NUMBER	DEFINITION		TYPE
1 12	48 character identifier message flight no., data etc.		alphabetic (single)
13	SPEC		alphabetic (single)
14	tape start GMT (first data record)	YR	
15		MO	
16		DAY	
17		HR	integer (single)
18		MIN	
19		SEC	
20 25	tape stop GMT (last data record)	YR	integer (single) SEC
26	number of parameters in dictionary		integer (single)

The dictionary record format shown in Table 2.2.3 informs the user of the available parameters and their relative location on the data records. The first word is an integer defining the number of parameters in the dictionary. The remaining N alphabetic words identify the relative location of the value of the parameter.

The data record format shown in Table 2.2.4 provides the double precision parameter data in a format where the value for a parameter is in the corresponding word location of that parameter symbol in the dictionary record. Note that the first word constitutes a last record flag.

2.2.2 Output Contents

A formatted Data Products file will be generated at user option. This file contains the same parameters at the same frequency as the binary data products file. It consists of a dictionary description followed by displays of each output record.

2.2.3 Parameter Groups

The ancillary parameters are divided into 17 groups with the option to include or omit each group. These parameter groups are described in Table 2.2.5. Note that the continuation flag (entry 1, dictionary name CONTINUE) is always included. Although any subset of these 17 groups may be selected, a fatal error will result if none are chosen or sufficient input information to compute parameters is not provided.

Table 2.2.3. Dictionary Record Format

DOUBLE WORD NUMBER	VARIABLE DEFINITION	TYPE
1	N = No. of parameters in dictionary	integer (double)
2	Symbol for parameter in location 2 on data record (1st variable)	alphabetic (double)
3	Symbol for parameter in location 3 on data record (2nd variable)	alphabetic (double)
.		
.		
.		
N + 1	Symbol for parameter in location N + 1 on data record (Nth variable)	alphabetic (double)

Note: Value for

Variable no. 1 is the double precision word 2 on data record.

Variable no. 2 is the double precision word 3 on data record.

.
.
.

Variable no. N is the double precision word N + 1 on data record.

Table 2.2.4. Data Record Format

WORD NUMBER	VARIABLE DEFINITION	TYPE
1	ITYPE 0 = continuing records	Double Precision
	1 = last record in file	
2	Value of parameter in location 2 on dictionary record (1st variable)	Double Precision
3	Value of parameter in location 3 on dictionary record	Double Precision
N + 1	Value of parameter in location N + 1 on dictionary record	Double Precision

Note: N is number of parameters read from previous dictionary record.

Table 2.2.5. Special Products Parameters

VARIABLE SET	PARAMETER GROUP NAME	SYMBOL	PARAMETER	UNITS
Time, Greenwich mean (GMT) Orbiter onboard	1	SGMTY		yr
		SGMTMO		month
		SGMTD		day
		SGMTH		hr
		SGMTM		min
		SGMTS		sec
Time, Greenwich mean (GMT) Ground - MCC UTC	2	GMTY		yr
		GMTMO		month
		GMTD		day
		GMTH		hr
		GMTM		min
		GMTS		sec
Time, ground elapsed (GET)	3	GETH		hr
		GETM		min
		GETS		sec
State Vector, Orbiter, in Aries mean of 1950 Cartesian coordinate System	4	XM	x	km
		YM	y	km
		ZM	z	km
		XDM	$\dot{x}$	km/sec
		YDM	$\dot{y}$	km/sec
		ZDM	$\dot{z}$	km/sec
State Vector, Orbiter, in free flyer LVLH Cartesian coordinates	5	SLVX	x	km
		SLVY	y	km
		SLVZ	z	km
		SLVXD	$\dot{x}$	km/sec
		SLVYD	$\dot{y}$	km/sec
		SLVZD	$\dot{z}$	km/sec
State Vector, Orbiter, in free flyer UVW Cartesian coordinates	6	SU	u	km
		SV	v	km
		SW	w	km
		SUD	$\dot{u}$	km/sec
		SVD	$\dot{v}$	km/sec
		SWD	$\dot{w}$	km/sec
Attitude, Orbiter, Euler angles, body axis with respect to Orbiter UVW coordinates	7	ALPHU	yaw	deg
		BETAU	pitch	deg
		PHIU	roll	deg

Table 2.2.5. Special Products Parameters (Continued)

VARIABLE SET	PARAMETER GROUP NAME	SYMBOL	PARAMETER	UNITS
Attitude, Orbiter, Transformation, Aries mean of 1950 to body matrix	8	A11		n.d.
		A12		n.d.
		A13	matrix	n.d.
		A21	elements	n.d.
		A22	row	n.d.
		A23	ordered	n.d.
		A31		n.d.
		A32		n.d.
		A33		n.d.
Attitude, Orbiter, quaternion, Aries mean of 1950 to body axes	9	Q1	scalar part	n.d.
		Q2		n.d.
		Q3	vector	n.d.
		Q4		n.d.
Attitude, rate, Orbiter, angular velocities and total rate, body about Aries mean of 1950 Cartesian coordinates	10	YDOTB	rate about z	deg/sec
		PDOTB	rate about y	deg/sec
		RDOTB	rate about x	deg/sec
		TDOTB	total rate	deg/sec
Look angle, and angle rates orbiter body to free flyer	11	AZ	yaw	deg
		EL	pitch	deg
		RAZ	yaw rate	deg/sec
		REL	pitch rate	deg/sec
Range, and range rate, between vehicle reference points	12	RANGE	range	km
		RRATE	range rage	km/sec
Simulation Flag	13	PSIM		n.d.
State Vector, free flyer, in Aries mean of 1950 Cartesian coordinates	14	PMX	x	km
		PMY	y	km
		PMZ	z	km
		PMXD	$\dot{x}$	km/sec
		PMYD	$\dot{y}$	km/sec
		PMZD	$\dot{z}$	km/sec
State Vector, free flyer, in Aries mean of 1950 Cartesian coordinates relative to Orbiter	15	PRMX	x	km
		PRMY	y	km
		PRMZ	z	km
		PRMXD	$\dot{x}$	km/sec
		PRMYD	$\dot{y}$	km/sec
		PRMZD	$\dot{z}$	km/sec

Table 2.2.5. Special Products Parameters (Continued)

VARIABLE SET	PARAMETER GROUP NAME	SYMBOL	PARAMETER	UNITS
State Vector, free flyer, in Orbiter UVW Cartesian coordinates	16	PU	u	km
		PV	v	km
		PW	w	km
		PUD	$\dot{u}$	km/sec
		PVD	$\dot{v}$	km/sec
		PWD	$\dot{w}$	km/sec
State Vector, free flyer, in Orbiter LVLH Cartesian coordinates	17	PLVX	x	km
		PLVY	y	km
		PLVZ	z	km
		PLVXD	$\dot{x}$	km/sec
		PLVYD	$\dot{y}$	km/sec
		PLVZD	$\dot{z}$	km/sec

2.3 SENSOR INPUT TAPE

The program **stop** (Sensor Tape Output Processor) generates two files in the Sensor Input Tape (SIT) and Sensor Environment Tape (SET) formats. These formats are described in reference 3 at the end of this Appendix. Both are input for the SENSOR program. The SIT file contains relative observation, attitude, sensed velocity, and vehicle state information. Each record has length of 80 integer words and consists of 59 parameters and one unused position per record. Table 2.3.1 describes the format and also provides associated units and data types. Note that single precision and integer data displace one integer word and double precision data displaces two integer words.

Table 2.3.1. Data Format for Sensor Input Tape

	PARAMETER DESCRIPTION	SOURCE/ M/SID	ENG. UNITS	RECORD LOCATION	RECORD TYPE
1.	Year	V90W4749C	Years	1	I
2.	Month	"	Months	2	I
3.	Day	"	Days	3	I
4.	Hour	"	Hours	4	I
5.	Minute	"	Minutes	5	I
6.	Seconds	"	Seconds	6	I
7.	GMT Time	"	Seconds	7	D
8.	M50 Shuttle - x	V90H4277C-79C	Feet	9	D
9.	- y	"	Feet	11	D
10.	- z	"	Feet	13	D
11.	- $\dot{x}$	V90L4281C-83C	Feet/Sec	15	D
12.	- $\dot{y}$	"	Feet/Sec	17	D
13.	- $\dot{z}$	"	Feet/Sec	19	D
14.	COAS Data Good Flag	V90X4849X	--	21	I
15.	COAS Horizontal Measurement	V90U4847C	Radians	22	S
16.	COAS Vertical Measurement	V90U4848C	Radians	23	S
17.	RNDZ Radar Angles Flag	V90X4901X	--	24	I
18.	RNDZ Radar Roll	V90U4893C	Degrees	25	S
19.	RNDZ Radar Pitch	V9074894C	Degrees	26	S
20.	RNDZ Range Flag	V90X4899X	--	27	I
21.	RNDZ Range Measurement	V90U4895C	Feet	28	S
22.	RNDZ Range Rate Flag	V90X4900X	--	29	I
23.	Rate Measurement	V90U4896C	Feet/Sec	30	S
24.	Star Tracker Data Good Flag	V90X4835X	--	31	I
25.	Star Tracker Horiz. Meas.	V90U4833C	Radians	32	S
26.	Star Tracker Vert. Meas.	V90U4834C	Radians	33	S
27.	Nav. Powered Flight Flag	V90X4264X	--	34	I
28.	M50 Accum. - x (CG ref.)	V90L8927C-29C	Feet/Sec	35	D

Table 2.3.1. Data Format for Sensor Input Tape (Continued)

	PARAMETER DESCRIPTION	SOURCE/ M/SID	ENG. UNITS	RECORD LOCATION	RECORD TYPE
31.	Current Orbiter Mass	V90U4254C	Slugs	41	D
32.	M50 Target - x	V90H4287C-89C	Feet	43	D
33.	- y	"	Feet	45	D
34.	- z	"	Feet	47	D
35.	- $\dot{x}$	V90L4291C-93C	Feet/Sec	49	D
36.	- $\dot{y}$	"	Feet/Sec	51	D
37.	- $\dot{z}$	"	Feet/Sec	53	D
38.	M50 to Body Quaternion	V95U0873C-76C	--	55	S
39.	"	"	--	56	S
40.	"	"	--	57	S
41.	"	"	--	58	S
42.	DA Threshold	V90A4747C	Micro g sec	59	D
43.	DAP Jet Flag	V90X5185X	--	61	I
44.	Unused		--	62	I
45.	Star Tracker Matrix	V90U4920C-28C	--	63	S
46.	"	"	--	64	S
47.	"	"	--	65	S
48.	"	"	--	66	S
49.-53.				67-71	S
54.	Star Tracker Time-tag	V90W4837C	Seconds	72	D
55.	COAS Matrix	V90U4857C-65C	--	74	S
56.	"	"	--	75	S
57.	"	"	--	76	S
58.	"	"	--	77	S
59.-63.				78-82	S
64.	COAS Time-tag	V90W4853C	Seconds	83	D
65.	Rendezvous Radar Quaternion	V90U4829C-32C	--	85	S
66.	"	"	--	86	S
67.	"	"	--	87	S
68.	"	"	--	88	S
69.	Rendezvous Radar Time-tag	V90W4841C	Seconds	89	D

2.4 SENSOR ENVIRONMENT TAPE (SET)

Each record has length of 104 integer words and contains 64 downlisted parameters and one unused space ordered as specified in Table 2.4.1. The table also provides associated units, and data types. Note that single precision and integer data displace one integer word and double precision data displace two integer words.

Table 2.4.1. Data Format for Sensor Environment Tape

	PARAMETER FSSR NAME	SOURCE/ M/SID	ENG. UNITS	RECORD LOCATION	TYPE
1.	DA.THRESHOLD	V90A4747C	f/s	1	S
2.	DVDISP(1)	V90L5181C	f/s	2	S
3.	DVDISP(2)	V90L5182C	f/s	3	S
4.	DVDISP(3)	V90L5183C	f/s	4	S
5.	E(1,1)	V90U4006C	n	5	D
6.	E(2,2)	V90U4027C	n	7	D
7.	E(3,3)	V90U4047C	n	9	D
8.	E(4,4)	V90U4067C	n	11	D
9.	E(5,5)	V90U4087C	n	13	D
10.	E(6,6)	V90U4108C	n	15	D
11.	E(1,2)	V90U4007C	n	17	D
12.	E(1,3)	V90U4008C	n	19	D
13.	E(1,4)	V90U4009C	n	21	D
14.	E(1,5)	V90U4011C	n	23	D
15.	E(1,6)	V90U4012C	n	25	D
16.	E(2,3)	V90U4028C	n	27	D
17.	E(2,4)	V90U4029C	n	29	D
18.	E(2,5)	V90U4030C	n	31	D
19.	E(2,6)	V90U4031C	n	33	D
20.	E(3,4)	V90U4048C	n	35	D
21.	E(3,5)	V90U4049C	n	37	D
22.	E(3,6)	V90U4050C	n	39	D
23.	E(4,5)	V90U4068C	n	41	D
24.	E(4,6)	V90U4069C	n	43	D
25.	E(10,10)	V90U4088C	n	45	D
26.	NAV.ANGLES.AIF	V90J4253C	n	47	I
27.	NAV.RANGE.AIF	V90J4268C	n	48	I
28.	NAV.RDOT.AIF	V90J4269C	n	49	I
29.	NAV.SV.SOURCE.FOR.UPP	V90X4977X	n	50	I
30.	NAV.DO.COVAR.REINIT	V90X4256X	n	51	I
31.	NAV.DO.FILTERED.TO.PROP	V90X4975X	n	52	I
32.	NAV.DO.FILTR.SLOW.RATE	V90X4257X	n	53	I

Table 2.4.1. Data Format for Sensor Environment Tape (Continued)

	PARAMETER FSSR NAME	SOURCE/ M/SID	ENG. UNITS	RECORD LOCATION	TYPE
33.	NAV.DO.ORB.TO.TGT	V90X4258X	n	54	I
34.	NAV.DO.OV.UPLINK	V90X4260X	n	55	I
35.	NAV.DO.PROP.TO.FILTERED	V90X4976X	n	56	I
36.	NAV.DO.TGT.TO.ORB	V90X4259X	n	57	I
37.	NAV.DO.TV.UPLINK	V90X426AX	n	58	I
38.	NAV.MEAS.ENABLE	V90X4262X	n	59	I
39.	NAV.MM.CODE	V90J4263C	n	60	I
40.	NAV.PWRD.FLT.NAV	V90X4264X	n	61	I
41.	NAV.RR.ANGLES.ENABLE	V90X4273X	n	62	I
42.	NAV.ST.ENABLE	V90X4274X	n	63	I
43.	REND.NAV.FLAG	V93X6220X	n	64	I
44.	T.CURRENT.FILT	V90W4749C	s	65	D
45.	TFOFF	V90W4960C	s	67	D
46.	TFON	V90W4959C	s	69	D
47.	T.ORB.STATE.UPDATE	V94W3727C	s	71	D
48.	T.TV.STATE.UPDATE	V90W4939C	s	73	D
49.	V.FORCE(1)	V90U4956C	LBF	75	S
50.	V.FORCE(2)	V90U4957C	LBF	76	S
51.	V.FORCE(3)	V90U4958C	LBF	77	S
52.	UNUSED			78	I
53.	T.LAST.FILT.TLM	V90W4285C	s	79	D
54.	R.FILT.TLM(1)	V90H4277C	f	81	D
55.	R.FILT.TLM(2)	V90H4278C	f	83	D
56.	R.FILT.TLM(3)	V90H4279C	f	85	D
57.	V.FILT.TLM(1)	V90L4281C	f/s	87	D
58.	V.FILT.TLM(2)	V90L4282C	f/s	89	D
59.	V.FILT.TLM(3)	V90L4283C	f/s	91	D
60.	R.TV.TLM(1)	V90H4287C	f	93	D
61.	R.TV.TLM(2)	V90H4289C	f	95	D
62.	R.TV.TLM(3)	V90H4289C	f	97	D
63.	V.TV.TLM(1)	V90L4291C	f/s	99	D
64.	V.TV.TLM(2)	V90L4292C	f/s	101	D
65.	V.TV.TLM(3)	V90L4293C	f/s	103	D

3.0 INTERNAL FILES

The Internal Files provide the communication between programs in RELBET. RELBET creates these files as outputs from various programs. User editing is possible through the editing processors. All internal files are in internal units (meter, kilogram, second, radian). Internal inertial coordinates are Aries mean of 1950 Cartesian, although other systems may be used for file storage or display.

3.1 GENERAL FORMAT

Internal RELBET files share a common overall format consisting of headers followed by data records that are called frames.

3.1.1 Format Compatibility

The RELBET internal files are generally compatible with the gff format described below and in reference 2. However some differences do occur.

Two sets of file routines exist in RELBET called the gff and gbfile routines. The gff routines manipulate files as specified in reference 4. The gbfile routine support the basic design of the so called gff files, however, they do not utilize the dictionary records though space is allowed for compatibility, and they do not set the program name/version number parameter hcver on the header record. In fact the gbfile routines being written in C do not use the IOP (I/O packet at all). Thus non-RELBET processors which read gff files may not be able to read files created by the gbfile routines.

3.1.2 Overall File Format

A gff file is defined by its format. There are two definition tables in the file, the header and the dictionary. The header provides file information and the dictionary provides data information. Following the header and dictionary is the data itself. Below is a diagram of a gff file.

Record	Description
1,n	Header (n = number of header records)
n+1,n+5	Dictionary
n+6	*beg
n+7,n+7+m	data frames (m = number of data records
n+7+m+1	*end

All records have the same record length. The user selects the record length by determining the required (or useful) length of a data frame. The Header must contain a fixed number of information words; therefore, the number of records needed for the Header is a variable.

By the nature of its structure, the Dictionary consists of 5 records. The record immediately preceeding the string of data records is filled with the work "*beg", and the last record is filled with the work "*end".

3.1.3 RELBET Data Frame Formats

Frames are of a fixed length for a given file, but the length may vary between file types. Although frame formats vary from file type to file type, they are composed of a fixed format header portion and a data portion. The standard frame header format is as follows:

Integer Word	Type	Contents
1-2	DP	Time-tag
3	CHAR*4	Frame ID
4	I	Edit status

The formats of the data portions are discussed under the individual files. As noted above the Begin and End Frames contain fill data:

"*beg" and "*end" respectively.

3.1.4 File I/O Packet

The file I/O packet, iop, is a gff communication table used by both input and output routines. This table carries pertinent file data across read/write interfaces. Table 3.1.4.1 defines the 41 integer words in iop.

Table 3.1.4.1. I/O Packet Contents

Word	Label	Type	Description
1	pname	c72	Name of gff file to create or open. Pname is a user-defined input to the open routines, gfnew and gfopen.
19	punit	i	Unit number on which to open the gff file. Punit is a user-defined input to the open routines, gfnew and gfopen.
20	pfrmsz	i	Record length. This will be the record length in integer words of the gff file records. The record length will also be stored in the file header. The header will be broken into records of this length and written to the gff file. The record length is a user-defined input to the create routine, gfnew. When the file is reopened, the open routine, gfopen, will check the header for the record length and then open the file with the proper record length.
21	phdrsz	i	Number of header words. The number of header records is a constant and is, at this time, 90 words. This parameter is here to prevent hard coding of the header size in the various gff routines. The number of header records is also stored in the file header. Phdrsz is set in the open routines, gfnew and gfopen.
22	pcrec	i	Current record number. Pcrec points to the current record in the gff file. When writing to the file, pcrec points to the last record written. When reading from the file, pcrec points to the last record read. Pcrec is always current, regardless of which gff routine was last used. The record number is not affected by the display routines.

Table 3.1.4.1. I/O Packet Contents (Continued)

Word	Label	Type	Description
23	prwflg	i	Read/write status. Prwflg defines the r/w ability of the file where a value of one is read only, and a value of two is read/write. When the file is being created by gfnew, the rwflg is set by the routine to a value of two. When the file is reopened with gfopen, the rwflg is a user-defined input. Any dependent on the value of the r/w flag.
24	pnrec	i	Number of records in the gff file. Pnrec is always current.
25	pstat	i	Error code returned from the gff routines. Pstat can contain two types of error codes - system i/o errors and gff errors. If the error is a system error, the error code will be a 900 number. If the error is a gff error, the error code will be negative. The possible gff errors are: 1. trying to access within the dictionary or the header 2. trying to access past the end of file 3. file does not have write permission 4. invalid time requested
26-27	ptoff	dp	Time offset. All timetags (timetags are assumed to be the first parameter in the data frame) stored in the gff file are stored in seconds since base time. The base time of the file is stored in the header. The base time of one gff file does not have to be the same base time of any other gff files used in a program, and it does not have to be the same as the program base time. The time offset is the difference between the program base time and file base time (tbase (prog) - tbase (file)). Upon writing a record, the timetag will be converted from seconds since program base time. The time offset is computed in the open routines, gfew and gfopen.
28	ptype	c4	File type. The file type is a user-defined, four character identifier used to identify the type of data in the file. It is an input to the create routine, gfnew, and is also stored in the header.
29-33	pbdate	i(5)	

Table 3.1.4.1. I/O Packet Contents (Continued)

Word	Label	Type	Description
34-35	pbsec	dp	Base date. The base time is stored in 5 integer words and one dp word. The integer words are the year, month, day, hour, and minute; the dp word is the seconds. All timetags stored in the file are stored as seconds since the base date. See the entry for ptoff for further information. The base date is a user-defined input to the open routine, gfnew.
36-37	pstrt	dp	Start time. This is the time of the first data record. The start time is defined when the *end frame is written to the file (gff routine gfend) and when an existing gff file is opened. If a gff file is changed, the start time is not guaranteed to be correct until the *end frame is written. The start time is stored as seconds since file base time and is also stored in the file header.
38-39	pstop	dp	Stop time. This is the time of the last data record. The stop time is defined when the *end frame is written to the file (gff routine gfend) and when an existing gff file is opened. If a gff file is changed, the stop time is not guaranteed to be correct until the *end frame is written. The stop time is stored as seconds since file base time and is also stored in the file header.
40	pnhrec	i	Number of header records. The header is 90 words of data and is written to the file "pfrmsz" words at a time. If the last record is not full, it will be garbage filled. There are routines which read and write the header. The number of header records is computed from phdrsz and pfrmsz (nhrec = int(hdrsz/frmsz + .5)) in the create routine, gfnew. This is stored in the header. See phdrsz for more details.
41	pndict	i	Number of dictionary records. The number of dictionary records is a constant 5 words. This parameter is here to prevent hard coding in the various gff routines. The number of dictionary records is also stored in the file header. Pndict is set in the open routines, gfnew and gfopen.

3.1.5 File Header Format

The file header consists of a set of records, each of which has the same word length as the data records. The header contains 90 (integer size) words plus as many fill words as are needed to complete the last header record. Table 3.1.5.1 summarizes the 90 integer words in the header.

Table 3.1.5.1. Header Contents

Word	Label	Type	Description
1	hnhrec	i	Number of header records. See iop - pnhrec for a description.
2	htype	c4	File type. See iop - ptype for a description.
3-4	hcdate	c8	Creation data. When a gff file is created, the currect data is stored as the creation date in the form yy/mm/dd.
5-8	hcver	c16	Program used for creation. The program name and version number which created the gff file, along with the version of the gff library is stored. The suggested form is "name-ver/gff-ver", where name is the name of the program (6 characters max) and ver is the version in the form Vn.m (Ex: trjpro-v2.2/gff1.0). The program name and version are user inputs to the create routine, gfnew, and the gff library verion is defined by the gff library in gfnew.
9-34	hcdes	c104	Creation description. The creation description is a user-defined input to the create routine, gfnew.
35-36	hudate	c8	Update date. Each time a gff file is reopened with a r/w status, the current date is stored as the update date. When the file is created, the update date is the same as the creation date. The update date is established in the close routine, gfcls.
37-40	huver	c16	Update program name and version. See HEADER - hcver for a description.

Table 3.1.5.1. Header Contents

Word	Label	Type	Description
41-66	hudes	c104	Update description. Each time a gff file is reopened with a r/w status, the update description is stored. When the file is created the update description is set to the creation description. The update description is established in the close routine, gfcls.
67-71			
72-73	hbdate	i(5)	File base date. See iop - pbdate, pbsec for a description.
74-75	hstrt	dp	Start time. See iop - pstrt for a description.
76-77	hstop	dp	Stop time. See iop - pstop for a description.
78	hnrec	i	Number of data records in the file. When the *end frame is written to the file, the number of data frames is computed and stored. This value will be correct as long as no new data records are written after the *end frame. New data frames appended to the file will automatically overwrite the *end frame, if one exists. Note that this is not exactly the same parameter as the iop parameter pnrec. In the iop, this parameter is the total number of records, whereas, in the header, it is the number of data frames. Hnrec is set when the *end frame is written.
79	hfrmsz	i	Record length. See iop - pfrmsz for a description.
80-90	hspec	i	File specific block. These 11 words are extra header space in case the user feels the need to put more in the header. The gff library only recognizes its existence; it does nothing with the information. The block is a user-defined input to the create routine, gfnew. It can have variables of any type stored within.

3.1.6 Dictionary Format

The dictionary contains the definition of each parameter in the data records. The definition is defined as the name of the parameter, its

dimension and type, and its length if the variable is character. Note that there records are not set with the gbfile routines, however, space is allocated for them.

The dictionary is written to the gff file when the file is created by the routine gfnew. The dictionary is input into gfnew in the following format:

```
character*20 dict(frmsz)
```

where frmsz is the number of words in a record. Each dict(n) is the entry for a single parameter, and each entry is in the form "ndtl", where

n is the parameter name (8 characters),
d is the dimension of the variable (Ex: 3 or 3,3) (8 characters),
t is the variable type (i,l,r,c) (1 character), and
l is the length of the variable if t=c (3 characters).

The dictionary is stored in the gff file transposed. The routine gfdict returns the dictionary in the c*20form. The dictionary is contained in exactly five records in the gff file. Each dictionary entry (dict(n)) uses one word from each of the five records, such that

rec 1-2 contain the name from each entry,
rec 3-4 contain the dimension of the variable,
rec 5 contains the type and length of the variable.

Figure 3.1.6 shows the relationship between the user-defined dictionary and the file dictionary.

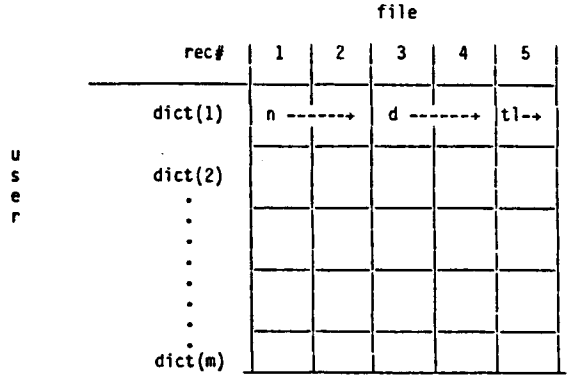


Figure 3.1.6. User/File Dictionary Relationship

3.2. SENSED VELOCITY FILE

The Sensed Velocity File contains time tables of IMU sensed velocities experienced by a vehicle. The file may be created from real time data or simulated. The sensed velocities are used for state propagation. Its frame format is as follows:

Integer Word	Type	Contents
1-4	MIX	Header
5-10	DP(3)	Vector for Sensed Velocity

3.3 ATTITUDE FILE

The Attitude File contains time tables of attitudes experienced by a vehicle. The file may be created from real time data or simulated. The trajectory processing routines use the attitudes for computing relative trajectories and residuals. Its frame format is as follows:

Integer Word	Type	Contents
1-4	MIX	Header
5-12	DP(4)	Quaternion for Attitude

3.4 OBSERVATION FILE

The Observation file contains time tables of observations associated with various on-board navigation sensors. The additional data slot is used by various applications. When the downlist formatter outputs an observation file, it places the onboard data good flag here. The Kalman Filter process expects the observation noise in this slot. This file may also contain residuals for editing or display purposes. The Observation file is used for relative trajectory estimation and residual computations. Its frame format is as follows:

Integer Word	Type	Contents
1-4	MIX	Header
5-6	DP	Additional Data
7-8	DP	Measurement
9-10	DP	Residual

3.5 EPHEMERIS FILE

An Ephemeris File contains the inertial trajectory for a vehicle. This file may be created from real time data or generated by simulation. Vehicle states are interpolated from this file. Its frames have the following format:

Integer Word	Type	Contents
1-4	MIX	Header
5-10	DP(3)	Position Vector
11-16	DP(3)	Velocity Vector

3.6 IMU ATTITUDE FILE

The IMU Attitude File is used by the IMU processor to generate a selected or average attitude file. It has the following format:

Integer Word	Type	Contents
1-4	MIX	Header
5-12	DP(4)	Quaternion IMU-1
13-20	DP(4)	Quaternion IMU-2
21-28	DP( )	Quaternion IMU-3
29-36	DP(4)	Quaternion Selected
37-38	I(2)	Source of Selection (spare)

3.7 IMU SENSED VELOCITY FILE

The IMU Sensed Velocity File is used by the IMU Processor to generate a selected or average sensed velocity file. It has the following frame format:

Integer Word	Type	Contents
1-4	MIX	Header
5-10	DP(3)	Vector IMU-1
11-16	DP(3)	Vector IMU-2
17-22	DP(3)	Vector IMU-3
23-28	DP(3)	Vector Selected
29-30	I(2)	Source of Selection (spare)

3.8 SOLUTION FILE

The Solution File contains such information as current solution state estimates and statistics. This file is an optional product of the Trajectory Processing processors. Frame lengths may vary from file to file depending on the options exercised by the processor, however, frame length is fixed in a given file. The solution has the following format:

Integer Word	Type	Contents
1-4	MIX	Header
5-4+2S	D	Solution Vector
5+2S-4+2S+D	DP(C)	Lower Triangular Covariance by Rows

Where

S = Solution size

D = S(S+1)/2.

3.9 RELATIVE TRAJECTORY FILE

The Relative Trajectory File contains relative trajectory information. It has the following frame format:

Integer Word	Type	Contents
1-4	MIX	Header
5-16	DP(6)	Fiducial State
17-28	DP(6)	Relative State

3.10 PLOT DISPLAY FILES

Table 3.10.1 summarizes the various options for output parameters from the program xqdisp. Here O designates the first vehicle or orbiter state vector, T the second vehicle or target state vector, and A the attitude of the first vehicle. All groups consist of six parameters. The coordinate systems are defined in the Engineering Manual (Reference 9, Section 2.0).

Plot file parameters are in internal units. For printed displays, the user may specify up to three systems of units for length, velocity, angle, time, mass, force, and acceleration. Parameter groups will be displayed in each of the specified systems of units. The record format for these files is as follows:

Integer Word	Type	Contents
1-4	MIX	Header
5-16	DP(6)	Coordinates

Table 3.10.1. Parameter Groups

NUMBER	ID	OBJECT	COORDINATES	REQUIRED INPUTS
1	SXYZ	0	M50 Cartesian	0
2	SELT	0	M50 Elements	0
3	SUVW	0	Target UVW	0, T
4	SLVH	0	Target LVLH	0, T
5	SSHL	0	Target Shell	0, T
6	TXYZ	T	M50 Cartesian	T
7	TELT	T	M50 Elements	T
8	TUVW	T	Orbiter UVW	0, T
9	TLVH	T	Orbiter LVLH	0, T
10	TSHL	T	Orbiter Shell	0, T
11	TBOD	T	Orbiter Spherical	0, T, A
12	IREL	T-0	M50 Cartesian	0, T
13	EREL	T-0	M50 Elements	0, T
14	APYR	A	M50 Euler Angles	A
15	AUVW	A	Orbiter UVW Euler Angles	0, A
16	RPRM	T-0	Range and Rate Magnitudes	0, T

In column 5,

A indicates orbiter attitude needed

0 indicates orbiter trajectory needed

T indicates target trajectory needed

3.11 COAS MATRIX FILE

This file contains frames of body to M50 rotation matrices time-tagged by the time of each COAS observation. Its frame format is as follows:

Integer Word	Type	Contents
1-4	MIX	Header
5-22	DP(9)	Components of Matrix by rows

3.12 COVARIANCE FILE

This file contains th UVW referenced sigmas associated with some filter estimate of some relative trajectory.

Integer Word	Type	Contents
1-4	MIX	Header
5-16	DP(6)	Sigmas for Base State
17-28	DP(6)	Sigmas for Relative State

3.13 BIAS FILE

This file contains the bias solutions and associated sigmas computed by one of the filters.

Integer Word	Type	Contents
1-4	MIX	Header
5-A	DP(S-12)	Bias Solutions
A+1-B	DP(S-12)	Bias Solution Sigmas

where s is the solution size

A = 4 + (S-12) *2

B = A + (-12) *2

3.14 RESIDUAL FILE

This file contains information from the Kalman filter as shown below:

Integer Word	Type	Contents
1-4	MIX	Header
5-6	DP	Edit Status (-1 or 1)
7-8	DP	Observation Value
9-10	DP	Residual Value
11-12	DP	Residual Sigma
13-14	DP	Bias Value
15-16	DP	Bias Sigma

3.15 QA FILES

The following files are created by the various QA processors:

The Attitude Angular Acceleration File

Integer Word	Type	Content
1-4	MIX	Header
5-10	DP(3)	Angular Acceleration
11-12	DP	Magnitude

The Sensed Acceleration File

Integer Word	Type	Content
1-4	MXI	Header
5-10	DP(3)	Sensed Acceleration
11-12	DP	Magnitude

The Master Noise File

Integer Word	Type	Content
1-4	MIX	Header
5-6	DP	End time of interval (Begin time is timetag)
7-8	DP	Average value of observation
9-28	DP(10)	Value of moise associated with various order of divided difference

The Selected Noise File

Integer Word	Type	Content
1-4	MIX	Header
5-6	DP	End time of intervals (Begin time is timetag)
7-8	DP	Average value of observation
9-10	DP	Selected value of noise for time interval

The Star File

Integer Word	Type	Content
1-4	MIX	Header
5-6	DP	Inertial Azimuth Angle
7-8	DP	Inertial Elevation Angle
9-10	DP	Angle between the previous pointing vector and current pointing vector

3.16 SENSOR FILE

This file contains downlist information needed to create the SENSOR tapes.

Integer Word	Type	Content
1-4	MIX	Header
5-6	DP	gmt
7-8	DP	COAS Data Good
9-12	DP(2)	COAS Angles
13-14	DP	Radar Angle Data Good
15-18	DP(2)	COAS Angle
19-20	DP	Radar Range Data Good
21-22	DP	Radar Angles
23-24	DP	Radar Range Rate Data Good
25-26	DP	Radar Range Rate
27-28	DP	Star Tracker Data Good
29-32	DP(2)	Star Tracker Angle
33-34	DP	NAV Power Flight Flag
35-40	DP(3)	Sensed Velocity
41-42	DP	Mass
43-50	DP(4)	Attitude
51-52	DP	Sensed Acceleration Threshold
53-54	DP	NAV DAP Jet Flag
55-72	DP(9)	Star Tracker Matrix
73-74	DP	Star Tracker Timetag
75-92	DP(9)	COAS Matrix
93-94	DP	COAS Timetag
95-102	DP(4)	Radar Quaternion
103-104	DP	Radr Timetag
105-110	DP(3)	DVDISP
111-152	DP(21)	Members of Covariance
153-188	DP(18)	NAV Flags
189-192	DP(2)	TOFF/TON Flags
193-196	DP(2)	Update Flags
197-202	DP(3)	V_force

Note for more details on contents see section 2.4.

4.0 TEXT FILES

Various units are used for display. Depending on the processor, the user may select up to four different units for short (TERM) display, nominal (OUT) display, detailed debug (BUGS), and graphic (PLOT) display. Details as to format and access of these displays depends upon the processor.

5.0 DATA BASE FILES

The text files described here are generally output by the qa processors. They share a common format which allows them to be merged, sorted, and searched by various UNIX and RELBET processors. Each line of text contains at least four fields separated by white space.

Field 1 - Timetag (begin time of some event in GMT seconds)
Field 2 - Primary key word (alpha) usually denoting data type
Field 3 - Secondary key word (alpha) usually denoting type of event
Field 4 - Duration of event in GMT seconds

The remaining fields are defined per processor output (see program manual entries).

REFERENCES

1. "Interface Control Document for ODRC Computer Compatible Tapes," CSC-1921, G. E. Hubacek/CSC, February 1984.
2. "RELBET Product Description (Update/May 1985)," TRW Report No. 85:W482.1-57, B. P. Huysman, L. A. Pieniazek, 15 May 1985.
3. "Sensor Tape Production Manual," TRW Report No. 86:W482.1-60, J. K. Knoedler, 3 June 1986.
4. "HP General File Format (GFF) User's Guide," TRW Report 39107-H-21-UX-00, D. Campbell, 14 December 1984.

APPENDIX II

CONTEXT FILES

```
#
#           #    #    #  #    #  #          #    #####
#           #    ##   #  #   #   #          #    #    #
#           #    # #  #  ####    #          #    #####
#           #    #  # #  #  #    #          #    #    #
#           #    #   ##  #   #   #          #    #    #
########    #    #    #  #    #  ######     #    #####
```

```
 1  BaseTime.h
 2  Gbfile.h
 3  Time.h
 4  anlists.h
 5  array.h
 6  cell.h
 7  fileio.h
 8  gbfile.x.h
 9  gbflags.h
10  gbfstruct.h
11  gbhdrec.h
12  gbmiscdef.h
13  gnamlist.h
14  gstack.h
15  gstack.typ.h
16  linput.h
17  lists.h
18  message.h
19  newcell.h
20  ptrlist.h
21  quat.h
22  ratetable.h
23  stop.h
24  timeline.h
```

```
1   total 60
2   -r--r--r--   1 relbet   RELBET       261 Dec 10 13:42 BaseTime.h
3   -r--r--r--   1 relbet   RELBET      2942 Dec 10 12:46 Gbfile.h
4   -r--r--r--   1 relbet   RELBET      1333 Dec 10 13:42 Time.h
5   -r--r--r--   1 relbet   RELBET       130 Dec 10 13:07 anlists.h
6   -r--r--r--   1 relbet   RELBET      1193 Dec 10 13:03 array.h
7   -r--r--r--   1 relbet   RELBET       339 Dec 10 12:59 cell.h
8   -r--r--r--   1 relbet   RELBET       122 Dec 10 12:42 fileio.h
9   -r--r--r--   1 relbet   RELBET       381 Dec 10 12:46 gbfile.x.h
10  -r--r--r--   1 relbet   RELBET       161 Dec 10 12:47 gbflags.h
11  -r--r--r--   1 relbet   RELBET      1498 Dec 10 12:47 gbfstruct.h
12  -r--r--r--   1 relbet   RELBET      1657 Dec 10 12:47 gbhdrec.h
13  -r--r--r--   1 relbet   RELBET       427 Dec 10 12:47 gbmiscdef.h
14  -r--r--r--   1 relbet   RELBET       642 Dec 10 13:07 gnamlist.h
15  -r--r--r--   1 relbet   RELBET       214 Dec 10 13:40 gstack.h
16  -r--r--r--   1 relbet   RELBET       435 Dec 11 09:48 gstack.typ.h
17  -r--r--r--   1 relbet   RELBET        73 Dec 10 12:58 linput.h
18  -r--r--r--   1 relbet   RELBET       625 Dec 10 12:59 lists.h
19  -r--r--r--   1 relbet   RELBET       500 Dec 10 13:07 message.h
20  -r--r--r--   1 relbet   RELBET        18 Dec 10 13:00 newcell.h
21  -r--r--r--   1 relbet   RELBET       968 Dec 10 13:07 ptrlist.h
22  -r--r--r--   1 relbet   RELBET       235 Dec 10 13:04 quat.h
23  -r--r--r--   1 relbet   RELBET       281 Dec 10 13:07 ratetable.h
24  -r--r--r--   1 relbet   RELBET       645 Dec 10 12:42 stop.h
25  -r--r--r--   1 relbet   RELBET       601 Dec 10 13:07 timeline.h
```

```
 1   #ifndef BASETIME_DOT_H
 2
 3
 4   extern CALTIME BaseCalTime;
 5   extern double JD_BaseTime;
 6   extern double GBeginTime;
 7   extern double GEndTime;
 8   extern double GDelTime;
 9   extern double LBeginTime;
10   extern double LEndTime;
11   extern double LDelTime;
12
13   #define BASETIME_DOT_H  1
14   #endif
```

```
#ifndef GBFILE_DOT_H

#include "Time.h"


typedef struct {
     double toff;       /*time offset: file time - prog time*/
     double tbegin;     /*start time of file*/
     double tend;       /*end time of file*/
     int fd;            /*file descriptor*/
     int rsize ;        /*data record size in bytes*/
     int origin;        /*byte corresponding to 1st record*/
     int ndatarec; /*number of data records in file*/
     long filepos;      /*current byte index*/
     int rec;            /*current record*/
     int  n_hrec; /*number of header records in file before start of
                   data.  Note 1srt data record is numbered 0*/
     int int_rsize; /*size of records in integer words*/
     int time_byte; /*byte at which time word occurs. Should b -1 if
                no timeword for file record structure*/
     CALTIME basetime; /*file base time*/
     double *time;      /*time word in in data buffer. Should be null
                if no timeword for file.*/
     char *data;     /*pointer to data buffer.  This buffer is used
                for read operations and is allocated when file is
                opened*/
     char *format;      /*format specification*/
     int status;
     char *textbuf;     /*pointer to test buffer. the following are
                stored in this buffer*/
     char mode;/*read/write mode*/
     char name[80];
   } GBFILE, *GBFILEPTR;



/*the header record is similar to the FORTRAN based GFF header and is
 maintained that way for compatiblity.  Note that characters are all
 four bytes longer so as to insure they may be treated as strings by
 null filling*/

/*application software should assure last bytes of character fields
are null, thus may be treated as string*/

/*time offsets handeled only if time word is in file data structure.
This is controled by time_byte and time. if time is null then no
time word is assumed. if time_byte is negative nominally -1, then
time infor is ingored and time is set accordingly.  See define for
NO_TBYTE to be sure of invalid value for time_byte*/

/* text buffer contains following:
     char *name;
     char *type; file type:  This is user defined. although it
         may be checked by the application software*
     char *c_date; creation date :
           Format is yy/mm/dd.*
```

```
57          char *c_prog; name of creating program*
58          char *c_text; description text for creation*
59          char *u_date; last update date :
60                Format is yy/mm/dd.*
61          char *u_prog; name of last updating program*
62          char *u_text; description text for last update*/
63
64
65    #ifndef GBFILE_DOT_DEF_ONLY
66
67
68
69    extern gbtoff();
70    extern gbwhead();
71    extern gbrhead();
72
73    extern gbpos();
74
75    extern char *gbread();
76
77    extern gbwrite();
78
79    extern char *gbdread();
80
81    extern gbdwrite();
82
83    extern GBFILE *makeGBF();
84    extern char *makeGBData();
85
86    extern GBFILEPTR gbopen();
87
88    extern GBFILEPTR gbnew();
89
90    extern gbclose();
91
92    extern gbfree();
93
94    extern freeGBF();
95
96    extern gbphead();
97
98    extern char *gbtime();
99
100   #endif
101
102
103   #define GBFILE_DOT_H    1
104   #endif
```

```
#ifndef TIME_DOT_H

typedef struct {          /*calender date*/
    int year;
    int month; /*1=Jan, 2=Feb, ...*/
    int day;
    int weekday; /*0=Sat, 1=Sun, ... */
  }  CALDATE, *CALDATEPTR;

 typedef struct {                /*hour,minute,second time*/
     int hour;
     int min;
     double sec;
  } HMSTIME, *HMSTIMEPTR;

 typedef struct {        /*calendar date and time*/
     CALDATE date;
     HMSTIME hms;
  } CALTIME, *CALTIMEPTR;


#ifndef TIME_DOT_DEF_ONLY

extern CALTIME *GetCurTime();
extern void fprtCurTime();
extern void prtCurTime();
extern CALTIME CurSysTime;
extern long GMTsec;
extern int GMTday;

extern double hms2sec();
extern HMSTIME *sec2hms();
extern int days1bc();
extern int days();
extern double etsec();
extern double jul2cal();
extern double juldate();
extern double jultime();
extern int std_time();
extern int mnthnum();

extern CALTIME *makeTime();
extern HMSTIME *makeHMS();
extern CALDATE *makeDate();

extern int setDate();
extern int setHMS();
extern int setTime();

extern void fprtdate();
extern void sprtdate();
extern void prtdate();

extern void fprthms();
extern void sprthms();
extern void prthms();
```

```
57
58   extern void fprtctime();
59   extern void sprtctime();
60   extern void prtctime();
61
62   extern void fprtsec();
63   extern void sprtsec();
64   extern void prtsec();
65
66   #define TIME_DOT_H      1
67
68   #endif
69   #endif
```

```
1   #ifndef ANLISTS_DOT_H
2
3   extern int addANLitem();
4   extern int prtANLitem();
5   extern void prtANList();
6
7   #define ANLISTS_DOT_H   1
8   #endif
```

```
 1
 2  #ifndef ARRAY_DOT_H
 3                    /* matrix.c */
 4  extern double *mxm() ; /* matrix.c */
 5  extern double *mxmc() ; /* matrix.c */
 6  extern double *mxv() ; /* matrix.c */
 7  extern double *mtxv() ; /* matrix.c */
 8  extern double *mt() ; /* matrix.c */
 9  extern double *mtxm() ; /* matrix.c */
10  extern double *mxmt() ; /* matrix.c */
11
12                    /* prtarray.c */
13  extern int prt3mat() ; /* prtarray.c */
14  extern int prt3vec() ; /* prtarray.c */
15  extern int prt3tmat() ; /* prtarray.c */
16  extern int fprtarray() ; /* prtarray.c */
17  extern int prtarray() ; /* prtarray.c */
18
19                    /* rmatrix.c */
20  extern double *rmxm() ; /* rmatrix.c */
21  extern double *rmxmc() ; /* rmatrix.c */
22  extern double *rmxv() ; /* rmatrix.c */
23
24                    /* vector.c */
25  extern double *vadd() ; /* vector.c */
26  extern double *vaddto() ; /* vector.c */
27  extern double vdist() ; /* vector.c */
28  extern double vdot() ; /* vector.c */
29  extern double *vfadd() ; /* vector.c */
30  extern double *vfaddto() ; /* vector.c */
31  extern double *vsub() ; /* vector.c */
32  extern double *vfmul() ; /* vector.c */
33  extern double vrss() ; /* vector.c */
34  extern double *vset() ; /* vector.c */
35  extern double *vzero() ; /* vector.c */
36  extern double vrunit() ; /* vector.c */
37  #define ARRAY_DOT_H 1
38  #endif
```

```
1   enum celltype {int_cell, double_cell, symbol_cell, function_cell, cons_cell};
2
3   typedef struct cell {
4           enum celltype type;
5           int mark;
6           union {
7                   int ivalue;
8                   double dvalue;
9                   char *svalue;
10                  struct cell *(*fvalue)();
11                  struct {
12                          struct cell *car;
13                          struct cell *cdr;
14                  } cons;
15          } val;
16  } CELL, *P_CELL;
17
18
19  typedef P_CELL (*P_FUNCTION)();
```

```
1   struct FILE_INFO {
2           char *name;
3           char *frmid;
4           char *(*function)();
5           char *gbf;
6           int option;
7           struct FILE_INFO *next;
8           };
```

```
1   extern gbtoff();
2   extern gbwhead();
3   extern gbrhead();
4
5   extern gbpos();
6
7   extern char *gbread();
8
9   extern gbwrite();
10
11  extern char *gbdread();
12
13  extern gbdwrite();
14
15  extern GBFILE *makeGBF();
16  extern char *makeGBData();
17
18  extern GBFILEPTR gbopen();
19
20  extern GBFILEPTR gbnew();
21
22  extern gbclose();
23
24  extern gbfree();
25
26  extern freeGBF();
27
28  extern gbphead();
29
30  extern char *gbtime();
```

```
1   /*flags and such for gbfile routines*/
2   #define ERRVAL          -1
3           /*error occured*/
4   #define NOERR           0
5           /*no error*/
6   #define NO_TBYTE        -1
7           /*no time word in data record*/
```

```
/*names for gbfile fields and text buffer locations*/
/*To use short hand for fields:
All entries have the string "GBS" preceeding the field names which
must be specified by a define or HS as the gbf structure. The define for
HS should be after the inclusion for this define block. For example
if gbf is a pointer to the structure, a define of
        #define GBS  (*gbf)
should follow these defines.
To use text locations:
A string "TXT" is defined as "GBS.textbuf +".
The pointer to an entry is then given by TXT <start byte def>,
for example  TXT NAME references the name of the file.*/

#define GBTXTBUFSIZE 520



        /*shorthand for fields*/

#define BDATE           GBS.basetime
#define BYR             GBS.basetime.date.year
#define BMO             GBS.basetime.date.month
#define BDA             GBS.basetime.date.day
#define BHR             GBS.basetime.hms.hour
#define BMIN            GBS.basetime.hms.min
#define BSEC            GBS.basetime.hms.sec
#define DATABUF         GBS.data
#define FDES            GBS.fd
#define DATFMT          GBS.format
#define FPOS            GBS.filepos
#define FRMSZ           GBS.int_rsize
#define NHREC           GBS.n_hrec
#define ORIG            GBS.origin
#define REC             GBS.rec
#define NREC            GBS.ndatarec
#define RSIZE           GBS.rsize
#define STAT            GBS.status
#define STRT            GBS.tbegin
#define STOP            GBS.tend
#define TBYTE           GBS.time_byte
#define TOFF            GBS.toff
#define TWORD           GBS.time

        /*byte offsets for text buffer entries*/

#define CDATE           86
#define CVER            330
#define CDES            110
#define FTYPE           80
#define NAME            0
#define UDATE           96
#define UVER            360
#define UDES            220


#define TXT             GBS.textbuf +
```

```
/*defines for contents of gbf header record field*/
/*contains current size of basic header record in bytes (HDRSIZE)
and byte offsets and lengths of each field.  To obtain pointer to
desired location add the  byte offset to the base pointer to
the header record buffer.  For example if HDR is defined as
"hdr_buffer +" where hdr_buffer is the header buffer base pointer,
the HDR HCDES is the pointer to the creation description field.
Names are as described in the gff documentation.*/

#define HDRSIZE         400
#define NFORMATREC      6

        /*%%byte locations of gff header fields*/

#define HNHREC      0
#define HTYPE       4
#define HCDATE      8
#define HCVER      16
#define HCDES      36
#define HUDATE    136
#define HUVER     144
#define HUDES     160
#define HBDATE    264
#define HBYR      264
#define HBMO      268
#define HBDA      272
#define HBHR      276
#define HBMIN     280
#define HBSEC     284
#define HSTRT     292
#define HSTOP     300
#define HNREC     308
#define HFRMSZ    312
#define HTBYTE    316


        /*%%byte lengths of gff header fields*/

#define LHNHREC     sizeof(int)
#define LHTYPE      4
#define LHCDATE     8
#define LHCVER     16
#define LHCDES    104
#define LHUDATE     8
#define LHUVER     16
#define LHUDES    104
#define LHBDATE     5 * sizeof(int)
#define LHBYR       sizeof(int)
#define LHBMO       sizeof(int)
#define LHBDA       sizeof(int)
#define LHBHR       sizeof(int)
#define LHBMIN      sizeof(int)
#define LHBSEC      sizeof( double)
#define LHSTRT      sizeof( double)
#define LHSTOP      sizeof( double)
#define LHNREC      sizeof(int)
```

```
57    #define LHFRMSZ      sizeof(int)
58    #define LHSPEC       sizeof(int)
59    #define LHTBYTE      sizeof(int)
```

```
1   /*miscellaneous defines for header manipulation*/
2
3   #define HDR  hdrbuf +
4   #define HDRBUFSIZE 520
5
6
7   #define GET_CUR_FPOS  {if((FPOS = lseek(FDES,0,1))< ORIG) FPOS = ORIG;}
8   #define RESTORE_FPOS {if(FPOS < ORIG) FPOS = ORIG; lseek(FDES,FPOS,0);}
9
10
11
12
13
14  #define WORDSIZE    sizeof(int)
15  #define TSIZE       sizeof(CALTIME)
16
17  #define CUR_YR          (CurTime->date).year
18  #define CUR_MO          (CurTime->date).month
19  #define CUR_DA          (CurTime->date).day
20
21
```

```
1   #ifndef GNAMLIST_DOT_H
2
3   typedef struct{
4     char *adress; /*adress of named item*/
5     char *name; /*name of item*/
6   } GNL_ITEM, *GNL_ITEM_PTR;
7
8   /*Structure for list of names and adresses.  It is assumed that end
9   of the list is marked by and entry with a null (0) adress.*/
10
11  typedef struct{
12    char *name; /*name of table*/
13    int max_length; /*max size of list*/
14    int cur_length;
15    int item_size;
16    int idcode;
17    char *items;
18    int (*search)();
19  } GNLIST, *GNLIST_PTR;
20
21  #ifndef GNAMLIST_DOT_DEF_ONLY
22
23  extern void fprtnlist();
24  extern GNLIST *Make_GNLIST();
25  extern char *GNLgetName();
26  extern int Add_GNList();
27
28  #endif
29  #define GNAMLIST_DOT_H  1
30  #endif
```

```
1   extern char *q_pop();
2   extern char *q_push();
3   extern char *gspop();
4   extern char *gspush();
5   extern char *gspeek();
6   extern char *gspoke();
7   extern char *gsempty();
8   extern char *gsfree();
9   extern P_GSTACK make_gstack();
```

```
1   typedef struct {
2           char *base;
3           char *front;
4           char *next;
5           int block_size;
6           int max;
7           int n_blocks;
8           } GSTACK, *P_GSTACK, QUEUE, *P_QUEUE  ;
9   /* A stack consists of blocks with fixed size, numbered from bottom to
10     top starting with empty stack number 0. The pointer to next block in
11     stack points to next available space. The pointer to front is basically
12     a Queue application and indicates the front of line for queu pop*/
```

C-1-2

```
1   typedef struct {
2           char *name;
3           char *loc;
4           int dim;
5           char *type
6   } INPUT;
```

```
enum celltype {int_cell, double_cell, symbol_cell, function_cell, cons_cell};

typedef struct cell {
        enum celltype type;
        int mark;
        union {
                int ivalue;
                double dvalue;
                char *svalue;
                struct cell *(*fvalue)();
                struct {
                        struct cell *car;
                        struct cell *cdr;
                } cons;
        } val;
} CELL, *P_CELL;


typedef P_CELL (*P_FUNCTION)();
P_CELL cons(), inumber(), dnumber(), symbol();
P_CELL car(), cdr();
int ivalue();
double dvalue();
char *svalue();
P_FUNCTION fvalue();
int isatom(), isinumber(), isdnumber(), issymbol(), isfunction(), eq();
P_CELL append();
int member(),length();
P_CELL locate();


extern P_CELL nil;
```

```
#ifndef MESSAGE_DOT_H

extern int fskip_lines();
extern int skip_lines();
extern int fprtStars();
extern int prtStars();

extern char *Stat_Msg;
extern int  Error_Count ;
extern int  Warning_Count;
extern int  StatErrExit;
extern void fprtFinish();
extern void prtFinish();
extern int MaxErrExit();
extern void err_hpcode();
extern void err_code();
extern void berror();
extern void bwarn();
extern int addefile();
extern int rmefile();
extern void setMaxErr();
extern char *Save_Str_Buf();

#endif
```

```
1   P_CELL newcell();
```

```
#ifndef PTRLIST_DOT_H

typedef struct{
  char *adress; /*adress of named item*/
  char *name; /*name of item*/
} NL_ITEM, *NL_ITEM_PTR;

/*Structure for list of names and adresses.  It is assumed that end
of the list is marked by and entry with a null (0) adress.*/

typedef struct{
  char *name; /*name of table*/
  int max_length; /*max size of list*/
  int cur_length;
  int item_size;
  int idcode;
  char *list;
  int (*search)();
} GLIST, *GLIST_PTR;



typedef struct {
   int n; /*current number of pointers in list*/
   int max; /*max number of pointers in list*/
   char **p; /*list of pointers.*/
  } PTRLIST, *PTRLISTPTR;

/* care should be taken to provide ample room in the list of
pointers. say 1 more than max and using the null pointer to
denote the end of the list as in strings*/


#ifndef PTRLIST_DOT_DEF_ONLY

extern int addptr();
extern int rmptr();
extern PTRLIST *makeplist();
extern void freeplist();

#endif


#define PTRLIST_DOT_H   1
#endif
```

```
1
2   #ifndef QUAT_DOT_H
3                   /* qnops.c */
4   extern int qxq() ; /* qnops.c */
5   extern int qcxq() ; /* qnops.c */
6   extern int qxqc() ; /* qnops.c */
7   extern int qintp() ; /* qnops.c */
8   extern int qvrot() ; /* qnops.c */
9   #define QUAT_DOT_H 1
10  #endif
```

```
 1  #ifndef RATETABLE_DOT_H
 2
 3  typedef struct{
 4    double time;
 5    double base;          /*base value*/
 6    double rate;          /*rate of flow per unit time*/
 7    } BASE_RATE_FLOW, *BASE_RATE_FLOW_PTR;
 8
 9  #ifndef RATETABLE_DOT_DEF_ONLY
10
11  double get_rate_table_value();
12
13  #endif
14  #define RATETABLE_DOT_H         1
15  #endif
```

```
typedef struct{
        float date[6];
        double gmt;
        double x_ft[6];
        int coas_dg;
        float coas[2];
        int rr_rp_dg;
        float rr_rp[2];
        int rr_r_dg;
        float rr_r;
        int rr_rt_dg;
        float rr_rt;
        int st_dg;
        float st[2];
        int nav_pff;
        double sv[3];
        double mass;
        double xt_ft[6];
        float att[4];
        double sa_thresh;
        int nav_dap_jf;
        float st_matrix[9];
        double st_time;
        float coas_matrix[9];
        double coas_time;
        float rr_quat[4];
        double rr_time;
        }SIT;

typedef struct{
        float sa_thresh;
        float dvdisp[3];
        double e[21];
        int nav_flags[18];
        double time;
        double tf[2];
        double t_update[2];
        float v_force[3];
        int unused;
        double filter_state[13];
        }SET;
```

```
#ifndef TIMELINE_DOT_H
typedef struct{
  char *table;          /* array of entries*/
  int last_item;        /*count start at 0*/
  int item_size;        /*number of bytes in each entry. All
                        are assumed to have the same size*/
  int time;             /*byte at which the time tag occurs, count
                        is assumed to start at 0*/
  int current_item;     /*index to current item*/
 } TIMELINE, *TIMELINE_PTR;

#ifndef TIMELINE_DOT_DEF_ONLY

extern char *bsearch_timeline();
extern char *lsearch_timeline();
extern char *isearch_timeline();
extern TIMELINE *MakeTimeLine();
extern void fprtTimeLine();

#endif
#define TIMELINE_DOT_H          1
#endif
```

```
c
c start file mtf.atm
c
c
c
c
c ***************************************************************
c start common block xxatm
c ***************************************************************
c
      DOUBLE PRECISION a62sc
c  a62 c param
c  atmospheric curve fit parameter
c
      DOUBLE PRECISION a62sh
c  a62 scale ht
c  atmospheric curve fit scale height
c
      DOUBLE PRECISION a62rr
c  a62 rr param
c  atmospheric curve fit parameter
c
      DOUBLE PRECISION a62rh
c  a62 ref dens
c  atmospheric density at sea level
c
      DOUBLE PRECISION a76a
c  a76 a param
c  1976 std atmos. empirical factor  a
c
      DOUBLE PRECISION a76b
c  1976 std atmos. empirical factor b
c  1976 std atmos. empirical factor  a
c
      DOUBLE PRECISION a76f10
c  a76 f10 flux
c  1976 std atmos. solar flux
c
      DOUBLE PRECISION a76rh1
c  a76 ref dens
c  1976 std atmos. ref. density  at scale height
c
      DOUBLE PRECISION a76a1
c  a76 a1 param
c  1976 std atmos. curve fit param
c
      DOUBLE PRECISION a76a2
c  a76 a2 param
c  1976 std atmos. curve fit param a2
c
      DOUBLE PRECISION a76a3
c  a76 a3 param
c  1976 std atmos. curve fit param a3
c
      DOUBLE PRECISION abmclg
c  bm cos lag
```

```
c  babb-mueller cosine of lag angle
c
      DOUBLE PRECISION abmslg
c  bm sin lag
c  babb-mueller sine of lag angle
c
      DOUBLE PRECISION abmgde
c  bm gdi
c  babb-mueller gdi exponent
c
      DOUBLE PRECISION abmalt
c  bm ref alt
c  babb-mueller ref. altitude
c
      DOUBLE PRECISION abm1( 3,2 )
c  bm fit params
c  babb-mueller curve fit params
c
      DOUBLE PRECISION abm0( 3 )
c  bm params 0
c  babb-mueller curve fit params
c
      DOUBLE PRECISION abmcda
c  bm c-densea
c  babb-mueller parameter c-densea
c
      DOUBLE PRECISION abmcb1
c  bm cbd1
c  babb-mueller parameter cbd1
c
      DOUBLE PRECISION abmcb2
c  bm cbd2
c  babb-mueller parameter cbd2
c
      DOUBLE PRECISION abmcm2
c  bm cbm2
c  babb-mueller parameter cbm2
c
      DOUBLE PRECISION abmrrf
c  bm rref
c  babb-mueller parameter rref
c
      COMMON /xxatm/ a62sc, a62sh, a62rr, a62rh, a76a, a76b, a76f10,
     * a76rh1, a76a1, a76a2, a76a3, abmclg, abmslg, abmgde, abmalt, abm1
     * , abm0, abmcda, abmcb1, abmcb2, abmcm2, abmrrf
c
c end block xxatm
c
c
c
c end file mtf.atm
c
```

```
c
c start file mtf.bias
c
c
c
c
c ************************************************************
c start common block xxbias
c ************************************************************
c
      DOUBLE PRECISION bias( 10 )
c  initial biases
c  at the beginning of the filter run, the observation biases
c  are initialized to the values in sbias
c
      DOUBLE PRECISION sgbias( 10 )
c  bias sigmas
c  at the beginning of the filter run, the observation bias
c  sigmas are initialized to the values in sgbias
c
      DOUBLE PRECISION timcon( 10 )
c  bias time constants
c  the time constants used in the propagation of the observation
c  biases (affects the decay of each computed bias)
c
      COMMON /xxbias/ bias, sgbias, timcon
c
c end block xxbias
c
c
c
c end file mtf.bias
c
```

```
c
c start file mtf.cct
c
c
c
c
c ***************************************************************
c start common block xxcct
c ***************************************************************
c
      INTEGER cct
c  cct info
c  indicates which tape drive the input cct is on
c
      INTEGER curyr
c  cct info
c   current year indicator on cct data records
c
      INTEGER datrec
c  cct info
c   number of words per data record
c
      INTEGER datscn
c  cct info
c   number of data records per scan
c
      INTEGER error( 5 )
c  cct info
c   error flag
c
      INTEGER lcynum
c  cct info
c   number of samples per cycle to process cct
c
      INTEGER nbuf( 3200 )
c  cct info
c   buffer to accept cct hdr and data records
c
      INTEGER numwrd
c  cct info
c   number of words per data record
c
      INTEGER numrec
c  cct info
c   number of records processed
c
      INTEGER o4000
c  cct info
c   the indicator of header record present on cct header records
c
      INTEGER scnrec
c  cct info
c   number of scans per data record
c
      COMMON /xxcct/ cct, curyr, datrec, datscn, error, lcynum, nbuf,
     * numwrd, numrec, o4000, scnrec
```

```
   57    c
   58    c end block xxcct
   59    c
   60    c
   61    c
   62    c end file mtf.cct
   63    c
```

```
c
c start file mtf.con
c
c
c
c
c ************************************************************
c start common block xxcon
c ************************************************************
c
      DOUBLE PRECISION clight
c  speed of light
c
c
      DOUBLE PRECISION halfpi
c  pi/2
c  one half pi.  computed form input value of pi
c
      DOUBLE PRECISION pi
c  pi
c  value of math constant pi
c
      DOUBLE PRECISION twopi
c   2*pi
c  two times pi.  computed from pi.
c
      COMMON /xxcon/ clight, halfpi, pi, twopi
c
c end block xxcon
c
c
c
c end file mtf.con
c
```

```
c
c start file mtf.data
c
c
c
c
c ************************************************************
c start common block xxdata
c ************************************************************
c
      INTEGER ldxtab
c  length of data ex table
c
c
      INTEGER dtxcld( 25 )
c  data x id table
c  ids of data that are excluded from processing
c
      DOUBLE PRECISION gwbtab( 2,25 )
c  wieghts and bias table
c  observation weights and bias information. on input
c
c gwbtab(1,*)= obs sigma in internal units ,
c
c gwbtab(2,*)= obs bias in internal units
c
      COMMON /xxdata/ ldxtab, dtxcld, gwbtab
c
c end block xxdata
c
c
c
c end file mtf.data
c
```

```
c
c start file mtf.dot
c
c
c
c
c ***************************************************************
c start common block xxdot
c ***************************************************************
c
      INTEGER iv
c  vehicle id
c
c
      INTEGER pvnton( 2 )
c  vent on flag
c
c
      INTEGER ivent( 2 )
c  current vent indices
c
c
      DOUBLE PRECISION mass
c  vehicle mass
c
c
      DOUBLE PRECISION mib( 3,3 )
c  inertial to body matrix
c
c
      DOUBLE PRECISION qib( 4 )
c  inertial to body quaternion
c
c
      DOUBLE PRECISION fvent( 3,2 )
c  vent force
c  m50 vent force for eac vehicle
c
      DOUBLE PRECISION senacc( 3,2 )
c  sensed accel
c  sensed acceleration in m50 frame as interpolated from
c  input files. one entry for each vehicle
c
     COMMON /xxdot/ iv, pvnton, ivent, mass, mib, qib, fvent, senacc
c
c end block xxdot
c
c
c
c end file mtf.dot
c
```

```
c
c start file mtf.dprm
c
c
c
c
c ***************************************************************
c start common block xxdprm
c ***************************************************************
c
      INTEGER spg( 20 )
c  display group flags
c  positive value specifies parameter groups to display
c  options ids
c  1.   sxyz  vehicle 1 m50 state
c  2.   selt  vehicle 1 m50 elements (a,e,i,node,perigee,tra)
c  3.   suvw  vehicle 1 in vehicle 2 uvw
c  4.   slvh  vehicle 1 in vehicle 2 lvlh
c  5.   sshl  vehicle 1 in vehicle 2 shell
c  6.   txyz  vehicle 2 m50 state
c  7.   telt  vehicle 2 m50 elements (a,e,i,node,perigee,tra)
c  8.   tuvw  vehicle 2 in vehicle 1 uvw
c  9.   tlvh  vehicle 2 in vehicle 1 lvlh
c  10.  tshl  vehicle 2 in vehicle 1 shell
c  11.  tbod  look angle to veh 2 from veh 1 in veh 1 body frame
c  12.  irel  m50 state of veh2 relative to veh1
c  13.  erel  m50 element of veh2 minus m50 elements of veh1
c  14.  apyr  inertial pitch-yaw-roll  (321 angles of input att quat)
c  15.  auvw  uvw pitch-yaw-roll (321 angles  of quaternion obtained  by
c            multiplying uvw -> m50 quat qui by the  input   attitude
c            conjugate q,i.e. qui * con(q) )
c  16.  rpvm  range,veh1 and veh2 position maginitudes,
c            range rate,veh1 and veh2 velocity magnitudes
c
c  Note: If a relative trajectories is used to generate any of the above
c
c       options both the vehicle and the target option must be set.
c
      COMMON /xxdprm/ spg
c
c end block xxdprm
c
c
c
c end file mtf.dprm
c
```

```
c
c start file mtf.dwncct
c
c
c
c
c ************************************************************
c start common block dwnfmt_cct
c ************************************************************
c
      INTEGER skip( 1 )
c   skip flag
c   indicates that first record on the input cct tape is to be skipped
c when set to 1
c
      COMMON /dwnfmt_cct/ skip
c
c end block dwnfmt_cct
c
c
c
c end file mtf.dwncct
c
```

```
c
c start file mtf.dwnfiles
c
c
c
c
c ************************************************************
c start common block dwnfmt_files
c ************************************************************
c
      CHARACTER*72 fname( 1 )
c  output gff file options for downlist formatter,
c for local interface between user and common block xxfils in subroutine
c                                    /
c dfnput
c
c
      CHARACTER*4 hdrid( 1 )
c  output gff file options for downlist formatter,
c for local interface between user and common block xxfils in subroutine
c
c dfnput
c
c
      CHARACTER*4 cfrmid( 10 )
c  frame ID,s for output gff RELBET file
c  up to 10 ID's may be input associated in the order in which
c they occur to the presence of numbers larger than 200 in the
c id_sequence. ie., whenever a number greater than 200 occurs in
c the id_sequence then associate the next frame ID on the list
c with the successive information until a new frame ID is indicated.
c
      INTEGER filid( 1 )
c  frame ID,s for output gff RELBET file
c
c
      INTEGER frsize( 3 )
c   frame size
c   the integer word size of the different portions of the frame
c
c (1)  the number of integer words including the time-tag which
c make up the header portion of the frame
c
c (2)  the number of integer words in data portion of the frame
c
c (3)  the number of dummy places to fill after the data portion
c of the frame in integer words
c
      INTEGER idseq( 200 )
c   id sequence
c   the id_sequence is set up to resemble as closely as possible
c the frame building process. The internal datum ids of the
c msids used to build the frames are arranged in a way which
c reflects their use in building the output frame as follows:
c
c - the first number must be the internal id of the msid
```

```
c representing a time value which is used as the
c time-tag of the output frame. this value is further
c negated to emphasize that this parameter is destined
c for the time-tag position of the header portion of
c the frame
c
c - the next number must be a number greater than 200
c representing the frame id to be placed on
c the frame_id position of the header portion of the
c frame. as mentioned above the actual frame id is
c input in the list cfrmid.
c
c - the remaining numbers represent the internal ids of the
c m/sid's whose values will appear on the data portion
c of the output frame in the order in which they are
c herein specified.
c
c EX: idseq = -74,1000,56,57,58;
c
      INTEGER dbflag( 1 )
c   process frames with bad data
c   when set non-zero indicates that file frames are to be output
c even if some data bad (note:if time-tag is bad no data output)
c
      INTEGER chgdef( 1 )
c   change the default values
c   non_zero indicates that the inputs for frsize , cfrmid and idseq
c are to be used instead of the defaults.
c Note: if chgdef is set then all settings of frsize , cfrmid
c and idseq must be set.
c
      INTEGER end( 1 )
c   end flag
c   indicates that current input block is the last files input block
c and the next input block is an msids input block
c
      COMMON /dwnfmt_files/ fname, hdrid, cfrmid, filid, frsize, idseq,
     * dbflag, chgdef, end
c
c end block dwnfmt_files
c
c
c
c end file mtf.dwnfiles
c
```

```
c
c start file mtf.dwnmsids
c
c
c
c
c ************************************************************
c start common block dwnfmt_msids
c ************************************************************
c
      CHARACTER*10 msid( 1 )
c
c  alphanumeric m/sid number associated with downlisted parameter on cc
c t
c
      INTEGER id_nav( 2 )
c
c  the internal datum id and flag indicating nav buffer parameter
c (1), data id has range 1 to 200
c (2), =0, not part of nav buffer: =1, part of nav buffer
c
      DOUBLE PRECISION off_sc_conv_lo_hi( 5 )
c
c  the double precision data for converting and checking m/sid values
c (1), offset term added to value
c (2), scale factor multiplied on value before validation check
c (3), conversion factor to achieve internal RELBET units
c (4), minimum expected value
c (5), maximum expected value
c default: = 0.0 , 1.0 , 1.0 , -1.e30 , 1.e30
c
      INTEGER end( 1 )
c
c   indicates that current input block is the last msids input block
c and the next input block is a cct input block
c
      COMMON /dwnfmt_msids/ msid, id_nav, off_sc_conv_lo_hi, end
c
c end block dwnfmt_msids
c
c
c
c end file mtf.dwnmsids
c
```

```
c
c start file mtf.erth
c
c
c
c
c ************************************************************
c start common block xxerth
c ************************************************************
c
      DOUBLE PRECISION muerth
c  earth mu
c  earth gravitational parameter
c
      DOUBLE PRECISION req
c  earth eq rad
c  equatorial radius of earth
c
      DOUBLE PRECISION rpol
c  earth pol rad
c
c
      DOUBLE PRECISION wei
c  earth inrl rot rate
c  inertial rotation rate of the earth
c
      DOUBLE PRECISION f1
c  earth shape const (1-ellipticity)(aero)
c  computed from flattening as f1 = 1 - flat
c
      DOUBLE PRECISION f2
c  earth shape const
c   computed from flattenning as f2=f1*f1-1
c
      DOUBLE PRECISION flat
c  earth flattening
c  flattenning of earth, ie the ratio of the polar to equatorial radius
c .
c computed from input radii
c
      COMMON /xxerth/ muerth, req, rpol, wei, f1, f2, flat
c
c end block xxerth
c
c
c
c end file mtf.erth
c
```

```
c
c start file mtf.file
c
c
c
c
c ***************************************************************
c start common block xxfile
c ***************************************************************
c
      CHARACTER*72 nama1
c  att 1 name
c  name for first attitude file
c
      INTEGER unta1
c  att 1 unit
c  unit for first attitude file
c
      INTEGER uzea1
c  att 1 use
c  usage code for first attitude file
c
      CHARACTER*72 nama2
c  att2 name
c  name for second attitude file
c
      INTEGER unta2
c  att2 unit
c  unit for second attitude file
c
      INTEGER uzea2
c  att 2 use
c  usage code for second attitude file
c
      CHARACTER*72 name1
c  eph 1 name
c  name for first ephemeris file
c
      INTEGER unte1
c  eph 1 unit
c  unit for first ephemeris file
c
      INTEGER uzee1
c  eph 1 use
c  usage code for first ephemeris file
c
      CHARACTER*72 name2
c  eph 2 name
c  name for second ephemeris file
c
      INTEGER unte2
c  eph 2 unit
c  unit for second ephemeris file
c
      INTEGER uzee2
c  eph 2 use
```

```
c  usage code for second ephemeris file
c
      CHARACTER*72 namd1
c  data 1 name
c  name for data file 1
c
      INTEGER untd1
c  data 1 unit
c  unit for data file 1
c
      INTEGER uzed1
c  data 1 use
c  usage code for data file 1
c
      CHARACTER*72 namd2
c  data 2 name
c  name for data file 2
c
      INTEGER untd2
c  data 2 unit
c  unit for data file 2
c
      INTEGER uzed2
c  data 2 use
c  usage code for data file 2
c
      CHARACTER*72 namv1
c  svel 1 name
c  name for sensed velocity file 1
c
      INTEGER untv1
c  svel 1 unit
c  unit for sensed velocity file 1
c
      INTEGER uzev1
c  svel 1 use
c  usage code for sensed velocity file 1
c
      CHARACTER*72 namv2
c  svel 2 name
c  name for sensed velocity file 2
c
      INTEGER untv2
c  svel 2 unit
c  unit for sensed velocity file 2
c
      INTEGER uzev2
c  svel 2 use
c  usage code for sensed velocity file 2
c
      CHARACTER*72 namr
c  rtrj name
c  name for relative trajectory file
c
      INTEGER untr
c  rtrj unit
```

```
113   c  unit for relative trajectory file
114   c
115            INTEGER uzer
116   c  rtrj use
117   c  usage code for relative trajectory file
118   c
119            CHARACTER*72 nams
120   c  sol name
121   c  name for solution file
122   c
123            INTEGER unts
124   c  sol unit
125   c  unit for solution file
126   c
127            INTEGER uzes
128   c  sol use
129   c  usage code for solution file
130   c
131            INTEGER tapeu
132   c  number for tape i
133   c
134   c
135            INTEGER filids( 3,2 )
136   c  file ids for vehicles
137   c
138   c
139           COMMON /xxfile/ nama1, unta1, uzea1, nama2, unta2, uzea2, name1,
140          * unte1, uzee1, name2, unte2, uzee2, namd1, untd1, uzed1, namd2,
141          * untd2, uzed2, namv1, untv1, uzev1, namv2, untv2, uzev2, namr,
142          * untr, uzer, nams, unts, uzes, tapeu, filids
143   c
144   c end block xxfile
145   c
146   c
147   c
148   c end file mtf.file
149   c
```

```
c
c start file mtf.fils
c
c
c
c ***********************************************************
c start common block xxfils
c ***********************************************************
c
      INTEGER filsta( 20 )
c  internal processing status for files
c
c
      INTEGER frmnum( 20 )
c  number of frames processed for files
c
c
      INTEGER frmsze( 3,20 )
c   frame after
c  the number of dummy places to fill after the data portion of the
c frame in integer words, the number of integer words in data portion
c of the frame,the number integer words including the time-tag which
c make up
c
      INTEGER datseq( 200,20 )
c   id sequence
c   input inf which is stored in datseq
c
      INTEGER dbproc( 20 )
c   process frames with bad data
c   when set non-zero indicates that file frames are to be output
c even if some data bad (note:if time-tag is bad no data output)
c
      INTEGER lastid( 20 )
c   position of the last id
c   indicates the length of each data sequence
c
      COMMON /xxfils/ filsta, frmnum, frmsze, datseq, dbproc, lastid
c
c end block xxfils
c
c
c
c end file mtf.fils
c
```

```
c
c start file mtf.gnrl
c
c
c
c
c ************************************************************
c start common block xxgnrl
c ************************************************************
c
      INTEGER bugs
c  debug file
c
c
      INTEGER in
c  input file unit designation
c
c
      INTEGER term
c  unit for short print
c
c
      INTEGER out
c  unit for output print
c
c
      INTEGER pbug( 19 )
c  debug flags
c
c
      INTEGER pfatal
c  fatal error flag
c  initialized to 0 in ingrss, set negative on calls to gerror.
c  -1   fatal error: program  should stop
c   0   no error condition
c  >0   program defined stop flag
c
      INTEGER interm
c  standard input unit
c  specifies unit for standard input.  set to 5 in ixgnrl
c
      COMMON /xxgnrl/ bugs, in, term, out, pbug, pfatal, interm
c
c end block xxgnrl
c
c
c
c end file mtf.gnrl
c
```

```
c
c start file mtf.graf
c
c
c
c
c ************************************************************
c start common block xxgraf
c ************************************************************
c
      INTEGER option
c  main option
c  designate main action:
c    neg      stop
c    non neg  plot
c    pos      use automatic scale of axes
c    zero     no automatic scale of axes. User must give min,max and
c             steps for x and y axes.(xmmx,ymmx,xyzstp)
c
      CHARACTER*60 xlabel
c  x axis label
c  specifies the label for x axis. The  first character cues special
c features.'!' label and tics on opposite  side of axis. '$' No axis
c and tick mark are drawn. Tick mark can be drawn if blank in quote
c is used.
c
      CHARACTER*60 ylabel
c  y axis label
c  specifies the label for y axis. See xlabel for detail.
c
      CHARACTER*60 zlabel
c  z axis label
c  specifies the label for z axis . See xlabel for detail.
c
      CHARACTER*60 title
c  main title
c  up to 4 titles may be specified. each title may be up to  60
c characters long. each title should start with and end with single
c quote. each title has a scale factor associated with.this factor
c specifies what factor of nominal character size the title is to be
c displayed with . a negative scale factor results in the title
c being underlined.
c
      CHARACTER*60 title2
c   title2
c  see title for description
c
      CHARACTER*60 title3
c   title3
c  see title for description
c
      CHARACTER*60 title4
c   title4
c  see title for description
c
      REAL ttlmul( 4 )
```

```
c  title scale
c  title scale: specifies the scale for title . A negative values cause
c
c the corresponding title to be underlined.
c
      CHARACTER*8 curves( 20 )
c  curves to plot
c  curves to plot. Up to 20 curves may be defined.  Up to 7 characters
c can be used for curves name. The last character is reserved for dollar
c
c sign.
c
      INTEGER pframe
c  frame flag
c  Plot frame flag : specifies  whether a frame is to be drawn around
c plot.
c    > 0 Draw frame
c   else  No frame
c
      INTEGER xygrid( 3 )
c  xy grid flags
c  xygrid flag : specifies the grid options.  The first two entries
c specify the grid frequency k for the x and y axes respectively.
c   K > 0 k grid line per tic mark
c   K = 0 No grid lines
c   K < 0 k tic marks per grid line
c The third entry specifies the line type. Options are same as for
c curve line types.
c
      CHARACTER*4 xyaxs( 2 )
c  xy axes option
c  specifies the character version of axes options. the options are:
c S for linear display scale, L for log, I for obsolute scale,
c P for Polar. Only the first two characters matter, e.g.,'IL', is
c linear in X and Log in Y.
c
      CHARACTER*4 ttlwd
c  title width option
c
c
      CHARACTER*4 lblwd
c  label width option
c
c
      CHARACTER*4 xyzln( 3 )
c  line options
c
c
      REAL xypage( 2 )
c  page size
c  the page size is the area which is taken up by the entire graphic
c display --both label and plot. it is specified in nominla inches,
c the first entry corresponding to the horizontal (x) dimension and
c the second to the vertical (y) dimension. the term nominal is used
c since the actual size depends upon the particular graphic device.
c the actual size may be smaller for hp . a typical page size is
```

```
c 11 by 8.5
c
      REAL porgin( 2 )
c  physical origin location
c  physical origin : specifies the location of plot origin in nominal
c inches from the lower left hand corner of the page . A negative
c value results in the program setting the plot origin.
c
      REAL xyarea( 2 )
c  subplot area
c  this is the area encompassed by the x and y axes. thus the value
c specifies the length of the x axis and the second value specifies
c the length of the y axis. it is specifies in nominal inches. the
c default value for subplot area is 6 by 6. a typical subplot area
c is 8.5 by 6.0 for hp screen.
c
      REAL frmthk
c  frame thickness
c  Specifies the scale for the frame thickness in nominal inches.
c If a value greater than zero is specified, the frame around the plot
c will be drawn with the approximate thickness.  Values greater than
c 0.125 are reset to this value.
c
      REAL gmrgin
c  grace margin
c  specifies a distance in nominal inches from the plot
c area border within which points will be plotted.   Positive values
c allow for points being plotted outside the plot area.Negative values
c ensure that point will lies within the plot area.
c
      REAL xyhite
c  xy label height fact
c  axis label scale : specifies for axis labels in multiple of .14
c
c inches.
c
      REAL xyznch( 3 )
c  axes scale per inch
c  specifies absolute scale of axes in axis units per inch .
c
      REAL xyzstp( 3 )
c  axes units per tic
c  specifies axis units per tic. this paramter is specified if
c and only if option = 0 is used(no scaling is done by program)
c
      REAL xmmx( 2 )
c  x axis min/max
c  this parameter is specified if and only if option = 0 is
c used. this implies no scaling is done by program.
c
      REAL ymmx( 2 )
c  y axis min/max
c  see xmmx for detail
c
      REAL zmmx( 2 )
c  z axis min/max
```

```
c  see xmmx for detail
c
      REAL xyang( 3 )
c  xyz label angles
c  specifies angle relative to horizontal for tic numbers. The default
c is 0.
c
      REAL plegnd
c  legend size
c  Legend scale : specifies scale factor for legend display as a
c multiple of standard character height of 0.14 inches. Options are :
c   > 0 Display legend width specified scale
c   else No legend
c
      REAL lgnpos( 2 )
c  legend position
c  Specified the position of the legend's upper hand corner in nominal
c inches form the plot origin.
c
      CHARACTER*60 xtitle( 4 )
c  internal titles
c  actual titles.  set up from input
c
      INTEGER nttl
c  number of titles
c  actual titles.  set up from input
c
      REAL kqsz( 4,3 )
c  axis size array
c  size array of min, max, tic interval, inch scale defined from
c input values.
c
      COMMON /xxgraf/ option, xlabel, ylabel, zlabel, title, title2,
     * title3, title4, ttlmul, curves, pframe, xygrid, xyaxs, ttlwd,
     * lblwd, xyzln, xypage, porgin, xyarea, frmthk, gmrgin, xyhite,
     * xyznch, xyzstp, xmmx, ymmx, zmmx, xyang, plegnd, lgnpos, xtitle,
     * nttl, kqsz
c
c end block xxgraf
c
c
c
c end file mtf.graf
c
```

```
c
c start file mtf.grav
c
c
c
c
c ************************************************************
c start common block xxgrav
c ************************************************************
c
      DOUBLE PRECISION rgrav
c  grav radius
c  geopotential model earth radius
c
      DOUBLE PRECISION cterms( 35 )
c  c harms
c  earth harmonics c22 thru c88
c
      DOUBLE PRECISION sterms( 35 )
c  s harms
c  earth harmonics s22 thru s88
c
      DOUBLE PRECISION jterms( 2:8 )
c  j harms
c  earth j harmonic terms
c
      COMMON /xxgrav/ rgrav, cterms, sterms, jterms
c
c end block xxgrav
c
c
c
c end file mtf.grav
c
```

```
c
c start file mtf.init
c
c
c
c
c ************************************************************
c start common block xxinit
c ************************************************************
c
      DOUBLE PRECISION trvint( 7,2 )
c  init t,r,v
c  initial time and state.  time is seconds since base time and
c  occurs in entry 1,i.  state is m50 cartesian and occurs as
c  position, velocity in terms 2-7,i.  here i refers to the vehicle
c
      COMMON /xxinit/ trvint
c
c end block xxinit
c
c
c
c end file mtf.init
c
```

```
c
c start file mtf.kal
c
c
c
c
c ************************************************************
c start common block xxkal
c ************************************************************
c
      DOUBLE PRECISION dtmax
c  max step
c  maximum allowed time step for kalman filter
c
      DOUBLE PRECISION edcrit( 25 )
c  edit criterion
c  if abs(resid) > sigma*edcrit for a given observation, then that
c observation is edited
c
      DOUBLE PRECISION var( 20 )
c  observation variances
c  These are the default observation sigmas reset to variances on
c initialization process.
c
      INTEGER lrmodl
c  noise model option
c  This option implements the lrbet3 range, range rate, and
c range bias noise models :=0,do not use;>0,use
c
      DOUBLE PRECISION ncons( 10 )
c  lrbet3 noise model constants
c  These are the noise model constants: (1-4),range
c ;(5-8),range bias;(9,10),range rate
c
      DOUBLE PRECISION undrwt( 2 )
c  underweighting options
c  undrwt(1) >0, implements the underweighting of the update
c by adjusting the computed residual variance by a factor of (1-undrwt).
c
c  Note: undrwt(1) =0, no underweighting
c  undrwt(2) , sets the criterion for underweighting .ie., whenever
c  RSS(relative position sigmas)**2>=undrwt(2)**2.
c
      INTEGER sxcld( 24 )
c  state exclusion flags
c  if the sxcld flag corresponding to an element in the filter state
c vector is 1, then that element is excluded from consideration in the
c filter processing
c
     COMMON /xxkal/ dtmax, edcrit, var, lrmodl, ncons, undrwt, sxcld
c
c end block xxkal
c
c
c
c end file mtf.kal
```

```
   57    c
```

```
c
c start file mtf.mas
c
c
c
c
c ***************************************************************
c start common block xxmas
c ***************************************************************
c
      INTEGER lmstab
c  length of mass table
c  index of highest entry in mass table
c
      DOUBLE PRECISION mastab( 3,10,2 )
c  mass tables
c  vehicle mass tables... contain up to 10 entries for 2 vehicles.
c   mastab(1,*,*)= start time of entry in seconds since base
c   mastab(2,*,*)= mass at start time
c   mastab(3,*,*)= rate of change of mass
c  entry is in effect from start time of entry to start time of next
c  entry
c
      COMMON /xxmas/ lmstab, mastab
c
c end block xxmas
c
c
c
c end file mtf.mas
c
```

```
c
c start file mtf.mast
c
c
c
c
c ************************************************************
c start common block xxmast
c ************************************************************
c
      INTEGER xbugs
c   unit number of debug file
c   unit number for the debug print file
c
      INTEGER hifile
c   (unused ) max unit allowed for internal files
c   default=10
c
      INTEGER hiunit
c   (unused ) highest unit
c   the highest unit number used for assigned files
c
      INTEGER maxfil
c   maximum number of files
c   the maximum number of files to be written to be used for dimension
c and looping purposes
c
      INTEGER maxtyp
c   maximum number of m/sid's
c   the maximum number of m/sid's to be processed on any file to be
c used for dimension and looping purposes
c
      INTEGER msnum
c   number of m/sid's to be processed
c
c
      INTEGER xin
c   input file
c
c
      INTEGER xout
c   output print file
c
c
      INTEGER xpbug( 20 )
c   debug print flags
c
c
      INTEGER tcrflt
c   time current file
c   internal id for t.current.filt and time-homo set processing flag
c
      DOUBLE PRECISION timbeg
c   begin time
c   begin time for data retrieval for files under construction
c
```

```
57          DOUBLE PRECISION timend
58    c   end time for data retrieval
59    c
60    c
61          DOUBLE PRECISION timoff
62    c   time off
63    c   delta difference to check time-tags against the record clock times
64    c
65          COMMON /xxmast/ xbugs, hifile, hiunit, maxfil, maxtyp, msnum, xin
66         * , xout, xpbug, tcrflt, timbeg, timend, timoff
67    c
68    c end block xxmast
69    c
70    c
71    c
72    c end file mtf.mast
73    c
```

```
c
c start file mtf.max
c
c
c
c
c ************************************************************
c start common block xxmax
c ************************************************************
c
      INTEGER maxveh
c  max number of vehicles
c
c
      INTEGER nveh
c  number of vehicles
c
c
     COMMON /xxmax/ maxveh, nveh
c
c end block xxmax
c
c
c
c end file mtf.max
c
```

```
c
c start file mtf.misc
c
c
c
c
c ************************************************************
c start common block xxmisc
c ************************************************************
c
      CHARACTER*60 jobdes( 2 )
c  job description
c
c
      INTEGER obsrvr
c  id of observor vehicle
c
c
      INTEGER target
c  id of target vehicle
c
c
      COMMON /xxmisc/ jobdes, obsrvr, target
c
c end block xxmisc
c
c
c
c end file mtf.misc
c
```

```
      c
      c start file mtf.moon
      c
      c
      c
      c
      c ***************************************************************
      c start common block xxmoon
      c ***************************************************************
      c
            DOUBLE PRECISION mumoon
      c  moon mu
      c  moon gravitational parameter
      c
            DOUBLE PRECISION dtmoon
      c  moon dt
      c  time step for frequency of moon ephemeris
      c
            DOUBLE PRECISION tmoo0
      c  moon state time
      c  base time for rvmoo0
      c
            DOUBLE PRECISION rvmoo0( 6 )
      c  init moon state
      c  m50 position of moon relative to earth used to generate moon
      c  ephemeris
      c
            INTEGER nmord
      c  moon intrp order
      c  number of ephemeris points used in interpolation of moon state
      c
           COMMON /xxmoon/ mumoon, dtmoon, tmoo0, rvmoo0, nmord
      c
      c end block xxmoon
      c
      c
      c
      c end file mtf.moon
      c
```

```
c
c start file mtf.msid
c
c
c
c
c ***************************************************************
c start common block xxmsid
c ***************************************************************
c
      INTEGER adrtbl( 200,3 )
c  address table
c  stores the location, number of samples,and data type for the
c desired msids
c
      INTEGER imsbuf( 500 )
c  msid buffer
c  integer buffer containing the alphanumeric msids
c
      INTEGER frmloc( 200 )
c  nav buffer flag
c  flag indicating parameter member of nav buffer on downlist
c
      INTEGER frmtim( 200 )
c  time-tag flag
c  indicates that msid is used as a time-tag
c
      INTEGER xdatid( 200 )
c  exclude data
c  indicates that msid is to be processed if set to 0
c
      DOUBLE PRECISION ddatum( 200 )
c  data buffer
c  stores the data values for desired msids from the cct
c
      INTEGER dstat( 200,2 )
c  data buffer
c  stores the validity of data values for desired msids from the cct
c
      DOUBLE PRECISION dofset( 200 )
c  offset value
c  value added to msid value for output
c
      DOUBLE PRECISION dscale( 200 )
c  scale value
c  value multiplied onto msid value before any validity checks made
c
      DOUBLE PRECISION dconv( 200 )
c  conversion value
c  value multiplied onto msid value for conversion
c
      DOUBLE PRECISION dvalid( 200,2 )
c  validity limits
c  low limit and hi limit for msid validity checks
c
      COMMON /xxmsid/ adrtbl, imsbuf, frmloc, frmtim, xdatid, ddatum,
```

```
     * dstat, dofset, dscale, dconv, dvalid
c
c end block xxmsid
c
c
c
c end file mtf.msid
c
```

```
c
c start file mtf.name
c
c
c
c
c ************************************************************
c start common block xxname
c ************************************************************
c
      CHARACTER*56 prnam
c  processor name
c
c
      INTEGER lprnam
c  length of processor name
c
c
      INTEGER prbase
c  processor baseline
c
c
      INTEGER prver
c  processor version
c
c
      CHARACTER*12 prgver
c  gff name/version
c
c
     COMMON /xxname/ prnam, lprnam, prbase, prver, prgver
c
c end block xxname
c
c
c
c end file mtf.name
c
```

```
c
c start file mtf.nflz
c
c
c
c
c ****************************************************************
c start common block xxnflz
c ****************************************************************
c
      INTEGER funit( 5 )
c  file units
c  specifies unit number for files
c
      CHARACTER*72 fname( 5 )
c  file names
c  specifies names of files used. uses are as follows
c for all
c
c    1        input first ephemeris or relative trajectory file
c
c    2        input second ephemeris or relative trajectory file
c
c **note that posx determine which vehicle corresponds to which file
c for xqdsp
c
c   3        input attitude file
c
c   4        output plot file     ,
c
c   5        not used
c for xcmpar
c
c   3        output ephemeris for vehicle 1
c
c   4        output ephemeris for vehicle 2
c
c   5        output relative trajectory of veh 2 with veh 1 base
c
      INTEGER bfopt
c  base file flag
c  specifies id of file to use as base.  output times correspond to
c the times of this file and other input information is interpolated
c to these times.  if this option is chosen, then the delta time in
c time in the times input specifies a minimum number of records
c between consecutive output times.
c the following options are valid
c
c   1        first ephemeris (xqdisp,xcmpar)
c
c   2        second ephemeris (xqdisp,xcmpar)
c
c   3        attitude file (xqdisp)
c
c   else     error
c
```

```
      INTEGER bfrec
c  base file record
c
c
      INTEGER curfl
c  current file index
c
c
      INTEGER basefl
c  base file index
c
c
      INTEGER flstat( 5 )
c  file status array
c  positive values indicate file is okay and open, negative open
c and error, 0 file not open
c
      INTEGER flx( 5 )
c  file data length
c  specifies the number of dp data words needed from the file
c
      INTEGER posx( 2 )
c  location for data
c  absolute value give input file id, either file 1 or 2.
c the sign specifies whether the state (+) or the relative
c state (-)
c
      INTEGER lordx( 2 )
c  interpolation flag
c  specifies interpolation option for file i.  options are
c
c   -1       from base file (set by program, not user input)
c
c   1        use two point position and velocity interpolation
c
c   2-10     use lagrangian interpolation with the specified number of p
c oints
c
      INTEGER xflg( 2 )
c  file used for state i
c   note that this is set to the file need by the vehicle in ndnflz and
c
c used in xqwget as the file index for the vehicle state. furthermore
c note that ndflz uses xqneed to determine whether or not the vehicle
c is considered.  this is set in xqkmpt1
c
      INTEGER xqneed( 3 )
c  state/att flags
c  positive value indicates respectively vehicle 1,2, or attitude
c is needed. set by xqkmpt1 and used by ndfnpt
c
      INTEGER print
c  print flag
c  positive value to generate print to unit out
c
      INTEGER plot
```

```
c  plot flag
c  positive value to save plot file to file 4
c
      INTEGER trjout( 3 )
c  traj flags
c  positive value to generate the specified file. note
c that this is used by xcmpar and ignored by xqdsp
c
c     1        generate ephemeris for vehicle 1
c
c     2        generate ephemeris for vehicle 2
c
c     3        generate relative trajectory, base =veh1, rel = vehicle 2
c
c
      COMMON /xxnflz/ funit, fname, bfopt, bfrec, curfl, basefl, flstat
     * , flx, posx, lordx, xflg, xqneed, print, plot, trjout
c
c end block xxnflz
c
c
c
c end file mtf.nflz
c
```

```
c
c start file mtf.ntrp
c
c
c
c ************************************************************
c start common block xxntrp
c ************************************************************
c
      INTEGER rbuf( 12,2 )
c  record buffer
c
c
      DOUBLE PRECISION xbuf( 120,2 )
c  traj intrp value table
c
c
      DOUBLE PRECISION xvalz( 12,2 )
c  traj valu table
c
c
      DOUBLE PRECISION fbuf( 120,2 )
c  intrp factor table
c
c
      DOUBLE PRECISION tbuf( 10,2 )
c  time table
c
c
     COMMON /xxntrp/ rbuf, xbuf, xvalz, fbuf, tbuf
c
c end block xxntrp
c
c
c
c end file mtf.ntrp
c
```

```
c
c start file mtf.obs
c
c
c
c
c ************************************************************
c start common block xxobs
c ************************************************************
c
      INTEGER pcoas
c  coas initialization
c
c
      INTEGER pdgrv
c  partials flag
c
c
      INTEGER pradar
c  radar initialization
c
c
      INTEGER ptrack
c  star tracker initialization
c
c
      DOUBLE PRECISION mibov( 3,3 )
c  attitude matrix for observor
c
c
      DOUBLE PRECISION qibov( 4 )
c  observor attitude quaternion
c
c
      DOUBLE PRECISION xi( 6,2 )
c  vehicle states
c
c
     COMMON /xxobs/ pcoas, pdgrv, pradar, ptrack, mibov, qibov, xi
c
c end block xxobs
c
c
c
c end file mtf.obs
c
```

```
c
c start file mtf.pkt
c
c
c
c
c ************************************************************
c start common block xxpkt
c ************************************************************
c
      INTEGER packet( 50,20 )
c  i/o packets
c  array of i/o packets for a maximum of 10 file in the trw
c gff format.  packet contents are described in the general file
c format (gff) users manual.  the particular index corresponding
c to a file is determined by the main program.  however certain
c applications utilizing the propagation modules (prop and force)
c should make the following identifications
c
c    file 1   input ephemeris 1 (traj or rel traj)
c
c    file 2   input ephemeris 2 (traj or rel traj)
c
c    file 3   attitude 1
c
c    file 4   attitude 2
c
c    file 5   sensed velocity 1
c
c    file 6   sensed velocity 2
c
c    file 7   data file 1
c
c    others   as needed by particular program
c
c note that not all the files need be opened or used in a
c particular program
c
      COMMON /xxpkt/ packet
c
c end block xxpkt
c
c
c
c end file mtf.pkt
c
```

```
c
c start file mtf.pmmx
c
c
c
c
c ************************************************************
c start common block xxpmmx
c ************************************************************
c
      REAL pmmx( 2,21 )
c  min/max values
c  min/max values for y axis
c
      COMMON /xxpmmx/ pmmx
c
c end block xxpmmx
c
c
c
c end file mtf.pmmx
c
```

```
c
c start file mtf.prnt
c
c
c
c
c ***********************************************************
c start common block xxprnt
c ***********************************************************
c
      INTEGER lpage
c  lines/page
c   specifies number of lines per page. if negative or
c zero, then no paging is provided.
c
      INTEGER npage
c  page number
c
c
      INTEGER lnz
c  line count
c
c
      INTEGER dlnz
c  line count increment
c
c
      INTEGER col80
c  column width option
c  positive value sets formats for 80 columns.
c otherwise, 130 columns.
c
      INTEGER header
c  header print option
c  positive value causes a header of time and scale
c information to be printed for each output time
c
      INTEGER nfrmat
c  format id
c  specifies output format.  options are
c
c    1     floating point
c
c    else  fixed point
c
       COMMON /xxprnt/ lpage, npage, lnz, dlnz, col80, header, nfrmat
c
c end block xxprnt
c
c
c
c end file mtf.prnt
c
```

```
c
c start file mtf.prop
c
c
c
c
c ***************************************************************
c start common block xxprop
c ***************************************************************
c
      INTEGER rvopt( 2 )
c  prop option
c  flag for how vehicle state is obtained options are
c
c    <0  interpolate
c
c     1  runge kutta integration
c
c     2  super g integration
c
c     e  error
c
      INTEGER pprpop( 2 )
c  propagation mode
c
c
      INTEGER paero( 2 )
c  aero force
c
c
      INTEGER pcb( 2 )
c  central body force
c
c
      INTEGER pdrag( 2 )
c  drag force
c
c
      INTEGER pharm( 2 )
c  harmonics force
c
c
      INTEGER pmoon( 2 )
c  moon force
c
c
      INTEGER prad( 2 )
c  solar radiation flag
c
c
      INTEGER psvel( 2 )
c  sensed velocity flag
c
c
      INTEGER psun( 2 )
c  sun force
```

```
c
c
      INTEGER pvent( 2 )
c  vent force
c
c
      DOUBLE PRECISION dtnom( 2 )
c  nom step
c  specifies largest step size in integration step
c
      COMMON /xxprop/ rvopt, pprpop, paero, pcb, pdrag, pharm, pmoon,
     * prad, psvel, psun, pvent, dtnom
c
c end block xxprop
c
c
c
c end file mtf.prop
c
```

```
c
c start file mtf.qcrv
c
c
c
c
c ***************************************************************
c start common block xxqcrv
c ***************************************************************
c
      INTEGER ncrvs
c  number of curves to plot
c
c
      CHARACTER*8 kname( 20 )
c  curve names
c  user defined curve names. Up to 7 characters can be used for curve
c
c names. The last character is reserved for dollar sign.
c
      CHARACTER*8 kparms( 3,20 )
c  parameter id's
c  parameter id's. It must match exactly in order to compute idxyz
c
      INTEGER imrk( 20 )
c  line option of curves
c  symbol frequency : Specify the frequency of plot symbols for the
c corresponding curve. For a value k :
c    k > 0 symbol every kth point. Line through each point.
c    k = 0 no symbols.  line through each point.
c    k < 0 symbol every kth point .  No line.
c
      INTEGER nlabel( 20 )
c  point label options
c  label frequency: Specifies the frequency of integer labels for
c curve points.
c    N < 0 label every Nth point in the sequence 1,2,3,...
c    N > 0 label every Nth point in the sequence 1,N,2N,..
c    N = 0 no labels
c
      INTEGER ksmbol( 20 )
c  curve plot symbol code
c  Specifies the plotting symbols for corresponding curve.
c For more detail on the different symbol codes ,see DISSPLA manual
c for symbols and their corresponding sequence numbers.
c
      CHARACTER*72 kfile( 20 )
c  curve file
c  Input file name. It specified where the input datas come from.
c Different curves can have different input files.
c
      INTEGER kedit( 20 )
c  edit plot options
c
c
      INTEGER kline( 20 )
```

```
c  line type code
c  Specifies the line type option for curve. Option are :
c   0 connected
c   1 Dot
c   2 Dash
c   3 Chained dot
c   4 Chained dot dash
c
        REAL kstep( 20 )
c  step size
c  curve interval : specifies the plot interval for curve i.e.,begin,
c end, and step size.  Times are specified as seconds from base time.
c If kstep > 0 then  use in time seconds, else use counts
c
        REAL kspan( 2,20 )
c  begin,end
c  begin and end time span (time or count depend)
c
        REAL psize( 20 )
c  plot symbol size
c  specifies the the size of the plot symbols.
c
        INTEGER crvz( 20 )
c  plot curves ids
c  indices of curves considered in current plot
c
        INTEGER idxyz( 3,20 )
c  parameter ids
c  indices of parameters corresponding to each curve
c these reflect the labels kparms
c
      COMMON /xxqcrv/ ncrvs, kname, kparms, imrk, nlabel, ksmbol, kfile
     * , kedit, kline, kstep, kspan, psize, crvz, idxyz
c
c end block xxqcrv
c
c
c
c end file mtf.qcrv
c
```

```
c
c start file mtf.qcur
c
c
c
c
c ***************************************************************
c start common block xxqcur
c ***************************************************************
c
      INTEGER cfl
c  current file id
c
c
      CHARACTER*4 kfid( 3 )
c  parm frame ids
c
c
      INTEGER kidat( 3 )
c  parm data indices
c
c
      REAL kqmmx( 2,3 )
c  parm min/max
c
c
      INTEGER kqpmx( 3 )
c  parm min/max flag
c
c
      REAL kqoset( 3 )
c  parm offsets
c
c
      REAL kqscal( 3 )
c  parm scale factors
c
c
      CHARACTER*12 kqunit( 3 )
c  parm unit names
c
c
      INTEGER kqdeln
c  count delta
c
c
      INTEGER kqnbeg
c  start count
c
c
      INTEGER kqnend
c  end count
c
c
      INTEGER kqnxtn
c  next count
```

```
c
c
      INTEGER kqtotn
c  total count
c
c
      INTEGER kqcurn
c  current count
c
c
      INTEGER kqtoti
c  total inclusions
c
c
      INTEGER kqcuri
c  current inclusions
c
c
      DOUBLE PRECISION kqdelt
c  time delta
c
c
      DOUBLE PRECISION kqtbeg
c  start time
c
c
      DOUBLE PRECISION kqtend
c  end time
c
c
      DOUBLE PRECISION kqnxtt
c  next time
c
c
      INTEGER kqrec
c  current record
c
c
      INTEGER frmall
c  frame id flag
c  positive value indicates that frame ids are not checked
c
      INTEGER kqedit
c  edit flag
c  specifies edit option for curve:
c   neg     plot edited points only
c   zero    ignore edit status
c   pos     plot unedited only
c
      INTEGER kqiop( 50 )
c  file i/o packet
c  specifies edit option for curve:
c   neg     plot edited points only
c   zero    ignore edit status
c   pos     plot unedited only
c
```

```
      COMMON /xxqcur/ cfl, kfid, kidat, kqmmx, kqpmx, kqoset, kqscal,
     * kqunit, kqdeln, kqnbeg, kqnend, kqnxtn, kqtotn, kqcurn, kqtoti,
     * kqcuri, kqdelt, kqtbeg, kqtend, kqnxtt, kqrec, frmall, kqedit,
     * kqiop
c
c end block xxqcur
c
c
c
c end file mtf.qcur
c
```

```
c
c start file mtf.qgen
c
c
c
c
c ***************************************************************
c start common block xxqgen
c ***************************************************************
c
      CHARACTER*4 qpdev
c  device for plot
c
c
      INTEGER qplevl
c  level of plot
c
c
      INTEGER qpdvup
c  device initialization
c
c
      INTEGER qperup
c  error unit initialization
c
c
      INTEGER popunt
c  dispop unit
c
c
      INTEGER ndim
c  dimension of current curve
c  the default ndim is 2.
c
      REAL sthite
c  standard symbol height
c
c
     COMMON /xxqgen/ qpdev, qplevl, qpdvup, qperup, popunt, ndim,
     * sthite
c
c end block xxqgen
c
c
c
c end file mtf.qgen
c
```

```
c
c start file mtf.qlbf
c
c
c
c
c ************************************************************
c start common block xxqlbf
c ************************************************************
c
      INTEGER ipkleg( 210 )
c  legend buffer
c
c
     COMMON /xxqlbf/ ipkleg
c
c end block xxqlbf
c
c
c
c end file mtf.qlbf
c
```

```
c
c start file mtf.qprm
c
c
c
c
c ************************************************************
c start common block xxqprm
c ************************************************************
c
      INTEGER pmxflg( 21 )
c  min/max options
c  specifies whether min/max check is performed for parameter.
c The scaled parameter with offset subtracted is compared against
c the values specified by xmmx,ymmx. Options are:
c   2  used in Ascale only. Ascale does not scale this parameter.
c   1  plot out range value at extremal values.
c   0  no check
c  -1  omit all out range points
c  -2  omit points above minimun,plot points below minimun at min values
c
c  -3  omit points above maximun,plot points below maximun at max values
c
c
      INTEGER pword( 21 )
c  index to data word
c  This is where the actual data resides. pword(21) is reserved for
c time and is set to zero.
c
      CHARACTER*4 pfid( 21 )
c  frame id's
c  specifies the frame type corresponding to the parameter. Only frame
c with the specified frame type will be used in obtaining the paramter.
c Special options are:
c       '????' wild card: consider all frames.
c       'time' time word wild card. When referenced by a curve, the frame
c               type of the other parameter is assumed and the time word of
c               that frame is plotted.
c       else   frame id from file must exact match
c
      CHARACTER*12 punits( 21 )
c  units names
c  specifies the units names for parameters
c
      CHARACTER*8 pname( 21 )
c  parameter names
c  user defined parameter names. pname(21) is reserved for TIME only.
c Up to 7 characters can be used for pname. The last character is
c reserved for dollar sign.
c
      REAL pscale( 21 )
c  scale factors
c  specifies the scale factor for parameters as internal units per
c display unit. the default scale is 1.
c
      REAL pofset( 21 )
```

```
c  offset
c  value that is subtracted from parameter before display.
c specified in the units determined by pscale.  note that min/max
c are checked after the offset is removed and thus should be relative
c to the offset
c
      COMMON /xxqprm/ pmxflg, pword, pfid, punits, pname, pscale, pofset
c
c end block xxqprm
c
c
c
c end file mtf.qprm
c
```

```
c
c start file mtf.rnpx
c
c
c
c
c ************************************************************
c start common block xxrnpx
c ************************************************************
c
      DOUBLE PRECISION trnp0
c  rnp time
c  specifies time in seconds from base time at which the
c base rnp matrix is anchored.
c
      DOUBLE PRECISION rnp0( 3,3 )
c  rnp matrix
c  true of date to m50 transformation matrix at base time.
c computed by program if abs( rnp0(1,1) )>1.2
c else input used
c
      DOUBLE PRECISION cdetut
c  et offset
c  specifies offset between ephemeris time and universal time
c
c    cdetut(1)= et - ut in seconds
c
c    cdetut(2)= rate of et - ut increase in seconds/second
c
      DOUBLE PRECISION alphah
c  base hour angle
c  base hour angle of earth.  computed upon initialization
c
       COMMON /xxrnpx/ trnp0, rnp0, cdetut, alphah
c
c end block xxrnpx
c
c
c
c end file mtf.rnpx
c
```

```
c
c start file mtf.rpst
c
c
c
c
c ***************************************************************
c start common block xxrpst
c ***************************************************************
c
      CHARACTER*72 infil
c  input obs file name
c
c
      CHARACTER*72 outfil
c  output obs file name
c
c
      DOUBLE PRECISION antang
c  antenna angle
c  antenna angle for radar entry in radians
c
      COMMON /xxrpst/ infil, outfil, antang
c
c end block xxrpst
c
c
c
c end file mtf.rpst
c
```

```
c
c start file mtf.scov
c
c
c
c
c ************************************************************
c start common block xxscov
c ************************************************************
c
      INTEGER iopt
c  input option on state covariances UVW
c  iopt = 1 indicates that the input target UVW covariance is
c referenced to the base state UVW coordinate frame.
c iopt = 0 indicates that the input target UVW covariance is
c referenced to the target inertial frame.( similar to Lear inputs)
c
      DOUBLE PRECISION rvcov( 6,6,2 )
c  position, velocity covariances
c  rvcov(1,1,1) is the position, velocity covariance for the
c base state.  rvcov(1,1,2) is the initial position, velocity
c covariance for the relative state.
c
      DOUBLE PRECISION sncon( 2 )
c  constants in the state noise computation
c  a value of 2.5648e-7 m**2/sec**3 corresponds to an uncertainty
c of 400 m downrange in the base state per orbit.  a value of
c .6412e-9 corresponds to an uncertainty of 20 m downrange in the
c relative state per orbit.  To increase the downrange error by a
c factor of n, multiply the corresponding sncon by n**2.
c
      COMMON /xxscov/ iopt, rvcov, sncon
c
c end block xxscov
c
c
c
c end file mtf.scov
c
```

```
c
c start file mtf.sen
c
c
c
c
c ***************************************************************
c start common block xxsen
c ***************************************************************
c
      INTEGER maxobs
c  maximum number of obs
c
c
      INTEGER maxsen
c  maximum number of sensors
c
c
      INTEGER icoas
c  coas id
c
c
      INTEGER iradar
c  rendevous radar id
c
c
      INTEGER itrky
c  y star tracker id
c
c
      INTEGER itrkz
c  z star tracker id
c
c
      INTEGER isen1
c  1st extra sensor id
c
c
      INTEGER isen2
c  2nd extra sensor id
c
c
      INTEGER isen3
c  3rd extra sensor id
c
c
      DOUBLE PRECISION rsob( 3,7 )
c  sensor offsets
c
c
      DOUBLE PRECISION qbs( 4,7 )
c  sensor attitude quaternions m50 to sensor
c
c
     COMMON /xxsen/ maxobs, maxsen, icoas, iradar, itrky, itrkz, isen1
     * , isen2, isen3, rsob, qbs
```

```
c
c end block xxsen
c
c
c
c end file mtf.sen
c
```

```
c
c start file mtf.sprm
c
c
c
c
c ***************************************************************
c start common block xxsprm
c ***************************************************************
c
      CHARACTER*40 gmsg
c  generic message
c  generic message in dpf header
c
      CHARACTER*8 seq
c  file sequence id
c  eight character identifier sequence unique to each product file.
c contents  are 'fnsqsfrv' where
c   fn   flight number
c   sq   starting sequence number for first file on tape
c   sf   sequence flag where
c       00=sq   only one sequence on file
c       01=sq   start of first sequence
c       nn=sq   start of sequence nn
c   rv   revision number
c
      INTEGER spg( 20 )
c  display group flags
c  positive value specifies parameter groups to display
c options are
c       1       orbiter gmt
c       2       mcc gmt
c       3       ground elased time
c       4       orbiter m50 state
c       5       orbiter in target lvlh
c       6       orbiter in target uvw
c       7       orbiter euler angles to uvw
c       8       orbiter m50 attitude matrix
c       9       orbiter m50 quaternion
c       10      orbiter attitude rate
c       11      look angle and rates to target
c       12      range and range rate
c       13      simulation flag (used for data drop out info instead)
c               1 = data in tracking intervals (default)
c               0 = data not in tracking intervals
c       14      target m50 state
c       15      target m50 relative state
c       16      target in orbiter uvw
c       17      target in orbiter lvlh
c
      INTEGER szid
c  header length
c  index of record size in header
c
      INTEGER hlng
c  length of header
```

```
c  number of integer words in header
c
      REAL dict( 3 )
c  dictionary record
c  dictionary record format
c
      REAL datbuf( 2 )
c  data buffer
c  data buffer used for creating Univac tape
c
      COMMON /xxsprm/ gmsg, seq, spg, szid, hlng, dict, datbuf
c
c end block xxsprm
c
c
c
c end file mtf.sprm
c
```

```
c
c start file mtf.sptm
c
c
c
c
c ****************************************************************
c start common block xxsptm
c ****************************************************************
c
      DOUBLE PRECISION pfreq
c  print frequency
c  nominal print frequency
c
      DOUBLE PRECISION sptol
c  tolerance
c   tolerance within which special time is printed
c
      DOUBLE PRECISION sptime( 20 )
c  special times
c  table of special print times as seconds since base time
c
      COMMON /xxsptm/ pfreq, sptol, sptime
c
c end block xxsptm
c
c
c
c end file mtf.sptm
c
```

```
c
c start file mtf.sun
c
c
c
c
c ***************************************************************
c start common block xxsun
c ***************************************************************
c
      DOUBLE PRECISION musun
c  sun's mu
c  sun gravitational parameter
c
      DOUBLE PRECISION dtsun
c  sun eph time step
c  specifies time between consecutive points of sun ephemeris
c
      DOUBLE PRECISION tsun0
c  sun base time
c  specifies time tag of rvsun0 as time since base time
c
      DOUBLE PRECISION rvsun0( 6 )
c  init sun state
c  m50 state of earth relative to sun used to generate
c  the sun ehemeris
c
      INTEGER nsord
c  sun intrp order
c  specifies number of points used in interpolation sun state
c  from ephemeris
c
      DOUBLE PRECISION ksun
c  1 au solar force
c  solar force on sphere  at a distance of 1 au
c
      DOUBLE PRECISION kear
c  earth reflect
c  earth reflection constant
c
      DOUBLE PRECISION kflux
c  flux
c  solar flux
c
      DOUBLE PRECISION kealb
c  albedo
c  earth albedo
c
      COMMON /xxsun/ musun, dtsun, tsun0, rvsun0, nsord, ksun, kear,
     * kflux, kealb
c
c end block xxsun
c
c
c
c end file mtf.sun
```

```
   57    c
```

```
c
c start file mtf.svbi
c
c
c
c
c ***************************************************************
c start common block xxsvbi
c ***************************************************************
c
      INTEGER psvb( 2 )
c  bias flag for file 1
c
c
      DOUBLE PRECISION svbtb( 8,10,2 )
c  bias table for file 1
c
c
     .COMMON /xxsvbi/ psvb, svbtb
c
c end block xxsvbi
c
c
c
c end file mtf.svbi
c
```

```
c
c start file mtf.timc
c
c
c
c
c ***************************************************************
c start common block xxtimc
c ***************************************************************
c
      INTEGER cmptim( 200 )
c  indicates whether the time associated with the  msid corresponding
c    with this array position is to be computed from the dow
c
c
      INTEGER skewid( 200 )
c   indicates the data record location of the proper skew for this msid
c
c
      REAL timdat( 200,2 )
c   time data from cct
c
c
      REAL rbfdat( 16 )
c   time data from cct
c
c
      INTEGER cycnum
c   the indicator of header record present on cct header records
c
c
      COMMON /xxtimc/ cmptim, skewid, timdat, rbfdat, cycnum
c
c end block xxtimc
c
c
c
c end file mtf.timc
c
```

```
c
c start file mtf.time
c
c
c
c
c ************************************************************
c start common block xxtime
c ************************************************************
c
      INTEGER date( 5 )
c  base date
c  base date year,month,day,hour,minute,second
c
      DOUBLE PRECISION dates
c  base sec
c  seconds in base date
c
      DOUBLE PRECISION tbegin
c  beg time
c  begin time as seconds since base time
c
      DOUBLE PRECISION tend
c  end time
c  end time as seconds since base time
c
      DOUBLE PRECISION delta
c  time step
c  time step
c
      INTEGER endopt
c  term opt
c  termination option
c
      DOUBLE PRECISION endval
c  term val
c  value for termination
c
      DOUBLE PRECISION tbase
c  base time
c
c
     COMMON /xxtime/ date, dates, tbegin, tend, delta, endopt, endval,
     * tbase
c
c end block xxtime
c
c
c
c end file mtf.time
c
```

```
c
c start file mtf.toff
c
c
c
c
c ************************************************************
c start common block xxtoff
c ************************************************************
c
      INTEGER ldate( 6 )
c  launch date
c  launch date as ymdhms
c
      INTEGER dtdate( 6 )
c  time bias date
c  date of mcc/shuttle time bias as ymdhms.  dtbias specifies bias
c  and rate
c
      DOUBLE PRECISION dtbias( 2 )
c  time bias and rate
c  time bias and drift rate for mcc/shuttle time bias.  the offset
c  is given by
c   gt = st + dtbias(1) + dtbias(2)*(st-tb)
c where
c   gt = ground time
c   st = shuttle time
c   tb=time tag of bias as
c  specified by dtdate (offset from base date)
c
      INTEGER qafreq
c  qa rec print freq
c  specifies frequency of print to print unit in terms of records.
c  this is not the same as the microfiche print which is at the same
c  frequency as that specified by delta for the tape.
c
      DOUBLE PRECISION tlapse
c  summary frequency (min)
c  specifies the number of minutes between status messages
c  that are displayed to the terminal. this value is converted to
c  seconds internally
c
      DOUBLE PRECISION bjds
c  shuttle base julian date
c
c
      DOUBLE PRECISION bjdg
c  ground base julian date
c
c
      DOUBLE PRECISION dtsg
c  s/g time factor
c
c
      DOUBLE PRECISION tlift
c  time of lift off
```

```
c
c
      INTEGER drive
c  tape drive
c
c
      INTEGER tape
c  tape option
c  if tape > 0 univac tape is produced
c
      INTEGER hpbin
c  hp binary file
c  if hpbin > 0 then hp binary file is produced
c
      DOUBLE PRECISION dtable( 50,2 )
c   data dropout table
c   data dropout time intervals table
c
      COMMON /xxtoff/ ldate, dtdate, dtbias, qafreq, tlapse, bjds, bjdg
     * , dtsg, tlift, drive, tape, hpbin, dtable
c
c end block xxtoff
c
c
c
c end file mtf.toff
c
```

```
c
c start file mtf.tols
c
c
c
c
c ************************************************************
c start common block xxtols
c ************************************************************
c
      DOUBLE PRECISION ctol
c  convergence tolerance
c
c
      DOUBLE PRECISION ctol1
c  convergence tolerance
c
c
      DOUBLE PRECISION invtol
c  matrix inversion tolerance
c
c
      DOUBLE PRECISION tol
c  error tolerance for sun
c
c
     COMMON /xxtols/ ctol, ctol1, invtol, tol
c
c end block xxtols
c
c
c
c end file mtf.tols
c
```

```
c
c start file mtf.usys
c
c
c
c
c ************************************************************
c start common block xxusys
c ************************************************************
c
      INTEGER usysex( 5 )
c  display scale flags
c  positive value indicates that print is to be scaled by
c corresponding system of units
c
      INTEGER nsysex
c  no of output unit systems
c
c
      INTEGER usysin
c  input unit system id
c  id of unit system to scale input with
c
      CHARACTER*4 usyst( 5 )
c  unit system names
c  user specified abbreviation for corresponding unit system
c
      CHARACTER*4 nsclz( 10,5 )
c  scale factor names
c  user specified names for the scale factors in each system
c of units.
c
      DOUBLE PRECISION usclz( 10,5 )
c  scale factors
c  scale factors as internal units per specified unit
c details are
c       1       length (meter default)
c       3       angle (radian default)
c       4       time  (sec default)
c
     COMMON /xxusys/ usysex, nsysex, usysin, usyst, nsclz, usclz
c
c end block xxusys
c
c
c
c end file mtf.usys
c
```

```
c
c start file mtf.vcx
c
c
c
c
c ************************************************************
c start common block xxvcx
c ************************************************************
c
      DOUBLE PRECISION aeroc( 2 )
c  drag numbers
c
c
      DOUBLE PRECISION cd( 2 )
c  drag coefficient
c
c
      DOUBLE PRECISION chord( 2 )
c  vehicle chord
c
c
      DOUBLE PRECISION c( 3,2 )
c  cylindrical fit parameters
c
c
      DOUBLE PRECISION d( 5,2 )
c  zutek fit parameters
c
c
      DOUBLE PRECISION kd( 2 )
c  drag multiplier
c
c
      DOUBLE PRECISION drgfac( 2 )
c  aerodynamic drag factor
c
c
      DOUBLE PRECISION rbarna( 3,2 )
c  actual cg pos rel to nominal cg
c
c
      DOUBLE PRECISION sref( 2 )
c  vehicle reference area
c
c
      DOUBLE PRECISION sarea( 2 )
c  area for solar rad
c
c
      DOUBLE PRECISION srflec( 2 )
c  vehicle reflectivity
c
c
      DOUBLE PRECISION solfac( 2 )
c  solar radiation factor
```

```
c
c
      INTEGER horder( 2 )
c  max order of harmonics
c
c
      INTEGER hdgree( 2 )
c  maximum degree of harmonics
c
c
      DOUBLE PRECISION sagate( 2 )
c  accel gate
c  threshold for sensed accelerations to be included in
c propagation.  if magnitude of the acceleration does not exceed
c this threshold, then the acceleration is omitted.
c
      COMMON /xxvcx/ aeroc, cd, chord, c, d, kd, drgfac, rbarna, sref,
     * sarea, srflec, solfac, horder, hdgree, sagate
c
c end block xxvcx
c
c
c
c end file mtf.vcx
c
```

```
c
c start file mtf.vnt
c
c
c
c
c ***************************************************************
c start common block xxvnt
c ***************************************************************
c
      DOUBLE PRECISION vnttab( 4,10,2 )
c  vent timeline
c  two timeline entries based on last index. max of 10 entries
c per timeline.  Each entry consists of the following
c   1   start time of vent
c   2   body x force
c   3   body y force
c   4   body z force
c
      COMMON /xxvnt/ vnttab
c
c end block xxvnt
c
c
c
c end file mtf.vnt
c
```

```
c
c start file mtf.writ
c
c
c
c
c ************************************************************
c start common block xxwrit
c ************************************************************
c
      CHARACTER*12 files( 10 )
c  available Datain files to write to
c
      INTEGER recpnt( 2,10,10 )
c  pointers to record numbers in a file
c
      INTEGER nfiles
c  number of available Datain files
c
      CHARACTER*12 vars( 10,10 )
c  names of variables in the files
c
      INTEGER nvars( 10 )
c  number of variables in each file
c
      COMMON /xxwrit/ files, recpnt, nfiles, vars, nvars
c
c end block xxwrit
c
c
c
c end file mtf.writ
c
```

```
c
c start file mtf.atm
c
c
c
c
c ************************************************************
c start common data block xxatm
c ************************************************************
c
      data a62sc / .2590478000d-03      /
      data a62sh / .1206650000d+06       /
      data a62rr / .1294000000d-07      /
      data a62rh / .1225053499d+01      /
      data a76a / .2046000000d-01      /
      data a76b / .8402100000d+00       /
      data a76f10 / .1010000000d+03       /
      data a76rh1 / .5299400000d-10       /
      data a76a1 / .5299400000d-10       /
      data a76a2 / .7949100000d+05       /
      data a76a3 / .8332520000d+06       /
      data abmclg / .7986355100d+00       /
      data abmslg / .6018150230d+00       /
      data abmgde / .1375000000d+01       /
      data abmalt / .2222395555d+06       /
      data abm1 / -.2515818000d+02,-.1166693668d-04 ,.9922058998d+04  ,
     *   -.2321397100d+02 , -.1625013451d-04 , .7868452230d+06   /
      data abm0 / -.1986449600d+00  ,.4152999344d-05 , -.3926329360d+05/
      data abmcda / .8999972860d+05       /
      data abmcb1 / .4604986877d-04       /
      data abmcb2 / -.4500000000d-04       /
      data abmcm2 / -.9896459900d+00       /
      data abmrrf / .1225004738d+01       /
c
c
c end block xxatm
c
c
c
c end file mtf.atm
c
```

```
    1   c
    2   c start file mtf.bias
    3   c
    4   c
    5   c
    6   c
    7   c ************************************************************
    8   c start common data block xxbias
    9   c ************************************************************
   10   c
   11         data sgbias / 50.d0,.01745d0,.01745d0,.1d0, .0002d0,.0002d0,
   12        * .0002d0,.0002d0, .00074d0,.00074d0   /
   13         data timcon / 400.d0,10000.d0,10000.d0,400.d0, 10000.d0,10000.d0,
   14        * 10000.d0,10000.d0, 400.d0,400.d0  /
   15   c
   16   c
   17   c end block xxbias
   18   c
   19   c
   20   c
   21   c end file mtf.bias
   22   c
```

```
c
c start file mtf.con
c
c
c
c
c ***************************************************************
c start common data block xxcon
c ***************************************************************
c
      data clight / 0.2997925d9  /
      data pi / .31415926535897932d1  /
c
c
c end block xxcon
c
c
c
c end file mtf.con
c
```

```
c
c start file mtf.dprm
c
c
c
c
c ****************************************************************
c start common data block xxdprm
c ****************************************************************
c
c
c
c end block xxdprm
c
c
c
c end file mtf.dprm
c
```

```
c
c start file mtf.erth
c
c
c
c
c ************************************************************
c start common data block xxerth
c ************************************************************
c
      data muerth / .3986012d15   /
      data req / .6378166d7    /
      data rpol / .635678428360710 7d7   /
      data wei / .729211514645921d-4    /
c
c
c end block xxerth
c
c
c
c end file mtf.erth
c
```

```
    1   c
    2   c start file mtf.file
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxfile
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxfile
   14   c
   15   c
   16   c
   17   c end file mtf.file
   18   c
```

```
c
c start file mtf.gnrl
c
c
c
c
c ****************************************************************
c start common data block xxgnrl
c ****************************************************************
c
      data bugs / 6  /
      data in / 69  /
      data term / 6  /
      data out / 6  /
      data pbug / 19*0  /
      data pfatal / 0  /
      data interm / 5  /
c
c
c end block xxgnrl
c
c
c
c end file mtf.gnrl
c
```

```
c
c start file mtf.graf
c
c
c
c
c ************************************************************
c start common data block xxgraf                             .
c ************************************************************
c
      data option / 1  /
      data ttlmul / 2.0 , 1.5 , 0.0 , 0.0  /
      data xygrid / 2*1,0  /
      data xypage / 11.0 , 8.5  /
      data porgin / 2*-1.0  /
      data xyarea / 8.5 , 6.0  /
      data xyhite / 1.2  /
      data xyang / 45.0 , 2*0.0  /
      data plegnd / 1.0  /
      data lgnpos / 8.47 , 3.5  /
c
c end file mtf.graf
c
```

```
c
c start file mtf.grav
c
c
c
c
c ***************************************************************
c start common data block xxgrav
c ***************************************************************
c
      data rgrav / 6378160.0d0   /
      data cterms / 0.d0, .155752d-05, .212763d-05, .304690d-06,
     * .957d-07,-.502698d-06, .738439d-07, .591298d-07, -.16838d-08,
     *-.460853d-07, .99182d-07, -.142322d-07, -.207839d-08,
     *.310069d-09, -.778802d-07, .682041d-08, .577916d-09,
     *-.152714d-11, -.170638d-09, .206637d-10, .17670d-06,
     * .281935d-07, .285733d-08, -.418909d-09, -.307997d-12,
     *-.179403d-10, .29525d-11, .214773d-07, .395567d-08,
     *-.58076d-09, -.200713d-09, -.102636d-10, -.150544d-11,
     *.175356d-12, -.986208d-13   /
      data sterms / 0.d0, -.880523d-06, .280994d-06, -.216784d-06,
     *.19946d-06,     -.462625d-06, .157940d-06, -.92433d-08,
     * .71686d-08,-.838408d-07, -.567829d-07, -.286286d-08, .646339d-09,
     *-.147513d-08, .296244d-07, -.437589d-07, .925271d-09,
     *-.152888d-08, -.421805d-09, -.174166d-10, .846618d-07,
     *.155828d-07, -.330286d-08, -.24187d-09, .348475d-10,
     * .708604d-11, -.125606d-11, .176579d-07, .691077d-08,
     *.18285d-09, .982345d-10, .11156d-10, .72419d-11,
     * .454672d-12, .861829d-13   /
      data jterms / 1.082637d-03, -2.539d-06, -1.617d-06, -2.34d-07,
     * 5.55d-07,     -3.48d-07, -2.09d-07  /
c
c
c end block xxgrav
c
c
c
c end file mtf.grav
c
```

```
    1   c
    2   c start file mtf.init
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxinit
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxinit
   14   c
   15   c
   16   c
   17   c end file mtf.init
   18   c
```

```
c
c start file mtf.kal
c
c
c
c
c ***************************************************************
c start common data block xxkal
c ***************************************************************
c
      data dtmax / 5.0d0  /
      data var / 30.d0,    .0027d0, .0027d0, .1d0, .5d-3,.5d-3,
     * .5d-4,.5d-4, .45d-3, .45d-3, 10*0.d0   /
      data ncons / 2400.d0,9000.d0,36000.d0,8.0d0, 15000.d0,93750.d0,
     *300000.d0,8.0d0, .1d0,.01d0  /
c
c
c end block xxkal
c
c
c
c end file mtf.kal
c
```

```
    1   c
    2   c start file mtf.mas
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxmas
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxmas
   14   c
   15   c
   16   c
   17   c end file mtf.mas
   18   c
```

```
c
c start file mtf.max
c
c
c
c
c ***************************************************************
c start common data block xxmax
c ***************************************************************
c
      data maxveh / 2  /
      data nveh / 1  /
c
c
c end block xxmax
c
c
c
c end file mtf.max
c
```

```
c
c start file mtf.misc
c
c
c
c
c ************************************************************
c start common data block xxmisc
c ************************************************************
c
      data jobdes / 2*'no job description given' /
      data obsrvr / 1  /
      data target / 2  /
c
c
c end block xxmisc
c
c
c
c end file mtf.misc
c
```

```
c
c start file mtf.moon
c
c
c
c
c ***************************************************************
c start common data block xxmoon
c ***************************************************************
c
c
c
c end block xxmoon
c
c
c
c end file mtf.moon
c
```

```
    1   c
    2   c start file mtf.name
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxname
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxname
   14   c
   15   c
   16   c
   17   c end file mtf.name
   18   c
```

```
c
c start file mtf.nflz
c
c
c
c
c ************************************************************
c start common data block xxnflz
c ************************************************************
c
      data funit / 21 , 22 , 23 , 24 , 25  /
c
c
c end block xxnflz
c
c
c
c end file mtf.nflz
c
```

```
    1   c
    2   c start file mtf.obs
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxobs
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxobs
   14   c
   15   c
   16   c
   17   c end file mtf.obs
   18   c
```

```
c
c start file mtf.prnt
c
c
c
c
c ************************************************************
c start common data block xxprnt
c ************************************************************
c
      data lpage / 0  /
      data col80 / 0  /
      data header / 0  /
      data nfrmat / 1  /
c
c
c end block xxprnt
c
c
c
c end file mtf.prnt
c
```

```
    1   c
    2   c start file mtf.prop
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxprop
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxprop
   14   c
   15   c
   16   c
   17   c end file mtf.prop
   18   c
```

```
c
c start file mtf.qcrv
c
c
c
c
c ************************************************************
c start common data block xxqcrv
c ************************************************************
c
      data kspan / 1.,2.d8, 1.,2.d8, 1.,2.d8, 1.,2.d8, 1.,2.d8,
     * 1.,2.d8, 1.,2.d8, 1.,2.d8,1.,2.d8, 1.,2.d8, 1.,2.d8, 1.,2.d8,
     * 1.,2.d8, 1.,2.d8, 1.,2.d8,1.,2.d8, 1.,2.d8, 1.,2.d8, 1.,2.d8,
     * 1.,2.d8  /
      data psize / 20*1.  /
c
c
c end block xxqcrv
c
c
c
c end file mtf.qcrv
c
```

```
    1   c
    2   c start file mtf.qgen
    3   c
    4   c
    5   c
    6   c
    7   c ****************************************************************
    8   c start common data block xxqgen
    9   c ****************************************************************
   10   c
   11         data qpdev / 'hp' /
   12         data sthite / 0.14  /
   13   c
   14   c
   15   c end block xxqgen
   16   c
   17   c
   18   c
   19   c end file mtf.qgen
   20   c
```

```
c
c start file mtf.qprm
c
c
c
c
c ************************************************************
c start common data block xxqprm
c ************************************************************
c
c
c end block xxqprm
c
c
c
c end file mtf.qprm
c
```

```
      c
      c start file mtf.rnpx
      c
      c
      c
      c ************************************************************
      c start common data block xxrnpx
      c ************************************************************
      c
            data rnp0 / 1.2d0 , 8*0.d0  /
            data cdetut / 0.d0  /
      c
      c
      c end block xxrnpx
      c
      c
      c
      c end file mtf.rnpx
      c
```

```
c
c start file mtf.rpst
c
c
c
c
c ************************************************************
c start common data block xxrpst
c ************************************************************
c
      data antang / 1.169370564  /
c
c
c end block xxrpst
c
c
c
c end file mtf.rpst
c
```

```
c
c start file mtf.scov
c
c
c
c
c ***************************************************************
c start common data block xxscov
c ***************************************************************
c
      data iopt / 0  /
      data rvcov / 140.d0
     * ,3*0.d0, -.94d0 ,2*0.d0, 1060.d0
     *    ,0.d0  , -.99d0 ,4*0.d0, 70.d0
     * ,4*0.d0, -.99d0 ,0.d0  , 1.22d0 ,2*0.d0, -.94d0 ,3*0.d0, .13d0
     * ,6*0.d0, .08d0, 140.d0,3*0.d0, -.94d0 ,
     *2*0.d0, 1060.d0       ,0.d0  , -.99d0 ,4*0.d0, 70.d0
     *    ,4*0.d0, -.99d0 ,0.d0  , 1.22d0        ,2*0.d0, -.94d0 ,3*0.d0,
     * .13d0         ,6*0.d0, .08d0         /
      data sncon / 1.603d-6, .6412d-6   /
c
c
c end block xxscov
c
c
c
c end file mtf.scov
c
```

```
c
c start file mtf.sen
c
c
c
c
c ***************************************************************
c start common data block xxsen
c ***************************************************************
c
      data maxobs / 16  /
      data maxsen / 4   /
      data icoas / 4  /
      data iradar / 1   /
      data itrky / 3  /
      data itrkz / 2   /
      data rsob / .139445122d2,.33933d1,-.176524d1,
     *.173104302d2,-.28194d0 , -.124206d1 , -.1649d2 ,-.525d1,.29d1,
     *.1386078d2,-.1408d1 ,-.322326d1, 3*0.0d0  /
      data qbs / .8338858225201d0 ,0.d0 , 0.d0,.5519369829422d0,
     * .0264059344345d0,.7065011118937d0 ,.7072182848987d0,
     *.00109582820l0d0, .0640725060379d0,-.0653106301822d0,
     *.7050333478320d0,-.7032476190239d0, .0d0 , .0d0 , .1d1 , .0d0 ,
     * 1.0d0, 3*0.0d0  /
c
c
c end block xxsen
c
c
c
c end file mtf.sen
c
```

```
    1   c
    2   c start file mtf.sprm
    3   c
    4   c
    5   c
    6   c
    7   c ***********************************************************
    8   c start common data block xxsprm
    9   c ***********************************************************
   10   c
   11   c
   12   c
   13   c end block xxsprm
   14   c
   15   c
   16   c
   17   c end file mtf.sprm
   18   c
```

```
c
c start file mtf.sptm
c
c
c
c
c ***********************************************************
c start common data block xxsptm
c ***********************************************************
c
      data pfreq / 4.0d0  /
      data sptime / 20*1.0d38  /
c
c
c end block xxsptm
c
c
c
c end file mtf.sptm
c
```

```
    1   c
    2   c start file mtf.sun
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxsun
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxsun
   14   c
   15   c
   16   c
   17   c end file mtf.sun
   18   c
```

```
c
c start file mtf.svbi
c
c
c
c ***********************************************************
c start common data block xxsvbi
c ***********************************************************
c
      data svbtb / 160*1.d20  /
c
c
c end block xxsvbi
c
c
c
c end file mtf.svbi
c
```

```
    1   c
    2   c start file mtf.timc
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxtimc
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxtimc
   14   c
   15   c
   16   c
   17   c end file mtf.timc
   18   c
```

```
c
c start file mtf.time
c
c
c
c
c ************************************************************
c start common data block xxtime
c ************************************************************
c
      data date / 1985 , 1 , 0 , 0 , 0  /
      data dates / 0.0  /
      data tbegin / -1.d30  /
      data tend / 1.d30  /
      data delta / 1  /
      data endopt / 0  /
      data endval / 1.d30  /
c
c
c end block xxtime
c
c
c
c end file mtf.time
c
```

```
    1   c
    2   c start file mtf.toff
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxtoff
    9   c ***************************************************************
   10   c
   11   c
   12   c
   13   c end block xxtoff
   14   c
   15   c
   16   c
   17   c end file mtf.toff
   18   c
```

```
c
c start file mtf.tols
c
c
c
c
c ***********************************************************
c start common data block xxtols
c ***********************************************************
c
c
c
c end block xxtols
c
c
c
c end file mtf.tols
c
```

```
c
c start file mtf.usys
c
c
c
c
c ************************************************************
c start common data block xxusys
c ************************************************************
c
      data usysex / 1,4*0  /
      data usysin / 0  /
      data usyst / 5*' ' /
      data nsclz / 50*' ' /
      data usclz / 50*1.d0  /
c
c
c end block xxusys
c
c
c
c end file mtf.usys
c
```

```
c
c start file mtf.vcx
c
c
c
c
c ************************************************************
c start common data block xxvcx
c ************************************************************
c
      data cd / 2.2 , 2.2  /
      data sref / 250.0 , 2.926  /
      data horder / 4 , 4  /
      data hdgree / 4 , 4  /
      data sagate / 0.00001369 , 0.0  /
c
c
c end block xxvcx
c
c
c
c end file mtf.vcx
c
```

```
    1   c
    2   c start file mtf.vnt
    3   c
    4   c
    5   c
    6   c
    7   c ***************************************************************
    8   c start common data block xxvnt
    9   c ***************************************************************
   10   c
   11         data vnttab / 80 * 1.0d20  /
   12   c
   13   c
   14   c end block xxvnt
   15   c
   16   c
   17   c
   18   c end file mtf.vnt
   19   c
```

```
c
c start file mtf.writ
c
c
c
c
c ************************************************************
c start common data block xxwrit
c ************************************************************
c
      data files / 'init' , 'time' , 8*' ' /
      data recpnt / 1 , 1 , 2 , 6 , 8 , 1 , 9 , 6 , 12*0, 7 , 1 , 8 ,
     * 1 , 11 , 1 , 174*0  /
      data nfiles / 2  /
      data vars / 'veh1 time' ,'veh1 state','veh2 time' ,'veh2 state',
     *  5*' ','tbegin' ,  'tend' ,'endval' , 88 * ' ' /
      data nvars / 4,3,8*0  /
c
c
c end block xxwrit
c
c
c
c end file mtf.writ
c
```

```
c
c start file mtf.atm
c
c
c start block xxatm
c
c
      call setin( a62sc, 'a62sc'//char(0), 'd*1'//char(0) )
      call setin( a62sh, 'a62sh'//char(0), 'd*1'//char(0) )
      call setin( a62rr, 'a62rr'//char(0), 'd*1'//char(0) )
      call setin( a62rh, 'a62rh'//char(0), 'd*1'//char(0) )
      call setin( a76a, 'a76a'//char(0), 'd*1'//char(0) )
      call setin( a76b, 'a76b'//char(0), 'd*1'//char(0) )
      call setin( a76f10, 'a76f10'//char(0), 'd*1'//char(0) )
      call setin( a76rh1, 'a76rh1'//char(0), 'd*1'//char(0) )
      call setin( a76a1, 'a76a1'//char(0), 'd*1'//char(0) )
      call setin( a76a2, 'a76a2'//char(0), 'd*1'//char(0) )
      call setin( a76a3, 'a76a3'//char(0), 'd*1'//char(0) )
      call setin( abmclg, 'abmclg'//char(0), 'd*1'//char(0) )
      call setin( abmslg, 'abmslg'//char(0), 'd*1'//char(0) )
      call setin( abmgde, 'abmgde'//char(0), 'd*1'//char(0) )
      call setin( abmalt, 'abmalt'//char(0), 'd*1'//char(0) )
      call setin( abm1, 'abm1'//char(0), 'd*6'//char(0) )
      call setin( abm0, 'abm0'//char(0), 'd*3'//char(0) )
      call setin( abmcda, 'abmcda'//char(0), 'd*1'//char(0) )
      call setin( abmcb1, 'abmcb1'//char(0), 'd*1'//char(0) )
      call setin( abmcb2, 'abmcb2'//char(0), 'd*1'//char(0) )
      call setin( abmcm2, 'abmcm2'//char(0), 'd*1'//char(0) )
      call setin( abmrrf, 'abmrrf'//char(0), 'd*1'//char(0) )
c
c end block xxatm
c
c
c
c end file mtf.atm
c
```

```
c
c start file mtf.bias
c
c
c start block xxbias
c
c
      call setin( bias, 'bias'//char(0), 'd*10'//char(0) )
      call setin( sgbias, 'sgbias'//char(0), 'd*10'//char(0) )
      call setin( timcon, 'timcon'//char(0), 'd*10'//char(0) )
c
c end block xxbias
c
c
c
c end file mtf.bias
c
```

```
    1   c
    2   c start file mtf.cct
    3   c
    4   c
    5   c start block xxcct
    6   c
    7   c
    8         call setin( cct, 'cct'//char(0), 'i*1'//char(0) )
    9         call setin( lcynum, 'lcynum'//char(0), 'i*1'//char(0) )
   10         call setin( o4000, 'o4000'//char(0), 'i*1'//char(0) )
   11   c
   12   c end block xxcct
   13   c
   14   c
   15   c
   16   c end file mtf.cct
   17   c
```

```
1   c
2   c start file mtf.con
3   c
4   c
5   c start block xxcon
6   c
7   c
8         call setin( clight, 'clight'//char(0), 'd*1'//char(0) )
9         call setin( pi, 'pi'//char(0), 'd*1'//char(0) )
10  c
11  c end block xxcon
12  c
13  c
14  c
15  c end file mtf.con
16  c
```

```
    1   c
    2   c start file mtf.data
    3   c
    4   c
    5   c start block xxdata
    6   c
    7   c
    8         call setin( dtxcld, 'dtxcld'//char(0), 'i*25'//char(0) )
    9         call setin( gwbtab, 'gwbtab'//char(0), 'd*50'//char(0) )
   10   c
   11   c end block xxdata
   12   c
   13   c
   14   c
   15   c end file mtf.data
   16   c
```

```
c
c start file mtf.dot
c
c
c start block xxdot
c
c
c
c end block xxdot
c
c
c
c end file mtf.dot
c
```

```
     1   c
     2   c start file mtf.dprm
     3   c
     4   c
     5   c start block xxdprm
     6   c
     7   c
     8         call setin( spg, 'spg'//char(0), 'i*20'//char(0) )
     9   c
    10   c end block xxdprm
    11   c
    12   c
    13   c
    14   c end file mtf.dprm
    15   c
```

```
c
c start file mtf.erth
c
c
c start block xxerth
c
c
      call setin( muerth, 'muerth'//char(0), 'd*1'//char(0) )
      call setin( req, 'req'//char(0), 'd*1'//char(0) )
      call setin( rpol, 'rpol'//char(0), 'd*1'//char(0) )
      call setin( wei, 'wei'//char(0), 'd*1'//char(0) )
c
c end block xxerth
c
c
c
c end file mtf.erth
c
```

```
c
c start file mtf.file
c
c
c start block xxfile
c
c
      call setin( nama1, 'nama1'//char(0), 's72*1'//char(0) )
      call setin( unta1, 'unta1'//char(0), 'i*1'//char(0) )
      call setin( uzea1, 'uzea1'//char(0), 'i*1'//char(0) )
      call setin( nama2, 'nama2'//char(0), 's72*1'//char(0) )
      call setin( unta2, 'unta2'//char(0), 'i*1'//char(0) )
      call setin( uzea2, 'uzea2'//char(0), 'i*1'//char(0) )
      call setin( name1, 'name1'//char(0), 's72*1'//char(0) )
      call setin( unte1, 'unte1'//char(0), 'i*1'//char(0) )
      call setin( uzee1, 'uzee1'//char(0), 'i*1'//char(0) )
      call setin( name2, 'name2'//char(0), 's72*1'//char(0) )
      call setin( unte2, 'unte2'//char(0), 'i*1'//char(0) )
      call setin( uzee2, 'uzee2'//char(0), 'i*1'//char(0) )
      call setin( namd1, 'namd1'//char(0), 's72*1'//char(0) )
      call setin( untd1, 'untd1'//char(0), 'i*1'//char(0) )
      call setin( uzed1, 'uzed1'//char(0), 'i*1'//char(0) )
      call setin( namd2, 'namd2'//char(0), 's72*1'//char(0) )
      call setin( untd2, 'untd2'//char(0), 'i*1'//char(0) )
      call setin( uzed2, 'uzed2'//char(0), 'i*1'//char(0) )
      call setin( namv1, 'namv1'//char(0), 's72*1'//char(0) )
      call setin( untv1, 'untv1'//char(0), 'i*1'//char(0) )
      call setin( uzev1, 'uzev1'//char(0), 'i*1'//char(0) )
      call setin( namv2, 'namv2'//char(0), 's72*1'//char(0) )
      call setin( untv2, 'untv2'//char(0), 'i*1'//char(0) )
      call setin( uzev2, 'uzev2'//char(0), 'i*1'//char(0) )
      call setin( namr, 'namr'//char(0), 's72*1'//char(0) )
      call setin( untr, 'untr'//char(0), 'i*1'//char(0) )
      call setin( uzer, 'uzer'//char(0), 'i*1'//char(0) )
      call setin( nams, 'nams'//char(0), 's72*1'//char(0) )
      call setin( unts, 'unts'//char(0), 'i*1'//char(0) )
      call setin( uzes, 'uzes'//char(0), 'i*1'//char(0) )
      call setin( filids, 'filids'//char(0), 'i*6'//char(0) )
c
c end block xxfile
c
c
c
c end file mtf.file
c
```

```
1   c
2   c start file mtf.fils
3   c
4   c
5   c start block xxfils
6   c
7   c
8         call setin( filsta, 'filsta'//char(0), 'i*20'//char(0) )
9         call setin( frmsze, 'frmsze'//char(0), 'i*60'//char(0) )
10        call setin( datseq, 'datseq'//char(0), 'i*4000'//char(0) )
11        call setin( dbproc, 'dbproc'//char(0), 'i*20'//char(0) )
12  c
13  c end block xxfils
14  c
15  c
16  c
17  c end file mtf.fils
18  c
```

```
c
c start file mtf.gnrl
c
c
c start block xxgnrl
c
c
      call setin( bugs, 'bugs'//char(0), 'i*1'//char(0) )
      call setin( in, 'in'//char(0), 'i*1'//char(0) )
      call setin( term, 'term'//char(0), 'i*1'//char(0) )
      call setin( out, 'out'//char(0), 'i*1'//char(0) )
      call setin( pbug, 'pbug'//char(0), 'i*19'//char(0) )
c
c end block xxgnrl
c
c
c
c end file mtf.gnrl
c
```

```
c
c start file mtf.graf
c
c
c start block xxgraf
c
c
      call setin( option, 'option'//char(0), 'i*1'//char(0) )
      call setin( xlabel, 'xlabel'//char(0), 's60*1'//char(0) )
      call setin( ylabel, 'ylabel'//char(0), 's60*1'//char(0) )
      call setin( zlabel, 'zlabel'//char(0), 's60*1'//char(0) )
      call setin( title, 'title'//char(0), 's60*1'//char(0) )
      call setin( title2, 'title2'//char(0), 's60*1'//char(0) )
      call setin( title3, 'title3'//char(0), 's60*1'//char(0) )
      call setin( title4, 'title4'//char(0), 's60*1'//char(0) )
      call setin( ttlmul, 'ttlmul'//char(0), 'r*4'//char(0) )
      call setin( curves, 'curves'//char(0), 's8*20'//char(0) )
      call setin( pframe, 'pframe'//char(0), 'i*1'//char(0) )
      call setin( xygrid, 'xygrid'//char(0), 'i*3'//char(0) )
      call setin( xyaxs, 'xyaxs'//char(0), 's4*2'//char(0) )
      call setin( ttlwd, 'ttlwd'//char(0), 's4*1'//char(0) )
      call setin( lblwd, 'lblwd'//char(0), 's4*1'//char(0) )
      call setin( xyzln, 'xyzln'//char(0), 's4*3'//char(0) )
      call setin( xypage, 'xypage'//char(0), 'r*2'//char(0) )
      call setin( porgin, 'porgin'//char(0), 'r*2'//char(0) )
      call setin( xyarea, 'xyarea'//char(0), 'r*2'//char(0) )
      call setin( frmthk, 'frmthk'//char(0), 'r*1'//char(0) )
      call setin( gmrgin, 'gmrgin'//char(0), 'r*1'//char(0) )
      call setin( xyhite, 'xyhite'//char(0), 'r*1'//char(0) )
      call setin( xyznch, 'xyznch'//char(0), 'r*3'//char(0) )
      call setin( xyzstp, 'xyzstp'//char(0), 'r*3'//char(0) )
      call setin( xmmx, 'xmmx'//char(0), 'r*2'//char(0) )
      call setin( ymmx, 'ymmx'//char(0), 'r*2'//char(0) )
      call setin( zmmx, 'zmmx'//char(0), 'r*2'//char(0) )
      call setin( xyang, 'xyang'//char(0), 'r*3'//char(0) )
      call setin( plegnd, 'plegnd'//char(0), 'r*1'//char(0) )
      call setin( lgnpos, 'lgnpos'//char(0), 'r*2'//char(0) )
c
c end block xxgraf
c
c
c
c end file mtf.graf
c
```

```
c
c start file mtf.grav
c
c
c start block xxgrav
c
c
      call setin( rgrav, 'rgrav'//char(0), 'd*1'//char(0) )
      call setin( cterms, 'cterms'//char(0), 'd*35'//char(0) )
      call setin( sterms, 'sterms'//char(0), 'd*35'//char(0) )
      call setin( jterms, 'jterms'//char(0), 'd*7'//char(0) )
c
c end block xxgrav
c
c
c
c end file mtf.grav
c
```

```
c
c start file mtf.init
c
c
c start block xxinit
c
c
      call setin( trvint, 'trvint'//char(0), 'd*14'//char(0) )
c
c end block xxinit
c
c
c
c end file mtf.init
c
```

```
    1   c
    2   c start file mtf.kal
    3   c
    4   c
    5   c start block xxkal
    6   c
    7   c
    8         call setin( dtmax, 'dtmax'//char(0), 'd*1'//char(0) )
    9         call setin( edcrit, 'edcrit'//char(0), 'd*25'//char(0) )
   10         call setin( var, 'var'//char(0), 'd*20'//char(0) )
   11         call setin( lrmodl, 'lrmodl'//char(0), 'i*1'//char(0) )
   12         call setin( ncons, 'ncons'//char(0), 'd*10'//char(0) )
   13         call setin( undrwt, 'undrwt'//char(0), 'd*2'//char(0) )
   14         call setin( sxcld, 'sxcld'//char(0), 'i*24'//char(0) )
   15  c
   16  c end block xxkal
   17  c
   18  c
   19  c
   20  c end file mtf.kal
   21  c
```

```
c
c start file mtf.mas
c
c
c start block xxmas
c
c
      call setin( mastab, 'mastab'//char(0), 'd*60'//char(0) )
c
c end block xxmas
c
c
c
c end file mtf.mas
c
```

```
c
c start file mtf.mast
c
c
c start block xxmast
c
c
      call setin( hifile, 'hifile'//char(0), 'i*1'//char(0) )
      call setin( hiunit, 'hiunit'//char(0), 'i*1'//char(0) )
      call setin( maxfil, 'maxfil'//char(0), 'i*1'//char(0) )
      call setin( maxtyp, 'maxtyp'//char(0), 'i*1'//char(0) )
      call setin( tcrflt, 'tcrflt'//char(0), 'i*1'//char(0) )
      call setin( timoff, 'timoff'//char(0), 'd*1'//char(0) )
c
c end block xxmast
c
c
c
c end file mtf.mast
c
```

```
c
c start file mtf.max
c
c
c start block xxmax
c
c
      call setin( nveh, 'nveh'//char(0), 'i*1'//char(0) )
c
c end block xxmax
c
c
c
c end file mtf.max
c
```

```
c
c start file mtf.misc
c
c
c start block xxmisc
c
c
      call setin( jobdes, 'jobdes'//char(0), 's60*2'//char(0) )
      call setin( obsrvr, 'obsrvr'//char(0), 'i*1'//char(0) )
      call setin( target, 'target'//char(0), 'i*1'//char(0) )
c
c end block xxmisc
c
c
c
c end file mtf.misc
c
```

```
c
c start file mtf.moon
c
c
c start block xxmoon
c
c
      call setin( mumoon, 'mumoon'//char(0), 'd*1'//char(0) )
      call setin( dtmoon, 'dtmoon'//char(0), 'd*1'//char(0) )
      call setin( tmoo0, 'tmoo0'//char(0), 'd*1'//char(0) )
      call setin( rvmoo0, 'rvmoo0'//char(0), 'd*6'//char(0) )
      call setin( nmord, 'nmord'//char(0), 'i*1'//char(0) )
c
c end block xxmoon
c
c
c
c end file mtf.moon
c
```

```
    1   c
    2   c start file mtf.msid
    3   c
    4   c
    5   c start block xxmsid
    6   c
    7   c
    8   c
    9   c end block xxmsid
   10   c
   11   c
   12   c
   13   c end file mtf.msid
   14   c
```

```
c
c start file mtf.name
c
c
c start block xxname
c
c
c
c end block xxname
c
c
c
c end file mtf.name
c
```

```
    1   c
    2   c start file mtf.nflz
    3   c
    4   c
    5   c start block xxnflz
    6   c
    7   c
    8         call setin( fname, 'fname'//char(0), 's72*5'//char(0) )
    9         call setin( bfopt, 'bfopt'//char(0), 'i*1'//char(0) )
   10         call setin( posx, 'posx'//char(0), 'i*2'//char(0) )
   11         call setin( lordx, 'lordx'//char(0), 'i*2'//char(0) )
   12         call setin( print, 'print'//char(0), 'i*1'//char(0) )
   13         call setin( plot, 'plot'//char(0), 'i*1'//char(0) )
   14         call setin( trjout, 'trjout'//char(0), 'i*3'//char(0) )
   15  c
   16  c end block xxnflz
   17  c
   18  c
   19  c
   20  c end file mtf.nflz
   21  c
```

```
c
c start file mtf.ntrp
c
c
c start block xxntrp
c
c
c
c end block xxntrp
c
c
c
c end file mtf.ntrp
c
```

```
    1   c
    2   c start file mtf.obs
    3   c
    4   c
    5   c start block xxobs
    6   c
    7   c
    8   c
    9   c end block xxobs
   10   c
   11   c
   12   c
   13   c end file mtf.obs
   14   c
```

```
c
c start file mtf.pkt
c
c
c start block xxpkt
c
c
c
c end block xxpkt
c
c
c
c end file mtf.pkt
c
```

```
    1   c
    2   c start file mtf.pmmx
    3   c
    4   c
    5   c start block xxpmmx
    6   c
    7   c
    8         call setin( pmmx, 'pmmx'//char(0), 'r*42'//char(0) )
    9   c
   10   c end block xxpmmx
   11   c
   12   c
   13   c
   14   c end file mtf.pmmx
   15   c
```

```
1   c
2   c start file mtf.prnt
3   c
4   c
5   c start block xxprnt
6   c
7   c
8         call setin( lpage, 'lpage'//char(0), 'i*1'//char(0) )
9         call setin( col80, 'col80'//char(0), 'i*1'//char(0) )
10        call setin( header, 'header'//char(0), 'i*1'//char(0) )
11        call setin( nfrmat, 'nfrmat'//char(0), 'i*1'//char(0) )
12  c
13  c end block xxprnt
14  c
15  c
16  c
17  c end file mtf.prnt
18  c
```

```
c
c start file mtf.prop
c
c
c start block xxprop
c
c
      call setin( rvopt, 'rvopt'//char(0), 'i*2'//char(0) )
      call setin( paero, 'paero'//char(0), 'i*2'//char(0) )
      call setin( pcb, 'pcb'//char(0), 'i*2'//char(0) )
      call setin( pdrag, 'pdrag'//char(0), 'i*2'//char(0) )
      call setin( pharm, 'pharm'//char(0), 'i*2'//char(0) )
      call setin( pmoon, 'pmoon'//char(0), 'i*2'//char(0) )
      call setin( prad, 'prad'//char(0), 'i*2'//char(0) )
      call setin( psvel, 'psvel'//char(0), 'i*2'//char(0) )
      call setin( psun, 'psun'//char(0), 'i*2'//char(0) )
      call setin( pvent, 'pvent'//char(0), 'i*2'//char(0) )
      call setin( dtnom, 'dtnom'//char(0), 'd*2'//char(0) )
c
c end block xxprop
c
c
c
c end file mtf.prop
c
```

```
c
c start file mtf.qcrv
c
c
c start block xxqcrv
c
c
      call setin( kname, 'kname'//char(0), 's8*20'//char(0) )
      call setin( kparms, 'kparms'//char(0), 's8*60'//char(0) )
      call setin( imrk, 'imrk'//char(0), 'i*20'//char(0) )
      call setin( nlabel, 'nlabel'//char(0), 'i*20'//char(0) )
      call setin( ksmbol, 'ksmbol'//char(0), 'i*20'//char(0) )
      call setin( kfile, 'kfile'//char(0), 's72*20'//char(0) )
      call setin( kedit, 'kedit'//char(0), 'i*20'//char(0) )
      call setin( kline, 'kline'//char(0), 'i*20'//char(0) )
      call setin( kstep, 'kstep'//char(0), 'r*20'//char(0) )
      call setin( kspan, 'kspan'//char(0), 'r*40'//char(0) )
      call setin( psize, 'psize'//char(0), 'r*20'//char(0) )
c
c end block xxqcrv
c
c
c
c end file mtf.qcrv
c
```

```
    1   c
    2   c start file mtf.qcur
    3   c
    4   c
    5   c start block xxqcur
    6   c
    7   c
    8   c
    9   c end block xxqcur
   10   c
   11   c
   12   c
   13   c end file mtf.qcur
   14   c
```

```
c
c start file mtf.qgen
c
c
c start block xxqgen
c
c
      call setin( qpdev, 'qpdev'//char(0), 's4*1'//char(0) )
      call setin( sthite, 'sthite'//char(0), 'r*1'//char(0) )
c
c end block xxqgen
c
c
c
c end file mtf.qgen
c
```

```
    1   c
    2   c start file mtf.qlbf
    3   c
    4   c
    5   c start block xxqlbf
    6   c
    7   c
    8   c
    9   c end block xxqlbf
   10   c
   11   c
   12   c
   13   c end file mtf.qlbf
   14   c
```

```
c
c start file mtf.qprm
c
c
c start block xxqprm
c
c
      call setin( pmxflg, 'pmxflg'//char(0), 'i*21'//char(0) )
      call setin( pword, 'pword'//char(0), 'i*21'//char(0) )
      call setin( pfid, 'pfid'//char(0), 's4*21'//char(0) )
      call setin( punits, 'punits'//char(0), 's12*21'//char(0) )
      call setin( pname, 'pname'//char(0), 's8*21'//char(0) )
      call setin( pscale, 'pscale'//char(0), 'r*21'//char(0) )
      call setin( pofset, 'pofset'//char(0), 'r*21'//char(0) )
c
c end block xxqprm
c
c
c
c end file mtf.qprm
c
```

```
c
c start file mtf.rnpx
c
c
c start block xxrnpx
c
c
      call setin( trnp0, 'trnp0'//char(0), 'd*1'//char(0) )
      call setin( rnp0, 'rnp0'//char(0), 'd*9'//char(0) )
      call setin( cdetut, 'cdetut'//char(0), 'd*1'//char(0) )
c
c end block xxrnpx
c
c
c
c end file mtf.rnpx
c
```

```
c
c start file mtf.rpst
c
c
c start block xxrpst
c
c
      call setin( infil, 'infil'//char(0), 's72*1'//char(0) )
      call setin( outfil, 'outfil'//char(0), 's72*1'//char(0) )
      call setin( antang, 'antang'//char(0), 'd*1'//char(0) )
c
c end block xxrpst
c
c
c
c end file mtf.rpst
c
```

```
c
c start file mtf.scov
c
c
c start block xxscov
c
c
      call setin( iopt, 'iopt'//char(0), 'i*1'//char(0) )
      call setin( rvcov, 'rvcov'//char(0), 'd*72'//char(0) )
      call setin( sncon, 'sncon'//char(0), 'd*2'//char(0) )
c
c end block xxscov
c
c
c
c end file mtf.scov
c
```

```
c
c start file mtf.sen
c
c
c start block xxsen
c                                    /
c
      call setin( maxobs, 'maxobs'//char(0), 'i*1'//char(0) )
      call setin( maxsen, 'maxsen'//char(0), 'i*1'//char(0) )
      call setin( icoas, 'icoas'//char(0), 'i*1'//char(0) )
      call setin( iradar, 'iradar'//char(0), 'i*1'//char(0) )
      call setin( itrky, 'itrky'//char(0), 'i*1'//char(0) )
      call setin( itrkz, 'itrkz'//char(0), 'i*1'//char(0) )
      call setin( isen1, 'isen1'//char(0), 'i*1'//char(0) )
      call setin( isen2, 'isen2'//char(0), 'i*1'//char(0) )
      call setin( isen3, 'isen3'//char(0), 'i*1'//char(0) )
      call setin( rsob, 'rsob'//char(0), 'd*21'//char(0) )
      call setin( qbs, 'qbs'//char(0), 'd*28'//char(0) )
c
c end block xxsen
c
c
c
c end file mtf.sen
c
```

```
c
c start file mtf.sprm
c
c
c start block xxsprm
c
c
      call setin( gmsg, 'gmsg'//char(0), 's40*1'//char(0) )
      call setin( seq, 'seq'//char(0), 's8*1'//char(0) )
      call setin( spg, 'spg'//char(0), 'i*20'//char(0) )
      call setin( dict, 'dict'//char(0), 'r*3'//char(0) )
      call setin( datbuf, 'datbuf'//char(0), 'r*2'//char(0) )
c
c end block xxsprm
c
c
c
c end file mtf.sprm
c
```

```
c
c start file mtf.sptm
c
c
c start block xxsptm
c
c
      call setin( pfreq, 'pfreq'//char(0), 'd*1'//char(0) )
      call setin( sptol, 'sptol'//char(0), 'd*1'//char(0) )
      call setin( sptime, 'sptime'//char(0), 'd*20'//char(0) )
c
c end block xxsptm
c
c
c
c end file mtf.sptm
c
```

```
c
c start file mtf.sun
c
c
c start block xxsun
c
c
      call setin( musun, 'musun'//char(0), 'd*1'//char(0) )
      call setin( dtsun, 'dtsun'//char(0), 'd*1'//char(0) )
      call setin( tsun0, 'tsun0'//char(0), 'd*1'//char(0) )
      call setin( rvsun0, 'rvsun0'//char(0), 'd*6'//char(0) )
      call setin( nsord, 'nsord'//char(0), 'i*1'//char(0) )
      call setin( ksun, 'ksun'//char(0), 'd*1'//char(0) )
      call setin( kear, 'kear'//char(0), 'd*1'//char(0) )
      call setin( kflux, 'kflux'//char(0), 'd*1'//char(0) )
      call setin( kealb, 'kealb'//char(0), 'd*1'//char(0) )
c
c end block xxsun
c
c
c
c end file mtf.sun
c
```

```
1   c
2   c start file mtf.svbi
3   c
4   c
5   c start block xxsvbi
6   c
7   c
8         call setin( psvb, 'psvb'//char(0), 'i*2'//char(0) )
9         call setin( svbtb, 'svbtb'//char(0), 'd*160'//char(0) )
10  c
11  c end block xxsvbi
12  c
13  c
14  c
15  c end file mtf.svbi
16  c
```

```
    1   c
    2   c start file mtf.timc
    3   c
    4   c
    5   c start block xxtimc
    6   c
    7   c
    8   c
    9   c end block xxtimc
   10   c
   11   c
   12   c
   13   c end file mtf.timc
   14   c
```

```
c
c start file mtf.time
c
c
c start block xxtime
c
c
      call setin( date, 'date'//char(0), 'i*5'//char(0) )
      call setin( dates, 'dates'//char(0), 'd*1'//char(0) )
      call setin( tbegin, 'tbegin'//char(0), 'd*1'//char(0) )
      call setin( tend, 'tend'//char(0), 'd*1'//char(0) )
      call setin( delta, 'delta'//char(0), 'd*1'//char(0) )
      call setin( endopt, 'endopt'//char(0), 'i*1'//char(0) )
      call setin( endval, 'endval'//char(0), 'd*1'//char(0) )
c
c end block xxtime
c
c
c
c end file mtf.time
c
```

```
c
c start file mtf.toff
c
c
c start block xxtoff
c
c
      call setin( ldate, 'ldate'//char(0), 'i*6'//char(0) )
      call setin( dtdate, 'dtdate'//char(0), 'i*6'//char(0) )
      call setin( dtbias, 'dtbias'//char(0), 'd*2'//char(0) )
      call setin( qafreq, 'qafreq'//char(0), 'i*1'//char(0) )
      call setin( tlapse, 'tlapse'//char(0), 'd*1'//char(0) )
      call setin( drive, 'drive'//char(0), 'i*1'//char(0) )
      call setin( tape, 'tape'//char(0), 'i*1'//char(0) )
      call setin( hpbin, 'hpbin'//char(0), 'i*1'//char(0) )
      call setin( dtable, 'dtable'//char(0), 'd*100'//char(0) )
c
c end block xxtoff
c
c
c
c end file mtf.toff
c
```

```
c
c start file mtf.tols
c
c
c start block xxtols
c
c
c
c end block xxtols
c
c
c
c end file mtf.tols
c
```

```
c
c start file mtf.usys
c
c
c start block xxusys
c
c
      call setin( usysex, 'usysex'//char(0), 'i*5'//char(0) )
      call setin( usysin, 'usysin'//char(0), 'i*1'//char(0) )
      call setin( usyst, 'usyst'//char(0), 's4*5'//char(0) )
      call setin( nsclz, 'nsclz'//char(0), 's4*50'//char(0) )
      call setin( usclz, 'usclz'//char(0), 'd*50'//char(0) )
c
c end block xxusys
c
c
c
c end file mtf.usys
c
```

```
c
c start file mtf.vcx
c
c
c start block xxvcx
c
c
      call setin( aeroc, 'aeroc'//char(0), 'd*2'//char(0) )
      call setin( cd, 'cd'//char(0), 'd*2'//char(0) )
      call setin( chord, 'chord'//char(0), 'd*2'//char(0) )
      call setin( c, 'c'//char(0), 'd*6'//char(0) )
      call setin( d, 'd'//char(0), 'd*10'//char(0) )
      call setin( kd, 'kd'//char(0), 'd*2'//char(0) )
      call setin( drgfac, 'drgfac'//char(0), 'd*2'//char(0) )
      call setin( rbarna, 'rbarna'//char(0), 'd*6'//char(0) )
      call setin( sref, 'sref'//char(0), 'd*2'//char(0) )
      call setin( sarea, 'sarea'//char(0), 'd*2'//char(0) )
      call setin( srflec, 'srflec'//char(0), 'd*2'//char(0) )
      call setin( solfac, 'solfac'//char(0), 'd*2'//char(0) )
      call setin( horder, 'horder'//char(0), 'i*2'//char(0) )
      call setin( hdgree, 'hdgree'//char(0), 'i*2'//char(0) )
      call setin( sagate, 'sagate'//char(0), 'd*2'//char(0) )
c
c end block xxvcx
c
c
c
c end file mtf.vcx
c
```

```
c
c start file mtf.vnt
c
c
c start block xxvnt
c
c
      call setin( vnttab, 'vnttab'//char(0), 'd*80'//char(0) )
c
c end block xxvnt
c
c
c
c end file mtf.vnt
c
```

```
c
c start file mtf.writ
c
c
c start block xxwrit
c
c
      call setin( files, 'files'//char(0), 's12*10'//char(0) )
      call setin( recpnt, 'recpnt'//char(0), 'i*200'//char(0) )
      call setin( nfiles, 'nfiles'//char(0), 'i*1'//char(0) )
      call setin( vars, 'vars'//char(0), 's12*100'//char(0) )
      call setin( nvars, 'nvars'//char(0), 'i*10'//char(0) )
c
c end block xxwrit
c
c
c
c end file mtf.writ
c
```

```
c hdrparm proc
c{d header index parameters
    c hbdate = start of base date y,m,d,h,m
          integer hbdate
          parameter (hbdate=67)
    c hbsec = seconds portion of base date
          integer hbsec
          parameter (hbsec=72)
    c hcdate = date of creation  yy/mm/dd  c*8
          integer hcdate
          parameter (hcdate=3)
    c hcver = version of program used for creation
    c stored as c16 (4 integer words)
          integer hcver
          parameter (hcver=5)
    c hcdes = job  description    of creation
    c 100 characters store as 25 integer words
          integer hcdes
          parameter (hcdes=9)
    c hfrmsz = frame length in integer words
          integer hfrmsz
          parameter (hfrmsz=79)
    c hnrec = number of records with data in it
          integer hnrec
          parameter (hnrec=78)
    c hnhrec = number of header records
          integer hnhrec
          parameter (hnhrec=1)
    c hstop = stop time
          integer hstop
          parameter (hstop=76)
    c hstrt = start time
          integer hstrt
          parameter (hstrt=74)
    c htype = file type
          integer htype
          parameter (htype=2)
    c hudate = date of creation  yy/mm/dd  c*8
          integer hudate
          parameter (hudate=35)
    c hudes = description of last update
    c 100 characters
          integer hudes
          parameter (hudes=41)
    c hspec = file spec block (extra 11 words)
          integer hspec
          parameter (hspec=80)
    c huver = version of update program (16characters)
          integer huver
          parameter (huver=37)
c}d
c end
```

```
 1   define PNAME  1
 2   define PUNIT  19
 3   define PFRMSZ 20
 4   define PHDRSZ 21
 5   define PCREC  22
 6   define PRWFLG 23
 7   define PNREC  24
 8   define PSTAT  25
 9   define PTOFF  26
10   define PTYPE  28
11   define PBDATE 29
12   define PBSEC  34
13   define PSTRT  36
14   define PSTOP  38
15   define PNHREC 40
16   define PNDICT 41
17   define iosccs "@(#)iopdefs      4.1 last update 86/12/12 16:24:32"
```

```
c iopparm proc
c{d i/o packet index parameters
    c pbdate = start of base date y,m,d,h,m
          integer pbdate
          parameter (pbdate=29)
    c pbsec = seconds portion of base date
          integer pbsec
          parameter (pbsec=34)
    c pcrec = current record number
          integer pcrec
          parameter (pcrec=22)
    c pfrmsz = frame size in integer words
          integer pfrmsz
          parameter (pfrmsz=20)
    c phdrsz = header size in integer words
          integer phdrsz
          parameter (phdrsz=21)
    c pnhrec = number of header records
          integer pnhrec
          parameter (pnhrec=40)
    c pname = start of file (32 characters)
          integer pname
          parameter (pname=1)
    c pndict = number of dictionary records
          integer pndict
          parameter (pndict=41)
    c pnrec = total number of records in file
          integer pnrec
          parameter (pnrec=24)
    c prwflg = read/write (uze) flag
          integer prwflg
          parameter (prwflg=23)
    c pstat = i/o status
          integer pstat
          parameter (pstat=25)
    c pstop = stop time
          integer pstop
          parameter (pstop=38)
    c pstrt = start time
          integer pstrt
          parameter (pstrt=36)
    c ptoff = time offset
          integer ptoff
          parameter (ptoff=26)
    c ptype = file type
          integer ptype
          parameter (ptype=28)
    c punit = unit number
          integer punit
          parameter (punit=19)
c}d
c end
```

```
      c pktparm proc
      c{d pktparm parameters
          c pktparm provides parameters for acces to the i/o packets
          c array packet
          c eph1 = first ephemeris file.
                integer eph1
                parameter (eph1=1)
          c eph2 = second ephemeris file.
                integer eph2
                parameter (eph2=2)
          c att1 = first attitude file.
                integer att1
                parameter (att1=3)
          c att2 = second attitude file.
                integer att2
                parameter (att2=4)
          c svl1 = first sensed velocity file.
                integer svl1
                parameter (svl1=5)
          c svl2 = second sensed velocity file.
                integer svl2
                parameter (svl2=6)
          c obs1 = first observation file.
                integer obs1
                parameter (obs1=7)
          c obs2 = second observation file.
                integer obs2
                parameter (obs2=8)
          c rel1 = first relative trajectory file.
                integer rel1
                parameter (rel1=9)
      c}d
      c end
```

```
1   c sfiltparam proc
2   c{d sfiltparam parameters
3       c nsol = dimension of solution vector
4             integer nsol
5             parameter (nsol = 22)
6   c}d
```

1 s.hdrparm

APPENDIX III

SUBROUTINE MANUALS

APPENDIX III SUBROUTINE MANUALS

This appendix provides manual entries for each subroutine file in the RELBET System. These serve as a quick reference to the subroutine function descriptions and provide definitions of the calling arguments used. The manual entries are organized according to the directory names of the associated code.

The entries follow a format standard to UNIX. As appropriate, they contain the following sections.

NAME: Names of all externally accessible identifiers followed by a brief description of the package.

SYNOPSIS: A quick summary of how to invoke the relevant functions and parameters. Includes types and arguments.

DESCRIPTION: A functional description of what the functions do and what the options are

OPTIONS: Description of the options when they are suitable for inclusion in the DESCRIPTION.

FILES: The files are used or assumed by the application.

EXAMPLE: Annotated examples of how to use the application.

COMMENTS: Miscellaneous comments. For example, rationales for the design or functions may appear here.

BUGS: Known problems.

DIAGNOSTICS: Warning and error messages, debug options.

SEE ALSO: References to related applications.

AUTHOR: The name of the responsible programmer.

ROUTINE CROSS REFERENCE

NAME	FILE	DIRECTORY
Add_GNList	"gnamlist.c"	Message
BaseCalTime	"BaseTime.c"	Time
BaseTime	"fileio.c"	Gbfile
Begin	"search.c"	Fman
CurSysTime	"systime.c"	Time
End	"search.c"	Fman
Error_Count	"stat_msg.c"	Message
File	"search.c"	Fman
FileBaseTime	"gbhead.c"	Gbfile
Frame3	"frame3d.c"	Coordinate
GBeginTime	"BaseTime.c"	Time
GDelTime	"BaseTime.c"	Time
GEndTime	"BaseTime.c"	Time
GMTday	"systime.c"	Time
GMTsec	"systime.c"	Time
GNLgetName	"gnamlist.c"	Message
GetCurTime	"systime.c"	Time
HBtoHS	"gbHBufMove.c"	Gbfile
HStoHB	"gbHBufMove.c"	Gbfile
Help	"search.c"	Fman
JD_BaseTime	"BaseTime.c"	Time
LBeginTime	"BaseTime.c"	Time
LDelTime	"BaseTime.c"	Time
LEndTime	"BaseTime.c"	Time
MakeTimeLine	"timelines.c"	Message
Make_GNLIST	"gnamlist.c"	Message
NTBLCK	"ntran.c"	Product
NTCLOSE	"ntran.c"	Product
NTEOF	"ntran.c"	Product
NTFILE	"ntran.c"	Product
NTOPEN	"ntran.c"	Product
NTREAD	"ntran.c"	Product
NTRITE	"ntran.c"	Product
NTRW	"ntran.c"	Product

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
NTRWRL	"ntran.c"	Product
Prime_id	"search.c"	Fman
Radius	"search.c"	Fman
Save_Str_Buf	"str_store.c"	Message
Second_id	"search.c"	Fman
StatErrExit	"stat_msg.c"	Message
Stat_Msg	"stat_msg.c"	Message
Time	"search.c"	Fman
UNVOUT	"unvout.c"	Mktape
UVW_Cart	"uvw.c"	Coordinate
Warning_Count	"stat_msg.c"	Message
a121q	"aijkq.b"	Coordinate
a123q	"aijkq.b"	Coordinate
a131q	"aijkq.b"	Coordinate
a132q	"aijkq.b"	Coordinate
a212q	"aijkq.b"	Coordinate
a213q	"aijkq.b"	Coordinate
a231q	"aijkq.b"	Coordinate
a232q	"aijkq.b"	Coordinate
a312q	"aijkq.b"	Coordinate
a313q	"aijkq.b"	Coordinate
a321q	"aijkq.b"	Coordinate
a323q	"aijkq.b"	Coordinate
a32t36	"a32t36.b"	Product
acnvrt	"acnvrt.b"	Math
adbl2i	"acnvrt.b"	Math
addANLitem	"anlists.c"	Message
addefile	"stat_msg.c"	Message
addptr	"ptrlists.c"	Message
aero	"aero.b"	Force
aerror	"ingrss.b"	Message
ai2dbl	"acnvrt.b"	Math
ai2rl	"acnvrt.b"	Math
aijkq	"aijkq.b"	Coordinate
amenu	"amenu.b"	Prompts

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
ang2	"ang2.b"	Math
angle_rate	"qaatt.c"	Fman
append	"lists.c"	Lists
arctan	"arctan.b"	Math
arl2i	"acnvrt.b"	Math
arshft	"arshft.b"	Math
ascale	"ascale.b"	Plot
automx	"automx.b"	Plot
awarn	"ingrss.b"	Message
berror	"stat_msg.c"	Message
binop	"linput.c"	Linput
bldout	"getout.b"	Downfor
bsearch_timeline	"timelines.c"	Message
bwarn	"stat_msg.c"	Message
c2shl	"c2shl.b"	Coordinate
car	"lists.c"	Lists
cart	"cart.b"	Coordinate
cbody	"cbody.b"	Force
cbugs	"xcoas.b"	Obs
cdr	"lists.c"	Lists
cdrag	"cdrag.b"	Force
cdtojd	"cmpdat.b"	Time
center	"center.b"	Files
char2i	"arshft.b"	Math
cinit	"xcoas.b"	Obs
clas	"clas.b"	Coordinate
clsfil	"obtfil.b"	Downfor
clsout	"getout.b"	Downfor
cmp2sg	"cmp2sg.c"	Fman
cmpdat	"cmpdat.b"	Time
cmvbit	"cmvbit.b"	Product
coash	"xcoas.b"	Obs
coasv	"xcoas.b"	Obs
combine	"obsnois.c"	Fman
comptw	"comptw.b"	Product

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
cons	"lists.c"	Lists
constm	"constm.b"	Celestial
cover_file	"qacover.c"	Fman
cros	"mx3ops.b"	Math
ctio	"tio.b"	Prompts
cvprop	"cvprop.b"	Filter
cyldr	"cyldr.b"	Force
datime	"datime.b"	Message
days	"days.b"	Time
days	"timedate.c"	Time
daysibc	"timedate.c"	Time
daysxx	"daysxx.b"	Time
dbl2i	"arshft.b"	Math
dbshft	"arshft.b"	Math
dciph	"dciph.b"	Prompts
dciphr	"dciphr.b"	Prompts
dctprt	"dctprt.b"	Gff
ddna	"ddna.b"	Noisanal
ddnois	"ddnois.b"	Noisanal
delout	"getout.b"	Downfor
dfdata	"dfdata.b"	Downfor
dfnput	"dfnput.b"	Downfor
dhms	"dhms.b"	Time
div	"linput.c"	Linput
dkal	"dkal.b"	Filter
dnumber	"lists.c"	Lists
dollar	"dollar.b"	Charutil
dpfmt	"dpfmt.b"	Qatape
dplot	"dplot.b"	Plot
dprod	"dprod.b"	Product
dprop	"dprop.b"	Propagate
drpst	"drpst.b"	Fman
dsmth	"dsmth.b"	Filter
dspxq	"dspxq.b"	Numdis
dtio	"tio.b"	Prompts

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
dvalue	"lists.c"	Lists
dvdt	"dvdt.b"	Propagate
dwnfmt	"dwnfmt.b"	Downfor
dxcmp	"dxcmp.b"	Numdis
dxqdsp	"dxqdsp.b"	Numdis
dxset	"dxset.b"	Product
eb2asc	"eb2asc.c"	Downfor
edit_frame	"filedit.c"	Fman
egrss	"ingrss.b"	Message
eph2_in	"mkinit.c"	Fman
eph2rel	"eph2rel.c"	Fman
eq	"lists.c"	Lists
err_code	"stat_msg.c"	Message
err_hpcode	"stat_msg.c"	Message
etsec	"timedate.c"	Time
euler	"euler.b"	Coordinate
fetchbits	"ibmcvt.c"	Downfor
fgbprt	"gbprtd.c"	Gbfile
file2io	"file2io.c"	Gbfile
file_size	"sln2rl.c"	Fman
fileinfo_io	"fileinfo_io.c"	Gbfile
fileio	"fileio.c"	Gbfile
fprtCurTime	"systime.c"	Time
fprtFinish	"stat_msg.c"	Message
fprtStars	"prt_util.c"	Message
fprtTimeLine	"timelines.c"	Message
fprtarray	"prtarray.c"	Math
fprtctime	"timeprint.c"	Time
fprtdate	"timeprint.c"	Time
fprtgnlist	"gnamlist.c"	Message
fprthms	"timeprint.c"	Time
fprtsec	"timeprint.c"	Time
freeGBF	"gbopen.c"	Gbfile
freeplist	"ptrlists.c"	Message
fskip_lines	"prt_util.c"	Message

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
fvalue	"lists.c"	Lists
gbclose	"gbopen.c"	Gbfile
gbdread	"gbio.c"	Gbfile
gbdwrite	"gbio.c"	Gbfile
gbfcls	"gbfio.c"	Gbfile
gbfcom	"gbfcom.c"	Fman
gbfdfc	"gbfio.c"	Gbfile
gbfnew	"gbfio.c"	Gbfile
gbfopn	"gbfio.c"	Gbfile
gbfops	"gbfio.c"	Gbfile
gbfphead	"gbprtd.c"	Gbfile
gbfree	"gbopen.c"	Gbfile
gbfwrt	"gbfio.c"	Gbfile
gbnew	"gbopen.c"	Gbfile
gbopen	"gbopen.c"	Gbfile
gbphead	"gbprtd.c"	Gbfile
gbpos	"gbio.c"	Gbfile
gbread	"gbio.c"	Gbfile
gbrhead	"gbhead.c"	Gbfile
gbtime	"gbtime.c"	Gbfile
gbtoff	"gbhead.c"	Gbfile
gbwhead	"gbhead.c"	Gbfile
gbwrite	"gbio.c"	Gbfile
gdisp	"gdisp.b"	Numdis
gdspop	"gdspop.b"	Numdis
genout	"genout.b"	Downfor
gerror	"iegrss.b"	Message
get	"get.b"	Files
get_next_frame	"file2io.c"	Gbfile
get_rate_table_value	"ratetable.c"	Message
getangle	"qastar.c"	Fman
getbit	"getbit.b"	Qatape
getbits	"eb2asc.c"	Downfor
getbits	"unvout.c"	Mktape
getdat	"getdat.b"	Downfor

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
gethdr	"gethdr.b"	Downfor
getin	"initlinput.c"	Linput
getm	"getm.b"	Force
getnxt	"getnxt.b"	Filter
getout	"getout.b"	Downfor
getrnp	"getrnp.b"	Celestial
getspg	"getspg.b"	Product
gfcls	"gfcls.b"	Gff
gfdict	"gfdict.b"	Gff
gfemsg	"gfemsg.b"	Gff
gfend	"gfend.b"	Gff
gff_in	"filedit.c"	Fman
gffdsp	"gffdsp.b"	Gff
gfnew	"gfnew.b"	Gff
gfopen	"gfopen.b"	Gff
gfread	"gfread.b"	Gff
gfrhdr	"gfrhdr.b"	Gff
gftime	"gftime.b"	Gff
gftims	"gftims.b"	Gff
gfwhdr	"gfwhdr.b"	Gff
gfwrit	"gfwrit.b"	Gff
gmenu	"qxmenu.b"	Prompts
gndsp	"gndsp.b"	Numdis
grfnpt	"pltnpt.b"	Plot
gsempty	"gstack.c"	Stacks
gsfree	"gstack.c"	Stacks
gspeek	"gstack.c"	Stacks
gspoke	"gstack.c"	Stacks
gspop	"gstack.c"	Stacks
gspush	"gstack.c"	Stacks
gwarn	"iegrss.b"	Message
harm	"harm.b"	Force
hdrprt	"hdrprt.b"	Gff
hms2ds	"hms2ds.b"	Time
hms2sec	"timedate.c"	Time

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
hptou5	"hptou5.b"	Product
i2char	"arshft.b"	Math
i2dbl	"arshft.b"	Math
i2real	"arshft.b"	Math
ibmcvt	"ibmcvt.c"	Downfor
ibshft	"arshft.b"	Math
icshft	"icshft.b"	Gff
ident	"mxops.b"	Math
ident3	"mx3ops.b"	Math
iegrss	"iegrss.b"	Message
imatq	"qnops.c"	Math
imatq	"qtnops.b"	Math
imove	"imove.b"	Qatape
imzero	"mxops.b"	Math
infmt	"infmt.b"	Qatape
ingrss	"ingrss.b"	Message
initin	"initlinput.c"	Linput
inumber	"lists.c"	Lists
invrs	"invrs.r"	RELBET/Filter
iopprt	"iopprt.b"	Gff
ioschk	"ioschk.b"	Message
irot	"qtnops.b"	Math
isatom	"lists.c"	Lists
isdnumber	"lists.c"	Lists
isearch_timeline	"timelines.c"	Message
isfunction	"lists.c"	Lists
isinumber	"lists.c"	Lists
issymbol	"lists.c"	Lists
itio	"tio.b"	Prompts
ivalue	"lists.c"	Lists
ivshft	"arshft.b"	Math
ivzero	"arshft.b"	Math
ixatm	"ixatm.b"	Input
ixbias	"ixbias.b"	Input
ixcon	"ixcon.b"	Input

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
ixdata	"ixdata.b"	Input
ixdprm	"ixdprm.b"	Input
ixerth	"ixerth.b"	Input
ixfile	"ixfile.b"	Input
ixgnrl	"ixgnrl.b"	Input
ixgraf	"ixgraf.b"	Input
ixgrav	"ixgrav.b"	Input
ixinit	"ixinit.b"	Input
ixkal	"ixkal.b"	Input
ixmas	"ixmas.b"	Input
ixmax	"ixmax.b"	Input
ixmisc	"ixmisc.b"	Input
ixmoon	"ixmoon.b"	Input
ixnflz	"ixnflz.b"	Input
ixprnt	"ixprnt.b"	Input
ixprop	"ixprop.b"	Input
ixqcrv	"ixqcrv.b"	Input
ixqgen	"ixqgen.b"	Input
ixqprm	"ixqprm.b"	Input
ixrpst	"ixrpst.b"	Input
ixscov	"ixscov.b"	Input
ixsen	"ixsen.b"	Input
ixsprm	"ixsprm.b"	Input
ixsptm	"ixsptm.b"	Input
ixsun	"ixsun.b"	Input
ixsvbi	"ixsvbi.b"	Input
ixtime	"ixtime.b"	Input
ixtoff	"ixtoff.b"	Input
ixusys	"ixusys.b"	Input
ixvcx	"ixvcx.b"	Input
ixvnt	"ixvnt.b"	Input
j2c	"j2c.b"	Time
j2ymd	"j2ymd.b"	Time
jdtocd	"cmpdat.b"	Time
jmpcomp	"qaranjmp.c"	Fman

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
jul2cal	"timedate.c"	Time
juldate	"timedate.c"	Time
jultime	"timedate.c"	Time
kalman	"kalman.b"	Filter
krelo	"krelo.b"	Filter
kreloc	"krelo.b"	Filter
lax	"lax.b"	Coordinate
length	"lists.c"	Lists
lgfac	"lgint.b"	Numdis
lgint	"lgint.b"	Numdis
linit	"lists.c"	Lists
linput	"linput.c"	Linput
lint	"lint.b"	Interpolate
lintin	"lintin.b"	Interpolate
locate	"lists.c"	Lists
lrntrp	"lrntrp.b"	Interpolate
lrntrp	"relntrp.c"	Interpolate
lsearch_timeline	"timelines.c"	Message
lsmin	"lsmin.b"	Math
lst	"lst.b"	Coordinate
lstfnd	"lstfnd.b"	Charutil
m2qsub	"m2qsub.b"	Math
main	"cmp2sg.c"	Fman
main	"eph2rel.c"	Fman
main	"fiche.c"	Fich
main	"filedit.c"	Fman
main	"gbfcom.c"	Fman
main	"mkinit.c"	Fman
main	"obsnois.c"	Fman
main	"qaatt.c"	Fman
main	"qacover.c"	Fman
main	"qanois.c"	Fman
main	"qaranjmp.c"	Fman
main	"qastar.c"	Fman
main	"qasv.c"	Fman

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
main	"rdwt.c"	Fman
main	"read_set.c"	Fman
main	"read_sit.c"	Fman
main	"rlvsrl.c"	Fman
main	"search.c"	Fman
main	"sln2rl.c"	Fman
main	"stop.c"	Fman
makeDate	"timemake.c"	Time
makeGBData	"gbmake.c"	Gbfile
makeGBF	"gbmake.c"	Gbfile
makeHMS	"timemake.c"	Time
makeTime	"timemake.c"	Time
make_gstack	"gstack.c"	Stacks
makeplist	"ptrlists.c"	Message
match	"match.b"	Downfor
matq	"qnops.c"	Math
matq	"qtnops.b"	Math
member	"lists.c"	Lists
mkinit	"mkinit.c"	Fman
mktape	"mktape.b"	Mktape
mmxchk	"mmxchk.b"	Plot
mnthnum	"timedate.c"	Time
moonup	"moopos.b"	Celestial
moopos	"moopos.b"	Celestial
msgdsp	"msgdsp.b"	Message
mshft3	"mx3ops.b"	Math
mshift	"mxops.b"	Math
mt	"matrix.c"	Math
mtran	"mxops.b"	Math
mtran3	"mx3ops.b"	Math
mtxm	"matrix.c"	Math
mtxm3	"mx3ops.b"	Math
mtxv	"matrix.c"	Math
mtxv	"mxops.b"	Math
mtxv3	"mx3ops.b"	Math

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
mult	"linput.c"	Linput
mvmul	"mxops.b"	Math
mvmul3	"mx3ops.b"	Math
mx3ops	"mx3ops.b"	Math
mxadd	"mxops.b"	Math
mxm	"matrix.c"	Math
mxmc	"matrix.c"	Math
mxmt	"matrix.c"	Math
mxmt3	"mx3ops.b"	Math
mxmul	"mxops.b"	Math
mxmul3	"mx3ops.b"	Math
mxops	"mxops.b"	Math
mxv	"matrix.c"	Math
mzero	"mxops.b"	Math
mzero3	"mx3ops.b"	Math
ndfnit	"ndnflz.b"	Dsputil
ndfnpt	"ndnflz.b"	Dsputil
ndhead	"ndhead.b"	Dsputil
ndnflz	"ndnflz.b"	Dsputil
ndpage	"ndpage.b"	Dsputil
newGBF	"gbhead.c"	Gbfile
newcell	"linput.c"	Linput
newcell	"newcell.c"	Linput
newpg	"msgdsp.b"	Message
ngrss	"ingrss.b"	Message
nil	"lists.c"	Lists
nitout	"getout.b"	Downfor
nkal	"nkal.b"	Filter
nlist	"linput.c"	Linput
nnois	"nnois.b"	Noisanal
normq	"qnops.c"	Math
nplot	"nplot.b"	Plot
nprod	"nprod.b"	Product
nprop	"nprop.b"	Propagate
nrmlzq	"nrmlzq.b"	Math

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
nrpst	"nrpst.b"	Fman
nsmth	"nsmth.b"	Filter
nstrng	"nstrng.b"	Charutil
nxcmp	"nxcmp.b"	Numdis
nxqdsp	"nxqdsp.b"	Numdis
nxtnd	"nxtnd.b"	Dsputil
nxtnd	"nxtnd.b"	Product
obcls	"obout.b"	Filter
obcode	"obcode.b"	Obs
obest	"obest.b"	Filter
obout	"obout.b"	Filter
obread	"obread.b"	Files
obtfil	"obtfil.b"	Downfor
obwrit	"obwrit.b"	Files
opnout	"getout.b"	Downfor
opnprp	"opnprp.b"	Propagate
out_bias	"sln2rl.c"	Fman
out_cov	"sln2rl.c"	Fman
out_rel	"sln2rl.c"	Fman
pchrat	"xradar.b"	Obs
pcpt	"pcpt.b"	Filter
pdsp	"tpdsp.b"	Message
pitch	"xradar.b"	Obs
plotx	"plotx.b"	Plot
pltnpt	"pltnpt.b"	Plot
pmenu	"qxmenu.b"	Prompts
pred	"pred.b"	Filter
print_record	"read_set.c"	Fman
print_record	"read_sit.c"	Fman
prodx	"prodx.b"	Product
prpnpt	"prpnpt.b"	Propagate
prt3mat	"prtarray.c"	Math
prt3tmat	"prtarray.c"	Math
prt3vec	"prtarray.c"	Math
prtANList	"anlists.c"	Message

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
prtANLitem	"anlists.c"	Message
prtCurTime	"systime.c"	Time
prtFinish	"stat_msg.c"	Message
prtStars	"prt_util.c"	Message
prtarray	"prtarray.c"	Math
prtctime	"timeprint.c"	Time
prtdate	"timeprint.c"	Time
prthms	"timeprint.c"	Time
prtin	"prt_input.c"	Linput
prtsec	"timeprint.c"	Time
pscls	"psout.b"	Filter
psout	"psout.b"	Filter
pstrz	"tpdsp.b"	Message
ptbl	"ptbl.b"	Filter
put72	"unvout.c"	Mktape
putchr	"unvout.c"	Mktape
putexp	"putexp.c"	Lists
pwr	"linput.c"	Linput
q121a	"qijka.b"	Coordinate
q123a	"qijka.b"	Coordinate
q131a	"qijka.b"	Coordinate
q132a	"qijka.b"	Coordinate
q212a	"qijka.b"	Coordinate
q213a	"qijka.b"	Coordinate
q231a	"qijka.b"	Coordinate
q232a	"qijka.b"	Coordinate
q2crv	"q2crvz.b"	Plot
q2crvz	"q2crvz.b"	Plot
q2draw	"q2draw.b"	Plot
q2ecrv	"q2crvz.b"	Plot
q2fnsh	"q2grph.b"	Plot
q2grph	"q2grph.b"	Plot
q2legn	"q2crvz.b"	Plot
q2lsto	"q2crvz.b"	Plot
q2msub	"q2msub.b"	Math

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
q312a	"qijka.b"	Coordinate
q313a	"qijka.b"	Coordinate
q321a	"qijka.b"	Coordinate
q323a	"qijka.b"	Coordinate
q_pop	"gstack.c"	Stacks
q_push	"gstack.c"	Stacks
qanois	"qanois.c"	Fman
qatape	"qatape.b"	Qatape
qcxq	"qnops.c"	Math
qcxq	"qtnops.b"	Math
qdevin	"qdevin.b"	Plot
qdevrl	"qdevin.b"	Plot
qdot	"qnops.c"	Math
qijka	"qijka.b"	Coordinate
qintp	"qnops.c"	Math
qintrp	"qintrp.b"	Interpolate
qpintx	"qpintx.b"	Plot
qplot	"qplot.b"	Plot
qplt2d	"qplt2d.b"	Plot
qpxget	"qpxget.b"	Plot
qrlptn	"qslptn.b"	Plot
qrot	"qrot.b"	Math
qslptn	"qslptn.b"	Plot
qtnops	"qtnops.b"	Math
qtoim	"qtnops.b"	Math
qtom	"qtnops.b"	Math
qvirot	"qnops.c"	Math
qvrot	"qnops.c"	Math
qwntrp	"qwntrp.b"	Interpolate
qxmenu	"qxmenu.b"	Prompts
qxq	"qnops.c"	Math
qxq	"qtnops.b"	Math
qxqc	"qnops.c"	Math
qxqc	"qtnops.b"	Math
qxstuf	"qxstuf.b"	Charutil

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
qxvalu	"qxvalu.b"	Charutil
range	"xradar.b"	Obs
ranrat	"xradar.b"	Obs
read_set	"read_set.c"	Fman
read_sit	"read_sit.c"	Fman
real2i	"arshft.b"	Math
ref_in	"filedit.c"	Fman
rel_in	"mkinit.c"	Fman
rl2beg	"rlfill.b"	Files
rl2end	"rlfill.b"	Files
rlback	"rlfill.b"	Files
rlbwds	"rledit.b"	Files
rlcls	"rlcls.b"	Files
rlcntr	"rlcntr.b"	Files
rldsp	"rldsp.b"	Files
rledit	"rledit.b"	Files
rlfill	"rlfill.b"	Files
rlfrnt	"rlfill.b"	Files
rlfwds	"rledit.b"	Files
rlhdsp	"rldsp.b"	Files
rlmid1	"rlmid1.b"	Files
rlnew	"rlnew.b"	Files
rlntrp	"rlntrp.b"	Interpolate
rlopen	"rlopen.b"	Files
r pdsp	"rldsp.b"	Files
rlread	"rlread.b"	Files
rltime	"rltime.b"	Files
rlvsrl	"rlvsrl.c"	Fman
rlwrit	"rlwrit.b"	Files
rmefile	"stat_msg.c"	Message
rmptr	"ptrlists.c"	Message
rmxm	"rmatrix.c"	Math
rmxmc	"rmatrix.c"	Math
rmxv	"rmatrix.c"	Math
rmzero	"mxops.b"	Math

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
roll	"xradar.b"	Obs
rolrat	"xradar.b"	Obs
rot	"qtnops.b"	Math
rotate	"rotate.b"	Celestial
rptost	"rptost.b"	Fman
rrrfile	"qaranjmp.c"	Fman
rst	"rst.b"	Coordinate
rtangl	"rtangl.b"	Obs
rtio	"tio.b"	Prompts
rtntrp	"relntrp.c"	Interpolate
rtntrp	"rtntrp.b"	Interpolate
runkut	"runkut.b"	Propagate
rvprop	"rvprop.b"	Propagate
rvshft	"arshft.b"	Math
rvup	"rvup.b"	Propagate
rvzero	"arshft.b"	Math
sclset	"sclset.b"	Plot
search	"search.c"	Fman
sec2hms	"timedate.c"	Time
secnds	"secnds.b"	Time
sensed_vel	"qasv.c"	Fman
setDate	"timemake.c"	Time
setHMS	"timemake.c"	Time
setMaxErr	"stat_msg.c"	Message
setTime	"timemake.c"	Time
setin	"initlinput.c"	Linput
setq	"linput.c"	Linput
sfilt	"sfilt.b"	Filter
sfinit	"sfinit.b"	Filter
sfout	"sfout.b"	Filter
sftrat	"xradar.b"	Obs
shaft	"xradar.b"	Obs
shoxq	"dspxq.b"	Numdis
sign	"sign.c"	Math
skip_lines	"prt_util.c"	Message

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
smmul3	"mx3ops.b"	Math
smooth	"smooth.r"	Filter
snoise	"snoise.b"	Filter
solrad	"solrad.b"	Force
spfmt	"spfmt.b"	Qatape
sphdr	"sphdr.b"	Product
sprtctime	"timeprint.c"	Time
sprtdate	"timeprint.c"	Time
sprthms	"timeprint.c"	Time
sprtsec	"timeprint.c"	Time
star_track	"qastar.c"	Fman
starz	"msgdsp.b"	Message
std_time	"timedate.c"	Time
stinit	"xtrack.b"	Obs
stm	"stm.b"	Filter
stmrv	"stmrv.b"	Filter
stop	"stop.c"	Fman
stprop	"stprop.b"	Filter
strsave	"strsave.c"	Lists
strstore	"strstore.c"	Lists
sub	"linput.c"	Linput
sum	"linput.c"	Linput
sunpos	"sunpos.b"	Celestial
sunup	"sunpos.b"	Celestial
superg	"superg.b"	Propagate
svalue	"lists.c"	Lists
svelbs	"svelbs.b"	Force
svftch	"svftch.b"	Force
svmul	"mxops.b"	Math
svmul3	"mx3ops.b"	Math
svntrp	"svntrp.b"	Interpolate
symbol	"lists.c"	Lists
symxv	"symxv.b"	Math
tangle	"tangle.b"	Obs
taperd	"taperd.c"	Downfor

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
tblchk	"tblchk.b"	Product
tblook	"tblook.b"	Force
tdsp	"tpdsp.b"	Message
tenrnd	"tenrnd.b"	Math
test_print	"qastar.c"	Fman
text_in	"filedit.c"	Fman
tio	"tio.b"	Prompts
tpdsp	"tpdsp.b"	Message
tpstrz	"tpdsp.b"	Message
trackh	"xtrack.b"	Obs
trackv	"xtrack.b"	Obs
trnion	"xradar.b"	Obs
trnrat	"xradar.b"	Obs
tstep	"tstep.b"	Propagate
tstrz	"tpdsp.b"	Message
twobod	"twobod.b"	Celestial
uax	"uax.b"	Coordinate
udpfmt	"udpfmt.b"	Product
uinfmt	"uinfmt.b"	Product
unop	"linput.c"	Linput
uspfmt	"uspfmt.b"	Product
ust	"ust.b"	Coordinate
usub	"linput.c"	Linput
uvw2m	"uvw2m.b"	Filter
uvwlvh	"uvwlvh.b"	Coordinate
uvwmat	"uvwmat.b"	Coordinate
vadd	"mxops.b"	Math
vadd	"vector.c"	Math
vadd3	"mx3ops.b"	Math
vaddto	"vector.c"	Math
valdat	"valdat.b"	Downfor
vdist	"vector.c"	Math
vdot	"mxops.b"	Math
vdot	"vector.c"	Math
vdot3	"mx3ops.b"	Math

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
vfadd	"vector.c"	Math
vfaddto	"vector.c"	Math
vfmul	"vector.c"	Math
vlist	"linput.c"	Linput
vlrest	"vlrest.b"	Filter
vnorm	"mxops.b"	Math
vnorm3	"mx3ops.b"	Math
vntfch	"vntfch.b"	Force
vrss	"vector.c"	Math
vrunit	"vector.c"	Math
vset	"vector.c"	Math
vshft3	"mx3ops.b"	Math
vshift	"arshft.b"	Math
vsub	"mxops.b"	Math
vsub	"vector.c"	Math
vsub3	"mx3ops.b"	Math
vunit	"mxops.b"	Math
vunit3	"mx3ops.b"	Math
vzero	"arshft.b"	Math
vzero	"vector.c"	Math
vzero3	"mx3ops.b"	Math
wndang	"wndang.b"	Force
write_file	"rdwt.c"	Fman
wrtout	"getout.b"	Downfor
x1blnk	"qxstuf.b"	Charutil
xblank	"qxstuf.b"	Charutil
xcmpar	"xcmpar.b"	Numdis
xcoas	"xcoas.b"	Obs
xdsp	"msgdsp.b"	Message
xdsp2	"msgdsp.b"	Message
xf8dmp	"xf8dmp.b"	Dsputil
xi2chr	"xi2chr.b"	Charutil
xnth	"xnth.b"	Charutil
xqdatz	"xqwtmz.b"	Prompts
xqdsp	"xqdsp.b"	Numdis

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
xqdsp1	"dspxq.b"	Numdis
xqdsph	"dspxq.b"	Numdis
xqkmp1	"xqkmpt.b"	Numdis
xqkmp2	"xqkmpt.b"	Numdis
xqkmpt	"xqkmpt.b"	Numdis
xqtime	"xqwtmz.b"	Prompts
xqwget	"xqwget.b"	Dsputil
xqwint	"xqwget.b"	Dsputil
xqwtmz	"xqwtmz.b"	Prompts
xradar	"xradar.b"	Obs
xtrack	"xtrack.b"	Obs
yesno	"yesno.b"	Prompts
ymd2j	"ymd2j.b"	Time
yyback	"lex.c"	Linput
yybgin	"lex.c"	Linput
yycrank	"lex.c"	Linput
yyerror	"linput.c"	Linput
yyestate	"lex.c"	Linput
yyextra	"lex.c"	Linput
yyfnd	"lex.c"	Linput
yyin	"lex.c"	Linput
yyinput	"lex.c"	Linput
yyleng	"lex.c"	Linput
yylex	"lex.c"	Linput
yylineno	"lex.c"	Linput
yylook	"lex.c"	Linput
yylsp	"lex.c"	Linput
yylstate	"lex.c"	Linput
yymatch	"lex.c"	Linput
yymorfg	"lex.c"	Linput
yyolsp	"lex.c"	Linput
yyout	"lex.c"	Linput
yyoutput	"lex.c"	Linput
yyprevious	"lex.c"	Linput
yysbuf	"lex.c"	Linput

ROUTINE CROSS REFERENCE (cont'd)

NAME	FILE	DIRECTORY
yysptr	"lex.c"	Linput
yysvec	"lex.c"	Linput
yytchar	"lex.c"	Linput
yytext	"lex.c"	Linput
yytop	"lex.c"	Linput
yyunput	"lex.c"	Linput
yyvstop	"lex.c"	Linput
zutek	"zutek.b"	Force

NAME

constm

SYNOPSIS

```
subroutine constm (sold,told,snew,tnew,part)
double precision sold (6)
double precision told
double precision snew (6)
double precision tnew
double precision part (6,6)
```

DESCRIPTION

constm

Computes a state transition matrix using mean conic approximation.

sold — old state (input)

told — time of old state (input)

snew — new state (input)

tnew — time of new state (input)

part — partial of new state with respect to new state (output)

NAME

getrnp

SYNOPSIS

```
subroutine getrnp (date,dates,cdetut,trnp,rnp,hangle)
integer date (5)
double precision dates
double precision cdetut (2)
double precision trnp
double precision rnp (3,3)
double precision hangle
```

DESCRIPTION

getrnp

Computes rnp matrix at given calendar date.

date year-month-day-hour-minute of base date (input)

dates seconds of base date (input)

cdetut julian ephermis to julian universal conversion constants (input)

trnp time after base date (in seconds) for rnp matrix (input)

rnp rotation nutation precision matrix (output)

hangle hour angle (output)

NAME

moopos, moonup

SYNOPSIS

```
subroutine moopos (t,r)
double precision t
double precision r (3)

entry moonup
```

DESCRIPTION

moopos

Computes the moon position by interpolation from the ephemeris file created by moonup.

t current time (input)

r moon position at current time (output)

moonup

Creates moon ephemeris based on xxmoon parameters.

NAME

rotate

SYNOPSIS

```
subroutine rotate (bjd,deltat,cdetut,np,rnp,rascm)
double precision bjd
double precision deltat
double precision cdetut (2)
double precision np (3,3)
double precision rnp (3,3)
double precision rascm
```

DESCRIPTION

rotate

Analytically generates the true hour angle of greenwich, the precession, nutation, precession-nutation, and geographic transformation matrices for the earth. All matrices are referenced to a base coordinate frame defined by the mean equator and vernal equinox of the earth at the epoch 1950.0.

bjd	bjd = base julian date
deltat	delta in hours from base julian date
cdetut	array of constants used in determining the difference between ephemeris time and universal time
np	nutation-precession matrix
rnp	rotation-nutation-precession matrix
rascm	greenwich mean hour angle

NAME

sunpos, sunup

SYNOPSIS

```
subroutine sunpos (t,r)
double precision t
double precision r (3)

entry sunup
```

DESCRIPTION

sunpos

Computes the sun position by interpolation from the ephemeris file created by sunup.

t current time (input)

r sun position at current time (output)

sunup

Creates sun ephemeris based on xxsun parameters.

NAME
twobod

SYNOPSIS
```
subroutine twobod (mu,tau,x,y)
double precision mu
double precision tau
double precision x (6)
double precision y (6)
```

DESCRIPTION
twobod
Propagates a vector using Goodyear's two universal variable scheme.

mu	gravitational parameter (input)
tau	time step (input)
x	initial state (input)
y	final state (output)

NAME

dollar

SYNOPSIS

```
subroutine dollar (x,lx)
integer lx
character*1 x (lx)
```

DESCRIPTION

dollar

Fills final blank characters of input string with dollar signs. Input lx is index of character to start at, returns 1st nonblank or non dollar sign index in lx. Note that last character is always a dollar sign.

lx length of string (input/output)

x character string (input/output)

NAME

lstfnd

SYNOPSIS

```
subroutine lstfnd (x,list,lx,ilist)
integer ilist
integer lx
character*1 x (lx)
character*i list (lx,ilist)
```

DESCRIPTION

lstfnd

Searches list for entry x. The search begins at the end (entry ilist), and continues to the beginning. On return ilist is thus the last index of the list with the name x or else 0, in which case there were no entries found.

ilist length of list on input, index of item on exit(input/output)

lx length of an entry in bytes (input)

x search item (input)

list list to search (input)

NAME
nstrng

SYNOPSIS
```
subroutine nstrng (number,string)
integer number
character*7 string
```

DESCRIPTION
nstrng
Converts an integer to a character string with the proper suffix.

number input integer

string output string

COMMENTS
The number is limited by 0 < number < 32687.

NAME

qxstuf, xblank, x1blnk

SYNOPSIS

```
subroutine qxstuf (l,p,i1)
integer l
character*1 p (l)
integer i1

entry xblank (l,p)

entry x1blnk (l,p,i1)
```

DESCRIPTION

qxstuf

Sets character array to blanks and finds first nonblank in string.

l — length of character string

p — character string to be blanked

i1 — index to non blank

xblank

Sets character array to blanks.

x1blnk

Finds 1st non blank in string.

NAME
qxvalu

SYNOPSIS
```
subroutine qxvalu (l,p,i1)
integer l
character*1 p (l)
integer i1
```

DESCRIPTION
qxvalu
Converts string to integer.

l length of character string (input)

p character string to be deciphered (input)

i1 index to non blank (output)

NAME
xi2chr

SYNOPSIS
```
subroutine xi2chr (l,p)
integer l
character*12 p
```

DESCRIPTION
xi2chr
Converts integer to character.

l length of character string (output)

p character string to be blanked (input)

NAME
xnth

SYNOPSIS
```
subroutine xnth (i,nth)
integer i
character*2 nth
```

DESCRIPTION
xnth
Obtains 2 character ordinal string from an integer.

i	integer (input)
nth	ordinal string (output)

NAME

aijkq, a121q, a123q, a131q, a132q, a212q, a213q, a231q, a232q, a312q, a313q, a321q, a323q

SYNOPSIS

```
subroutine aijkq (alpha,q)
double precision alpha (3)
double precision q (4)

entry a121q (alpha,q)

entry a123q (alpha,q)

entry a131q (alpha,q)

entry a132q (alpha,q)

entry a212q (alpha,q)

entry a213q (alpha,q)

entry a231q (alpha,q)

entry a232q (alpha,q)

entry a312q (alpha,q)

entry a313q (alpha,q)

entry a321q (alpha,q)

entry a323q (alpha,q)
```

DESCRIPTION

aijkq

Converts euler angles to quaternion elements.

alpha euler angles in sequence.1=roll,2=pitch,3=yaw (input)

q quaternion elements generated from the euler sequence (output)

a121q

Performs a roll-pitch-roll to quaternion operation..

a123q

Performs a roll-pitch-yaw to quaternion operation.

a131q

Performs a roll-yaw-roll to quaternion operation.

a132q

Performs a roll-yaw-pitch to quaternion operation.

a212q

Performs a pitch-roll-pitch to quaternion operation.

a213q

Performs a pitch-roll-yaw to quaternion operation.

a231q
Performs a pitch-yaw-roll to quaternion operation.

a232q
Performs a pitch-yaw-pitch to quaternion operation.

a312q
Performs a yaw-roll-pitch to quaternion operation.

a313q
Performs a yaw-roll-yaw to quaternion operation.

a321q
Performs a yaw-pitch-roll to quaternion operation.

a323q
Performs a yaw-pitch-yaw to quaternion operation.

NAME

c2shl

SYNOPSIS

```
subroutine c2shl (x,y,s)
double precision x (6)
double precision y (6)
double precision s (6)
```

DESCRIPTION

c2shl

Obtains modified shell coordinates from two cartesian vectors.

x reference vehicle state (input)

y target vehicle state (input)

s modified shell coordinates in order of x,y,z.xdot,ydot,zdot (output)

NAME
cart

SYNOPSIS
```
subroutine cart (mu,c,x)
double precision mu
double precision c (6)
double precision x (6)
```

DESCRIPTION
cart
Converts clasical elements to cartesian elements.

mu gravitational parameter (input)

c classical elements in following order
1. semimajor axis or semilatus rectum (if parabolic)
2. eccentricity
3. inclination
4. ascending node
5. arguement of perigee
6. true anomaly (input)

x cartesian position and velocity (output)

COMMENTS
If eccentricity is one, the orbit is assumed to be parabolic and the first element is treated as the semilatus rectum rather than the semimajor axis.

NAME
clas

SYNOPSIS
```
subroutine clas (mu,x,c)
double precision mu
double precision c (6)
double precision x (6)
```

DESCRIPTION
clas
Converts cartesian elements into classical elements.

mu gravitational parameter (input)

c classical elements in order:
1. semimajor axis or semilatus rectum
2. eccentricity
3. inclination
4. ascendinding node
5. arguement of perigee
6. true anomaly (output)

x cartesian state (input)

NAME

euler

SYNOPSIS

```
subroutine euler (kop,n,alpha,q,rmat)
integer kop
integer n
double precision alpha (3)
double precision q (4)
double precision rmat (3,3)
```

DESCRIPTION

euler

Expresses a rotation as a sequence of euler angles, a set of quaternion elements, and a set of transformation matrix elements. Given one of these, euler generates the other two. It must be told the input type and the required euler angle sequence.

kop — the input type ...
1 = euler angles
2 = quaternion
3 = transformation matrix

n — the euler sequence (three digits), where ...
1 = roll
2 = pitch
3 = yaw
e.g. 132 = roll, yaw, pitch

alpha — the euler angles which describe the rotation. they may be either the input or an output.

q — the quaternion which describes the rotation.it may be either the input or an output

rmat — the transformation matrix which describes the rotation. it may be either the input or an output.

NAME

Frame3 - forms orthoganal frame from 3 dimensional vectors x,y,z.

SYNOPSIS

```
int Frame3(frame,x,y)
double *frame,*x,*y;
```

DESCRIPTION

Frame3

forms orthoganal frame from 3 dimensional vectors x,y,z. The X axis is along x, the Z axis along x cross y and the Y completes the triad. A 1 is returned if x and y are not perpendicular and the frame is trivialized. Else a 0 is returned. The axes vectors are stored in order X,Y,Z. Interpreted as a matrix stored by rows, the frame is thus the transformation that takes coordinates in the frame defining the vectors x and y to the coordinates in the frame they define.

NAME

lax

SYNOPSIS

```
subroutine lax (s,m,omega)
double precision s (6)
double precision m (3,3)
double precision omega (3)
```

DESCRIPTION

lax

Computes LVLH axes given a geocentric cartesian state as

s input state vector in geocentric cartesian coordinates

m rotation matrix. This rotates inertial to LVLH (output)

```
row1 = unit (V x R) x Z
row2 = Z x X
row3 = - unit (R)
```

omega velocity correction vector (Vel x Pos/(r x r))(output)

NAME
lst

SYNOPSIS
```
subroutine lst (xi,xo,lm,omega)
double precision xi (6)
double precision xo (6)
double precision lm (3,3)
double precision omega (3)
```

DESCRIPTION
lst
Computes lvlh state given inertial state and inertial to lvlh matrix.

xi input inertial state

xo output lvlh state where
PosO = lm * Posi
VelO = lm * (Veli + Posi x omega)

lm transformation matrix from inertial to lvlh coordinates

omega velocity correction vector (input)

NAME

qijka, q121a, q123a, q131a, q132a, q212a, q213a, q231a, q232a, q312a, q313a, q321a, q323a

SYNOPSIS

```
subroutine qijka (q,alpha)
double precision q (4)
double precision alpha (3)

entry q121a (q,alpha)

entry q123a (q,alpha)

entry q131a (q,alpha)

entry q132a (q,alpha)

entry q212a (q,alpha)

entry q213a (q,alpha)

entry q231a (q,alpha)

entry q232a (q,alpha)

entry q312a (q,alpha)

entry q313a (q,alpha)

entry q321a (q,alpha)

entry q323a (q,alpha)
```

DESCRIPTION

qijka

Converts quaternion elements into euler angle elements.

q quaternion (input)

alpha euler angles (output) 1=roll,2=pitch,3=yaw generated from the quaternion

q121a

Performs a roll-pitch-roll operation.

q123a

Performs a roll-pitch-yaw operation.

q131a

Performs a roll-yaw-roll operation.

q132a

Performs a roll-yaw-pitch operation.

q212a

Performs a pitch-roll-pitch operation.

q213a

Performs a pitch-roll-yaw operation.

q231a
Performs a pitch-yaw-roll operation.

q232a
Performs a pitch-yaw-pitch operation.

q312a
Performs a yaw-roll-pitch operation.

q313a
Performs a yaw-roll-yaw operation.

q321a
Performs a yaw-pitch-roll operation.

q323a
Performs a yaw-pitch-yaw operation.

NAME

rst

SYNOPSIS

```
subroutine rst (x1,x2,xb)
double precision x1 (6)
double precision x2 (6)
double precision xb (24)
```

DESCRIPTION

rst

Computes relative states. Output buffer is x1, x1-x2, x2, x2-x1.

x1 state one (input)

x2 state two (input)

xb state buffer (output)

NAME
uax

SYNOPSIS
```
subroutine uax (s,m)
double precision s (6)
double precision m (3,3)
```

DESCRIPTION
uax
Computes u,v,w axes, given a geocentric cartesian state.

s input state vector in geocentric cartesian coordinates

m rotation matrix. This rotates cartesis to UVW (output)
row1 = unit (Pos)
row2 = W x U
row3 = unit (Pos x Vel)

NAME

ust

SYNOPSIS

```
subroutine ust (xi,xo,um)
double precision xi (6)
double precision xo (6)
double precision um (3,3)
```

DESCRIPTION

ust

Computes uvw state given relative inertial state and inertial to uvw matrix.

xi inertial state (input)

xo uvw state (output)

um transformation matrix from inertial to uvw coordinates (input)

NAME

UVW_Cart - UVW coordinates

SYNOPSIS

```
double *UVW_Cart(UVW,Xtrgt,Xbase)
double *UVW,*Xtrgt,*Xbase;
```

DESCRIPTION

UVW_Cart

Returns pointer to UVW position and velocity of target with respect to a base state (position, velocity). If UVW is null, a local buffer is used for the coordinates, otherwise the space indicated by UVW is used.

NAME

uvwlvh

SYNOPSIS

```
subroutine uvwlvh (xu,w,xl)
double precision xu (6)
double precision w
double precision xl (6)
```

DESCRIPTION

uvwlvh

Computes lvlh coordinates from uvw info.

xu uvw coordinates (input)

w angular rate of local vertical frame (input)

xl lvlh state (output)

NAME
uvwmat

SYNOPSIS
```
subroutine uvwmat (x,m,r,w)
double precision x (6)
double precision m (3,3)
double precision r
double precision w
```

DESCRIPTION
uvwmat
Computes uvw to inertial matrix.

x inertial state (input)

m uvw to inertial matrix (output)
col1 = unit(Pos)
col2 = col1 x col3
col3 = unit(Pos x V)

r radius of vehicle(input)

w angular rate of local vertical frame norm(Pos x Vel)/(r * r) (output)

NAME
dfdata

SYNOPSIS
block data dfdata

DESCRIPTION
dfdata
Intializes common blocks.

NAME
dfnput

SYNOPSIS
```
subroutine dfnput (status)
integer status
```

DESCRIPTION
dfnput
processes the inputs

status the input processing error flag

NAME
dwnfmt

SYNOPSIS
program dwnfmt

DESCRIPTION
dwnfmt
Drives the download processing.

dwnfmt < Input_file > Output_file

Files BUGS and OUTPUT are also created at execution level to contain debug and additional summary print.

NAME

eb2asc, getbits - routine to convert ebcdic to ascii

SYNOPSIS

```
eb2asc(strng1,strng2,m)
int *strng1;
int *strng2;
int *m;

getbits(w,s,n)
int n;
int s;
unsigned w;
```

DESCRIPTION

eb2asc

C routine to convert ebcdic to ascii

strng1 string of characters to be convert (input)

strng2 string of characters (output)

m number of characters to be convert (input)

getbits

return (right adjusted) the n-bit field of w that begins at position s

n number of bits (input)

s begin postion (input)

w working buffer (output)

NAME

genout

SYNOPSIS

```
subroutine genout (file)
integer file
```

DESCRIPTION

genout

Builds the nominal file frames and record buffers and directs the output file construction.

file the output file designation used for looping

COMMENTS

1. All data frames to all output files depend on the organization of the data ids.

2. The data is expected to be contained on the CCT in either 36 bit or 72 bit word lengths.

3. Only data for which the CCT address table has non-zero entries will be processed.

NAME

getdat

SYNOPSIS

```
subroutine getdat (recfnd)
integer recfnd
```

DESCRIPTION

getdat

Obtains parameters for orbiter CCT downlist tape and calls a subroutine to reformat intc the internal files.

recfnd flag indicationg data scan records found(output)

COMMENTS

1. The subroutine gethdr will have already found the first record of the first data scan so that the cct is currently positioned at the beginning of the second record.

2. The begin scan will be determined by the input timbeg which will be compared to the time-tag values found in each scan in order to find the scan from which data processing may begin.

3. The end scan will be determined by the input timend which will be compared to the time-tag values found in each scan in order to find the scan in which data processing will end.

NAME
gethdr

SYNOPSIS
```
subroutine gethdr (recfnd)
integer recfnd
```

DESCRIPTION
gethdr
Reads the headers off of the downlist CCT to obtain information on the msid's to be processed.

recfnd the record status flag

NAME

getout, opnout, nitout, bldout, delout, wrtout, clsout

SYNOPSIS

```
subroutine getout (datid,file,fnam,frmid,frmsta,hdrid,
 recsta,unitid,wrdfrm)
double precision dtim
integer datid
integer file
character fnam
character*4 frmid
integer frmsta
character hdrid
integer recsta
integer unitid
integer wrdfrm

entry opnout (file,fnam,hdrid,wrdfrm,recsta)

entry nitout (frmid,file)

entry bldout (datid,file,frmsta)

entry delout (datid,file)

entry wrtout (file,recsta)

entry clsout (file,recsta)
```

DESCRIPTION

getout

Builds frames, deletes frames, collects frames into records and writes records to file.

dtim data scan time

datid loop counter on the data id relative current file

file the output file designation

fnam the character file name of the output file

frmid dummy variable

frmsta the frame status flag

hdrid the character type of output file

recsta the record status flag

unitid the unit number associated with current file

wrdfrm the number of words per frame excluding the 4 header words

opnout

Opens all output files and writes header records. Stores copy of i/o packet for each output file.

nitout

Initializes frame buffer.

bldout

Builds frame buffer.

delout
Deletes current frame locates next valid entry in data sequence.

wrtout
Writes frame to record.

clsout
Closes output files.

COMMENTS

1. All data frames to all output files depend on the organization of the data ids.

2. The data is expected to be contained on the CCT in either 36 bit or 72 bit word lengths.

3. Only data for which the CCT address table has non-zero entries will be processed.

NAME

ibmcvt, fetchbits - routine to convert IBM DP data to HP DP data

SYNOPSIS

```
ibmcvt(buf,n,type)
int *buf;
int *n;
int *type;

fetchbits(x,p,k)
unsigned x;
int p;
int k;
```

DESCRIPTION

ibmcvt

C routine to convert IBM DP data to HP DP data

buf	array containing number to convert
n	no. entries in array
type	type of conversion 0 - sp ; 1 - dp

fetchbits

function to get k-bits from word(right adjustified)

x	working location(output)
p	begin position (input)
k	number of bits (input)

NAME

match

SYNOPSIS

```
subroutine match (msfnd)
integer msfnd
```

DESCRIPTION

match

Matches msid's from the current header record to the namelist inputs associated with desired output file. For each matched msid fill in a table indexed by internal ordering to provide the number of data samples, address relative the data scan of the first sample in the scan.

msfnd current count of matched m/sid's

NAME

obtfil, clsfil

SYNOPSIS

```
subroutine obtfil

entry clsfil
```

DESCRIPTION

obtfil

Drives the process which will accumulate the data for output frames and write them to files.

clsfil

Closes all files.

COMMENTS

1. All data frames to all output files depend on the organization of the data ids.

2. The data is expected to be contained on the CCT in either 36 bit or 72 bit word lengths.

3. Only data for which the CCT address table has non-zero entries will be processed.

NAME

taperd - buffer input routine for tape

SYNOPSIS

```
taperd(buf,n,unit)
int *buf;
int *n;
int *unit;
```

DESCRIPTION

taperd

read tape info routine

buf	buffer(output)
n	number of words(input)
unit	tape drive unit(input)

NAME
valdat

SYNOPSIS
```
subroutine valdat (relcyc,pointr)
integer pointr
integer relcyc
```

DESCRIPTION
valdat
Finds data associated with various msid's at addresses specified in the table constructed in match. Converts data to internal units and constructs table.

pointr the pointer to the current scan in nbuf

relcyc the relative cycle number currently being processed

COMMENTS
1. All data frames to all output files depend on the organization of the data ids.

2. The data is expected to be contained on the CCT in either 36 bit or 72 bit word lengths.

3. Only data for which the CCT address table has non-zero entries will be processed.

NAME

ndhead

SYNOPSIS

```
subroutine ndhead (u,ir,t,x,e)
integer u
integer ir
double precision t
character*4 x
integer e
```

DESCRIPTION

ndhead

Prompts input of format page layout.

u	display unit
ir	current record (output)
t	time tag (input/output)
x	frame id (output)
e	edit flag (output)

NAME

ndnflz, ndfnpt, ndfnit

SYNOPSIS

```
subroutine ndnflz (nf,stat,tbegin,tend,t,dt)
integer nf
integer stat
double precision tbegin
double precision tend
double precision t
double precision dt

entry ndfnpt (nf,stat)

entry ndfnit (nf,tbegin,tend,dt,t,stat)
```

DESCRIPTION

ndnflz

Sets up input files for numeric display.

nf	number of files needed (input)
stat	status flag - set negative if error encountered (output)
tbegin	begin time (input)
tend	end time (input)
t	time tag (input)
dt	time step (input)

ndfnpt

Sets up input files for numeric display.

ndfnit

Inititializes input files to begin time and sets base file options.

COMMENTS

File 3 is assumed to be an attitude file.

NAME
ndpage

SYNOPSIS
subroutine ndpage

DESCRIPTION
ndpage
Checks if new page is needed, if so then new page number written.

NAME

nxtnd

SYNOPSIS

```
subroutine nxtnd (t,dt,nstep,tend,done)
double precision t
double precision dt
integer nstep
double precision tend
integer done
```

DESCRIPTION

nxtnd

Sets next output time either from a base file (bfopt non-zero), or by specified time step dt (bfopt=0), done is set to 1 if past end time tend, else set to 0. If nstep > 0, then edit status is not checked. If bfopt > 0, then base file is position to next unedited point.

t next time (input/output)

dt step size (input)

nstep step for running off base file (input)

tend stop time (input)

done done flag (output)

COMMENTS

Base file data must be less than 100 words.

NAME

xf8dmp

SYNOPSIS

```
subroutine xf8dmp (pfmt,u,iop,ir)
integer pfmt
integer u
integer iop (40)
integer ir
```

DESCRIPTION

xf8dmp

Provides octal dump of standard file format record in 80 or 130 column format.

pfmt format flag: - means 130 col, else 80 (input)

u display unit (input)

iop i/o packet (input/output)

ir current record (input)

NAME

xqwget, xqwint

SYNOPSIS

```
subroutine xqwget (t,x1,x2,q,w,s)
double precision t
double precision x1 (6)
double precision x2 (6)
double precision q (4)
double precision w (3)
integer s

entry xqwint
```

DESCRIPTION

xqwget

Obtains trajectory and attitude information from files. Entry xqwint initialize time buffer and should be called upon entry.

t desired time (input)

x1 first state (output)

x2 2nd state (output)

q attitude quaternion (output)

w angular velocity (output)

s status word: set to -1 if error encountered (output)

xqwint

Initializes time buffer.

NAME

main - generate a tape which can be read by the fich reader

SYNOPSIS

```
void main(argc,argv)
int argc;
char *argv[];
```

DESCRIPTION

main

generate a tape which can be read by the fich reader. The format of the tape is 9-track, 1600 bpi, ASCII, 132 character fixed length records. The program fiche can generate a multi-reel file. However, the fiche processor will treat each tape as an individual file. It requires three user inputs which are put on the command line.

usage : fiche file run_id tape_drive where
file - name of print file to put on the tape
runid - Univac type runid 660???
tape_drive - 0 or 1

NAME

center

SYNOPSIS

```
subroutine center (iop,tget,t,x,n,nrec,status)
integer iop (*)
double precision tget
double precision t (2)
double precision x (n,2)
integer n
integer nrec
integer status
```

DESCRIPTION

center

Returns data bracketing a given time.

iop	i/o packet (input/output).
tget	given time (input).
t	bracketing times (input/output).
x	data associated with t (input/output).
n	number of double precision data words per frame (input).
nrec	record number (input/output).
status	i/o status flag (output).

NAME

get

SYNOPSIS

```
subroutine get (iop,fintrp,tget,t,x,status)
integer iop (*)
double precision tget
double precision t (2)
double precision x (*)
integer status
```

DESCRIPTION

get

Interpolates data to desired time using routine fintrp. The interpolation routine must be of the form:

```
fintrp(t1,x1,t2,x2,t,x)
```

where

```
t1 = first time
x1 = data associated with t1
t2 = second time
x2 = data associated with t2
t  = desired time
x  = interpolated data at t (output)
```

iop i/o packet (input/output)/

tget time tag of desired pos.,vel. (input).

t bracketing time tags used (output).

x data interpolated to desired time (output).

status i/o status (output).

NAME

obread

SYNOPSIS

```
subroutine obread (iop,nrec,t,id,ed,obias,obs,res,status)
integer iop (*)
integer nrec
double precision t
character id
integer ed
double precision obias
double precision obs
double precision res
integer status
```

DESCRIPTION

obread

RELBET read interface to GFF routines.

iop	i/o packet (input/output)
nrec	desired record number (input)
t	time tag (output)
id	frame id (output)
ed	edit status (output)
obias	observation bias (output)
obs	observation (output)
res	residual (output)
status	i/o status (output)

NAME

obwrit

SYNOPSIS

```
subroutine obwrit (iop,nrec,t,id,ed,obias,obs,res,status)
integer iop (*)
integer nrec
double precision t
character id
integer ed
double precision obias
double precision obs
double precision res
integer status
```

DESCRIPTION

obwrit

RELBET write interface to GFF routines.

iop	i/o packet (input/output)
nrec	desired record number (input)
t	time tag (input)
id	frame id (input)
ed	edit status (input)
obias	observation bias (input)
obs	observation (input)
res	residual (input)
status	i/o status (output)

NAME
rlcls

SYNOPSIS
```
subroutine rlcls (iop,status)
integer iop (*)
integer status
```

DESCRIPTION
rlcls
Closes GFF file of standard RELBET type.

iop i/o packet (input/output).

status i/o status (output)

NAME

rlcntr

SYNOPSIS

```
subroutine rlcntr (iop,tm,n,rec,t,e,stat)
integer iop (*)
double precision tm
integer n
integer rec (n)
double precision t (n)
double precision e (*)
integer stat
```


DESCRIPTION

rlcntr

Centers array of unedited points from RELBET GFF about a desired time tm. The times and records of the points are returned. Warning messages are issued if the array cannot be centered because an end of file is encountered. Error Messages are issued if not enough points are found. The routine assumes that the times have been initialized on the first call so that the last time in the array is less than the first time, or was initialized by first calling rlmid1.

iop — i/o package for file (input)

tm — desired time (input)

n — number of frames in output array(input)

rec — records corresponding to entries in e (input/output)

t — array of time tags for entries (input/output)

e — array buffer for entries (input/output)

stat — status flag; >=0 is good, -1 is eof, e is error (output)

NAME

rldsp, rlpdsp, rlhdsp

SYNOPSIS

```
subroutine rldsp (p,u)
integer p (45)
integer u

entry rlpdsp (p,u)

entry rlhdsp (p,u)
```

DESCRIPTION

rldsp

Displays input/output packet to specified unit.

p i/o packet (input)

u display unit number (input)

rlpdsp

Displays i/o packet info such as unit, type, name, size and time span.

rlhdsp

Displays i/o packet information such as unit, type, name, header, time span, size, creation and update dates.

NAME

rledit, rlfwds, rlbwds

SYNOPSIS

```
subroutine rledit (iop,rec,t,fid,e,stat)
integer iop (*)
integer rec
double precision t
character fid
double precision e (*)
integer stat

entry rlfwds (iop,rec,t,fid,e,stat)

entry rlbwds (iop,rec,t,fid,e,stat)
```

DESCRIPTION

rledit

Entries to obtain next unedited frame. Note that status: -1 for end of file +1 edit value for okay < -10 for error

iop i/o package for file (input)

rec records corresponding to entries in e (input/output)

t array of time tags for entries (input/output)

fid frame id (input/output)

e array buffer for entries (input/output)

stat status flag: >=0 is good, -1 is eof, e is error (output)

rlfwds

Reads forward to the next frame or the end of file.

rlbwds

Reads backward to the next frame or the begining.

NAME

rlfill, rl2end, rlback, rl2beg, rlfrnt

SYNOPSIS

```
subroutine rlfill (iop,n,iep,fnd,rec,t,e,stat)
double precision e (*)
integer fnd
integer iep
integer iop (*)
integer n
integer rec (n)
integer stat
double precision t (n)

entry rl2end (iop,n,iep,fnd,rec,t,e,stat)

entry rlback (iop,n,fnd,rec,t,e,stat)

entry rl2beg (iop,n,iep,fnd,rec,t,e,stat)

entry rlfrnt (iop,n,fnd,rec,t,e,stat)
```

DESCRIPTION

rlfill

Entry points for filling up frame arrays of unedited points: rlback and rl2end attempt to fill entries in array after specified index. If unsuccessful rlfrnt and rlback error exit. If unsuccessful rl2end and rl2beg move the array all the way to the end or front respectively and reset the index iep to the new start or end index note that the number of length of each entry in the array must be the same as the frame length or an error will occur. Index use and start is as follows: Input rl2end: iep=start of data, filling starts at iep+fnd+1 rl2beg: iep=current 1st index, filling starts at iep-1 output rl2end iep=current 1st index, filled from iep to n rl2beg iep=current last index, filled from 1 to iep rlfrnt filled from n to n-fnd rlback filled from 1 to fnd

e — array buffer for entries (input/output)

fnd — number of points found (input/output)

iep — pointer to end of e (input)

iop — i/o package for file (input)

n — number of entries to be found (input)

rec — records corresponding to entries in e (input/output)

stat — status flag: >=0 is good, -1 is eof, e is error (output)

t — array of time tags for entries (input/output)

rl2end

Attempts to fill entries in array from iep to n.

rlback

Attempts to fill entries in array at fnd+1.

rl2beg

Attempts to fill entries in array from 1 to iep.

rlfrnt
Attempts to fill entries in array from n to n-fnd

NAME

rlmid1

SYNOPSIS

```
subroutine rlmid1 (iop,tm,n,fnd,rec,t,e,stat)
integer iop (*)
double precision tm
integer n
integer fnd
integer rec (n)
double precision t (n)
double precision e (*)
integer stat
```

DESCRIPTION

rlmid1

Centers an array of frames about a desired time from scratch.

iop	i/o package for file (input)
tm	desired time (input)
n	number of entries to be found(input)
fnd	number of points found(input/output)
rec	records corresponding to entries in e(input/output)
t	array of time tags for entries(input/output)
e	array buffer for entries (input/output)
stat	status flag: >=0 is good, -1 is eof, e is error(output)

NAME

rlnew

SYNOPSIS

```
subroutine rlnew (iop,name,type,unt,ndata,status)
integer iop (*)
character name
character type
integer unt
integer ndata
integer status
```

DESCRIPTION

rlnew

Opens a new GFF file in accordance with the standard RELBET usage.

iop i/o packet (output).

name file name (input).

type file type (input).

unt unit number (input).

ndata number of double precision data items in frame (input).

status i/o status (output).

NAME
rlopen

SYNOPSIS
```
subroutine rlopen (iop,name,unt,uze,status)
integer iop (*)
character name
integer unt
integer uze
integer status
```

DESCRIPTION
rlopen
Opens a old GFF file in accordance with the standard RELBET usage.

iop i/o packet (output).

name file name (input).

unt unit number (input).

uze use flag (input).

status i/o status (output).

NAME

rlread

SYNOPSIS

```
subroutine rlread (iop,nrec,t,id,ed,data,status)
integer iop (*)
integer nrec
double precision t
character id
integer ed
double precision data (*)
integer status
```

DESCRIPTION

rlread

RELBET read from GFF files.

iop i/o packet (input/output).

nrec desired record number (input).

t time tag (output).

id frame id (output).

ed edit status (output).

data output data from frame (output).

status i/o status (output).

NAME
rltime

SYNOPSIS
```
subroutine rltime (iop,t,nrec,status)
integer iop (*)
double precision t
integer nrec
integer status
```

DESCRIPTION
rltime
RELBET interface to GFF routine gftime. Returns record directly before first record or end of file and big time tag (+- 1e30).

iop i/o packet (input/output).

t time tag (output).

nrec desired record number (input).

status i/o status (output).

NAME

rlwrit

SYNOPSIS

```
subroutine rlwrit (iop,nrec,t,id,ed,data,status)
integer iop (*)
integer nrec
double precision t
character id
integer ed
double precision data (*)
integer status
```

DESCRIPTION

rlwrit

RELBET write to GFF files

iop i/o packet (input/output).

nrec desired record number (input).

t time tag (input).

id frame id (input).

ed edit status (input).

data data from frame (input).

status i/o status (output).

COMMENTS

Differs from gfwrit in that writing past the end of the file is not an error, even if append mode is not specified and cannot write to header.

NAME
cvprop

SYNOPSIS
```
subroutine cvprop (told,xold,tnew,xnew,cov)
double precision told
double precision xold (nsol)
double precision tnew
double precision xnew (nsol)
double precision cov (nsol,nsol)
```

DESCRIPTION
predict covariance

cvprop

told	old time (input).
xold	old state (input).
tnew	new time (input).
xnew	new state (input).
cov	covariance (input/output).

NAME
dkal

SYNOPSIS
block data dkal

DESCRIPTION
dkal
This block data routine initializes the common used by this program

NAME
dsmth

SYNOPSIS
block data dsmth

DESCRIPTION
dsmth
This block data routine initializes the common used by this program

NAME

getnxt

SYNOPSIS

```
subroutine getnxt (tnxt,pob,ob,obtyp,obsig,pstop)
double precision ob
character*4 obtyp
double precision obsig
double precision tnxt
```

DESCRIPTION

getnxt

get next time for sequential filter.

ob observation (output).

obtyp observation type (output).

obsig observation sigma (output). flag to tell of observation time (output). flag to tell if stop time (output).

tnxt time of observation (output).

NAME
kalman

SYNOPSIS
```
subroutine kalman (t,x,c,ob,y,dydx,v,edcrit,r,redit,uwterm)
double precision t
double precision x (nsol)
double precision c (nsol,nsol)
double precision ob
double precision y
double precision dydx (nsol)
double precision v
double precision edcrit
double precision r
double precision uwterm
```

DESCRIPTION
kalman
kalman filter.

t current time (input).

x state vector (input).

c covariance (input/output).

ob true observation (input).

y computed observation (input).

dydx partials of observation w.r.t. state (input).

v residual variance (input).

edcrit edit criterion (input).

r residual (output). residual edit flag (output).

uwterm the uderweighting term

NAME
krelo, kreloc

SYNOPSIS
```
subroutine krelo (t,x)
double precision t
double precision x (12)

entry kreloc
```

DESCRIPTION

krelo

kalman relative filter traj output routine for sequential filter

t filter time (input).

x filter state (input).

kreloc

close output file

NAME

invrs

SYNOPSIS

```
subroutine invrs (n,a,ai,eps)
integer n
double precision a(n,n)
double precision ai(n,n)
double precision eps
```

DESCRIPTION

invrs

matrix convert routine for smooth

n dimension of matrix

a matrix to be inverted (input).

ai(n,n) inverted matrix (output).

eps threshold for singularity (input).

NAME
nkal

SYNOPSIS
subroutine nkal

DESCRIPTION
nkal
This subroutine fetches the user inputs for the common blocks needed by this program

NAME
nsmth

SYNOPSIS
subroutine nsmth

DESCRIPTION
nsmth
This subroutine fetches the user inputs for the common blocks needed by this program

NAME

obest

SYNOPSIS

```
subroutine obest (t,x,obtyp,obsig,y,dydx,v,e,nob)
double precision e
double precision t
double precision x (nsol)
character*4 obtyp
double precision obsig
double precision y
double precision dydx (nsol)
double precision v
integer nob
```

DESCRIPTION

obest

observation estimate for sequential filter.

e	edit criterion (output)
t	time (input).
x	state (input).
obtyp	observation type (input).
obsig	observation sigma (input).
y	estimated observation (output).
dydx	partials of observations w.r.t. state (output).
v	variance of observation (output).
nob	observation number.

NAME

obout. obcls

SYNOPSIS

```
subroutine obout (t,obtyp,redit,ob,r,v,b,bv)
double precision t
character*4 obtyp
double precision ob
double precision r
double precision v
double precision b
double precision bv

entry obcls
```

DESCRIPTION

obout

output routine for sequential filter

t filter time (input).

obtyp frame id of current obs edit status of observation

ob observation valu

r residual value

v value of residual variance

b solution bias

bv bias variance

obcls

close output file

NAME

pcpt

SYNOPSIS

```
subroutine pcpt (p,c)
double precision p (nsol,nsol)
double precision c (nsol,nsol)
```

DESCRIPTION

pcpt

perform operation phi*c*phitranspose.

p state transition matrix (input).

c covaraince (input/output).

NAME

pred

SYNOPSIS

```
subroutine pred (t,x,c,tnxt)
double precision t
double precision x (nsol)
double precision c (nsol,nsol)
double precision tnxt
```

DESCRIPTION

pred

predict state, covariance at next time for sequential filter.

t old time (input/output).

x state (input/output).

c covariance (input/output).

tnxt next time (input).

NAME

psout, pscls

SYNOPSIS

```
subroutine psout (t,x)
double precision t
double precision x (12)

entry pscls
```

DESCRIPTION

psout

output routine for sequential filter

t filter time (input).

x filter state (input).

pscls

close output file

NAME

ptbl

SYNOPSIS

```
integer function ptbl (t,tbl)
double precision t
double precision tbl (*)
```

DESCRIPTION

ptbl

Returns a pointer to a time increasing table to the entry with the maximum time tag at or before a given time. Sentinal time tags are expected at the beginning and end of the table, less than and greater than, respectively, any expected time tag.

t desired time (input).

tbl time increasing table (input).

NAME
sfilt

SYNOPSIS
program sfilt

DESCRIPTION
sfilt
RELBET Kalman filter driver

NAME

sfinit

SYNOPSIS

```
subroutine sfinit (t,x,c)
double precision t
double precision x (nsol)
double precision c (nsol,nsol)
```

DESCRIPTION

sfinit

initialize sequential filter

t initial time (output).

x initial state (output).

c initial covariance (output).

NAME

sfout

SYNOPSIS

```
subroutine sfout (t,x,c,pstop)
double precision t
double precision x (nsol)
double precision c (nsol,nsol)
```

DESCRIPTION

sfout

output routine for sequential filter

t filter time (input).

x filter state (input).

c covariance matrix (input). stop flag (input).

NAME

smooth

SYNOPSIS

```
subroutine smooth (infil,outfil,cpfreq,cnrec)
character*72 infil
character*72 outfil
character*72 cpfreq
character*72 cnrec
```

DESCRIPTION

smooth

relbet smoother

infil input file name

outfil output file name

cpfreq print frequency

cnrec number of records

NAME

snoise

SYNOPSIS

```
subroutine snoise (told,tnew,c)
double precision told
double precision tnew
double precision c (nsol,nsol)
```

DESCRIPTION

snoise

add state noise to covariance for relbet sequential filter

told last time (input).

tnew new time (input).

c covariance (input/output).

NAME

stm

SYNOPSIS

```
subroutine stm (phi,told,xold,tnew,xnew)
double precision phi (nsol,nsol)
double precision told
double precision xold (nsol)
double precision tnew
double precision xnew (nsol)
```

DESCRIPTION

stm

state transition matrix

phi state transition matrix (output).

told begin time (input).

xold state at old time (input).

tnew end time (input).

xnew state at new time (input).

NAME

stmrv

SYNOPSIS

```
subroutine stmrv (phi,dt,xold,xnew)
double precision dt
double precision phi (6,6)
double precision xnew (6)
double precision xold (6)
```

DESCRIPTION

stmrv

state transition matrix for position and velocity in orbital motion

dt delta time (input).

phi state transition matrix (output).

xnew next position and velocity (input).

xold last position and velocity (input).

NAME

stprop

SYNOPSIS

```
subroutine stprop (told,t,x)
double precision told
double precision t
double precision x (nsol)
```

DESCRIPTION

stprop

state propagation

told old time (input).

t new time (input).

x new state (input/output).

NAME
uvw2m

SYNOPSIS
```
subroutine uvw2m (c)
double precision c (nsol,nsol)
```

DESCRIPTION
uvw2m
this routine rotates uvw coordinates to m50

```
c       the full solution state covariance
```

NAME
vlrest

SYNOPSIS
```
subroutine vlrest (t,obtyp,ob,v)
double precision t
character*4 obtyp
double precision ob
double precision v
```

DESCRIPTION
vlrest
lrbet3 variance estimate for sequential filter. for range, range rate, and range bias

t time (input).

obtyp observation type (input).

ob the actual observation of range (input)

v variance of observation (output).

NAME

main, cmp2sg - compare relative difference to computed sigmas

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

char *cmp2sg(frame1,frame2a,frame2b)
char *frame1;
char *frame2a;
char *frame2b;
```

DESCRIPTION

main

cmp2sg reads in two files including a relative trajectory difference file and a file containing the computed sigmas associated with some filtering process in the same frame format as a relative trajectory file

-r__ indicates file name of relative traj file
-c__ indicates file name of sigma data file
-t__ is the threshold of comparison (default is 3 indicating that 3-sigma is a good comparison)

NOTE: Output text files are CMP_POS for position info and CMP_VEL for velocity info

cmp2sg

frame1 the current frame from Relative difference file 1

frame2a the frame from relative sigma file 2 <= time of frame1

frame2b the frame from relative sigma file 2 > time of frame1

NAME
drpst

SYNOPSIS
block data drpst

DESCRIPTION
drpst
This block data routine initializes the common used by this program

NAME

main, eph2rel - reads two ephemeris files and creates relative trajectory file as output

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

char *eph2rel(frame1,frame2a,frame2b)
double *frame1;
double *frame2a;
double *frame2b;
```

DESCRIPTION

main

eph2rel reads two ephemeris files and creates a relative trajectory file output. The first ephemeris_file is the base file name. The second ephemeris_file2 is the file to be interpolated for the relative portion of the frame. The output relative trajectory file is assigned the name of ephemeris_file_REL.

NAME

main, text_in, gff_in, ref_in, edit_frame - gff file editor

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *text_in(inframe,infile)
char *inframe;
char *infile;

char *gff_in(inframe,reference,end)
char *inframe;
char *reference;
char *end;

char *ref_in(reference,infile)
char *reference;
char *infile;

char *edit_frame(inframe,reference)
char *inframe;
char *reference;
```

DESCRIPTION

the following parameters are set by command line arguments

main

Filedit reads in one of two edit sources to be used to edit a specified file depending upon certain program dependent criterion.

Multiple frame ids can be specified to indicate the frames to be edited .

USAGE:

filedit <-a__><-c><-t__><-o__><-p__><-f__><-g__><-h__><-i__><-j__>

where

-a__ Provides the edit value to be applied.
Zero is the same as default.
Negative input indicates editted frames are to be uneditted.
Positive input indicates uneditted frames are to be editted.
When used with -c indicates that only frames bearing the specified edit value are effected.
Default is 999 .

-t__ Indicates file name wherein edit intervals are specified by two text fields :
The first field should be begin time
The second field is an end time of edit interval
Frames associated with specified (<-p__>) frame_ids are editted within text input time intervals
Default is to process whole file

-c Indicates that the edit process will change the edit status of all frames

possessing the desired edit value input
using the <-a> option or defaulted to 999.
For example in the default case frames
with 999 receive -999 if the proper edit
criterion are met. In this usage
negative values may be used with <-a>.
The object is to undo a previous edit
operation or redo a previous operation.
This option can be used with both <-t>
and <-f> options and has the same purpose

-o__ indicates file name of gff file to be edited
If unused or same as the reference file name
then the reference file is to be edited.

-p__ indicates list of frame_ids to be effected
on the file to be editted

-f__ indicates file name of reference gff file
whose frames are used in edit criterion
of input file frames.
If same as the file to be edited file name
then the reference file is to be edited.

-h__ indicates criterion for edit to be
determined from frames on the reference file
necessary only if -i__ option specified
in which case a double precision value
is expected which is treated as a sort
of threshold to be interperted as
ignored if -i option not used

-i__ indicates frame pos source of edit criterion
for frames on the reference file.
if not specified edit status portion assumed
and target frames will be editted if the
edit status is set to edit
of the specified frame of input file
integer input expected which indicates double
precision location on data portion of frame

-g__ indicates frame_id source of edit criterion
for frames on the reference file
(only one allowed).
Ignored if reference file is to be edited.

-j__ indicates the type of test used to compare
input threshold with reference file frame
information as criterion for edit,
=0: abs(frame info) >= abs(threshold)
>0: frame info >= threshold
<0: frame info <= threshold
ignored if -i option not used
NOTE: if the frame being tested
was a valid frame before the test
then the results of the test will
change the value of the edit status
flag in some cases even if the result
shows valid data, however in this
case the frame will still be valid.

gff_in
inframe

current frame from file 1 the file to be editted

reference the comparison frame from file 2 <= inframe

end not used

this function edits the input frame on the basis of the frame from the reference file which has the greatest time-tag <= to the input frame time_tag and returns the output frame .

ref_in
reference the input frame from the reference file

infile unused

this function edits the reference frame based on info in the reference frame and returns the output frame

edit_frame
inframe
input frame to be editted

reference the reference frame

this function examines the reference frame to determine the edit status of the input frame and returns the output frame

NAME

main, gbfcom - merge files

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

void gbfcom(out_name,base_name,merge_name)
char *out_name,*base_name,*merge_name;
```

DESCRIPTION

main

gbfcom reads two files and performs a merge using the first input file as a base file and the second file as a merge file ie., if two frames with identical frame ids occur at the same time then the base file frame is discarded and the merge file frame used the new file is created in the first argument

Usage: gbfcom out in1 in2

NAME

main, rel_in, eph2_in, mkinit - creates text file from two traj files or a rel traj file

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

char *rel_in(frame1,p_char)
char *frame1;
char *p_char;

char *eph2_in(frame1,frame2a,frame2b)
double *frame1;
double *frame2a;
double *frame2b;

char *mkinit(time,base,target)
double *time,*base,*target;
```

DESCRIPTION

main

mkinit reads two trajectory files or one relative trajectory file and writes out text to be used for init linput input

NAME
nrpst

SYNOPSIS
subroutine nrpst

DESCRIPTION
nrpst
This subroutine fetches the user inputs for the common blocks needed by this program

NAME

main, combine - obsnois merges the obs data with the noise data into one file

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *combine(frame,frame2,frame3)
char *frame;
char *frame2;
char *frame3;
```

DESCRIPTION

main

obsnois merges observation data and noise data into one file. The noise data is stored in the first location of obs data. Usage:

```
obsnois obsfile noisefile
```

The output file is a binary file with _ON attached to the first input file (obsfile). If any of the input files is unavailable, an error message is generated.

Note: User needs to run obsnois repeatly for each frame id until all of the desirable types are completed. The output file is the input file for the next run. For example:
We have the following :

```
obsnois f8obs Obs_ddnois_Na
```

The output file is f8obs_ON. For the next run, we will have

```
obsnois f8obs_ON Obs_ddnois_Nb
```

Now the output file is f8obs_ON_ON.

combine

frame — the current frame

frame2 — the beginning of the bracket frame

frame3 — the end of the bracket frame

merges the obs data with the noise data. The noise data is stored in the first location of the data portion.

NAME

main, angle_rate - Generates angular acceleration magnitude text summary and gff file.

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *angle_rate(frame,ptr)
char *ptr;
char *frame;
```

DESCRIPTION

main

qaatt provides output to stdout documenting attitude files

Usage:

```
qaatt <-e><-b____>  file1,....
   where -e___ is the maximum acceptable angular acc magnitude
             (default .10)
         -b___ is the mimimum angular acc to determine an event
             (default .0005)
```

The options are parseo and the appropriate variables are set.

If a file is unavailable, an error message is generated.

qaatt creates an output gff with the angular acceleration vector and magnitude between the two input angular rates:

the frame id for this file is "attr." Also, for any point > e and any point > b, a record is written to stdout containing time,"attr","valid" or "edit", duration ,and magnitude.

angle_rate

computes angular rate and documents

ptr character pointer,not used

frame current frame of data

NAME

main, cover_file - Generates a coverage history summary

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *cover_file(frame,ptr)
char *ptr;
char *frame;
```

DESCRIPTION

main

qacover provides output to stdout summarizing the coverage of a specified data type from a gff file. Usage:

```
qacover <-t___><-f____>  file1.....
      where -t___ is the minimum time gap for lost data(default 30s)
            -f___ frame id (default no check on frame id)
```

The options are parsed and the appropriate variables are set. The files are parsed and output text is written until the last file has been processed. If a file is unavailable, an error message is generated and the next file is processed. The output text contains the start time of the interval, the frame id, data indicator (valid,lost or edit), the duration of the interval and the number of points in the interval.

cover_file

generates coverage history

ptr character pointer, not used

frame current data frame

NAME

main, qanois - generate textfile for noise statistic and binary file for graphic

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *qanois(frame,ptr)
char *frame;
char *ptr;
```

DESCRIPTION

main

qanois provides output to stdout documenting noise statistics for obs gff file and binary file for graphic. The input is from ddnois. Usage :

```
qanois <-f___><-i___><-q___><-m___>inputfile
     where -f___ is the frame id
           -i___ is the index to difference table to use for
              noise source(default 6th number )
           -q___ is the quantization value
              default see above
                 -m___ is the minimum number of points acceptable for
              noise consideration (default see above)
```

The options are parsed and the appropriate variables are set. If a file is unavailable, an error message is generated. The output text has the following format:

```
Begin_time  Data_type  "noise"  duration  quantized  average_obs  number_of
of_noise                value    measurement   points   interval Ex: 1000.0
aran        noise     384.0      .01          50.0          90   1384.0       aran
noise   768.0    .03        90.0         180
```

Interpretation:

. At 1000.0 Range data noise average .01 meters for 384.0 seconds with an average range measurement of 50 meters and a total of 90 observation marks
. etc...

The output binary file used the following naming convention :
Inputfile_name plus suffix _N plus
first letter of the frame id desired

The frame of the output binary file is described as follows:

Standard header where edit status indicates number of points in the noise interval
The data portion of frame provides the following info
End time of noise interval
Average value of observation
Computed noise selected by Index

Note: User needs to run qanois repeatly for each frame id until all of the desirable types are finished.

the following defines depend on the format of the input frame

qanois

frame	current frame of data
ptr	character pointer, not used

NAME

main, rrrfile, jmpcomp - Generates a radar range versus range rate difference history

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *rrrfile(frame)
char *frame;

char *jmpcomp(frame,ptr)
char *ptr;
char *frame;
```

DESCRIPTION

main

qaranjmp provides output to stdout documenting jumps in range compared with range computed from range rate. The following intervals are documented.

```
If range is <= Rbound:  15 <= diff < 30
                        30 <= diff
          >  Rbound:    30 <= diff <  70
                        70 <= diff <  100
                        100 <= diff
```

```
Usage:
qaranjmp <-r___><-g__>  file1,....
where:          -r___ is Rbound (default 6000 meters)
              -g___ is maximum allowable gap for computation of
                    of jump in range (default 50 seconds)
```

The options are parsed and the appropriate variables are set.
If a file is unavailable, an error message is generated.
An interim gff file is created with the same name as the input file suffixed with RJ. This file contains a record with the range and range rate in one record and no other observations. This file can be removed after program execution; it has a frame id of "rand".
For each point that falls in the previously defined interval a record is written to stdout containing start time of interval, "jump", "range", duration of interval, and magnitude of the jump.

rrrfile

creates a range and range rate obs file

frame	current frame of data off obs file

jmpcomp

computes interval of range jump

ptr	character pointer, not used
frame	current frame of data from r/rr file

NAME

main, star_track, getangle, test_print - Star Tracker obs history

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *star_track(frame,frame1,frame2)
char *frame;
char *frame1,*frame2;

char *getangle(frame,infile)
char *frame;
char *infile;

void test_print(now,angle,end)
double *now;
double angle;
double *end;
```

DESCRIPTION

main

qastar provides output to stdout summarizing the star observations on an observation file. Usage:

qastar <-p><-y><-l___><-u___><-t____><-q___> obsfile <attfile>

where -p use the gff created from previous run of qastar (name input via obsfile) to recompute printed output (attfile not used).

where -y use the Y star tracker instead of the Z.

where -l___ defines the delta angle reflecting the low noise threshold setting for spotting stars

where -u___ defines the delta angle reflecting the high noise threshold setting for spotting stars

where -t___ defines maximum time difference allowable between input quaternions and observations

An input observation file(obsfile) and attitude file(attfile) are required inputs (except when using the <-p> option, see above.

The Z star tracker is assumed unless the -y option is requested.

The options are parsed and the appropriate variables are set.

Output text is written for each interval where two consecutive observations are fixed on the same inertially fixed object. This interval is accumulated.

The text includes the following information: time of beginning, "star","edit",duration of interval, average angle over the interval. An output gff file is created with the name the same as the input obs file plus the suffix _ST. The data on the file is:time of the end of interval, "star",mode,azimuth,elevation and angle.

NOTE: The angle placed on each frame is the angle between the pointing vectors represented by the previous valid frame and the current frame.

star_track

determines star obs

frame current data from obs file

frame2 current data from attitude file that span frame

getangle
frame current data from inertial star tracker angle off file

infile not used

retrieves the inertial pointing angle difference

test_print
now time tag of current vector

angle angle measure between last inertial pointing vector and current vector

end
time tag of last vector

this function tests the angle tolerance for printed output

NAME

main, sensed_vel - Summarizes the sensed velocity file and creates a sensed acceleration file

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *sensed_vel(frame,ptr)
char *ptr;
char *frame;
```

DESCRIPTION

main

qasv provides output to stdout documenting edited sensed velocities and burn intervals. Usage:

```
qasv <-e___><-b____>  file1,....
     where -e___ is the maximum acceptable sensed acceleration
                 (default 100)
           -b___ is the mimimum burn sensed acceleration
                 (default .00369)
```

The options are parsed and the appropriate variables are set. The files are parsed and output text is written until the last file has been processed. If a file is unavailable, an error message is generated and the next file is processed. A gff file is generated containing the sensed acceleration vectors and the magnitude of the sensed acceleration at times of events. The frame id for this file is "sacc."

For each point >e or >b a record is written to stdout containing start time, "valid" or "edit", "sacc", delta time, and magnitude of the sensed acceleration. The intervals are not accumulated

sensed_vel

computes sensed accel. & documents interval

ptr — pointer, not used

frame — current data for file

NAME

main, write_file - write a subset of a file with user specified frame id and time

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *write_file(frame,ptr)
char *ptr;
char *frame;
```

DESCRIPTION

main

Rdwt read in a file and write to the file with user specified frame and time. Usage:

```
rdwt <-f____><-b___><-e___> file
        where -f___ frame id (default no check on frame id)
           -b___ begin time (default to file begin time)
           -e___ end time (default to file end time)
```

The options are parsed and the appropriate variables are set. The file is parsed and output binary file is written. If input file is unavailable, an error message is generated.

write_file

ptr — pointer, not used

frame — current frame write to a file with user specified time and frame id

NAME

main, read_set, print_record - reads SET file which is output from sensor tape output processor

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

void read_set()

char *print_record(time,count)
double time;
int *count;
```

DESCRIPTION

main

read_set reads in a binary file created by the stop processor and provides a printed output option over a specified time span

-f__ indicates file name of SET file
-b__ indicates begin time in gmt seconds
-e__ indicates last time in gmt seconds
-p__ indicates print frequency option
Nonpositive means first and last only

NAME

main, read_sit, print_record - read the SIT data output by stop

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

void read_sit()

char *print_record(time,count)
double time;
int *count;
```

DESCRIPTION

main

read_sit reads in a binary file created by the stop processor and provides a printed output option over a specified time span

-f__ indicates file name of SIT file
-b__ indicates begin time in gmt seconds
-e__ indicates last time in gmt seconds
-p__ indicates print frequency option
Nonpositive means first and last only

NAME

main, rlvsrl

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

char *rlvsrl(frame1,frame2a,frame2b)
double *frame1;
double *frame2a;
double *frame2b;
```

DESCRIPTION

main

rlvsrl reaos two relative trajectory files and determines the diference between states in UVW coordinates

NAME
rptost

SYNOPSIS
program rptost

DESCRIPTION
rptost
Converts roll/pitch to shaft/trunnion reference.

NAME

File, Time, Radius, Begin, End, Prime_id, Second_id, Help, main, search - gff file procor

SYNOPSIS

```
FILE *File;

double Time,Radius,Begin,End;

char *Prime_id[],*Second_id[];

int Help;

int main(argc,argv)
int argc;
char **argv;

void search()
```

DESCRIPTION

main

this program extracts intervals of desired info frames from the text files produced by the relbet qa processors where the frames have the general form as follows:

time-tag(number) ID(string) INFO_TYPE(string) duration(number) number ... the information used on each frame comes from the first three fields but the complete frame is read in and output if applicable

invoke this function:

search -h -f___ -t___ -r___ -b___ -e___ -p___ ... -s___ ...

where:

-h - indicates help option which displays primary and dependent labels available for extraction
The use of this option precludes all others
-f - indicates the file name of the input text file
-t - is the origin time of interest
-r - is the radius about the origin which encloses the time interval of interest
-b - is the begin time of the specified interval of interest
-e - is the end time of the specified interval of interest
-p - indicates the list of frame IDs desired
-s - indicates the list of INFO_TYPEs desired associated with the list of IDs

NAME

main, out_rel, out_cov, out_bias, file_size

SYNOPSIS

```
int main(argc,argv)
int argc;
char **argv;

char *out_rel(inframe)
char *inframe;

char *out_cov(inframe)
char *inframe;

char *out_bias(inframe)
char *inframe;

int file_size(file_id)
char *file_id;
```

DESCRIPTION

main

sln2rl reads an arbitrary sized (nsol) solution file output by either the kalmar or smoothing filter and generates

- relative trajectory files for solution states
- covariance file in the form of UVW relative trajectory files where the noise sigmas representing the base state solution and relative state solution in UVW of the base solution state are presented as a 12- element vector and the bias sigmas follow
- bias solution file containing the bias solution for as many obs solved for per frame

```
sln2rl <-r__> file1 <-c__> file2 <-b__> file3 <-n__> nsol
       <-s__> file4
```

where -r__ indicates create the rel traj
-c__ indicates create the covariance soln file
-b__ indicates create the bias soln file
-n__ is the size of the solution vector (default 16)
-s__ is the input solution file

NAME

main, stop - sensor tape output processor

SYNOPSIS

```
main(argc,argv)
int argc;
char **argv;

char *stop(frame1,frame2a,frame2b)
char *frame1;
char *frame2a;
char *frame2b;
```

DESCRIPTION

main

stop reads in two files including a relative trajectory file and a sensor data file which is created by the downlist program specifically for the purpose of writing two binary files of SENSOR input data called SIT and SET

-r__ indicates file name of relative traj file
-s__ indicates file name of sensor data file

stop

frame1 the current frame from Sensor file 1

frame2a the frame from relative traj file 2 <= time of frame1

frame2b the frame from relative traj file 2 > time of frame1

NAME

aero

SYNOPSIS

```
subroutine aero (rv,rvt,rsun,rnp,a)
double precision rv (3)
double precision rvt (6)
double precision rsun (3)
double precision rnp (3,3)
double precision a (3)
```

DESCRIPTION

aero

Executive for aerooynamic effects on m50 acceleration and body angular acceleration.

rv m50 position of vehicle vector (input)

rvt true of date state (input)

rsun m50 position of sun vector (input)

rnp m50 to true of date matrix (input)

a m50 accelaration (ouput)

NAME
cbody

SYNOPSIS
```
subroutine cbody (mu,r,a)
double precision mu
double precision r (3)
double precision a (3)
```

DESCRIPTION
cbody
Computes central body acceleration.

mu gravitational parameter (input)

r position relative to central body (input)

a accelaration (output)

NAME

cdrag

SYNOPSIS

```
subroutine cdrag (w,ad,f)
double precision w (3)
double precision ad
double precision f (3)
```

DESCRIPTION

cdrag

Computes constant area drag force.

w	wind in body coordinates (input)
ad	density of atmosphere (input)
f	drag force (output)

NAME
cyldr

SYNOPSIS
```
subroutine cyldr (wb,rho,fb)
double precision fb (3)
double precision rho
double precision wb (3)
```

DESCRIPTION
cyldr
compute force and torque due to aerodynamic forces on a cylinder

fb drag force in body coordinates

rho atmospheric density

wb wind velocity in body coordinates

NAME
getm

SYNOPSIS
```
subroutine getm (t,m)
double precision t
double precision m
```

DESCRIPTION
getm
Fetches mass from mass table.

t time mass is desired (input)

m vehicle mass (output)

NAME

harm

SYNOPSIS

```
subroutine harm (rg,gg)
double precision rg (3)
double precision gg (3)
```

DESCRIPTION

harm

Computes acceleration due to harmonic expansion terms in gravitational field of central body.

rg position vector of reference vehicles id earth-fixed coordinates (input)

gg acceleration in earth-fixed coordinates (output)

COMMENTS

Maximum 8th degree, 8th order.

NAME
solrad

SYNOPSIS
```
subroutine solrad (rv,rs,acc)
double precision rv (3)
double precision rs (3)
double precision acc (3)
```

DESCRIPTION
solrad
Computes acceleration due to solar radiation pressure.

rv m50 position of vehicle (input)

rs m50 position of sun (input)

acc acceleration (output)

NAME

svelbs

SYNOPSIS

```
subroutine svelbs (v,t,m,as)
integer v
double precision t
double precision m (3,3)
double precision as (3)
```

DESCRIPTION

svelbs

Computes bias and unbias of sensed acceleration.

v	vehicle id (input)
t	desired time (input)
m	body to inertial matrix (input)
as	sensed acceleration (input/output)

NAME

svftch

SYNOPSIS

```
subroutine svftch (ivloc,inc,tc,tnew)
integer ivloc
integer inc
double precision tc
double precision tnew
```

DESCRIPTION

svftch

Computes sensed acceleration.

ivloc vehicle id (input).

inc step direction (input).

tc current time (input).

tnew next time (output).

NAME

tblook

SYNOPSIS

```
subroutine tblook (imx,ltb,tb,t,i)
integer imx
integer ltb
double precision tb (ltb,imx)
double precision t
integer i
```

DESCRIPTION

tblook

Finds first entry in a time increasing table that is at or before a given time. The table is double precison with the first slot the time. If all entries are after the time, an index of zero is returned.

imx max index to table (input)

ltb double word length of table entry (input)

tb table (input)

t desired time (input)

i index to table (input/output)

COMMENTS

The index must be initialized to a value less than or equal to the maximum index value.

NAME
vntfch

SYNOPSIS
```
subroutine vntfch (v,tc,tnew)
integer v
double precision tc
double precision tnew
```

DESCRIPTION
vntfch
Fetches vent force of each vehicle.

v vehicle id (input)

tc current time (input)

tnew new time (input/output)

NAME

wndang

SYNOPSIS

```
subroutine wndang (w,ws,sb,cb,sa,ca)
double precision w (3)
double precision ws
double precision sb
double precision cb
double precision sa
double precision ca
```

DESCRIPTION

wndang

Computes sines and cosines of sideslip and attack angles.

w	wind in body coordinates(input)
ws	wind speed(output)
sb	sine of side slip angle(output)
cb	cosine of side slip angle(output)
sa	sine of angle attack(output)
ca	cosine of angle attack(output)

NAME

zutek

SYNOPSIS

```
subroutine zutek (wb,rho,fb)
double precision wb (3)
double precision rho
double precision fb (3)
```

DESCRIPTION

zutek

Computes aerodynamic force and torque using zuteck curve fit.

wb wind velocity in body coordinates (input)

rho atmospheric density (input)

fb drag force in body coordinates (output)

NAME

file2io, get_next_frame - function for creating bracket times from second file.

SYNOPSIS

```
char *file2io(inframe,file_info)
char *inframe;
char *file_info;

double *get_next_frame()
```

DESCRIPTION

file2io

inframe — data frame of previouslly read gff

file_info — file information of input gff

file2io sends the data frames which have times bracketting the time of the input frame time to the function specified in the input file structure. Note if no bracket interval available then nulls are sent. Warning: The presence of static variables makes this function an unlikely candidate for general use except for the purpose for which it was written. This purpose is to open and manipulate one file per execution in responce to being invoked by fileio

get_next_frame

get the next frame , check first for end of file , check next for proper frame id , check next for edit status if necessary

NAME

fileinfo_io - a gbf input/output routine for recursive FILE_INFO structures

SYNOPSIS

```
char *fileinfo_io(frame,file_info)
char *frame;
char *file_info;
```

DESCRIPTION

fileinfo_io

frame — current frame from driver file

file_info — FILE_INFO structure pointer see fileio.h

This routine enables the programmer to utilize the fileio process to obtain each frame from some input file and perform a multiple set of functions each of which result in a frame output to a specific file. The output process of fileio is not invoked. The power of this process is in the recursive data structure FILE_INFO (see fileio.h).

NAME

BaseTime, fileio - gff file input and output routine

SYNOPSIS

```
CALTIME BaseTime;

int fileio(in_file,out_suffix,out_size)
struct FILE_INFO *in_file;
char *out_suffix;
int out_size;
```

DESCRIPTION

fileio

in_file — structure defining input file

out_suffix — suffix to be cateneated to input file name for new file

out_size — size of output data record:
if <0, same as input;
if =0, no output file;
if >0, output size.

fileio reads a gff file and creates a new gff file with data that is a function of the input file at a subset of the time points on the input file.

NAME

HStoHB, HBtoHS - move gbf header infor to and from buffer

SYNOPSIS

```
void HStoHB(hdrbuf,gbf)
GBFILE *gbf;
char *hdrbuf;

void HBtoHS(gbf,hdrbuf)
GBFILEPTR gbf;
char *hdrbuf;
```

DESCRIPTION

HStoHB

shifts information from header structure to buffer. The update date field is set to the current date in both structures.

HBtoHS

shifts information from buffer to header structure. The time offset, record size, origin, and time word pointer are also computed

NAME

gbfnew, gbfwrt, gbfopn, gbfcls, gbfdfc, gbfops - C interface between FORTRAN and gbfile C routines

SYNOPSIS

```
void gbfnew(file,fname,hdrid,jobdes,wrdfrm,date,dates,status)
int *file;
char *fname;
char *hdrid;
char *jobdes;
int *wrdfrm;
int *date;
double *dates;
int *status;

void gbfwrt(file,dfrbuf,status)
int *file;
double *dfrbuf;
int *status;

void gbfopn(file,fname,status)
int *file;
char *fname;
int *status;

void gbfcls(file,status)
int *file;
int *status;

void gbfdfc(file,dfrbuf,status)
int *file;
double *dfrbuf;
int *status;

void gbfops(fname)
char *fname;
```

DESCRIPTION

gbfnew

file	internal id
fname	input file name
hdrid	input header id
jobdes	input header description
wrdfrm	number of bytes per frame
date	calendar base date- yr, mo, day, hr, min
dates	seconds of base date
status	status flag

general new gbf create function

gbfwrt

general write

gbfopn
general open

gbfcls
general close

gbfdfc
close function for use with downlist processor to close files then reopen and display header and return begin/end times

gbfops
open the file name for output display in place of stderr used if accessing the function gbfdfc above

NAME

FileBaseTime, newGBF, gbtoff, gbwhead, gbrhead - gbfile structure and header operations

SYNOPSIS

```
CALTIME FileBaseTime;

GBFILE *newGBF(name,size,tbyte,tbase,story,format)
int size,tbyte;
CALTIME *tbase;
char *name,*story,*format;

gbtoff(gbf)
GBFILE *gbf;

gbwhead(gbf)
GBFILEPTR gbf;

gbrhead(gbf)
GBFILEPTR gbf;
```

DESCRIPTION

newGBF

forms new gbstructure and sets input fields. Returns null pointer is error encounted

gbtoff

set time offset for gbfile If reference basetime is not set, it is set to file basetime and the time offset is null

gbwhead

writes header information stored in current buffer to file record

gbrhead

reads header record

NAME

gbpos, gbread, gbwrite, gbdread, gbdwrite - i/o of gbfiles

SYNOPSIS

```
gbpos(gbf,rec)
GBFILE *gbf;
unsigned rec;

char *gbread(gbf)
GBFILE *gbf;

int gbwrite(gbf,data)
GBFILE *gbf;
char *data;

char *gbdread(gbf,rec)
GBFILE *gbf;
unsigned rec;

int gbdwrite(gbf,rec,data)
GBFILE *gbf;
unsigned rec;
char *data;
```

DESCRIPTION

gbpos
positions file pointer to desired record of referenced gbfile

gbread
reads current record of referenced gbfile and returns pointer to data in record. Pointer should be recast by caller to desired record type. Errors result in invalid pointer value (0)

gbwrite
writes to current record of referenced gbfile returning the number of the bytes written. Note that the file is posed at the next record. errors result in negative return values.

gbdread
reads desired record of referenced gbfile and returns pointer to data in record. pointer should be recast by caller to desired reocrd type. errors result in invalid pointer value (0)

gbdwrite
like gbwrite but the file is first positioned to the specified record.

NAME

makeGBF, makeGBData - gbfile structure allocations

SYNOPSIS

```
GBFILE *makeGBF(name)
char *name;

char *makeGBData(gbf)
GBFILE *gbf;
```

DESCRIPTION

makeGBF

allocate space for GBFILE structure and set specified fields. Errors result in null pointer being returned. Fields are not set if pointers are null

makeGBData

allocate space for GBFILE data buffer. The size of a record is also computed. If size is not a multiple of sizeof(int), the fields are reset so that rsize = sizeof(int)*int_rsize. Errors result in null pointer being returned. Fields are not set if pointers are null

NAME

gbclose, freeGBF, gbfree, gbopen, gbnew - open, close and free of gbfiles

SYNOPSIS

```
void gbclose(gbf)
GBFILEPTR gbf;

void freeGBF(gbf)
GBFILEPTR gbf;

void gbfree(gbf)
GBFILEPTR gbf;

GBFILE *gbopen(name,mode)
char *name;
char *mode;

GBFILEPTR gbnew(name,size,tbyte,tbase,story,format,mode)
char *name;
char *mode;
int size;
int tbyte;
char *story,*format;
CALTIME *tbase;
```

DESCRIPTION

gbclose
closes file and updates header if file had a write key set

freeGBF
frees all space allocated to gbfstructure. WARNING dire things may happen if file was not opened with gbopen since it is assumed that malloc was used to allocate space.

gbfree
closes file and frees all space allocated to it. WARNING dire things may happen if file was not opened with gbopen since it is assumed that malloc was used to allocate space.

gbopen
name file path name

opens file specified by name with designated mode: read only read/write, or append. File must previously exist or error will result. returns pointer to file structure and assigns necessary space such as buffer space and space for the file packet. Read/write modes are:

r,R =read only (default)
b,B =read and write
a,A =append

errors return invalid pointer value (0) and are as follows: bad read/write mode, unable to open file, unable to assign enough buffer space.
note that the file is positioned to the first record (record 0) and the input buffer contains this record. NOTE only the first character of the mode is considered and the default is read only.

NOTE: mode is type (char *) not char, a null value (0) selects the default.

gbnew

name file path name

size record size in bytes

opens file specified by name with designated mode: read only read/write, or append. file must previously exist or error will result. returns pointer to file structure and assigns necessary space such as buffer space and space for the file packet.

read/write modes are:

b,B =read and write (default)

w,W = write only (causes error with gbclose)

errors return invalid pointer value (0) and are as follows: bad read/write mode, unable to open file, unable to assign enough buffer space,

note that the file is positioned to the first record (record 0) and the input buffer contains this record. NOTE only the first character of the mode is considered and the default is read/write.

NOTE: if the write only option is used, header records are not correcly updated on closing the file. Invoking gbclose or gbfree to close to will result in an error.

NAME

gbfphead, fgbprt, gbphead - display of gbfile structures

SYNOPSIS

```
void gbfphead(file,gbf)
FILE *file;
GBFILEPTR gbf;

void fgbprt(file,gbf)
GBFILEPTR gbf;
FILE *file;

void gbphead(gbf)
GBFILEPTR gbf;
```

DESCRIPTION

gbfphead
displays current header structure to specified file

fgbprt
displays current header structure to specified file

gbphead
displays header to standard out

NAME

gbtime - gb time

SYNOPSIS

```
char *gbtime(gbf,time)
GBFILE *gbf;
double time;
```

DESCRIPTION

gbtime

positions file to last time before designated time and return pointer to data in buffer. If time before start of file, the first record is selected (record number 0) and if after the end time of the file, the last record is selected. Errors result in an invalid data data pointer (0). The output pointer snould be recast to the data structure type by the calling function. file search consists of binary search after checking that the time is within file bounds and an initial estimate of the desired record record is obained by linear interpolation

NAME

dctprt

SYNOPSIS

```
subroutine dctprt (iop,frame,out)
integer iop (*)
integer frame (*)
integer out
```

DESCRIPTION

dctprt

Prints the dictionary from a GFF file.

iop i/o packet (input/output)

frame record buffer (input/output)

out print file unit number (output)

NAME

gfcls

SYNOPSIS

```
subroutine gfcls (iop,stdout,filout,frame,dscrpt,prgver,status)
integer iop (*)
integer stdout
integer filout
integer frame (*)
character dscrpt
character prgver
integer status
```

DESCRIPTION

gfcls

Closes a GFF file.

iop i/o packet

stdout standard output unit number

filout file output unit number

frame frame buffer

dscrpt update description

prgver program name and version

status i/o status

NAME
gfdict

SYNOPSIS
```
subroutine gfdict (iop,frame,dict)
integer iop (*)
integer frame (*)
character*20 dict (*)
```

DESCRIPTION
gfdict
Returns the dictionary from the GFF file in a usable form.

iop	i/o packet
frame	record buffer (input/output)
dict	dictionary (output)

NAME

gfemsg

SYNOPSIS

```
subroutine gfemsg (xx,p,stdout,filout)
integer xx
integer p (*)
integer stdout
integer filout
```

DESCRIPTION

gfemsg

Displays error messages to the output file.

xx	error type (inpput)
p	i/o packet
stdout	standard output unit number
filout	file output unit number

NAME
gfend

SYNOPSIS
```
subroutine gfend (iop,stdout,filout,frame,status)
integer iop (*)
integer stdout
integer filout
integer frame (*)
integer status
```

DESCRIPTION
gfend
Updates the file header and append the *end frame to the GFF file.

iop i/o packet(input/output)

stdout standard out unit number(input)

filout print file unit number(input)

frame frame buffer(input)

status i/o status(output)

NAME

gffdsp

SYNOPSIS

```
subroutine gffdsp (iop,out,frame,dict)
integer iop (*)
integer out
integer frame (*)
character dict (*)
```

DESCRIPTION

gffdsp

Displays file information.

iop	i/o packet
out	unit number of the output file
frame	record buffer
dict	dictionary

NAME

gfnew

SYNOPSIS

```
subroutine gfnew (iop,stdout,filout,name,unit,type,dscrpt,
prgver,fdate,fsec,pdate,psec,frmsz,frame,spcblk,dict,status)
integer iop (*)
integer stdout
integer filout
character name
integer unit
character type
character dscrpt
character prgver
integer fdate (5)
double precision fsec
integer pdate (5)
double precision psec
integer frmsz
integer frame (frmsz)
integer spcblk (11)
character dict (frmsz)
integer status
```

DESCRIPTION

gfnew

Creates a new GFF file by doing the following: 1) Creates the i/o packet for GFF inter-library communications. 2) Creates the file header and write it to the GFF file. 3) Creates the *beg frame and write it to the GFF file.

iop	i/o packet (input/output)
stdout	standard output unit number (input)
filout	print file unit number (input)
name	file name (input)
unit	GFF file unit number (input)
type	file type (input)
dscrpt	file description (input)
prgver	program name and version
fdate	file base date - ymdhm (input)
fsec	file base date - sec (input)
pdate	program base date - ymdhm (input)
psec	program base date - sec (input)
frmsz	frame size (input)
frame	output record (file output)
spcblk	block of information particular to the file type (input)
dict	input dictionary (input)
status	i/o status (output)

NAME

gfopen

SYNOPSIS

```
subroutine gfopen (iop,stdout,filout,name,unit,pdate,psec,rwflg,
frame,status)
integer iop (*)
integer stdout
integer filout
character name
integer unit
integer frame (*)
integer pdate (5)
double precision psec
integer rwflg
integer status
```

DESCRIPTION

gfopen

Opens an existing GFF file and set up the i/o packet.

iop	i/o packet
stdout	standard output unit number
filout	print file unit number
name	name of GFF file to open
unit	unit number on which to open the GFF file
frame	frame buffer
pdate	program base date (ymdhm)
psec	program base date (sec)
rwflg	read/write flag
status	i/o status

NAME

gfread

SYNOPSIS

```
subroutine gfread (iop,stdout,filout,status,timtag,recnum)
integer iop (*)
integer stdout
integer filout
integer status
double precision timtag
integer recnum
```

DESCRIPTION

gfread

Converts the timetag from prgram base time to file base time and returns the record number of the frame.

iop io packet

stdout standard output unit number

filout print file unit number

status output status

timtag timetag

recnum current record number

NAME

gfrhdr

SYNOPSIS

```
subroutine gfrhdr (hdr,iop,hdrsz,frame)
integer hdrsz
integer hdr (hdrsz)
integer iop (*)
integer frame (*)
```

DESCRIPTION

gfrhdr

Reads the header from the GFF file.

hdrsz number of words in the header (input)

hdr header to be written to the GFF file (output)

iop i/o packet

frame frame buffer

NAME

gftime

SYNOPSIS

```
subroutine gftime (iop,stdout,filout,time,index,status)
integer iop (*)
integer stdout
integer filout
double precision time
integer index
integer status
```

DESCRIPTION

gftime

Returns the index of the first record whose time is less than the time specified. If the time specified is past the end of file, then the index of the last frame is returned. If the time specified is before the beginning of the file, then an error is returned.

iop	i/o packet
stdout	standard output unit number
filout	print file unit number (output)
time	time to search for (input/output)
index	index the the record requested (input/output)
status	error status (output)

NAME

gftims

SYNOPSIS

```
subroutine gftims (iop,stdout,filout,timei,timeo,index,status)
integer iop (*)
integer filout
integer stdout
double precision timei
double precision timeo
integer index
integer status
```

DESCRIPTION

gftims

Returns the index of the first record whose time is less than the time specified. If the time specified is past the end of file, then the index of the last frame is returned. If the time specified is before the beginning of the file, then an error is returned.

iop	i/o packet
filout	print file unit number
stdout	standard output unit number
timei	program time to find (input)
timeo	file time found (output)
index	index the the record requested (input/output)
status	error status (output)

NAME

gfwhdr

SYNOPSIS

```
subroutine gfwhdr (hdr,iop,hdrsz)
integer hdrsz
integer hdr (hdrsz)
integer iop (*)
```

DESCRIPTION

gfwhdr

Writes the header to the GFF file.

hdrsz number of words in the header (input)

hdr header to be written to the GFF file (output)

iop i/o packet

NAME

gfwrit

SYNOPSIS

```
subroutine gfwrit (iop,stdout,filout,status,timtag,recnum)
integer iop (*)
integer stdout
integer filout
integer status
double precision timtag
integer recnum
```

DESCRIPTION

gfwrit

Converts the timetag from program base time to file base time and returns the record number of the frame.

iop io packet

stdout standard output unit number

filout print file unit number

status output status (output)

timtag timetag (input/output)

recnum current record number (output)

NAME

hdrprt

SYNOPSIS

```
subroutine hdrprt (iop,out)
integer iop (*)
integer out
```

DESCRIPTION

hdrprt

Prints the header record from an GFF file.

iop i/o packet

out output unit number

NAME
icshft

SYNOPSIS
```
subroutine icshft (n,a.d)
integer n
integer a (n)
integer d (n)
```

DESCRIPTION
icshft
Moves a word from location to another. This would typically be from a character to an integer.

n dimension of a and e (input)

a word to move from (input)

d word to move to (output)

NAME

iopprt

SYNOPSIS

```
subroutine iopprt (out,ip)
integer out
integer ip (*)
```

DESCRIPTION

iopprt

Prints the i/o packet.

out output unit number

ip i/o packet

NAME
ixatm

SYNOPSIS
subroutine ixatm

DESCRIPTION
ixatm
Input of common block xxatm.

NAME
ixbias

SYNOPSIS
subroutine ixbias

DESCRIPTION
ixbias
Input of common block xxbias.

NAME
ixcon

SYNOPSIS
subroutine ixcon

DESCRIPTION
ixcon
Input of common block xxcon.

NAME
ixdata

SYNOPSIS
subroutine ixdata

DESCRIPTION
ixdata
Input of common block xxdata.

NAME

ixdprm

SYNOPSIS

subroutine ixdprm

DESCRIPTION

ixdprm

Input of common block xxdprm.

NAME
ixerth

SYNOPSIS
subroutine ixerth

DESCRIPTION
ixerth
Input of common block xxerth.

NAME
ixfile

SYNOPSIS
subroutine ixfile

DESCRIPTION
ixfile
Input of common block xxfile.

NAME
ixgnrl

SYNOPSIS
subroutine ixgnrl

DESCRIPTION
ixgnrl
Input of common block data.

NAME
ixgraf

SYNOPSIS
subroutine ixgraf

DESCRIPTION
ixgraf
Input of common block xxgraf.

NAME
ixgrav

SYNOPSIS
subroutine ixgrav

DESCRIPTION
ixgrav
Input of common block data.

NAME
ixinit

SYNOPSIS
subroutine ixinit

DESCRIPTION
ixinit
Input of common block xxinit.

NAME
ixkal

SYNOPSIS
subroutine ixkal

DESCRIPTION
ixkal
Input of common block xxkal.

NAME
ixmas

SYNOPSIS
subroutine ixmas

DESCRIPTION
ixmas
Input of common block xxmas.

NAME
ixmax

SYNOPSIS
subroutine ixmax

DESCRIPTION
ixmax
Input of common block xxmax.

NAME
ixmisc

SYNOPSIS
subroutine ixmisc

DESCRIPTION
ixmisc
Input of common block xxmisc.

NAME
ixmoon

SYNOPSIS
subroutine ixmoon

DESCRIPTION
ixmoon
Input of common block data.

NAME
ixnflz

SYNOPSIS
subroutine ixnflz

DESCRIPTION
ixnflz
Input of common block xxnflz.

NAME
ixprnt

SYNOPSIS
subroutine ixprnt

DESCRIPTION
ixprnt
Input of common block xxprnt.

NAME
ixprop

SYNOPSIS
subroutine ixprop

DESCRIPTION
ixprop
Input of common block xxprop.

NAME
ixqcrv

SYNOPSIS
subroutine ixqcrv

DESCRIPTION
ixqcrv
Input of common block xxqcrv.

NAME
ixqgen

SYNOPSIS
subroutine ixqgen

DESCRIPTION
ixqgen
Input of common block xxqgen.

NAME
ixqprm

SYNOPSIS
subroutine ixqprm

DESCRIPTION
ixqprm
Input of common block xxqprm.

NAME
ixrpst

SYNOPSIS
subroutine ixrpst

DESCRIPTION
ixrpst
Input for common block xxrpst.

NAME

ixscov

SYNOPSIS

subroutine ixscov

DESCRIPTION

ixscov

Input of common block xxscov.

NAME
ixsen

SYNOPSIS
subroutine ixsen

DESCRIPTION
ixsen
Input of common block data.

NAME

ixsprm

SYNOPSIS

subroutine ixsprm

DESCRIPTION

ixsprm

Input of common block xxsprm.

NAME
ixsptm

SYNOPSIS
subroutine ixsptm

DESCRIPTION
ixsptm
Input of common block xxsptm.

NAME
ixsun

SYNOPSIS
subroutine ixsun

DESCRIPTION
ixsun
Input of common block xxsun.

NAME
ixsvbi

SYNOPSIS
subroutine ixsvbi

DESCRIPTION
ixsvbi
Input of common block xxsvbi.

NAME

ixtime

SYNOPSIS

subroutine ixtime

DESCRIPTION

ixtime

Input of common block xxtime.

NAME
ixtoff

SYNOPSIS
subroutine ixtoff

DESCRIPTION
ixtoff
Input of common block xxtoff.

NAME
ixusys

SYNOPSIS
subroutine ixusys

DESCRIPTION
ixusys
Input of common block xxusys.

NAME
ixvcx

SYNOPSIS
subroutine ixvcx

DESCRIPTION
ixvcx
Input of common block xxvcx.

NAME
ixvnt

SYNOPSIS
subroutine ixvnt

DESCRIPTION
ixvnt
Input of common block xxvnt.

NAME

lint

SYNOPSIS

```
subroutine lint (n,x,w,fac,z)
integer n
double precision x (1)
double precision w
double precision fac (1)
double precision z
```

DESCRIPTION

lint

Lagrangian interpolation.

n order of the interpolation to be used (input)

x the table of known arguments (input)

w the argument to which the function value is to be interpolated (input)

fac table of lagrangian factors (input)

z the interpolated value (output)

COMMENTS

The table of lagrangian factors must have been initialized by a call to the subroutine lintin.

NAME

lintin

SYNOPSIS

```
subroutine lintin (n,x,y,fac)
integer n
double precision x (1)
double precision y (1)
double precision fac (1)
```

DESCRIPTION

lintin

Initializes lagrangian factors for interpolation.

n order of interpolation to be used (input)

x table of known arguments of the function corresponding to y (input)

y table of known values of the function corresponding to x (input)

fac table of lagrangian factors (output)

NAME

lrntrp

SYNOPSIS

```
subroutine lrntrp (t1,x1,t2,x2,t,x)
double precision t1
double precision x1 (6)
double precision t2
double precision x2 (6)
double precision t
double precision x (6)
```

DESCRIPTION

lrntrp

LEAR interpolation method for position and velocity.

t1 first known time (input).

x1 known state at time t1 (input).

t2 second known time (input).

x2 known state at time t2 (input).

t desired time for interpolation (input).

x interpolated state for time t (output).

NAME

qintrp

SYNOPSIS

```
subroutine qintrp (t1,q1,t2,q2,t,q)
double precision t1
double precision q1 (4)
double precision t2
double precision q2 (4)
double precision t
double precision q (4)
```

DESCRIPTION

qintrp

Interpolates a quaternion from two input quaternions.

t1 time of attitude 1 (input)

q1 quaternion for interpolation (input)

t2 time of attitude 2 (input)

q2 quaternion for interpolation (input)

t time of desired attitude (input)

q attitude quaternion (output)

NAME

qwntrp

SYNOPSIS

```
subroutine qwntrp (t,t1,q1,t2,q2,q,w)
double precision t
double precision t1
double precision q1 (4)
double precision t2
double precision q2 (4)
double precision q (4)
double precision w (3)
```

DESCRIPTION

qwntrp

Interpolates a quaternion and rate from two input quaternions.

t	time of desired attitude (input)
t1	time of attitude 1 (input)
q1	quaternion for interpolation (input)
t2	time of attitude 2 (input)
q2	quaternion for interpolation (input)
q	attitude quaternion (output)
w	average angular rate (output)

NAME

rtntrp, lrntrp - interpolation methods

SYNOPSIS

```
double *rtntrp(t,x,t1,x1,t2,x2)
double t1;
double *x1;
double t2;
double *x2;
double t;
double *x;

double *lrntrp(t,x,t1,x1,t2,x2)
double t1;
double *x1;
double t2;
double *x2;
double t;
double *x;
```

DESCRIPTION

rtntrp
LEAR interpolation method for relative states

lrntrp
Lagrangian interpolation

NAME

rlntrp

SYNOPSIS

```
subroutine rlntrp (iop,tm,n,rec,t,f,e,z,stat)
integer iop (*)
double precision tm
integer n
integer rec (n)
double precision t (n)
double precision f (*)
double precision e (*)
double precision z (n)
integer stat
```

DESCRIPTION

rlntrp

Lagrangian interpolation of RELBET GFF Centers n pt ephem if neccessary however center routine must be initialized. This may be done on the initial call by setting the last time in the array is less than the 1st it may also be done by a call to rlmid1.

iop	i/o package for file (input)
tm	desired time (input)
n	number of entries to be found (input)
rec	records corresponding to entries in e (input/output)
t	array of time tags for entries
f	interpolation factors (input/output)
e	array buffer for entries (input/output)
z	interpolated value (length is same as file data frame length (output)
stat	status flag: >=0 is good, -1 is eof, e is error (output)

NAME

rtntrp

SYNOPSIS

```
subroutine rtntrp (t1,x1,t2,x2,t,x)
double precision t1
double precision x1 (12)
double precision t2
double precision x2 (12)
double precision t
double precision x (12)
```

DESCRIPTION

rtntrp

LEAR interpolation method for relative states.

t1 first known time (input).

x1 known state at time t1 (input).

t2 second known time (input).

x2 known state at time t2 (input).

t desired time for interpolation (input).

x interpolated state for time t (output).

NAME

svntrp

SYNOPSIS

```
subroutine svntrp (t1,v1,t2,v2,t,a)
double precision t1
double precision v1 (3)
double precision t2
double precision v2 (3)
double precision t
double precision a (3)
```

DESCRIPTION

svntrp

Interpolates acceleration from two sensed velocities.

t1 known first time (input).

v1 known first velocity (input).

t2 known second time (input).

v2 known second velocity (input).

t desired time (input).

a desired acceleration (output).

NAME

initin, setin, getin - initialize the structure needed by the LINPUT process

SYNOPSIS

```
void initin()

void setin(loc,name,dimtype)
char *loc;
char *name;
char *dimtype;

void getin()
```

DESCRIPTION

initin

initialize maximum memory allocation for INPUT structure array

setin

set the values for the current INPUT entry and increment to next entry position

getin

get the inputs from the user's standard in file

NAME

yyleng, yymorfg, yytchar, yyin, yyout, yyestate, yylex, yyvstop, yycrank, yysvec, yytop, yybgin, yymatch, yyextra, yylineno, yytext, yylstate, yylsp, yyolsp, yysbuf, yysptr, yyfnd, yyprevious, yylook, yyback, yyinput, yyoutput, yyunput

SYNOPSIS

```
int yyleng;

int yymorfg;

int yytchar;

FILE *yyin,*yyout;

struct yysvf *yyestate;

yylex()

int yyvstop[];

struct yywork yycrank[];

struct yysvf yysvec[];

struct yywork *yytop;

struct yysvf *yybgin;

uchar yymatch[];

uchar yyextra[];

int yylineno;

uchar yytext[];

struct yysvf *yylstate[],**yylsp,**yyolsp;

uchar yysbuf[];

uchar *yysptr;

int *yyfnd;

int yyprevious;

yylook()

yyback(p,m)
int *p;

yyinput()

yyoutput(c)
int c;

yyunput(c)
int c;
```

DESCRIPTION

```
int
  end of yylex

yyextra
  UNISRC_ID @(#)27.2   85/02/26

yyolsp
  char yysbuf[YYLMAX];
   * char *yysptr = yysbuf;
   * ***** nls8 *****

yyinput
  the following are only used in the lex library
```

NAME

nlist, vlist, linput, yyerror, newcell, setq, binop, unop, sum, sub, usub, mult, div, pwr

SYNOPSIS

```
P_CELL nlist,vlist;

linput(input)
INPUT *input;

int yyerror(s)
char *s;

P_CELL newcell()

P_CELL setq(n,v)
P_CELL n,v;

P_CELL binop(op,a,b)
P_FUNCTION op;
P_CELL a,b;

P_CELL unop(op,a)
P_FUNCTION op;
P_CELL a;

P_CELL sum(a,b)
P_CELL a,b;

P_CELL sub(a,b)
P_CELL a,b;

P_CELL usub(a)
P_CELL a;

P_CELL mult(a,b)
P_CELL a,b;

P_CELL div(a,b)
P_CELL a,b;

P_CELL pwr(a,b)
P_CELL a,b;
```

DESCRIPTION

linput

L-input linput provides for L-input for other programs. The single input is an array of type INPUT. INPUT is a structure of the form

```
{ char *name; char *loc; int dim; char *type}
```

name is the name as it should appear in the user's input, loc is a pointer to where the input should be copied at the end of the input phase. dim is the number of items to be stored, and type is the type of the data (currenty "int", "double", "float", "char", or "string" or "string#"). Note that for "string" types, dim is the number of strings, not the length of the string however, for "string#" types the # is decoded as the length of the string and a test is made to verify that the input string is consitent with the expected length. The final entry in the input array should have a (char *)0 name.

newcell
The newcell function for lists. Currently newcell simply calls on calloc when needed, and makes no attempt at reclaiming storage no longer needed.

setq
assign value to namelist

binop
binary operations

unop
unary operations

sum
sum

sub
difference

usub
unary minus

mult
multiply

div
divide

pwr
power

NAME
newcell

SYNOPSIS

```
P_CELL newcell()
```

DESCRIPTION
newcell
allocate space

NAME

prtin - print the contents of the input blocks one variable at a time

SYNOPSIS

```
void prtin(ioc,name,dimtype)
char *ioc;
char *name;
char *dimtype;
```

DESCRIPTION

prtin

print the contents of the input blocks one variable at a time

NAME

nil, linit, cons, inumber, dnumber, symbol, isatom, isinumber, isdnumber, issymbol, isfunction, eq, car, cdr, ivalue, dvalue, fvalue, svalue, append, member, length, locate

SYNOPSIS

```
P_CELL nil;

int linit(cell_alloc)
P_CELL (*cell_alloc)();

P_CELL cons(a,b)
P_CELL a,b;

P_CELL inumber(n)
int n;

P_CELL dnumber(d)
double d;

P_CELL symbol(s)
char *s;

int isatom(p)
P_CELL p;

int isinumber(p)
P_CELL p;

int isdnumber(p)
P_CELL p;

int issymbol(p)
P_CELL p;

int isfunction(p)
P_CELL p;

int eq(p1,p2)
P_CELL p1,p2;

P_CELL car(p)
P_CELL p;

P_CELL cdr(p)
P_CELL p;

int ivalue(p)
P_CELL p;

double dvalue(p)
P_CELL p;

P_FUNCTION fvalue(p)
P_CELL p;

char *svalue(p)
P_CELL p;

P_CELL append(x,y)
P_CELL x,y;

int member(x,l)
```

```
P_CELL x,l;

int length(l)
P_CELL l;

P_CELL locate(x,l,m)
P_CELL x,l,m;
```

DESCRIPTION

linit
initialization

cons
cons cell constructor

inumber
integer number cell constructor

dnumber
double number cell constructor

symbol
symbol cell constructor

isatom
predicate: is cell atom?

isinumber
predicate: is cell integer?

isdnumber
predicate: is cell double

issymbol
predicate: is cell symbol?

isfunction
predicate: is cell function?

eq
predicate: are atoms equal?

car
return car of list

cdr
return cdr of list

ivalue
return ivalue of cell

dvalue
return dvalue of cell

fvalue
return fvalue of cell

svalue
return svalue of cell

append
append list to list

NAME

putexp

SYNOPSIS

```
int putexp(e)
P_CELL e;
```

DESCRIPTION

putexp

output expression in symbolic form

NAME

strsave

SYNOPSIS

```
char *strsave(s)
char *s;
```

DESCRIPTION

strsave

save string s somewhere

NAME

strstore

SYNOPSIS

```
char *strstore(t)
char *t;
```

DESCRIPTION

strstore

store string uniquely in array

NAME

acnvrt, ai2rl, ai2dbl, arl2i, adbl2i

SYNOPSIS

```
subroutine acnvrt (i,v,iv1,rv1)
integer i
integer iv1 (i)
real rv1 (i)
double precision v (i)

entry ai2rl (i,iv1,rv1)

entry ai2dbl (i,iv1,v)

entry arl2i (i,rv1,iv1)

entry adbl2i (i,v,iv1)
```

DESCRIPTION

acnvrt

Converts vector arrays of one type to another type.

i dimension indexes (input)

iv1 integer vectors (input/output)

rv1 real vector (input/output)

v double precision vectors (output)

ai2rl

Shifts integer vector to real vector.

ai2dbl

Shifts integer vector to double precision vector.

arl2i

Shifts real vector to integer vector.

adbl2i

Shifts double precision vector to integer vector.

NAME
ang2

SYNOPSIS
```
double precision function ang2 (x)
double precision x
```

DESCRIPTION
ang2
Reduces angles outside the range -pi to pi into that range.

x input angle in radians which is to be put into the range of -pi to pi.(input)

NAME

arctan

SYNOPSIS

```
double precision function arctan (a,b)
double precision a
double precision b
```

DESCRIPTION

arctan

Returns zero if both arguments are zero and computes datan2 otherwise.

a input argument

b input argument

NAME

arshft, vzero, rvzero, ivzero, vshift, ivshft, rvshft, ibshft, dbshft, i2char, i2real, i2dbl, char2i, real2i, dbl2i

SYNOPSIS

```
subroutine arshft (i,v,v2,v3,iv1,iv2,rv1,xv,rv2)
integer i
double precision v (i)
double precision v2 (i)
double precision v3 (i)
integer iv1 (i)
integer iv2 (i)
real rv1 (i)
character*4 xv (i)
real rv2 (i)

entry vzero (i,v)

entry rvzero (i,rv1)

entry ivzero (i,iv1)

entry vshift (i,v,v2)

entry ivshft (i,iv1,iv2)

entry rvshft (i,rv1,rv2)

entry ibshft (i,iv1,iv2)

entry dbshft (i,v2,v3)

entry i2char (i,iv1,xv)

entry i2real (i,iv1,rv1)

entry i2dbl (i,iv1,v)

entry char2i (i,xv,iv1)

entry real2i (i,rv1,iv1)

entry dbl2i (i,v,iv1)
```

DESCRIPTION

arshft

Moves data between arrays of different types (no conversion).

i	dimension index (input)
v	vectors (input/output)
v2	double precision vectors (input/output)
v3	double precision vectors (output)
iv1	integer vectors (input/output)
iv2	integer vectors (input/output)
rv1	real vector (input/output)
xv	character array (input/output)

rv2 real vector (input/output)

vzero
Zeros the vector.

rvzero
Zeros the real vector.

ivzero
Zeros the integer vector.

vshift
Shifts the double precision vector v into v2.

ivshft
Shifts the integer vector iv1 to iv2.

rvshft
Shifts the real vector rv1 into rv2.

ibshft
Shifts an integer array starting at last element.

dbshft
Shifts an double array starting at last element.

i2char
Shifts integer vector to character.

i2real
Shifts integer vector to real vector.

i2dbl
Shifts integer vector to double precision vector.

char2i
Shifts integer vector to character.

real2i
Shifts integer vector to real vector.

dbl2i
Shifts integer vector to double precision vector.

NAME

lsmin

SYNOPSIS

```
subroutine lsmin (n,a,w,invtol)
integer n
double precision a (*)
double precision w (*)
double precision invtol
```

DESCRIPTION

lsmin

Inverts positive definite symmetric matrix stored as lower triangular vector.

n	dimension of matrix (input)
a	positive definite symetric matrix stored as lower triangular array (input)
w	inverse of a (output)
invtol	tolerance for singularity (input)

NAME

m2qsub

SYNOPSIS

```
subroutine m2qsub (m,q)
double precision m (3,3)
double precision q (4)
```

DESCRIPTION

m2qsub

Converts a rotational matrix to a quaternion.

m rotational matrix (input)

q quaternion to be generated (output)

NAME

mxm, mxmc, mxv, mtxv, mt, mtxm, mxmt - Square matrix operations

SYNOPSIS

```
double *mxm(a,b,c,dim)
int dim;
double *a,*b,*c;

double *mxmc(a,b,c,dim)
int dim;
double *a,*b,*c;

double *mxv(y,a,x,dim)
int dim;
double *y,*a,*x;

double *mtxv(y,a,x,dim)
int dim;
double *y,*a,*x;

double *mt(a,b,dim)
int dim;
double *a,*b;

double *mtxm(a,b,c,dim)
int dim;
double *a,*b,*c;

double *mxmt(a,b,c,dim)
int dim;
double *a,*b,*c;
```

DESCRIPTION

mxm
set a = b c, i.e., form matrix product of b and c and saves in a. Matrices are assumed to be stored by ROWS not columns.

mxmc
form matrix product of b and c and saves in a. Matrices are assumed to be stored by COLUMNS not rows

mxv
multiply vector x by square matrix a and store at y. (y = a x). Matrices is assumed to be stored by ROWS not columns.

mtxv
multiply vector x by transpose of square matrix a and store at y. (y = a x). Matrix is assumed to be stored by ROWS not columns.

mt
gets the transpose of matrix b and stores in matrix a: i.e a = bT

mtxm
set a = bT c,i.e., form matrix product of b transpose and c and save in a. Matrices are assumed to be stored by ROWS not columns.

mxmt
set a = b cT, i.e., form matrix product of b and c transpose and saves in a. Matrices are assumed to be stored by ROWS not columns.

NAME

mx3ops, mvmul3, mtxv3, mshft3, mzero3, ident3, smmul3, mxmul3, mtxm3, mxmt3, mtran3, vadd3, svmul3, vsub3, cros, vnorm3, vunit3, vzero3, vdot3, vshft3

SYNOPSIS

```
subroutine mx3ops
double precision m1 (3,3)
double precision m2 (3,3)
double precision m3 (3,3)
double precision v1 (3)
double precision v2 (3)
double precision v3 (3)
double precision x

entry mvmul3 (m1,v1,v2)

entry mtxv3 (m1,v1,v2)

entry mshft3 (m1,m2)

entry mzero3 (m1)

entry ident3 (m1)

entry smmul3 (x,m1,m2)

entry mxmul3 (m1,m2,m3)

entry mtxm3 (m1,m2,m3)

entry mxmt3 (m1,m2,m3)

entry mtran3 (m1,m2)

entry vadd3 (v1,v2,v3)

entry svmul3 (x,v1,v2)

entry vsub3 (v1,v2,v3)

entry cros (v1,v2,v3)

entry vnorm3 (v1,x)

entry vunit3 (v1,v2)

entry vzero3 (v1)

entry vdot3 ( v1,v2,x)

entry vshft3 (v1,v2)
```

DESCRIPTION

mx3ops

Various 3 dimensional matrix and vector operations.

m1 double precision matrices (i/o)

m2 double precision matrices (i/o)

m3 double precision matrices (i/o)

v1 double precision vectors (i/o)

v2 double precision vectors (i/o)

v3 double precision vectors (i/o)

x scalar (input)

mvmul3
Does vector and matrix multiplication. (V2=M1*V1)

mtxv3
Multiplies vector by transpose of matrix. (V2=transpose(M1)*V1)

mshft3
Sets one 3x3 matrix equal to another. (M2=M1)

mzero3
Zeroes out a 3x3 matrix. (M1=0)

ident3
Changes a square matrix to a identity. (M1=I)

smmul3
Multiplies a scalar by a 3x3 matrix. (M2=x*M1)

mxmul3
Does matrix multiplication. (M3=M1*M2)

mtxm3
Does matrix multiplication, first matrix is transposed. (M3=transpose(M1)*M2)

mxmt3
Does matrix multiplication, second matrix is transposed. (M3=M1*transpose(M2))

mtran3
Does matrix transpose. (M2=transpose(M1))

vadd3
Does 3-vector addition. (V3=V1+V2)

svmul3
Does scalar and vector multiplication. (V2=x*V1)

vsub3
Does vector subtraction. (V3=V1-V2)

cros
Computes cross product. (V3=V1xV2)

vnorm3
Computes vector magnitude. (x=nor(V1))

vunit3
Normilizes the vector. (V2=unit(V1))

vzero3
Zeroes out each component of a vector. (V1=0)

vdot3
Computes the dot product of a vector. (c=V1*V2)

vshft3
Sets one 3-vector equal to another. (V2=V1)

NAME

mxops, mxmul, mvmul, mtxv, mxadd, mshift, mtran, mzero, imzero, rmzero, ident, vadd, vsub, vdot, vnorm, vunit, svmul

SYNOPSIS

```
subroutine mxops (i,j,p,rm1,im1,m1,m2,m3,m,mt,v1,v2,v3,x,v,mm)
integer i
integer j
integer p
real rm1 (i,j)
integer im1 (i,j)
double precision m1 (i,j)
double precision m2 (j,p)
double precision m3 (i,p)
double precision m (i,j)
double precision mt (j,i)
double precision v1 (j)
double precision v2 (i)
double precision v3 (i)
double precision x
double precision v (i)
double precision mm (i,j)

entry mxmul (i,j,p,m1,m2,m3)

entry mvmul (i,j,m1,v1,v2)

entry mtxv (i,j,m,v2,v1)

entry mxadd (i,j,m1,m,mm)

entry mshift (i,j,m1,m)

entry mtran (i,j,m1,mt)

entry mzero (i,j,m1)

entry imzero (i,j,im1)

entry rmzero (i,j,rm1)

entry ident (i,j,m1)

entry vadd (i,v,v2,v3)

entry vsub (i,v,v2,v3)

entry vdot (i,v,v2,x)

entry vnorm (i,v,x)

entry vunit (i,v,v2)

entry svmul (i,x,v,v2)
```

DESCRIPTION

mxops

Performs various matrix and vector operations.

i dimension index (input)

j dimension index (input)

p	dimension index (input)
rm1	real matrix (output)
im1	integer matrix (output)
m1	double precision matrices (input)
m2	double precision matrices (input)
m3	m3 double precision matrices (output)
m	double precision matrices (input/output)
mt	double precision matrices (output)
v1	double precision vectors (input/output)
v2	double precision vectors (input/output)
v3	double precision vectors (output)
x	scalar (output)
v	double precision vectors (input)
mm	double precision matrices (output)

mxmul
Performs matrix multiplication.

mvmul
Performs vector, matrix multiplication.

mtxv
Performs matrix calculation.

mxadd
Performs matrix addition.

mshift
Shifts a matrix.

mtran
Transposes a matrix.

mzero
Zeroes a double precision matrix.

imzero
Zeroes an integer matrix.

rmzero
Zeroes a real matrix.

ident
Performs matrix identity operation.

vadd
Performs vector addition.

vsub
Performs vector subtraction.

vdot
Performs dot product.

vnorm
Performs vector normalization.

vunit
Performs vector unitization.

svmul
Performs vector multiplication by scaler.

NAME

nrmlzq

SYNOPSIS

```
subroutine nrmlzq (q)
double precision q (4)
```

DESCRIPTION

nrmlzq

Normalizes i/o quaternion q for rot and mat entry points.

q quaternion to be normalized (input/output)

NAME

prt3mat, prt3vec, prt3tmat, fprtarray, prtarray

SYNOPSIS

```
int prt3mat(m,dscp)
double *m;
char *dscp;

int prt3vec(v,dscp)
double *v;
char *dscp;

int prt3tmat(m,dscp)
double *m;
char *dscp;

int fprtarray(fl,dim,row_length,fmt,a)
FILE *fl;
int dim,row_length;
char *fmt;
double *a;

int prtarray(dim,row_length,fmt,a)
int dim,row_length;
char *fmt;
double *a;
```

DESCRIPTION

The description field dscp is printed as a string followed by 1 newline.

prt3mat
display 3 by 3 matrix m to stdout in 21.14e format

prt3vec
display 3 vector v to stdout in 21.14e format

prt3tmat
display 3 by 3 transformation matrix m to stdout. A fixed point format is used with the assumption that the matrix is normalized.

fprtarray
print double precision array of dimension dim to specified file. row_length specifies the number of elements per line that are to be printed. If row_length is negative, each row is preceed by a count of the number of elements that have been thus far printed. A null value of row_length results in 5 elements per row being printed. fmt specifies a format with which to print a single element. If fmt is null, then the %15g format is used. Note that format should have a leading tab or space separator.

prtarray
as fprtarray except to standard out <stdout>

NAME

q2msub

SYNOPSIS

```
subroutine q2msub (q,q0,m)
double precision q (4)
double precision q0
double precision m (3,3)
```

DESCRIPTION

q2msub

Converts a quaternion into a rotational matrix.

q	quaternion to be converted (input)
q0	positive part of q(1) (input)
m	rotational matrix to be made (output)

NAME

qxq, qcxq, qxqc, qintp, qvrot, qvirot, qdot, imatq, matq, normq - quaternion operations

SYNOPSIS

```
double *qxq(q3,q1,q2)
QUATERNION q3,q1,q2;

double *qcxq(q3,q1,q2)
QUATERNION q3,q1,q2;

double *qxqc(q3,q1,q2)
QUATERNION q3,q1,q2;

double *qintp(q3,t3,q1,t1,q2,t2)
QUATERNION q3,q1,q2;
double t3,t1,t2;

double *qvrot(vout,q,v)
double vout[],v[];
QUATERNION q;

double *qvirot(vout,q,v)
double vout[],v[];
QUATERNION q;

double *qdot(qdt,q,w)
double qdt[];
QUATERNION q;
double w[];

double *imatq(q,t)
QUATERNION q;
double t[][];

double *matq(q,t)
QUATERNION q;
double t[][];

double *normq(q)
QUATERNION q;
```

DESCRIPTION

qxq
forms quaternion product, Sets q3 equal to q1 times q2.

qcxq
Sets q3 to q1 conjugate times q2.

qxqc
Sets q3 equal to q1 times q2 conjugate.

qintp
performs a linear interpolation between q1 and q2 defined at times t1 and t2 respectively to get the quaternion q3 defined at t3.

qvrot
Sets vout = qc x v x q; rotates the input vector to the system defined by the quaternion q.

qvirot
Sets vout = q x v x qc; rotates the input vector to the system defined by the quaternion q.

qdot
w angular velocity in body coordinates

compute time derivative of quaternion

imatq
Given the inverse of the matrix transformation from F to G, compute the quaterion of the transformation from F to G. Code converted directly from the Pascal of the Orbital Flight Simulation Utility Software Unit Specifications (S. W. Wilson).

matq
call imatq and conjugate the quaternion

normq
call imatq and conjugate the quaternion

NAME

qrot

SYNOPSIS

```
subroutine qrot (q,q0,v,x)
double precision q (4)
double precision q0
double precision v (3)
double precision x (3)
```

DESCRIPTION

qrot

Performs rotations of quaternions.

q	quaternion (input)
q0	quaternionn (input)
v	vector (input)
x	vector (output)

NAME

qtnops, qxq, qcxq, qxqc, qtom, qtoim, rot, irot, imatq, matq

SYNOPSIS

```
subroutine qtnops
double precision m (3,3)
double precision q (4)
double precision q0
double precision q1 (4)
double precision q2 (4)
double precision q3 (4)
double precision v (3)
double precision x (3)

entry qxq (q1,q2,q3)

entry qcxq (q1,q2,q3)

entry qxqc (q1,q2,q3)

entry qtom (q,m)

entry qtoim (q,m)

entry rot (q,v,x)

entry irot (q,v,x)

entry imatq (m,q)

entry matq (m,q)
```

DESCRIPTION

qtnops

Performs various quaternion operations. Determines an orthogonal matrix represented by a quaternion. Transforms the components of a vector in one coordinate system to those in another coordinate system rotated with respect to the first in a way defined by the versor q.

m	orthogonal matrix (output)
q	quaternion (input)
q0	local variable
q1	quaternion (input)
q2	quaternion (input)
q3	quaternion (output)
v	input vector (input)
x	output vector (output)

qxq

Sets q3 equal to q1 time q2.

qcxq

Sets q3 to q1 conjugate times q2.

qxqc

Sets q3 equal to qi times q2 conjugate.

qtom
Converts q to matrix m.

qtoim
Computes matrix of inverse rotation represented by q.

rot
Uses q to transform v to x.

irot
Uses conjugate of q to transform v to x.

imatq
Converts matrix m to inverse quaternion q.

matq
Converts matrix m to quaternion q.

COMMENTS
1) The input quaternion must be normalized. 2) Input matrix must be orthonormal.

NAME

rmxm, rmxmc, rmxv - double array operations

SYNOPSIS

```
double *rmxm(a,b,c,rowb,colb,colc)
int rowb,colb,colc;
double *a,*b,*c;

double *rmxmc(a,b,c,rowb,colb,colc)
int rowb,colb,colc;
double *a,*b,*c;

double *rmxv(y,a,x,dim)
int dim;
double *y,*a,*x;
```

DESCRIPTION

rmxm
set a = b c, i.e., form matrix product of b and c and saves in a. Matrices are assumed to be stored by ROWS not columns.

rmxmc
form matrix product of b and c and saves in a. Matrices are assumed to be stored by COLUMNS not rows

rmxv
set y = a y, i.e., form product of matrix a and vector b and save in y. Matrix is
assumed to be stored by ROWS not columns.

NAME

sign - sign operation

SYNOPSIS

```
double sign(a,b)
double a,b;
```

DESCRIPTION

sign

perform sign change operation

NAME

symxv

SYNOPSIS

```
subroutine symxv (ds,dx,s,x,y)
integer ds
integer dx
double precision s (ds)
double precision x (dx)
double precision y (dx)
```

DESCRIPTION

symxv

Multiplies a symmetric matrix stored lower triangularly by rows times a vector.

ds dimension of array containing symmetric matrix (input)

dx dimension of vector (input)

s array containing symmetric matrix: lower triangular portion stored by rows (input)

x vector to be multiplied (input)

y product vector (output)

NAME

tenrnd

SYNOPSIS

```
subroutine tenrnd (place,x)
integer place
real x
```

DESCRIPTION

tenrnd

Rounds off values.

place decimal place in which to round. 1 rounds to second digit, etc (input)

x value to round (input/output)

COMMENTS

Rounds small values to zero.

NAME

vadd, vaddto, vdist, vdot, vfadd, vfaddto, vsub, vfmul, vrss, vset, vzero, vrunit - double array operations

SYNOPSIS

```
double *vadd(a,b,c,dim)
int dim;
double *a,*b,*c;

double *vaddto(a,b,dim)
int dim;
double *a,*b;

double vdist(a,b,dim)
int dim;
double *a,*b;

double vdot(a,b,dim)
int dim;
double *a,*b;

double *vfadd(a,b,factor,c,dim)
int dim;
double *a,*b,*c,factor;

double *vfaddto(a,factor,b,dim)
int dim;
double *a,*b,factor;

double *vsub(a,b,c,dim)
int dim;
double *a,*b,*c;

double *vfmul(a,factor,b,dim)
int dim;
double *a,*b,factor;

double vrss(a,dim)
int dim;
double *a;

double *vset(a,b,dim)
int dim;
double *a,*b;

double *vzero(a,dim)
int dim;
double *a;

double vrunit(unit,a,b,dim)
int dim;
double *unit,*a,*b;
```

DESCRIPTION

vadd
array addition: set a = b + c

vaddto
add array to array b, i.e. a = a+b.

vdist
  returns the root sum square or Euclidean distance between the double arrays
  a and b

vdot
  returns dot product of a and b

vfadd
  set a = b + factor*c

vfaddto
  multiply array b by factor and add result to array a. i.e., a = a + f*b.

vsub
  set a = b - c

vfmul
  set a = factor*b

vrss
  returns root sum square norm of a

vset
  set array a equal to b

vzero
  set array a equal to 0

vrunit
  returns the root sum square or Euclidean distance between the double arrays
  a and b and stores unit vector from a to b in unit.

NAME

addANLitem, prtANLitem, prtANList - Adress and Name lists

SYNOPSIS

```
int addANLitem(adress,name)
char *adress,*name;

int prtANLitem(adress,string)
char *adress,*string;

void prtANList()
```

DESCRIPTION

addANLitem

Add entry to adress/name list. Returns item number (starting at 1) if okay. Returns 0 if list is full

prtANLitem

searches adress/name lists and displays name and adress to stdout. Returns item number (starting at 1) if item found, 0 if not. If string is not null, then the string is printed in front of the name followed by a colon and a space.

prtANList

display adress/name list to stdout

NAME
datime

SYNOPSIS
```
subroutine datime (unit)
integer unit
```

DESCRIPTION
datime
Displays processor name, version, date and time.

unit output unit (input)

COMMENTS
Input processor name must be less than 57 characters.

NAME

GNLgetName, fprtgnlist, Make_GNLIST, Add_GNList - manipulation of namelist structures

SYNOPSIS

```
char *GNLgetName(adress,nlist)
GNL_ITEM *nlist;
char *adress;

void fprtgnlist(file,nlist,dscp)
FILE *file;
GNL_ITEM *nlist;
char *dscp;

GNLIST *Make_GNLIST(list,name,maxlength,items)
GNLIST *list;
char *name,*items;
int maxlength;

int Add_GNList(adress,dscp,list)
char *adress,*dscp;
GNLIST *list;
```

DESCRIPTION

GNLgetName
find entry in namelist with given adress and return the name. Null (0) name pointer is returned if entry not found.

fprtgnlist
display namelist to stdout. List is preceeded by description dscp and consists of octal adress, name followed by newline

Make_GNLIST
initialize and set up a namelist of adresses and names. Space is allocated for the list structure list and the list buffer items if the pointers are null. Else the given space is used. Errors result in a null pointer being returned.

Acd_GNList
add entry to Namelist. Returns current number of items in list. If list is full, -1 is returned.

NAME

iegrss, gwarn, gerror

SYNOPSIS

```
subroutine iegrss (p,stdout,filout)
character p
integer stdout
integer filout

entry gwarn (p,stdout,filout)

entry gerror (p,stdout,filout)
```

DESCRIPTION

iegrss

Gwarn and gerror display warnings and errors and keep the count of warnings and errors encountered.

p message string (input)

stdout unit for terminal print (input)

filout unit for print output (input)

gwarn

Displays warning and accummulates counts.

gerror

Displays error message and accummulates counts.

NAME

ingrss, ngrss, awarn, aerror, egrss

SYNOPSIS

```
subroutine ingrss
character p

entry ngrss

entry awarn (p)

entry aerror (p)

entry egrss
```

DESCRIPTION

ingrss

Contains entry points that provide for the start and stop of all RELBET processors. Ingrss is a generic starting point for all routines. It displays the processor name, version, and the current date and time via a call to the routine datime. It also reads the first input image that specifies the execution mode and the calling processor for the subordinate execution mode. Egrss is a generic stopping point for relbet routines. It displays a termination message and schedules the theexecution of the calling processor if the execution was in the subordinate mode. Awarn and aerror display warnings and errors and keep the count of warnings and errors encountered.

p message string (input)

ngrss

displays name and version tc terminal.

awarn

Displays warning and accummulates counts.

aerror

Displays error message and accummulates counts.

egrss

Displays termination messages.

NAME

ioschk

SYNOPSIS

```
subroutine ioschk (u,ios,s)
integer u
integer ios
integer s
```

DESCRIPTION

ioschk

Checks the i/o status word for a file and sets status flag to 0=okay, -1=eof, -11=error. if eof or error is detected, a message is sent to terminal and print displays.

u unit number.

ios i/o status word from iostat clause.

s output status word: 0=okay, -1=end of file, -11=error

NAME

msgdsp, xdsp, xdsp2, starz, newpg

SYNOPSIS

```
subroutine msgdsp (x,u1,u2)
character x
integer u1
integer u2

entry xdsp (x,u1)

entry xdsp2 (x,u1,u2)

entry starz (u1)

entry newpg (u1)
```

DESCRIPTION

msgdsp

Displays messages to specified units. Xdsp display character string to specified unit. Xdsp2 displays character string to two units unless they are the same. The string is up to a '$' or to 80 characters if no '$' is encountered. Starz display asteriscs and newpg flips a page.

x string to display (input)

u1 1st display unit (input)

u2 2nd display unit (input)

xdsp

Displays string x to unit u1.

xdsp2

Displays string x to unit u1,u2.

starz

Displays a line of asteriscs.

newpg

Flips a page in display file.

NAME

fskip_lines, skip_lines, fprtStars, prtStars - display utilities

SYNOPSIS

```
fskip_lines(fdes,lines)
FILE *fdes;
int lines;

skip_lines(lines)
int lines;

fprtStars(fdes,nlines)
FILE *fdes;
int nlines;

prtStars(nlines)
int nlines;
```

DESCRIPTION

fskip_lines
skips specified number of lines by print newlines to fdes

skip_lines
send specified number of newlines to stdout

fprtStars
Prints nlines of 60 stars followed by a cr to specified stream

prtStars
Prints nlines of 60 stars followed by a cr to stdout

NAME

addptr, rmptr, makeplist, freeplist - maintain pointer lists

SYNOPSIS

```
int addptr(plist,ptr)
PTRLIST *plist;
char *ptr;

int rmptr(plist,ptr)
PTRLIST *plist;
char *ptr;

PTRLIST *makeplist(max,list)
int max;
char **list;

void freeplist(ptrlist)
PTRLIST *ptrlist;
```

DESCRIPTION

Removes and adds entries to pointer list. If entry not in list remove has no effect. If entry in list then add has no effect. returns the number of remaining pointers in the list unless add would result in the max being exceeded in which case -1 is returned The end of the list is always marked by a null pointer and if these functions are used to edit the list there will be no duplication. removes only remove the first occurence from the beginning of the list and then slide other entries down

addptr

add ptr to plist returning location of ptr in list list is searched from beginning until null pointer or desired pointer is found. The ptr is added is there is room and skipped if already there If no room then the lenth of the list is set to the max and -1 is returned.

rmptr

remove ptr to plist returning number of list elements left list is searched from beginning until null or desired ptr found any remaining list elements are then shifted down and the number of pointers remaining is returned

makeplist

creates a pointer list structure and returns a pointer to it. The list storage is also allocated and the contents in the input list are stored in this list up to the designated maximum max. If the input list is a null pointer then no initialization occurs. The new list structure thus has its private list independent of the input list. Errors result in a null list pointer being returned

freeplist

frees storage for structure ptrlist. Dire events may happen if the structure was not made with malloc or makeplist value of 0 is always returned

NAME

get_rate_table_value - fetch value from rate table

SYNOPSIS

```
double get_rate_table_value(time,table)
double time;
TIMELINE_PTR table;
```

DESCRIPTION

get_rate_table_value

fetch value from rate table. Value is interpolated even if time is before start of table.

NAME

Stat_Msg, Error_Count, Warning_Count, StatErrExit, fprtFinish, prtFinish, err_hpcode, err_code, berror, bwarn, addefile, rmefile, setMaxErr - status message display utilities

SYNOPSIS

```
char Stat_Msg[];

int Error_Count;

int Warning_Count;

int StatErrExit;

void fprtFinish(fdes)
FILE *fdes;

void prtFinish()

void err_hpcode(f)
FILE *f;

void err_code(f)
FILE *f;

void berror(msg,code)
char *msg;
int code;

void bwarn(msg,code)
char *msg;
int code;

int addefile(file)
FILE *file;

int rmefile(file)
FILE *file;

int setMaxErr(err,warn,exfunc)
int err,warn;
int (*exfunc)();
```

DESCRIPTION

Displays error and warning messages to files specified by a list of status display files.

The message is used as a format in a call to fprintf so that the integer code may be incorporated in the message. If the 1st character of the message is a semicolon ';' then the current system error code and its description is also written. If the first character is a colon ':' the the HP code errinfo is written a s well as the system code. In these cases the initial letter is dropped. Messages are written to a list of files default for which is stderr and stdout however they may be changed by the functions addefile and rmefile. A maximum of 3 files may be specified, other ones are ignored.

The function setMaxErr sets the maximum number of errors and warnings that are issued before a call to a termination routine is made. Null or negative values (default) result in no max error count check. The termination function Exit is specified by the call. A null value results in the

routine MaxErrExit being used for the Exit.

fprtFinish
writes final message to specified stream. Message includes error and warning counts.

prtFinish
writes final message to stderr

err_hpcode
displays HP error code errinfo(2) to file

err_code
displays system error code to file see errno(2) for details. The code and description are both displayed.

berror
prints message to all the error displays and increments the error count

bwarn
prints message to all the error displays and increments the warning count

addefile
add file to list of display files. returns total number of display files. -1 returned if file cannot be added

rmefile
remove file from display files. returns total number of display files.

setMaxErr
specifies the max error count and max warning count. The function exfunc is invoked if counts exceeded. The default of Exit (exit) is set if exfunc is null.

NAME

Save_Str_Buf - storing strings

SYNOPSIS

```
char *Save_Str_Buf()
```

DESCRIPTION

Save_Str_Buf

stores contents of Str_Buf and returns pointer to saved string. Allocates room for storage if neccesary.If room cannot be allocated, then as much of the string as possible is stored. Note that the last character of Str_Buf (at index Str_Buf_Size) is set to the null character. Thus a maximum of Str_Buf_Size characters may be stored. The string is stored until a null character is found thus the terminating null should be included. A null pointer is returned if the string cannot be stored.

NAME

bsearch_timeline, lsearch_timeline, isearch_timeline, MakeTimeLine, fprtTimeLine - timeline information

SYNOPSIS

```
char *bsearch_timeline(time,timeline)
double time;
TIMELINE *timeline;

char *lsearch_timeline(time,timeline)
double time;
TIMELINE *timeline;

char *isearch_timeline(index,timeline)
int index;
TIMELINE *timeline;

TIMELINE *MakeTimeLine(timeline,table,n_items,time_byte,rec_size)
TIMELINE *timeline;
char *table;
int n_items,time_byte,rec_size;

void fprtTimeLine(timeline)
TIMELINE *timeline;
```

DESCRIPTION

bsearch_timeline
performs binary search of timeline table returning pointer to last entry with a time less than or equal to the specified time. The current_item member of the timeline structure is set as a side effect. If time is before initial entry, null pointer (0) is returned and the current_item is set to -1.

lsearch_timeline
performs linear search of timeline table returning pointer to last entry with a time less than or equal to the specified time. The search begins at item indicated by current_item member. If this member is negative, a binary search is conducted instead. Error values and side effects are the same as for bsearch_timeline.

isearch_timeline
performs an inspection of timeline table returning pointer to entry positioned at the input index from the beginning of the table. The search begins at item indicated by the index. If this number is out of range of the table an error value is returned

MakeTimeLine
Sets up a timeline structure. If either the timeline or table pointer is null, then space is allocated via malloc and calloc. Thus these structures may be freed. Errors result in null pointer being returned

fprtTimeLine
displays timeline structure to stdout

NAME

tpdsp, tdsp, pdsp, tpstrz, tstrz, pstrz

SYNOPSIS

```
subroutine tpdsp (msg)
character msg

entry tdsp (msg)

entry pdsp (msg)

entry tpstrz

entry tstrz

entry pstrz
```

DESCRIPTION

tpdsp
Displays messages to print and terminal units.

msg message string (input/output)

tdsp
Displays message to terminal unit.

pdsp
Display message to nominal print unit out.

tpstrz
Displays string of asteriscs to terminal and print file.

tstrz
Displays string of asteriscs to terminal.

pstrz
Displays string of asteriscs to print file.

NAME
mktape

SYNOPSIS
program mktape

DESCRIPTION
mktape
This program prompts user for inputs required to process an hp9000 created binary file with mixed type data words. Output is a univac FORTRAN V readable tape in the same mixed order.

NAME

UNVOUT, getbits, put72, putchr - routine to build UNIVAC tape file

SYNOPSIS

```
UNVOUT(rec,fmt,bufout)
int *rec;
int *fmt;
int *bufout;

getbits(x,p,k)
unsigned x;
int p,k;

put72(w,b,bo)
int *bo;
char *w,*b;

putchr(w,buf,bits)
int *bits;
char w,*buf;
```

DESCRIPTION

UNVOUT

Routine to build UNIVAC tape file. This routine is machine depended

rec — array containing output data

fmt — array containing output record specifications

bufout — output buffer

getbits

function to get bits

put72

function to put 72 bits in output buffer

b — define as char to eliminate shift

putchr

function to put character in output buffer

NAME

ddna

SYNOPSIS

```
subroutine ddna (x,t,npts,maxord,ddn)
double precision x (*)
double precision t (*)
integer npts
integer maxord
double precision ddn (10)
```

DESCRIPTION

ddna

Variate divided difference noise analysis.

x	data (input).
t	times for each datum (input).
npts	number of elements in x,t (input).
maxord	maximum order of differences to be used (input).
ddn	variate difference noise values up to 10th order (output).

NAME
ddnois

SYNOPSIS
program ddnois

DESCRIPTION
ddnois
Driver for noise analysis using variate divided difference.

NAME
nnois

SYNOPSIS
subroutine nnois (fnam,fnamo,nobs,obaray,obdelt)

DESCRIPTION
nnois
This subroutine fetches the user inputs for the common blocks needed by this program input file name output file name number of observations to process array of frame ids array of time interval lengths w/r to obs type to process for noise computation

NAME

dspxq, xqdsp1, xqdsph, shoxq

SYNOPSIS

```
subroutine dspxq (t,is,if,d,stat)
double precision t
integer is
integer if
double precision d (6)
integer stat

entry xqdsp1 (stat)

entry xqdsph (t)

entry shoxq (t,xpg,if,d,is)
```

DESCRIPTION

dspxq

Entry points for initialization and performing xqdsp output.

t time tag (input/output)

is scale option: 1=lllvvv,2=lnaaaa,3=lvaaww,4=aaawww where l=leng,v=vel,a=ang,n=n.c.,w=ang rate (input)

if parameter index(input/output)

d data buffer(input/output)

stat status flag (output)

xqdsp1

Initializes compute flags and check for errors.

xqdsph

Initializes display heading for parameter group.

shoxq

Performs output for xqkmpt.

NAME
dxcmp

SYNOPSIS
block data dxcmp

DESCRIPTION
dxcmp
This block data routine initializes the common used by this program

NAME
dxqdsp

SYNOPSIS
block data dxqdsp

DESCRIPTION
dxqdsp
This block data routine initializes the common used by this program

NAME
gdisp

SYNOPSIS
program gdisp

DESCRIPTION
gdisp
Executive for generic display.

NAME

gdspop

SYNOPSIS

```
subroutine gdspop (stat,rec,count,d)
integer stat
integer rec
integer count
double precision d (0:200)
```

DESCRIPTION

gdspop

Solicits options for display of files by frames.

stat status word (i/o)

rec record number (input/output)

count number of frames to display (input/output)

d record data (output)

NAME
gndsp

SYNOPSIS
```
subroutine gndsp (stat,idz)
integer stat
integer idz (20)
```

DESCRIPTION
gndsp
Displays selected frames of a standard format file.

stat status word (i/o)

idz idz of parameters to display (input/output)

NAME

lgint, lgfac

SYNOPSIS

```
subroutine lgint (m,n,td,t,eval,f,z)
integer m
integer n
double precision eval (m,n)
double precision f (m,n)
double precision t (n)
double precision td
double precision z (m)

entry lgfac (m,n,t,eval,f)
```

DESCRIPTION

lgint

lagrangian interpolation (lgint) of n m-vectors. set up of factors (lgfac)

m	length of each entry in double words
n	order of interpolation(input)
eval	array n-point buffer (input)
f	array of interpolation factors (input/output)
t	array of time tags for entries
td	desired time (input)
z	output array (output)

lgfac

set up of factors

COMMENTS

if time tag is within 14 digits of span, the no interpolation output is set to nearby point. Factors must be precomputed by lgfac or similar routine

NAME
nxcmp

SYNOPSIS
subroutine nxcmp

DESCRIPTION
nxcmp
This subroutine fetches the user inputs for the common blocks needed by this program

NAME
nxqdsp

SYNOPSIS
subroutine nxqdsp

DESCRIPTION
nxqdsp
This subroutine fetches the user inputs for the common blocks needed by this program

NAME
xcmpar

SYNOPSIS
program xcmpar

DESCRIPTION
xcmpar
driver for comparison of trajectories. output may be 2 trajectories or relative trajectory

NAME
xqdsp

SYNOPSIS
program xqdsp

DESCRIPTION
xqdsp
Driver for display of trajectory and attitude information.

NAME

xqkmpt, xqkmp1, xqkmp2

SYNOPSIS

```
subroutine xqkmpt (t,x1,x2,q,w,stat)
double precision t
double precision x1 (6)
double precision x2 (6)
double precision q (4)
double precision w (3)
integer stat

entry xqkmp1 (stat)

entry xqkmp2 (t,x1,x2,q,w,stat)
```

DESCRIPTION

xqkmpt

Computes display and plot parameters for given time in xqdsp.

t	time tag (input)
x1	1st vehicle state (input)
x2	2nd vehicle state (input)
q	attitude quaternion (input)
w	attitude rate (input)
stat	status flag (output)

xqkmp1

Initializes compute flags and check for errors.

xqkmp2

Initializes, computes, and displays required parameters.

NAME

obcode

SYNOPSIS

```
subroutine obcode (nam,id,pxclud)
character*4 nam
integer id
integer pxclud
```

DESCRIPTION

obcode

Decodes frame id for observation file and checks whether the observation has been excluded.

nam obs frame id (input)

id data type id (output)

pxclud exlcusion flag (output)

NAME

rtangl

SYNOPSIS

```
subroutine rtangl (xtsi,ua,ub,g,gr,gv)
double precision xtsi (6)
double precision ua (3)
double precision ub (3)
double precision g
double precision gr (3)
double precision gv (3)
```

DESCRIPTION

rtangl

Computes angles and partials for angle observations whose tangent is ratio of the projection of a range vector onto two sensor axes. Such angles include rendezous radar roll and shaft, and coas and startracker horizontal and vertical angles. tan(g) = r*ua / r*ub

xtsi inertial target relative to sensor position and velocity(input)

ua first unit axis vector (input)

ub second unit axis vector (input)

g observation angle (output)

gr partial of angle with respect to target position (output)

gv partial of angle with respect to target velocity (output)

COMMENTS

1) Assumes that the partials are initialized to zero before the call to this routine. 2) Assumes that the angle rates have been compensated for the angular motion of the observing vehicle. 3) The axes vectors must be normalized and have the correct sense. 4) Thus they are not neccessarily the unit vectors for the sensor.

NAME

tangle

SYNOPSIS

```
subroutine tangle (xtsi,ua,ub,g,gr)
double precision xtsi (6)
double precision ua (3)
double precision ub (3)
double precision g
double precision gr (3)
```

DESCRIPTION

tangle

Computes angles and partials for angle observations whose tangent is ratio of the projection of a range vector onto two sensor axes. Such angles include rendezous radar roll and shaft, and coas and startracker horizontal and vertical angles. tan(g) = r*ua / r*ub

xtsi inertial target relative to sensor position and velocity (input)

ua first unit axis vector (input)

ub second unit axis vector (input)

g observation angle (output)

gr partial of angle with respect to target position (output)

COMMENTS

1) Assumes that the partials are initialized to zero before the call to this routine. 2) The axes vectors must be normalized and have the correct sense. 3) Thus they are not neccessarily the unit vectors for the sensor.

NAME

xcoas, coash, coasv, cinit, cbugs

SYNOPSIS

```
subroutine xcoas (g,gr)
double precision g
double precision gr (3)

entry ccash (g,gr)

entry coasv (g,gr)

entry cinit

entry cbugs
```

DESCRIPTION

xcoas

Computes coas angle observations and partials.

g — angle observation (output)

gr — partial of observation with respect to the target inertial position (output)

coash

Computes horizontal angle and required partials.

coasv

Compute vertical angle and required partials.

cinit

Initializes.

cbugs

Writes debug information.

NAME

xradar, range, ranrat, shaft, sftrat, roll, rolrat, trnion, trnrat, pitch, pchrat

SYNOPSIS

```
subroutine xradar (g,gr,gv)
double precision g
double precision gr (3)
double precision gv (3)

entry range (g,gr)

entry ranrat (g,gr,gv)

entry shaft (g,gr)

entry sftrat (g,gr,gv)

entry roll (g,gr)

entry rolrat (g,gr,gv)

entry trnion (g,gr)

entry trnrat (g,gr,gv)

entry pitch (g,gr)

entry pchrat (g,gr,gv)
```

DESCRIPTION

xradar

Computes rendevous radar observations and partials.

g — Computesd observation (output)

gr — partial of observation with respect to target vehicle inertial position (output)

gv — partial of observation with respect to target vehicle inertial velocity (output)

range

Computes range.

ranrat

Computes range rate.

shaft

Computes shaft.

sftrat

Computes shaft rate.

roll

Computes roll.

rolrat

Computes roll rate.

trnion
Computes trunnion.

trnrat
Computes trunnion rate.

pitch
Computes pitch.

pchrat
Computes pitch rate.

COMMENTS
Assumes rates have been compensated for observing frame angular motion.

NAME

xtrack, trackh, trackv, stinit

SYNOPSIS

```
subroutine xtrack (itrack,g,gr)
integer itrack
double precision g
double precision gr (3)

entry trackh (itrack,g,gr)

entry trackv (itrack,g,gr)

entry stinit
```

DESCRIPTION

xtrack

Computes star tracker angle observations and partials.

itrack star tracker sensor id (input)

g angle observation (output)

gr partial of observation with respect to the target inertial position (output)

trackh

Computes horizontal angle and required partials.

trackv

Computes vertical angle and required partials.

stinit

Initializes.

NAME
ascale

SYNOPSIS
program ascale

DESCRIPTION
ascale
Determines minima and maxima of specified parameters in specified files.

NAME
automx

SYNOPSIS
subroutine automx

DESCRIPTION
automx
Sets minimum and maximum of axes based on parameter min//max.

COMMENTS
Considers only those curves that are designated.

NAME
dplot

SYNOPSIS
block data dplot

DESCRIPTION
dplot
This block data routine initializes the common used by this program

NAME

mmxchk

SYNOPSIS

```
subroutine mmxchk (i,pmmx,mmx,rchck)
integer i
integer pmmx
real mmx (2)
real rchck
```

DESCRIPTION

mmxchk

Checks value against min/max. Value is omitted, or set to min/max depending upon input flag.

i delete flag (set to -1 if point deletes else left alone) (output)

pmmx min/max options for output parameters (input) 2=recompute,+1=set to within input minmax, -1=omit if outside input minmax,-3=clip if outside file minmax, else=set to file minmax last entry specifies

mmx array of minimum and maximum values for output parameters.(input/output)

rchck test value for indepenoent parameter (input/output) note that is in internal units.

NAME
nplot

SYNOPSIS
subroutine nplot

DESCRIPTION
nplot
This subroutine fetches the user inputs for the common blocks needed by this program

NAME
plotx

SYNOPSIS
program plotx

DESCRIPTION
plotx
Main driver for graphic routines.

NAME

pltnpt, grfnpt

SYNOPSIS

```
subroutine pltnpt

entry grfnpt
```

DESCRIPTION

pltnpt

obtain general inputs for plot routine and initialize various parameters such as labels, titles, axes, curves,time default, and parameter id.

grfnpt

obtain graph specific input

NAME

q2crvz, q2crv, q2ecrv, q2lsto, q2legn

SYNOPSIS

```
subroutine q2crvz (ikurve,i,j)
integer ikurve
integer j
integer i

entry q2crv (ikurve)

entry q2ecrv (ikurve)

entry q2lsto (ikurve,i,j)

entry q2legn (i)
```

DESCRIPTION

q2crvz

Entry points for 2d graphics. Sets up preliminaries for curve plot, finish drawing curve, store legend information, and display legend.

ikurve curve index (input/output)

j number of calls for current curve.(input)

i current count of how many times curve segment has been drawn . (input)

q2crv

Sets up preliminaries for curve plot.

q2ecrv

Finishes drawing curve.

q2lsto

Stores legend information.

q2legn

Displays legend.

NAME

q2draw

SYNOPSIS

```
subroutine q2draw (x,nx,imark,nlbl,ncnt)
real x (1000,2)
integer nx
integer imark
integer nlbl
integer ncnt
```

DESCRIPTION

q2draw

Entry points for 2d graphics. Specifies the curves, lines and nlabels.

x buffer for values (input)

nx number of points in array (input)

imark mark options for plot curve (input/output)

nlbl label flag for plot curves (input) <0 means label every nth point starting at 1 >0 means label every nth point using file sequence no.

ncnt current index count at beginning of curve (input/output)

NAME

q2grph, q2fnsh

SYNOPSIS

subroutine q2grph

entry q2fnsh

DESCRIPTION

q2grph

Entry points for 2d graphics. Specifies the general layout for the plot. Sets and specifies the page size, grace area, subplot area, heading, angles x and y axes, labels ...etc. Defines physical set up of plot.

q2fnsh

Finishes up a plot.

NAME

qdevin, qdevrl

SYNOPSIS

```
subroutine qdevin

entry qdevrl
```

DESCRIPTION

qdevin

Initializes and releases plot device.

qdevrl

Releases devise.

NAME

qpintx

SYNOPSIS

```
subroutine qpintx (krv,stat)
integer krv
integer stat
```

DESCRIPTION

qpintx

Initializes and fetches data for plotting.

krv curve id (input)

stat file io status (input/output)

NAME
qplot

SYNOPSIS
subroutine qplot

DESCRIPTION
qplot
Draws a plot: broken out for segmentation purposes.

NAME

qplt2d

SYNOPSIS

subroutine qplt2d

DESCRIPTION

qplt2d

Initializes variables such as stop flag, terminal, curves, plotting array and finishes up curves.

NAME

qpxget

SYNOPSIS

```
subroutine qpxget (x,pcont)
real x (1000,2)
integer pcont
```

DESCRIPTION

qpxget

Initializes and fetches data for plotting.

x output array (output)

pcont status flag (input/output) o=last request done,>0 continue with request,<0 error

NAME

qslptn, qrlptn

SYNOPSIS

```
subroutine qslptn (dotdsh)
integer dotdsh

entry qrlptn (dotdsh)
```

DESCRIPTION

qslptn

Entry points to set and reset line patterns.

dotdsh line type (input/output)

qrlptn

Finishes up drawing curve.

NAME
sclset

SYNOPSIS
```
subroutine sclset (rmin,rmax,step)
real rmin
real rmax
real step
```

DESCRIPTION
sclset
Adjusts input rmin/rmax value to integral steps. The step size is chosen to give approx 10 steps form min to max.

rmin rminimum value (input)

rmax rmaximum value (input)

step step size (output)

NAME

a32t36

SYNOPSIS

```
subroutine a32t36 (temp,ctemp)
integer temp (3)
integer ctemp (3)
```

DESCRIPTION

a32t36

Move 32 bits to 36 bits.

temp input buffer (input)

ctemp temporary buffer(output)

COMMENTS

Must not be compiled with range checking.

NAME

cmvbit

SYNOPSIS

```
subroutine cmvbit (n,in,iw,ib,itemp,ow.ob)
integer n
integer in (3)
integer iw
integer ib
integer itemp (3)
integer ow
integer ob
```

DESCRIPTION

cmvbit

This is a compound version of getbit, which allows specification of ow and ob, thus permitting restack of one array into anotner. This program extract n consecutive bits from the array in, beginning with the initial word iw and initial bit ib, and store in the ow,ob portion of tne array itemp. n can be any number of bits, or 18, 36, or 72 for univac created records. itemp is treated as 32 bit worc for current applications.

n	number of bits (input)
in	initial array (input)
iw	initial worc (input)
ib	initital bits (input)
itemp	temporary storage (output)
ow	output word (output)
ob	output bits (output)

COMMENTS

Must not be compiled witn range checking.

NAME

comptw

SYNOPSIS

```
subroutine comptw (rdata,itype,ktypes,kwords,num)
real rdata (3)
integer*4 num (50)
integer*4 itype (50)
integer*4 ktypes
integer*4 kwords
```

DESCRIPTION

comptw

This subroutine computes the data types and the number of words to be written in each data type.

rdata data

num number of data

itype types

ktypes data types

kwords number of words

NAME
dprod

SYNOPSIS
block data dprod

DESCRIPTION
dprod
This block data routine initializes tne common used by this program

NAME

dxset

SYNOPSIS

```
subroutine dxset (spg,dx,np)
integer spg (20)
character*8 dx (200)
integer np
```

DESCRIPTION

dxset

Forms dictionary for RELBET special products.

spg parameter group flags (input)

dx dictionary buffer (output)

np number of parameters in output (1+number of parameters)

NAME
getspg

SYNOPSIS
```
subroutine getspg (dindex,t,x1,x2,q,w,spg,d,count)
integer count
integer dindex
double precision t
double precision x1 (6)
double precision x2 (6)
double precision q (4)
double precision w (3)
double precision d (200)
integer spg (20)
```

DESCRIPTION
getspg
Computes required RELBET product parameters.

count	countflag (output)
dindex	index for tracking intervals (input)
t	time tag (input)
x1	1st vehicle state (input)
x2	2nd vehicle state (input)
q	attitude quaternion (input)
w	attitude rate (input)
d	data buffer (output)
spg	parameter group flags (input)

NAME

hptou5

SYNOPSIS

```
subroutine hptou5 (iop,inrec,maxrec,npar)
integer iop
integer inrec (400)
integer maxrec
integer npar
```

DESCRIPTION

hptou5

Writes the header record, dictionary record and data records on to the tape. Output is a univac FORTRAN V readable tape in the same mixed order.

iop type of calls (input/output) 1 initialization call,open tape drive,and write header 2 dictionary call 3 data call 4 wrap up call

inrec input record

maxrec maximum record count(output)

npar number of parameters (input)

NAME
nprod

SYNOPSIS
subroutine nprod

DESCRIPTION
nprod
This subroutine fetches the user inputs for the common blocks needed by this program

NAME

NTCLOSE, NTEOF, NTFILE, NTOPEN, NTREAD, NTRITE, NTRW, NTRWRL, NTBLCK - FORTRAN callable routines that perform various tape manipulations

SYNOPSIS

```
void NTCLOSE(chanl,status)
int *chanl;
int *status;

void NTEOF(chanl,status)
int *chanl;
int *status;

void NTFILE(chanl,fcount,status)
int *chanl;
int *fcount;
int *status;

void NTOPEN(chanl,drive,status)
int *chanl;
int *drive;
long int *status;

void NTREAD(chanl,nobytes,buffer,status,overflow)
int *chanl;
int *nobytes;
char *buffer;
long int *status;
long *overflow;

void NTRITE(chanl,nobytes,buffer,status)
int *chanl;
int *nobytes;
char *buffer;
int *status;

void NTRW(chanl,status)
int *chanl;
int *status;

void NTRWRL(chanl,status)
int *chanl;
int *status;

void NTBLCK(chanl,fcount,status)
int *chanl;
int *fcount;
int *status;
```

DESCRIPTION

These functions are FORTRAN callable routines that perform various tape manipulations. The FORTRAN calling sequence for NTCLOSE,NTEOF,NTRW and NTRWRL is

```
call NAME(chanl,status)
```

where all the arguments are integers. The chanl is the file descriptor obtained from a call to ntopen. If the request is performed successfully, status returns a zero, otherwise status is positive and contains the system error number (errno). If an error occurrs, an error message is printed on the standard error device (stderr or unit 7). The FORTRAN calling sequences for NTFILE,NTOPEN,NTREAD,NTBLCK,and NTRITE are

```
call NTFILE(chanl,fcount,status)
```

```
call NTOPEN(chanl,drive,status)
call NTREAD(chanl,nobytes,buffer,status,overflow)
call NTRITE(chanl,nobytes,buffer,status)
call NTBLCK (chanl,fcount,status)
```

For detail calling arguments description, see individual function below.

Restrictions : maximum of 20 file descriptors may be open at any one time including
stdin (unit 5)
stdout (unit 6)
stderr (unit 7)
NOTE: the functions are designed only for raw magnetic tape with 7970E tape drive. files used : /dev/rmt1 assumed,but no specific code in this routine.

NTCLOSE
tape file close routine.

NTEOF
write an end of file (EOF) mark on the magnetic tape.

NTFILE
position a tape file forward or backward over an end of file marks. The FORTRAN calling sequence is call ntfile (chanl,fcount,status) where all the arguments are integers. The number of EOF marks to pass over is fcount, where fcount positive implies forward, negative implies backward motion.

NTOPEN
FORTRAN callable tape file open routine for multiple (i.e. two) tape drives files used : /dev/rmt0 /dev/rmt1

NTREAD
reads one physical block per call where
status specifies error flag or actual count of words transferred
nobytes specifies number of bytes to transfer;
buffer specifies pointer to user supplied buffer array;
overflow specifies the count of bytes ignored in physical.
restrictions :
1. file must be opened by a C call to open().
2. user supplied buffer must be a singly subscripted integer array large enough to contain the largest physcial block .
3. involves possibly machine dependent code for HP-9000 & HP-UX

NTRITE
Write a record to tape. The user supplies the record to be written in the integer array buffer, the number of characters (bytes) to transfer in nobytes, and the file descriptor obtained from a previous call to ntopen in chanl The FORTRAN syntax is: call ntrite(chanl,nobytes,buffer,status) restrictions: The user must supply a large enough buffer for the of bytes to be transferred. At present, no internal error handling is provided. For example, upon encountering the end of tape, some fraction of the buffer is written to tape. However, the tape is not backspaced to its original position.

NTRW
rewind the magnetic

NTRWRL
rewind and release the magnetic tape

NTBLCK
routine to position a tape file forward or backward over n physical blocks. The number of blocks to pass over is fcount, where fcount positive implies forward, negative implies backward motion.

NAME

nxtnd

SYNOPSIS

```
subroutine nxtnd (t,dt,nstep,tend,done)
double precision t
double precision dt
integer nstep
double precision tend
integer done
```

DESCRIPTION

nxtnd

Sets next output time either from a base file (bfopt non-zero) or by specified time step dt (bfopt=0). Done is set to 1 if past end time tend, else set to 0. If nstep>0, then edit status is not checked. If bfopt>0, then base file is position to next unedited point.

t next time (input/output)

dt step size (input)

nstep step for running off base file (input)

tend stop time (input)

done done flag (output)

COMMENTS

Base file data must be less than 100 words.

NAME
prodx

SYNOPSIS
program prodx

DESCRIPTION
prodx
Executive for RELBET tape product output.

NAME
sphdr

SYNOPSIS
```
subroutine sphdr (u,x,r)
integer u
character*4 x (26)
integer r
```

DESCRIPTION
sphdr
Displays header and record for RELBET DPF generation.

u display unit (input)

x header message (input/output)

r number of record

NAME

tblchk

SYNOPSIS

```
integer function tblchk (i,timtag,onoff)
integer i
double precision timtag
integer onoff
```

DESCRIPTION

tblchk

Checks to see if time tag is in tracking interval.

i index (input)

timtag time tag (input)

onoff 1 if in tracking intervals (output) =-1 not in tracking intervals

NAME
udpfmt

SYNOPSIS
```
subroutine udpfmt (temp,dpnew)
integer temp (3)
integer dpnew (3)
```

DESCRIPTION
udpfmt
Converts 64 bit HP9000 double precision word to 72 bit UNIVAC.

temp input buffer

dpnew new storage variable (output)

COMMENTS
Must not be compiled with range checking.

NAME

uinfmt

SYNOPSIS

```
subroutine uinfmt (input,int)
integer input (2)
integer int (2)
```

DESCRIPTION

uinfmt

Moves bits.

input input

int output location

COMMENTS

Must not be compiled with range checking.

NAME
uspfmt

SYNOPSIS
```
subroutine uspfmt (temp,spnew)
integer temp (3)
integer spnew (2)
```

DESCRIPTION
uspfmt
Converts 32 bit HP9000 single precision to 36 bit UNIVAC.

temp input storage

spnew output storage

COMMENTS
Must not be compiled with range checking.

NAME

amenu

SYNOPSIS

```
subroutine amenu (nidz,xidz,idz)
integer nidz
character xidz (nidz)
integer idz (nidz)
```

DESCRIPTION

amenu

Sets flags according to prompt menu.

nidz number of input variables (input)

xidz names of options (input)

idz on/off flags +1 on, -1 off (input/output)

NAME

dciph

SYNOPSIS

```
subroutine dciph (prompt,type,nv,iv,rv,dv)
character prompt
character type
integer nv
integer iv (*)
real rv (*)
double precision dv (*)
```

DESCRIPTION

dciph

Read input character string and translates character string input to numerical values.

prompt display prompt (input)

type data type code (input)

nv number of values desired(input)

iv integer value array (input)

rv real value array (input/output)

dv double precision value array (input/output)

NAME

dciphr

SYNOPSIS

```
subroutine dciphr (type,lx,xin,nv,iv,rv,dv,erf)
character type
integer lx
character*1 xin (*)
integer nv
integer iv (*)
real rv (*)
double precision dv (*)
integer erf
```

DESCRIPTION

dciphr

Reads input character string and translates character string input to numerical values.

type data type code (input)

lx length of string (input)

xin character array (input)

nv number of values desired (input)

iv integer value array (input/output)

rv real value array (input/output)

dv double precision value array (input/output)

erf error/stop flag (output)

COMMENTS

Input string is considered to a max of 80 characters.

NAME

qxmenu, pmenu, gmenu

SYNOPSIS

```
subroutine qxmenu (nc,mnunam,np,pnmz,isel,xsel)
integer nc
character*32 mnunam
integer np
character*8 pnmz (np)
integer isel
character*4 xsel

entry pmenu (nc,mnunam,np,pnmz,isel,xsel)

entry gmenu (nc,mnunam,np,pnmz,isel,xsel)
```

DESCRIPTION

qxmenu

Solicits menu prompt from input list.

nc number of letters to check (input)

mnunam name of menu (input)

np number of items in menu

pnmz names of prompt items (input)

isel selected id (output)

xsel first four characters of selected item (output)

pmenu

Initializes count and selection index, display option menu and choice and interpret response.

gmenu

Obtains menu input.

NAME

tio, ctio, itio, dtio, rtio

SYNOPSIS

```
subroutine tio (p,l,iv,dv,rv,str)
character p
integer l
integer iv (l)
double precision dv (l)
real rv (l)
character str

entry ctio (p,l,str)

entry itio (p,l,iv)

entry dtio (p,l,dv)

entry rtio (p,l,rv)
```

DESCRIPTION

tio

Interactive terminal input/output.

p	prompt string (input)
l	number of values or length of string(input)
iv	integer value array (input/output)
dv	double precision value array(input/output)
rv	real value array (input/output)
str	string for input/output (input/output)

ctio

Character entry point for dciph call.

itio

Integer entry point for dciph call.

dtio

Double precision entry point for dciph call.

rtio

Real entry point for dciph call.

NAME

xqwtmz, xqdatz, xqtime

SYNOPSIS

```
subroutine xqwtmz (p,ymdhm,sec)
character p
integer ymdhm (6)
double precision sec

entry xqdatz (p,ymdhm,sec)

entry xqtime (p,sec)
```

DESCRIPTION

xqwtmz

Interactive input of dates and times.

p prompt string (input/output)

ymdhm year,month,day,hour minute (input/output)

sec seconds (input/output)

xqdatz

Obtains a date in year,month,day.hour,minute,second.

xqtime

Obtains an time via hour,minute,second input.

NAME

yesno

SYNOPSIS

```
subroutine yesno (prompt,pyes)
character prompt
integer pyes
```

DESCRIPTION

yesno

Prompts for a yes or no input.

prompt prompt string(input)

pyes response flag: + = yes, -=no (output)

NAME
dprop

SYNOPSIS
block data dprop

DESCRIPTION
dprop
This block data routine initializes the common used by this program

NAME
dvdt

SYNOPSIS
```
subroutine dvdt (t,rv,a)
double precision t
double precision rv (6)
double precision a (3)
```

DESCRIPTION
dvdt
Computes acceleration for a vehicle.

t current time (input)

rv vehicle position and velocity (input)

a acceleration (output)

COMMENTS
Vent force and start and stop times must be determined by driver.

NAME
nprop

SYNOPSIS
subroutine nprop

DESCRIPTION
nprop
This subroutine fetches the user inputs for the common blocks needed by this program

NAME
opnprp

SYNOPSIS
```
subroutine opnprp (fatals)
integer fatals
```

DESCRIPTION
driver for opening the files needed for propagation

opnprp
fatals fatal error count (output)

COMMENTS
does not incorporate full error checks

NAME
prpnpt

SYNOPSIS
subroutine prpnpt

DESCRIPTION
prpnpt
obtains input for the propagation module

NAME
runkut

SYNOPSIS
```
subroutine runkut (t,dt,x)
double precision t
double precision dt
double precision x (6)
```

DESCRIPTION
runkut
Uses 4th order runge-kutta integrator to propagate position and velocity one time step.

t initial time (input/output)

dt total time step (input)

x position and velocity (input/output)

NAME

rvprop

SYNOPSIS

```
subroutine rvprop (tout,vehid,trv)
double precision tout
integer vehid
double precision trv (7)
```

DESCRIPTION

rvprop

Obtains a state vector at desired time by 1) interpolation. 2) runge-kutta integration of input state. 3) super-g integration of input state.

tout required output time (input)

vehid vehicle id (input)

trv vehicle time, position, velocity (input/output)

NAME

rvup

SYNOPSIS

```
subroutine rvup (c,s,u,ac,dr)
double precision c
double precision s (6)
double precision u (6)
double precision ac (3)
double precision dr (3)
```

DESCRIPTION

rvup

Updates position and velocity.

c	time factor (input)
s	input state array (input)
u	updated state array (output)
ac	current acceleration (input)
dr	save position derivative (input)

NAME

superg

SYNOPSIS

```
subroutine superg (hstep,tcur,s)
double precision hstep
double precision tcur
double precision s (6)
```

DESCRIPTION

superg

Super G integrator.

hstep current step size (input)

tcur old time (output)

s state whose position and velocity are to be advanced (output)

NAME

tstep

SYNOPSIS

```
subroutine tstep (inc,iv,tend,tc.dt,tnew)
```

DESCRIPTION

tstep

Obtains next time for integration step.

NAME

dpfmt

SYNOPSIS

```
subroutine dpfmt (temp,dpnew)
integer temp (3)
integer dpnew (2)
```

DESCRIPTION

dpfmt

convert 72 bit unviac dp to 64 hp

temp dp buffer(input)

dpnew hp dp buffer (output)

NAME

getbit

SYNOPSIS

```
subroutine getbit (n,in,iw,ib,itemp)
integer ib
integer in (3)
integer itemp (3)
integer iw
integer n
```

DESCRIPTION

getbit

extract n consecutive bits from the array in, beginning with the initial word iw and initial bit ib, and store in the first portion of the array itemp. n is usually expected to be 18, 36, or 72 for univac created records. itemp is treated as 32 bit word for current applications.

ib initial bit (input/output)

in input array

itemp temp buffer (output)

iw initial word (input/output)

n consecutive bit (input/output)

NAME

imove

SYNOPSIS

```
subroutine imove (input,iword1,ibit1,len,iout,iword2,ibit2)
integer ibit1
integer ibit2
integer input (*)
integer iword1
integer iword2
integer len
integer iout (*)
```

DESCRIPTION

imove

This subroutine moves a field of bits from one position in the input array to another position in the output array. The subroutine allows for the input and output arrays to be the same array, and allows the source field to overlap the destination field, by checking whether the move is to the left (i.e. lower subscripts) or to the right, and so starting the move at the left or right, respectively. The fields may span word boundaries. The user is cautioned to identify the arrays by their first subscript, using the calling arguments to define the positions of interest within the arrays, as trickery may defeat the program design with unpredictable results. In common with all refmt utility routines, imove allows the user to specify the word length and integer negative number conventions desired for both input and output, which need not be the same. All internal calculations are done using HP-9000 integer (32 bit two's complement) words. However, the user may choose to reference the input to some other computer, e.g. Univac 36 bit one's complement. In this way, the users program can read Univac data using Univac subscripting.

ibit1 bit position in input(iword1) for start of move (internal & output,input & updted)

ibit2 bit position in iout(iword2) for start of move (internal,input)

input array of input data (internal,input)

iword1 subscript to input array using wrdln1 length words for start of move (internal,input)

iword2 subscript to output array using wrdln2 length words for start of move (internal,input)

len no. of bits remaining to be transferred (internal,input)

iout array of output data (output,computed)

COMMENTS

All calculations of the beginning and ending bit positions within the arrays must result in integer numbers. For example using Univac input word length (36 bits per word), the max array size is 59,652,323 which defies the imagination! THE USER IS RESPONSIBLE FOR PROTECTING AGAINST SUBSCRIPTS OUT OF RANGE!!!! Several Fortran intrinsic functions which are not part of ANSI Fortran 77 are used by this program.

NAME
infmt

SYNOPSIS
```
subroutine infmt (temp,int)
integer temp (2)
integer int
```

DESCRIPTION
infmt
convert univac integer to hp integer

temp input buffer

int output buffer

NAME

qatape

SYNOPSIS

program qatape

DESCRIPTION

qatape

This program reads the data product (binary) tape created on Univac. It prints the first and last page if print is set to zero & prints every page if print is set to 1. It prints every n records if print is set to 2. This program also creates a binary disc file(HP format) when print is set to 3.

NAME
spfmt

SYNOPSIS
```
subroutine spfmt (temp,spnew)
integer temp (3)
integer spnew (1)
```

DESCRIPTION
spfmt
converts 36 bit univac sp to 32 bit hp

temp temp buffer (input/output)

spnew single precision buffer (output)

NAME

gspop, gspush, gspeek, gspoke, gsempty, gsfree, make_gstack, q_pop, q_push - stack operations

SYNOPSIS

```
char *gspop(stack)
P_GSTACK stack;

char *gspush(stack,block)
P_GSTACK stack;
char *block;

char *gspeek(stack,offset)
P_GSTACK stack;
int offset;

char *gspoke(stack,block,offset)
P_GSTACK stack;
char *block;
int offset;

char *gsempty(stack)
P_GSTACK stack;

char *gsfree(stack)
P_GSTACK stack;

P_GSTACK make_gstack(stack,b_size,m_block,buffer)
P_GSTACK stack;
int b_size;
int m_block;
char *buffer;

char *q_pop(queue)
P_QUEUE queue;

char *q_push(queue,block)
P_QUEUE queue;
char *block;
```

DESCRIPTION

gspop

Pop the top block of stack. Return pointer found in top block. Return null pointer if stack is empty. Decrement block count and pointer to top block. No blocks are moved, in particular top block.

gspush

Push block on top of stack, return pointer to new top, return null pointer if no room. increment pointer to top block and block count.

gspeek

To read contents of particular block in stack. Offset indicates location in stack,

>=0, displacement from top

< 0, displacement from bottom.

Return pointer to start of requested block. Return null pointer if out-of-range.

gspoke

Place contents of block into stack replacing particular stack block. Offset indicates location in stack,

>=0, displacement from top
< 0, displacement from bottom.
Return null pointer if out-of-range, non null otherwise.

gsempty
Empty contents of stack.Set top of stack to zero. Return null pointer if error else return begin of stack

gsfree
Free allocation for stack. Return null pointer

make_gstack
b_size block size

m_block maximum number of blocks

buffer pointer to where stack is to begin Create new stack beginning at location indicated by buffer. having block size of b_size, and maximum number of blocks m_block. Return null stack pointer when error. Return pointer to new stack. If input for pointer to stack is null then space is allocated according to stack size. If onput for pointer to char is null then space is allocated according to m_block and b_size.

q_pop
Pop the front block of queue. Return pointer to old front block. Return null pointer if queue is empty. Decrement block count. Adjust pointer to front block. No blocks are moved, in particular front block.

q_push
Push block on top of queue, return pointer to new top, return null pointer if no room. Increment pointer to top block and block count.

NAME

BaseCalTime, JD_BaseTime, GBeginTime, GEndTime, GDelTime, LBeginTime, LEndTime, LDelTime - Base and time control

SYNOPSIS

```
CALTIME BaseCalTime;

double JD_BaseTime;

double GBeginTime;

double GEndTime;

double GDelTime;

double LBeginTime;

double LEndTime;

double LDelTime;
```

DESCRIPTION

These parameters are provided with the idea that GBeginTime < GEndTime and that the nominal Begin and End times default to these values. They provide added safety in the case where BeginTime and EndTime may be changed during execution.

BaseCalTime
Time from which all times are measured.
Assumed to be UT in Gregorian Calendar

JD_BaseTime
Julian date of BaseCalTime

GBeginTime
Global start time in seconds from BaseCalTime.
No processing is considered beyond this time

GEndTime
As BeginTime but at end.

GDelTime
Global Time step in seconds

These parameters are provided with the idea that they specify working time spans. The EndTime may be less than the BeginTime for backwards time processing. They should default to the Global times.

LBeginTime
Start time in seconds from BaseCalTime.

LEndTime
As BeginTime but specifies end.

NAME

cmpdat, cdtojd, jdtocd

SYNOPSIS

```
subroutine cmpdat (date,djul)
double precision date (6)
double precision djul

entry cdtojd (date,djul)

entry jdtocd (djul,date)
```

DESCRIPTION

cmpdat

Does true calculations for: 1) Calendar date to Julian date. 2) Julian date to Calendar date.

date the calendar date- yr, mo, day, hr, min, sec (input/output)

djul julian date (assumed at noon) (input/output)

cdtojd

Converts a given calender date to its corresponding Julian date.

jdtocd

Converts double precision julian date to year,month,day,hour, minute,seconds.

COMMENTS

Date must be before year 2000 and after 1984.

NAME
days

SYNOPSIS
```
subroutine days (dv1,dv2,d)
integer dv1 (3)
integer dv2 (3)
integer d
```

DESCRIPTION
days
Computes the number of days between two calendar dates.

dv1 date vector (y,m,d) for first date

dv2 date vector (y,m,d) for second date

d number of days difference between dates

COMMENTS
1. Algorithm works for years after 1582.

2. Implementation assumes that years between 0 and 200 are relative to year 1900.

NAME

daysxx

SYNOPSIS

```
function daysxx (y,m)
integer y
integer m
```

DESCRIPTION

daysxx

Computes leap-year correction to days.

y input year

m input month

NAME
dhms

SYNOPSIS
```
subroutine dhms (t,dhm,sec)
double precision t
integer dhm (3)
real sec
```

DESCRIPTION
dhms
Converts a time in seconds to days, hours, minutes, seconds.

t time tag (input)

dhm days, hours, and minutes (output)

sec seconds (output)

COMMENTS
Time tag must be in seconds.

NAME
hms2ds

SYNOPSIS
```
subroutine hms2ds (itime,te)
integer itime (3)
double precision te
```

DESCRIPTION
hms2ds
Converts hours minutes and seconds to double precision seconds.

itime integer hours, minutes, seconds (input)

te double precision seconds (output)

NAME

j2c

SYNOPSIS

```
subroutine j2c (t,jb,tv,sec)
double precision t
integer jb
integer tv (5)
real sec
```

DESCRIPTION

j2c

Takes julian base plus offset in seconds and gives back integer year,month,day,hour,min and real seconds.

t input time in seconds relative to base date

jb base julian day (input)

tv integer time vector (y:m:d:h:m) input

sec real seconds (output)

NAME

j2ymd

SYNOPSIS

```
subroutine j2ymd (jd,ymd)
integer jd
integer ymd (3)
```

DESCRIPTION

j2ymd

Computes calendar year month day from universal Julian date.

jd julian date (input)

ymd year month day vector (output)

NAME

secnds

SYNOPSIS

```
subroutine secnds (date1,date2,secs)
double precision date1 (6)
double precision date2 (6)
double precision secs
```

DESCRIPTION

secnds

Determines the difference in seconds between two calendar times.

date1 1st date as double words (input)

date2 2nd date as double words (input)

secs seconds between the two dates (output)

NAME

CurSysTime, GMTsec, GMTday, GetCurTime, fprtCurTime, prtCurTime - system time

SYNOPSIS

```
CALTIME CurSysTime;

long GMTsec;

int GMTday;

CALTIME *GetCurTime()

void fprtCurTime(file)
FILE *file;

void prtCurTime()
```

DESCRIPTION

Module for obtaining current time as witnessed by system. The variables CurSysTime, GMTsec, and GMTday respectively contain the system GMT date, secons since 1970 and days since 1970. These variables are set upon calling the the function GetCurTime. They are not updated otherwise and may accordingly be stale. Note in particular that fprtCurTime merely prints what is in CurSysTime and does not update the value.

GetCurTime
Sets external variable to curent GMT as witnessed by system. Returns pointer to the variable CurSysTime

fprtCurTime
Display current system time in full format. See fprtctime for details.

prtCurTime
Display current system time in full format to stdout.

NAME

hms2sec, sec2hms, days1bc, days, etsec, jul2cal, juldate, jultime, std_time, mnthnum - time format conversions

SYNOPSIS

```
double hms2sec(hms)
HMSTIME *hms;

HMSTIME *sec2hms(hms,sec)
HMSTIME *hms;
double sec;

days1bc(date)
CALDATE *date;

days(date,base)
CALDATE *date,*base;

double etsec(to,from)
CALTIME *to,*from;

double jul2cal(jdate,date)
double jdate;
CALDATE *date;

double juldate(date)
CALDATE *date;

double jultime(c)
CALTIME *c;

std_time(time)
CALTIME *time;

mnthnum(month)
char *month;
```

DESCRIPTION

hms2sec
Return seconds specified by hms structure

sec2hms
Convert seconds to hour minute second format. Returns the pointer to HMSTIME structure. Note no space is allocated.

days1bc
Number of days from Ojan1BC using the Gregorian calendar. %CDates are valid from 15Oct1582 in Catholic countries (previous day was 4Oct1582 in old style) and from 14Sep1752 in British countries (US included) (previous day was 2Sep1752 Old style)

days
Number of days from base date to date . Dates must be in Gregorian calendar. Note that day of week field is set in this computation.

etsec
Returns elapsed time in seconds from CALTIME from to CALTIME to.

jul2cal
Converts input Julian date to calendar date, returning the number of fractional hours left in the julian time. Note that whole julian dates

occur at noon and calendar dates are Gregorian.

juldate
Returns julian date in days. Date is assumed to correspond to midnight thus result is julian date of midnight. Method is to add julian date of Ojan1BC to number of elapsed days.

jultime
Returns julian time in days from calendar time. Note the time portion is included so that the time tag does not always end with a .5.

std_time
Ensures time is in normalized format. ie, right month, day, etc. Note that the day of the week field is set in this normalization.

mnthnum
Returns number of month based on deciphering the characters in month. Differences between upper and lower case are ignored. Characters are checked until a match is found , e.g jun or aP are june and april. If no match then 0 is returned

NAME

makeTime, makeHMS, makeDate, setDate, setHMS, setTime - Allocate and set time structures

SYNOPSIS

```
CALTIME *makeTime(n)
int n;

HMSTIME *makeHMS(n)
int n;

CALDATE *makeDate(n)
int n;

setDate(date,year,month,day)
CALDATE *date;
int year,month,day;

setHMS(hms,hour,min,sec)
HMSTIME *hms;
int hour,min;
double sec;

setTime(time,year,month,day,hour,min,sec)
CALTIME *time;
int year,month,day;
int hour,min;
double sec;
```

DESCRIPTION

Create and initialize time structures. calloc is used for storage allocation whence structures may be freed with free. Null pointers are returned when errors are encountered.

makeTime
Allocates space for specified number of CALTIME structures. Returns pointer to a CALTIME structure.

makeHMS
Allocates space for specified number of HMSTIME structures. Returns pointer to initial HMSTIME structure.

makeDate
Allocates space for specified number of CALDATE structures. Returns pointer to initial CALDATE structure.

setDate
Set date structure to specified values. Note that day of week is not set.

setHMS
Set HMSTIME structure to specific values

setTime
Set time structure to specified values. Note that day of week field is not set.

NAME

fprtdate, sprtdate, prtdate, fprthms, sprthms, prthms, fprtctime, sprtctime, prtctime, fprtsec, sprtsec, prtsec - display formatted time information

SYNOPSIS

```
void fprtdate(file,date)
FILE *file;
CALDATE *date;

void sprtdate(s,date)
char *s;
CALDATE *date;

void prtdate(date)
CALDATE *date;

void fprthms(file,hms)
FILE *file;
HMSTIME *hms;

void sprthms(s,hms)
char *s;
HMSTIME *hms;

void prthms(hms)
HMSTIME *hms;

void fprtctime(file,time)
FILE *file;
CALTIME *time;

void sprtctime(s,time)
char *s;
CALTIME *time;

void prtctime(time)
CALTIME *time;

void fprtsec(file,sec)
FILE *file;
double sec;

void sprtsec(s,sec)
char *s;
double sec;

void prtsec(sec)
double sec;
```

DESCRIPTION

Note that the file print routines do not add newlines nor do they flush the print buffers. The displays to stdout out do however.

fprtdate
Formatted print of date to file in Dayofweek Monthday, month year format.

sprtdate
Formatted print of date to file in Dayofweek Monthday, month year format.

prtdate
Formatted print of date to stdout

fprthms
Formatted print of hms to file in hr:min:sec format.

sprthms
Formatted print of hms to string in hr:min:sec format.

prthms
Formatted print of hms to stdout in hr:min:sec format

fprtctime
Formatted print of calendar time to specified file.

sprtctime
Formatted print of calendar time to specified string.

prtctime
Formatted print of calendar time to stdout

fprtsec
Formatted print of hms to file in hr:min:sec format.

sprtsec
Formatted print of hms to string in hr:min:sec format.

prtsec
Formatted print of hms to stdout in hr:min:sec format

NAME
ymd2j

SYNOPSIS
```
subroutine ymd2j (ymd,jd)
integer ymd (3)
integer jd
```

DESCRIPTION
ymd2j
Computes Julian universal date given calendar year,month,day.

ymd calendar year month day (input) year is either calendar year or calendar year mod 1900

jd julian date (output)