RELIENT 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
RELBEH 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
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

1.0 INTRODUCTION ................................................................. 1-1
1.1 IDENTIFICATION ............................................................... 1-2
1.2 REQUIRED BACKGROUND ................................................... 1-3
1.3 HARDWARE/SOFTWARE ENVIRONMENT ...................................... 1-4
1.4 TERMINOLOGY AND CONVENTIONS ......................................... 1-4

2.0 REFERENCES ......................................................................... 2-1

3.0 FUNCTIONAL OVERVIEW .................................................... 3-1
3.1 GENERAL OVERVIEW .......................................................... 3-1
3.2 CCT DATA FORMATTING ..................................................... 3-3
3.3 DATA PREPARATION ............................................................ 3-6
3.4 TRAJECTORY ESTIMATION .................................................. 3-9
3.5 PRODUCTS GENERATION ...................................................... 3-11

4.0 DESIGN OVERVIEW ............................................................ 4-1
4.1 BASIC PROGRAM STRUCTURE .............................................. 4-1
4.2 RELBET STANDARD FILE FORMATS ..................................... 4-2
4.3 INCLUDE FILES ................................................................. 4-3
4.4 SUBROUTINE MODULES ...................................................... 4-4
4.5 LINPUT INPUT SCHEME ...................................................... 4-4

5.0 DIRECTORY STRUCTURE .................................................... 5-1
5.1 CONTROL DIRECTORIES ..................................................... 5-1
5.2 PROGRAM DIRECTORIES ..................................................... 5-1
5.3 CONTEXT DIRECTORIES ...................................................... 5-3
5.4 UTILITY DIRECTORIES ....................................................... 5-3
4.5.1 File Routines ............................................................... 5-3
4.5.2 Input Manipulation Routines .......................................... 5-4
4.5.3 Math Routines ............................................................. 5-4
4.5.4 Propagation Routines .................................................... 5-4
4.5.5 Data Structure Manipulations ......................................... 5-4
4.5.6 Observation Computation .............................................. 5-5
# TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0 INSTALLATION AND MAINTENANCE</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1 INSTALLATION</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1.1 Creating a RELBET Master Directory</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1.2 Reading the Delivery Tape</td>
<td>6-1</td>
</tr>
<tr>
<td>6.1.3 Tailoring the System to Your Environment</td>
<td>6-2</td>
</tr>
<tr>
<td>6.1.3.1 Graphic Display Device Configuration</td>
<td>6-2</td>
</tr>
<tr>
<td>6.1.3.2 Tools Configuration</td>
<td>6-2</td>
</tr>
<tr>
<td>6.1 MAINTENANCE</td>
<td>6-2</td>
</tr>
<tr>
<td>6.2.1 Configuration Control</td>
<td>6-3</td>
</tr>
<tr>
<td>6.2.2 Recreating Programs</td>
<td>6-3</td>
</tr>
<tr>
<td>6.2.3 Modifying Common, Input, Default, etc., Blocks</td>
<td>6-3</td>
</tr>
<tr>
<td>6.2.4 Documentation</td>
<td>6-3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPENDIX I</td>
<td>I-1</td>
</tr>
<tr>
<td>APPENDIX II</td>
<td>II-1</td>
</tr>
<tr>
<td>APPENDIX III</td>
<td>III-1</td>
</tr>
<tr>
<td>APPENDIX IV</td>
<td>IV-1</td>
</tr>
</tbody>
</table>
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 MAINTENANCE: 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:

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

It incorporates the following features not available in RELBET 2.0:

- Organization of standard runstreams into shell scripts
- Enhanced user input scheme via linput
- Increased data QA programs
- A text file QA data base
It lacks the following features available in RELBET 2.0:

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

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

- Source code (configured in SCCS format)
- Relocatable subroutine code
- Executable programs
- Program creation directives (makefile's)
- Maintenance tools
- 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

<table>
<thead>
<tr>
<th>Area</th>
<th>Applicable Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>General source code</td>
<td>FORTRAN, BOX, ratfor, C, Bourne Shell</td>
</tr>
<tr>
<td>input source</td>
<td>C, lex(1), yacc(1)</td>
</tr>
<tr>
<td>Graphics display</td>
<td>DISSPLA</td>
</tr>
<tr>
<td>Configuration Control</td>
<td>sccs(1)</td>
</tr>
<tr>
<td>Creating programs</td>
<td>make(1)</td>
</tr>
<tr>
<td>Common blocks</td>
<td>mtf utilities</td>
</tr>
<tr>
<td>Input blocks</td>
<td>mtf utilities</td>
</tr>
<tr>
<td>Documentation</td>
<td>nroff(1), man(7), cstrip, txman, blist, clist, mtf utilities</td>
</tr>
</tbody>
</table>

1.3 HARDWARE/SOFTWARE ENVIRONMENT

RELBET 4.0 assumes the following hardware configuration:

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

RELBET 4.0 assumes the following software environment:

- RELBET 4.0
- HPUCX operating system version 5.1
- 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.

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 basic steps:

- Data Collection
- Input Data Formatting
- Input Data Preparation
- Trajectory Estimation
- 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 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 CGO11X CCT (Computer Compatible Tape). (CGO11X 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 \texttt{downfmt} 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:

- **Attitude file**: contains selected quaternions and is used to determine orbiter attitude. The following discussion uses the abbreviation "ATT" for this file.

- **Observation file**: contains relative observations. The following discussion uses the abbreviation "OBS" for this file.

- **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.

- **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 target ephemerides.

- **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 necessary 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 preferred for filtering because they theoretically 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 necessary. 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_qaddata` 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_qaddata` to edit the attitude, sensed velocity, and observation files for bad data. Several shell programs facilitate 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>
<thead>
<tr>
<th>Name</th>
<th>Text File</th>
<th>Gff File</th>
</tr>
</thead>
<tbody>
<tr>
<td>qaatt</td>
<td>identifies attitude manuevers and bad points on the attitude file</td>
<td>angular momenta and magnitudes between consecutive attitudes</td>
</tr>
<tr>
<td>qaranjmp</td>
<td>identifies inconsistencies in range and range rate to detect sudden biases or equipment reconfiguration</td>
<td></td>
</tr>
<tr>
<td>qastar</td>
<td>identifies false targets that the star tracker may have locked onto</td>
<td>star file containing azimuth, elevation, and angle for each star tracker observation</td>
</tr>
<tr>
<td>qasv</td>
<td>identifies burns as jumps in the magnitudes of the sensed acceleration computed from the sensed velocity and bad data</td>
<td>sensed acceleration file</td>
</tr>
<tr>
<td>qanois</td>
<td>documents the noise for each observation type</td>
<td>noise files for each observation</td>
</tr>
</tbody>
</table>
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 sln2r1, cmp2sg, and rlvsr1 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 sln2r1, cmp2sg, and rlvsr1 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 sln2r1 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

ONLY IF RELATIVE EPHEMERIS NOT AVAILABLE

ORBITER EPHEM

TARGET EPHEM

COMBINE EPHEMS

SENSOR DATA

RELATIVE EPHEM

SENSOR TAPE OUTPUT PROGRAM

STOP

SIT

SET
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 features of the design and implementation are:

- Isolation of logic and data to well defined routines or blocks
- Data structuring via blocks of functionally related variables
- Extensive use of include directives to provide consistent context
- A common library of utility routines organized in functional modules
- A standardized binary file format that facilitates interprogram interfaces
- 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 than 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 consistent 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 involving 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 consistency 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 through a set of C language routines in the directory Llnput. 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:

- Control Directories that contain executables and input data
- Program Directories that contain program unique code
- Context Directories that contain data structure information such as common blocks and parameters
- 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.

<table>
<thead>
<tr>
<th>Control</th>
<th>Executable and production shell scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make</td>
<td>Make directives used to recreate executables</td>
</tr>
<tr>
<td>Test</td>
<td>Sample inputs and shell scripts</td>
</tr>
<tr>
<td>Tools</td>
<td>Maintenance tools</td>
</tr>
<tr>
<td>Manual</td>
<td>Program user manuals</td>
</tr>
<tr>
<td>Pmanual</td>
<td>Routine Manuals</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Directory</th>
<th>Program Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downfor</td>
<td>dwnfmt</td>
<td>Downlist Formatter</td>
</tr>
<tr>
<td></td>
<td>rptost</td>
<td>Roll/pitch to shaft trunnion observation convention</td>
</tr>
<tr>
<td>Filter</td>
<td>sfilt</td>
<td>Kalman filter</td>
</tr>
<tr>
<td></td>
<td>smooth</td>
<td>Smoothing Filter</td>
</tr>
<tr>
<td>Fich</td>
<td>fiche</td>
<td>Create microfiche</td>
</tr>
<tr>
<td>Fman</td>
<td>cmp2sg</td>
<td>Compare state difference to sigma</td>
</tr>
<tr>
<td></td>
<td>eph2rel</td>
<td>Convert ephemerides to relative trajectory</td>
</tr>
<tr>
<td></td>
<td>filedit</td>
<td>File editor</td>
</tr>
<tr>
<td></td>
<td>gbfcom</td>
<td>Combine gff files</td>
</tr>
<tr>
<td></td>
<td>mkinit</td>
<td>Make the init input block</td>
</tr>
<tr>
<td></td>
<td>obsnois</td>
<td>Place noise data on obs file</td>
</tr>
<tr>
<td></td>
<td>qaatt</td>
<td>QA the attitude file</td>
</tr>
<tr>
<td></td>
<td>qacover</td>
<td>QA the data coverage</td>
</tr>
<tr>
<td></td>
<td>qanois</td>
<td>QA the data noise</td>
</tr>
<tr>
<td></td>
<td>qaranjmp</td>
<td>QA the range observations</td>
</tr>
<tr>
<td></td>
<td>qastar</td>
<td>QA the star tracker obs</td>
</tr>
<tr>
<td></td>
<td>qasv</td>
<td>QA the sensed velocity</td>
</tr>
<tr>
<td></td>
<td>rdwt</td>
<td>Read file then write file</td>
</tr>
<tr>
<td></td>
<td>read_set</td>
<td>Read SET file and print</td>
</tr>
<tr>
<td></td>
<td>read_sit</td>
<td>Read SIT file and print</td>
</tr>
<tr>
<td></td>
<td>rlvsrc</td>
<td>Compare relative trajectory files</td>
</tr>
<tr>
<td></td>
<td>search</td>
<td>Extract data from event data base</td>
</tr>
<tr>
<td></td>
<td>sln2r1</td>
<td>Strip relative trajectory from solution file</td>
</tr>
<tr>
<td></td>
<td>stop</td>
<td>Create SENSOR output tapes</td>
</tr>
<tr>
<td>Mktape</td>
<td>mktape</td>
<td>Create output UNIVAC tape</td>
</tr>
<tr>
<td>Noisanal</td>
<td>ddnois</td>
<td>Compute obs data noise</td>
</tr>
<tr>
<td>Numdis</td>
<td>gdisp</td>
<td>General gff display</td>
</tr>
<tr>
<td></td>
<td>xcmpar</td>
<td>Compare two ephemeris files</td>
</tr>
<tr>
<td></td>
<td>xqdisp</td>
<td>State/Attitude Parameter Display</td>
</tr>
<tr>
<td>Plot</td>
<td>ascale</td>
<td>Auto scale</td>
</tr>
<tr>
<td></td>
<td>plotx</td>
<td>Graphic display of gff files</td>
</tr>
<tr>
<td>Product</td>
<td>prodx</td>
<td>Special product generation</td>
</tr>
<tr>
<td>Qatape</td>
<td>qatape</td>
<td>QA special product tape</td>
</tr>
</tbody>
</table>
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**
  - Input 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 synonymous 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 original 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 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 necessary 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 arguments 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 control 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 bcmans, 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

## 1.0 INTRODUCTION

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-1</td>
</tr>
</tbody>
</table>

## 2.0 EXTERNAL FILES

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 DOWNLIST CCT</td>
<td>I-1</td>
</tr>
<tr>
<td>2.2 RELBET PRODUCT FILE</td>
<td>I-2</td>
</tr>
<tr>
<td>2.2.1 Binary Data Products File</td>
<td>I-2</td>
</tr>
<tr>
<td>2.2.2 Output Contents</td>
<td>I-5</td>
</tr>
<tr>
<td>2.2.3 Parameter Groups</td>
<td>I-5</td>
</tr>
<tr>
<td>2.3 SENSOR INPUT TAPE</td>
<td>I-11</td>
</tr>
<tr>
<td>2.4 SENSOR ENVIRONMENT TAPE (SET)</td>
<td>I-14</td>
</tr>
</tbody>
</table>

## 3.0 INTERNAL FILES

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 GENERAL FORMAT</td>
<td>I-17</td>
</tr>
<tr>
<td>3.1.1 Format Compatibility</td>
<td>I-17</td>
</tr>
<tr>
<td>3.1.2 Overall File Format</td>
<td>I-17</td>
</tr>
<tr>
<td>3.1.3 RELBET Data Frame Formats</td>
<td>I-18</td>
</tr>
<tr>
<td>3.1.4 File I/O Packet</td>
<td>I-19</td>
</tr>
<tr>
<td>3.1.5 File Header Format</td>
<td>I-22</td>
</tr>
<tr>
<td>3.1.6 Dictionary Format</td>
<td>I-23</td>
</tr>
<tr>
<td>3.2 SENSED VELOCITY FILE</td>
<td>I-25</td>
</tr>
<tr>
<td>3.3 ATTITUDE FILE</td>
<td>I-25</td>
</tr>
<tr>
<td>3.4 OBSERVATION FILE</td>
<td>I-25</td>
</tr>
<tr>
<td>3.5 EPHEMERIS FILE</td>
<td>I-26</td>
</tr>
<tr>
<td>3.6 IMU ATTITUDE FILE</td>
<td>I-26</td>
</tr>
<tr>
<td>3.7 IMU SENSED VELOCITY FILE</td>
<td>I-27</td>
</tr>
<tr>
<td>3.8 SOLUTION FILE</td>
<td>I-27</td>
</tr>
<tr>
<td>3.9 RELATIVE TRAJECTORY FILE</td>
<td>I-28</td>
</tr>
<tr>
<td>3.10 PLOT DISPLAY FILES</td>
<td>I-28</td>
</tr>
<tr>
<td>3.11 COAS MATRIX FILE</td>
<td>I-30</td>
</tr>
<tr>
<td>3.12 STAR TRACKER MATRIX FILE</td>
<td>I-30</td>
</tr>
<tr>
<td>3.13 SUPPLEMENTARY SIT FILE</td>
<td>I-30</td>
</tr>
<tr>
<td>3.14 ENVIRONMENT FILE</td>
<td>I-31</td>
</tr>
<tr>
<td>3.15 QA FILES</td>
<td>I-31</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3-16 SENSOR FILES</td>
<td>I-32</td>
</tr>
<tr>
<td>4.0 TEXT FILES</td>
<td>I-35</td>
</tr>
<tr>
<td>5.0 DATA BASE FILES</td>
<td>I-36</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>I-37</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

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

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

<table>
<thead>
<tr>
<th>INPUT</th>
<th>Downlist CCTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTPUT</td>
<td>RELBET Product Tape</td>
</tr>
<tr>
<td></td>
<td>RELBET Microfiche Text</td>
</tr>
<tr>
<td></td>
<td>SENSOR Input Tape (SIT)</td>
</tr>
<tr>
<td></td>
<td>SENSOR Environment Tape (SET)</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>RECORD</th>
<th>INFORMATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tape Identifier</td>
</tr>
<tr>
<td>2</td>
<td>Parameter Dictionary</td>
</tr>
<tr>
<td>3</td>
<td>Data</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>Monotonically Increasing</td>
</tr>
<tr>
<td>.</td>
<td>Time</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>L</td>
<td>Data</td>
</tr>
</tbody>
</table>
Table 2.2.2. File Identifier Record Format

<table>
<thead>
<tr>
<th>SINGLE WORD NUMBER</th>
<th>DEFINITION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48 character identifier message</td>
<td>alphabetic (single)</td>
</tr>
<tr>
<td></td>
<td>flight no., data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>SPEC</td>
<td>alphabetic (single)</td>
</tr>
<tr>
<td>14</td>
<td>tape start GMT (first data record)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>HR</td>
<td>integer (single)</td>
</tr>
<tr>
<td>18</td>
<td>MIN</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>SEC</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>tape stop GMT (last data record)</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>number of parameters in dictionary</td>
<td>integer (single)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>DOUBLE WORD NUMBER</th>
<th>VARIABLE DEFINITION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$N = \text{No. of parameters in dictionary}$</td>
<td>integer (double)</td>
</tr>
<tr>
<td>2</td>
<td>Symbol for parameter in location 2 on data record (1st variable)</td>
<td>alphabetic (double)</td>
</tr>
<tr>
<td>3</td>
<td>Symbol for parameter in location 3 on data record (2nd variable)</td>
<td>alphabetic (double)</td>
</tr>
<tr>
<td>$N + 1$</td>
<td>Symbol for parameter in location $N + 1$ on data record (Nth variable)</td>
<td>alphabetic (double)</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>WORD NUMBER</th>
<th>VARIABLE DEFINITION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ITYPE 0 = continuing records</td>
<td>Double Precision</td>
</tr>
<tr>
<td></td>
<td>1 = last record in file</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Value of parameter in location 2 on dictionary record (1st variable)</td>
<td>Double Precision</td>
</tr>
<tr>
<td>3</td>
<td>Value of parameter in location 3 on dictionary record</td>
<td>Double Precision</td>
</tr>
<tr>
<td>N + 1</td>
<td>Value of parameter in location N + 1 on dictionary record</td>
<td>Double Precision</td>
</tr>
</tbody>
</table>

Note: \( N \) is number of parameters read from previous dictionary record.
Table 2.2.5. Special Products Parameters

<table>
<thead>
<tr>
<th>VARIABLE SET</th>
<th>PARAMETER GROUP NAME</th>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time, Greenwich mean (GMT)</td>
<td>1</td>
<td>SGMTY</td>
<td>yr</td>
<td></td>
</tr>
<tr>
<td>Orbiter onboard</td>
<td></td>
<td>SGMTMO</td>
<td>month</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGMTD</td>
<td>day</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGMTH</td>
<td>hr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGMTM</td>
<td>min</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SGMTS</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Time, Greenwich mean (GMT)</td>
<td>2</td>
<td>GMTY</td>
<td>yr</td>
<td></td>
</tr>
<tr>
<td>Ground - MCC UTC</td>
<td></td>
<td>GMTMO</td>
<td>month</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMTD</td>
<td>day</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMTH</td>
<td>hr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMTM</td>
<td>min</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GMTS</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>Time, ground elapsed (GET)</td>
<td>3</td>
<td>GETH</td>
<td>hr</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETM</td>
<td>min</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GETS</td>
<td>sec</td>
<td></td>
</tr>
<tr>
<td>State Vector, Orbiter, in Aries mean of 1950 Cartesian coordinate System</td>
<td>4</td>
<td>XM</td>
<td>x</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YM</td>
<td>y</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZM</td>
<td>z</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XDM</td>
<td>(\hat{x})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>YDM</td>
<td>(\hat{y})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZDM</td>
<td>(\hat{z})</td>
<td>km/sec</td>
</tr>
<tr>
<td>State Vector, Orbiter, in free flyer LVLH Cartesian coordinates</td>
<td>5</td>
<td>SLVX</td>
<td>x</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLVV</td>
<td>y</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLVZ</td>
<td>z</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLVXD</td>
<td>(\hat{x})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLVYD</td>
<td>(\hat{y})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLVZD</td>
<td>(\hat{z})</td>
<td>km/sec</td>
</tr>
<tr>
<td>State Vector, Orbiter, in free flyer UVW Cartesian coordinates</td>
<td>6</td>
<td>SU</td>
<td>u</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SV</td>
<td>v</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SW</td>
<td>w</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUD</td>
<td>(\hat{u})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVD</td>
<td>(\hat{v})</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SWD</td>
<td>(\hat{w})</td>
<td>km/sec</td>
</tr>
<tr>
<td>Attitude, Orbiter, Euler angles, body axis with respect to Orbiter UVW coordinates</td>
<td>7</td>
<td>ALPHU</td>
<td>yaw</td>
<td>deg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BETAU</td>
<td>pitch</td>
<td>deg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PHIU</td>
<td>roll</td>
<td>deg</td>
</tr>
<tr>
<td>VARIABLE SET</td>
<td>PARAMETER GROUP</td>
<td>SYMBOL</td>
<td>PARAMETER</td>
<td>UNITS</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>--------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>Attitude, Orbiter, Transformation, Aries mean of 1950 to body matrix</td>
<td>8</td>
<td>A11</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A12</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A13 matrix</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A21 elements</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A22 row</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A23 ordered</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A31</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A32</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A33</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td>Attitude, Orbiter, quaternion, Aries mean of 1950 to body axes</td>
<td>9</td>
<td>Q1</td>
<td>scalar part</td>
<td>n.d.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3 vector</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td>Attitude, rate, Orbiter, angular velocities and total rate, body about Aries mean of 1950 Cartesian coordinates</td>
<td>10</td>
<td>YDOTB</td>
<td>rate about z</td>
<td>deg/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PDOTB</td>
<td>rate about y</td>
<td>deg/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RDOTB</td>
<td>rate about x</td>
<td>deg/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDOTB</td>
<td>total rate</td>
<td>deg/sec</td>
</tr>
<tr>
<td>Look angle, and angle rates orbiter body to free flyer</td>
<td>11</td>
<td>AZ yaw</td>
<td>deg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>EL pitch</td>
<td>deg</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RAZ yaw rate</td>
<td>deg/sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>REL pitch rate</td>
<td>deg/sec</td>
<td></td>
</tr>
<tr>
<td>Range, and range rate, between vehicle reference points</td>
<td>12</td>
<td>RANGE range</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RRATE range rate</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td>Simulation Flag</td>
<td>13</td>
<td>PSIM</td>
<td>n.d.</td>
<td></td>
</tr>
<tr>
<td>State Vector, free flyer, in Aries mean of 1950 Cartesian coordinates</td>
<td>14</td>
<td>PMX x</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMY y</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMZ z</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMXD ŷ</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMYD ý</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMZD ź</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td>State Vector, free flyer, in Aries mean of 1950 Cartesian coordinates relative to Orbiter</td>
<td>15</td>
<td>PRMX x</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRMY y</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRMZ z</td>
<td>km</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRMXD ŷ</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRMYD ý</td>
<td>km/sec</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRMZD ź</td>
<td>km/sec</td>
<td></td>
</tr>
</tbody>
</table>
Table 2.2.5. Special Products Parameters (Continued)

<table>
<thead>
<tr>
<th>VARIABLE SET</th>
<th>PARAMETER GROUP NAME</th>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>State Vector, free flyer, in Orbiter UVW Cartesian coordinates</td>
<td>16</td>
<td>PU</td>
<td>u</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PV</td>
<td>v</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PW</td>
<td>w</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PUD</td>
<td>ū</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PVD</td>
<td>ŭ</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PWD</td>
<td>ĵ</td>
<td>km/sec</td>
</tr>
<tr>
<td>State Vector, free flyer, in Orbiter LVLH Cartesian coordinates</td>
<td>17</td>
<td>PLVX</td>
<td>x</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLVY</td>
<td>y</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLVZ</td>
<td>z</td>
<td>km</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLVXD</td>
<td>ẋ</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLVYD</td>
<td>ẏ</td>
<td>km/sec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PLVZD</td>
<td>ĵ</td>
<td>km/sec</td>
</tr>
</tbody>
</table>
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 displaced two integer words.
Table 2.3.1. Data Format for Sensor Input Tape

<table>
<thead>
<tr>
<th>PARAMETER DESCRIPTION</th>
<th>SOURCE/M/SID</th>
<th>ENG. UNITS</th>
<th>RECORD LOCATION</th>
<th>RECORD TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Year</td>
<td>V90W4749C</td>
<td>Years</td>
<td>1</td>
<td>I</td>
</tr>
<tr>
<td>2. Month</td>
<td>&quot;</td>
<td>Months</td>
<td>2</td>
<td>I</td>
</tr>
<tr>
<td>3. Day</td>
<td>&quot;</td>
<td>Days</td>
<td>3</td>
<td>I</td>
</tr>
<tr>
<td>4. Hour</td>
<td>&quot;</td>
<td>Hours</td>
<td>4</td>
<td>I</td>
</tr>
<tr>
<td>5. Minute</td>
<td>&quot;</td>
<td>Minutes</td>
<td>5</td>
<td>I</td>
</tr>
<tr>
<td>6. Seconds</td>
<td>&quot;</td>
<td>Seconds</td>
<td>6</td>
<td>I</td>
</tr>
<tr>
<td>7. GMT Time</td>
<td>&quot;</td>
<td>Seconds</td>
<td>7</td>
<td>D</td>
</tr>
<tr>
<td>8. M50 Shuttle - x</td>
<td>V90H4277C-79C</td>
<td>Feet</td>
<td>9</td>
<td>D</td>
</tr>
<tr>
<td>9. - y</td>
<td>&quot;</td>
<td>Feet</td>
<td>11</td>
<td>D</td>
</tr>
<tr>
<td>10. - z</td>
<td>&quot;</td>
<td>Feet</td>
<td>13</td>
<td>D</td>
</tr>
<tr>
<td>11. - x</td>
<td>V90L4281C-83C</td>
<td>Feet/Sec</td>
<td>15</td>
<td>D</td>
</tr>
<tr>
<td>12. - y</td>
<td>&quot;</td>
<td>Feet/Sec</td>
<td>17</td>
<td>D</td>
</tr>
<tr>
<td>13. - z</td>
<td>&quot;</td>
<td>Feet/Sec</td>
<td>19</td>
<td>D</td>
</tr>
<tr>
<td>14. COAS Data Good Flag</td>
<td>V90X4849X</td>
<td>--</td>
<td>21</td>
<td>I</td>
</tr>
<tr>
<td>15. COAS Horizontal Measurement</td>
<td>V90U4847C</td>
<td>Radians</td>
<td>22</td>
<td>S</td>
</tr>
<tr>
<td>16. COAS Vertical Measurement</td>
<td>V90U4848C</td>
<td>Radians</td>
<td>23</td>
<td>S</td>
</tr>
<tr>
<td>17. RNDZ Radar Angles Flag</td>
<td>V90X4901X</td>
<td>--</td>
<td>24</td>
<td>I</td>
</tr>
<tr>
<td>18. RNDZ Radar Roll</td>
<td>V90U4893C</td>
<td>Degrees</td>
<td>25</td>
<td>S</td>
</tr>
<tr>
<td>19. RNDZ Radar Pitch</td>
<td>V9074894C</td>
<td>Degrees</td>
<td>26</td>
<td>S</td>
</tr>
<tr>
<td>20. RNDZ Range Flag</td>
<td>V90X4899X</td>
<td>--</td>
<td>27</td>
<td>I</td>
</tr>
<tr>
<td>21. RNDZ Range Measurement</td>
<td>V90U4895C</td>
<td>Feet</td>
<td>28</td>
<td>S</td>
</tr>
<tr>
<td>22. RNDZ Range Rate Flag</td>
<td>V90X4900X</td>
<td>--</td>
<td>29</td>
<td>I</td>
</tr>
<tr>
<td>23. Rate Measurement</td>
<td>V90U4896C</td>
<td>Feet/Sec</td>
<td>30</td>
<td>S</td>
</tr>
<tr>
<td>24. Star Tracker Data Good Flag</td>
<td>V90X4835X</td>
<td>--</td>
<td>31</td>
<td>I</td>
</tr>
<tr>
<td>27. Nav. Powered Flight Flag</td>
<td>V90X4264X</td>
<td>--</td>
<td>34</td>
<td>I</td>
</tr>
<tr>
<td>28. M50 Accum. - x</td>
<td>V90L8927C-29C</td>
<td>Feet/Sec</td>
<td>35</td>
<td>D</td>
</tr>
</tbody>
</table>

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

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

I-13
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

<table>
<thead>
<tr>
<th>PARAMETER SOURCE/ENGINE RECORD</th>
<th>ENG. UNITS</th>
<th>RECORD LOCATION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSSR NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. DA.THRESHOLD</td>
<td>V90A4747C</td>
<td>a/s</td>
<td>S</td>
</tr>
<tr>
<td>2. DVDISP(1)</td>
<td>V90L5181C</td>
<td>a/s</td>
<td>S</td>
</tr>
<tr>
<td>3. DVDISP(2)</td>
<td>V90L5182C</td>
<td>a/s</td>
<td>S</td>
</tr>
<tr>
<td>4. DVDISP(3)</td>
<td>V90L5183C</td>
<td>a/s</td>
<td>S</td>
</tr>
<tr>
<td>5. E(1,1)</td>
<td>V90U4006C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>6. E(2,2)</td>
<td>V90U4027C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>7. E(3,3)</td>
<td>V90U4047C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>8. E(4,4)</td>
<td>V90U4067C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>9. E(5,5)</td>
<td>V90U4087C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>10. E(6,6)</td>
<td>V90U4108C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>11. E(1,2)</td>
<td>V90U4007C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>12. E(1,3)</td>
<td>V90U4008C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>13. E(1,4)</td>
<td>V90U4009C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>14. E(1,5)</td>
<td>V90U4011C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>15. E(1,6)</td>
<td>V90U4012C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>16. E(2,3)</td>
<td>V90U4028C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>17. E(2,4)</td>
<td>V90U4029C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>18. E(2,5)</td>
<td>V90U4030C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>19. E(2,6)</td>
<td>V90U4031C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>20. E(3,4)</td>
<td>V90U4048C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>21. E(3,5)</td>
<td>V90U4049C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>22. E(3,6)</td>
<td>V90U4050C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>23. E(4,5)</td>
<td>V90U4068C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>24. E(4,6)</td>
<td>V90U4069C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>25. E(10,10)</td>
<td>V90U4088C</td>
<td>a/s</td>
<td>D</td>
</tr>
<tr>
<td>26. NAV.ANGLES.AIF</td>
<td>V90J4253C</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>27. NAV.RANGE.AIF</td>
<td>V90J4268C</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>28. NAV.RDOT.AIF</td>
<td>V90J4269C</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>29. NAV.SV.SOURCE.FOR.UPP</td>
<td>V90X4977X</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>30. NAV.DO.COVAR.REINIT</td>
<td>V90X4256X</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>31. NAV.DO.FILTERED.TO.PROP</td>
<td>V90X4975X</td>
<td>a/s</td>
<td>I</td>
</tr>
<tr>
<td>32. NAV.DO.FILTR.SLOW RATE</td>
<td>V90X4257X</td>
<td>a/s</td>
<td>I</td>
</tr>
</tbody>
</table>
### Table 2.4.1. Data Format for Sensor Environment Tape (Continued)

<table>
<thead>
<tr>
<th>PARAMETER SOURCE/</th>
<th>ENG. RECORD</th>
<th>LOCATION</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>M/SID NAME</td>
<td>UNITS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33. NAV.DO.ORB.TO.TGT V90X4258X</td>
<td>n</td>
<td>54</td>
<td>I</td>
</tr>
<tr>
<td>34. NAV.DO.OV.UPLINK V90X4260X</td>
<td>n</td>
<td>55</td>
<td>I</td>
</tr>
<tr>
<td>35. NAV.DO.PROP.TO.FILTERED V90X4976X</td>
<td>n</td>
<td>56</td>
<td>I</td>
</tr>
<tr>
<td>36. NAV.DO.TGT.TO.ORB V90X4259X</td>
<td>n</td>
<td>57</td>
<td>I</td>
</tr>
<tr>
<td>37. NAV.DO.TV.UPLINK V90X426AX</td>
<td>n</td>
<td>58</td>
<td>I</td>
</tr>
<tr>
<td>38. NAV.MEAS.ENABLE V90X4262X</td>
<td>n</td>
<td>59</td>
<td>I</td>
</tr>
<tr>
<td>39. NAV.MM.CODE V90J4263C</td>
<td>n</td>
<td>60</td>
<td>I</td>
</tr>
<tr>
<td>40. NAV.PWRD.FLT.NAV V90X4264X</td>
<td>n</td>
<td>61</td>
<td>I</td>
</tr>
<tr>
<td>41. NAV.RR.ANGLES.ENABLE V90X4273X</td>
<td>n</td>
<td>62</td>
<td>I</td>
</tr>
<tr>
<td>42. NAV.ST.ENABLE V90X4274X</td>
<td>n</td>
<td>63</td>
<td>I</td>
</tr>
<tr>
<td>43. REND.NAV.FLAG V93X6220X</td>
<td>n</td>
<td>64</td>
<td>I</td>
</tr>
<tr>
<td>44. T.CURRENT.FILT V90W4749C</td>
<td>s</td>
<td>65</td>
<td>D</td>
</tr>
<tr>
<td>45. TFOFF V90W4960C</td>
<td>s</td>
<td>67</td>
<td>D</td>
</tr>
<tr>
<td>46. TFON V90W4959C</td>
<td>s</td>
<td>69</td>
<td>D</td>
</tr>
<tr>
<td>47. T.ORB.STATE.UPDATE V94W3727C</td>
<td>s</td>
<td>71</td>
<td>D</td>
</tr>
<tr>
<td>48. T.TV.STATE.UPDATE V90W4939C</td>
<td>s</td>
<td>73</td>
<td>D</td>
</tr>
<tr>
<td>49. V_FORCE(1) V90U4956C</td>
<td>LBF</td>
<td>75</td>
<td>S</td>
</tr>
<tr>
<td>50. V_FORCE(2) V90U4957C</td>
<td>LBF</td>
<td>76</td>
<td>S</td>
</tr>
<tr>
<td>51. V_FORCE(3) V90U4958C</td>
<td>LBF</td>
<td>77</td>
<td>S</td>
</tr>
<tr>
<td>52. UNUSED</td>
<td></td>
<td>78</td>
<td>I</td>
</tr>
<tr>
<td>53. T.LAST.FILT.TLM V90W4285C</td>
<td>s</td>
<td>79</td>
<td>D</td>
</tr>
<tr>
<td>54. R.FILT.TLM(1) V90H4277C</td>
<td>f</td>
<td>81</td>
<td>D</td>
</tr>
<tr>
<td>55. R.FILT.TLM(2) V90H4278C</td>
<td>f</td>
<td>83</td>
<td>D</td>
</tr>
<tr>
<td>56. R.FILT.TLM(3) V90H4279C</td>
<td>f</td>
<td>85</td>
<td>D</td>
</tr>
<tr>
<td>57. V.FILT.TLM(1) V90L4281C</td>
<td>f/s</td>
<td>87</td>
<td>D</td>
</tr>
<tr>
<td>58. V.FILT.TLM(2) V90L4282C</td>
<td>f/s</td>
<td>89</td>
<td>D</td>
</tr>
<tr>
<td>59. V.FILT.TLM(3) V90L4283C</td>
<td>f/s</td>
<td>91</td>
<td>D</td>
</tr>
<tr>
<td>60. R.TV.TLM(1) V90H4287C</td>
<td>f</td>
<td>93</td>
<td>D</td>
</tr>
<tr>
<td>61. R.TV.TLM(2) V90H4289C</td>
<td>f</td>
<td>95</td>
<td>D</td>
</tr>
<tr>
<td>62. R.TV.TLM(3) V90H4289C</td>
<td>f</td>
<td>97</td>
<td>D</td>
</tr>
<tr>
<td>63. V.TV.TLM(1) V90L4291C</td>
<td>f/s</td>
<td>99</td>
<td>D</td>
</tr>
<tr>
<td>64. V.TV.TLM(2) V90L4292C</td>
<td>f/s</td>
<td>101</td>
<td>D</td>
</tr>
<tr>
<td>65. V.TV.TLM(3) V90L4293C</td>
<td>f/s</td>
<td>103</td>
<td>D</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,n</td>
<td>Header ( n = ) number of header records</td>
</tr>
<tr>
<td>( n+1,n+5 )</td>
<td>Dictionary</td>
</tr>
<tr>
<td>( n+6       )</td>
<td>*beg</td>
</tr>
<tr>
<td>( n+7,n+7+m )</td>
<td>data frames ( m = ) number of data records</td>
</tr>
<tr>
<td>( n+7+m+1 )</td>
<td>*end</td>
</tr>
</tbody>
</table>

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 preceding 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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>DP</td>
<td>Time-tag</td>
</tr>
<tr>
<td>3</td>
<td>CHAR*4</td>
<td>Frame ID</td>
</tr>
<tr>
<td>4</td>
<td>I</td>
<td>Edit status</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Word</th>
<th>Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pname</td>
<td>c72</td>
<td>Name of gff file to create or open. Pname is a user-defined input to the open routines, gfnew and gfopen.</td>
</tr>
<tr>
<td>19</td>
<td>punit</td>
<td>i</td>
<td>Unit number on which to open the gff file. Punit is a user-defined input to the open routines, gfnew and gfopen.</td>
</tr>
<tr>
<td>20</td>
<td>pfrmsz</td>
<td>i</td>
<td>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.</td>
</tr>
<tr>
<td>21</td>
<td>phdrsz</td>
<td>i</td>
<td>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.</td>
</tr>
<tr>
<td>22</td>
<td>pcrec</td>
<td>i</td>
<td>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.</td>
</tr>
</tbody>
</table>
Table 3.1.4.1. I/O Packet Contents (Continued)

<table>
<thead>
<tr>
<th>Word</th>
<th>Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>prwflg</td>
<td>i</td>
<td>Read/write status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>24</td>
<td>pnrec</td>
<td>i</td>
<td>Number of records in the gff file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pnrec is always current.</td>
</tr>
<tr>
<td>25</td>
<td>pstat</td>
<td>i</td>
<td>Error code returned from the gff routines.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. trying to access within the dictionary or the header</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. trying to access past the end of file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. file does not have write permission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. invalid time requested</td>
</tr>
<tr>
<td>26-27</td>
<td>ptoff</td>
<td>dp</td>
<td>Time offset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>28</td>
<td>ptype</td>
<td>c4</td>
<td>File type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>29-33</td>
<td>pbdate</td>
<td>i(5)</td>
<td></td>
</tr>
</tbody>
</table>
Table 3.1.4.1. I/O Packet Contents (Continued)

<table>
<thead>
<tr>
<th>Word</th>
<th>Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>34-35</td>
<td>pbsec</td>
<td>dp</td>
<td>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. 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.</td>
</tr>
<tr>
<td>36-37</td>
<td>pstrt</td>
<td>dp</td>
<td>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.</td>
</tr>
<tr>
<td>38-39</td>
<td>pstop</td>
<td>dp</td>
<td>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.</td>
</tr>
<tr>
<td>40</td>
<td>pnhec</td>
<td>i</td>
<td>Number of header records. The header is 90 words of data and is written to the file &quot;pfrmsz&quot; 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.</td>
</tr>
<tr>
<td>41</td>
<td>pndict</td>
<td>i</td>
<td>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.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Word</th>
<th>Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hnhrec</td>
<td>i</td>
<td>Number of header records. See iop - pnhrec for a description.</td>
</tr>
<tr>
<td>2</td>
<td>htype</td>
<td>c4</td>
<td>File type. See iop - ptype for a description.</td>
</tr>
<tr>
<td>3-4</td>
<td>hcdate</td>
<td>c8</td>
<td>Creation data. When a gff file is created, the current data is stored as the creation date in the form yy/mm/dd.</td>
</tr>
<tr>
<td>5-8</td>
<td>hcvver</td>
<td>c16</td>
<td>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 &quot;name-ver/gff-ver&quot;, 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 version is defined by the gff library in gfnew.</td>
</tr>
<tr>
<td>9-34</td>
<td>hcdes</td>
<td>c104</td>
<td>Creation description. The creation description is a user-defined input to the create routine, gfnew.</td>
</tr>
<tr>
<td>35-36</td>
<td>hudate</td>
<td>c8</td>
<td>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.</td>
</tr>
<tr>
<td>37-40</td>
<td>huver</td>
<td>c16</td>
<td>Update program name and version. See HEADER - hcvver for a description.</td>
</tr>
</tbody>
</table>
Table 3.1.5.1. Header Contents

<table>
<thead>
<tr>
<th>Word</th>
<th>Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41-66</td>
<td>hudes</td>
<td>c104</td>
<td>Update description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>67-71</td>
<td>hbdate</td>
<td>i(5)</td>
<td>File base date.</td>
</tr>
<tr>
<td>72-73</td>
<td></td>
<td></td>
<td>See iop - pbdate, pbsec for a description.</td>
</tr>
<tr>
<td>74-75</td>
<td>hstrt</td>
<td>dp</td>
<td>Start time.</td>
</tr>
<tr>
<td>76-77</td>
<td>hstop</td>
<td>dp</td>
<td>Stop time.</td>
</tr>
<tr>
<td>78</td>
<td>hnrec</td>
<td>i</td>
<td>Number of data records in the file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>79</td>
<td>hfrmsz</td>
<td>i</td>
<td>Record length.</td>
</tr>
<tr>
<td>80-90</td>
<td>hspec</td>
<td>i</td>
<td>File specific block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

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](image)
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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-10</td>
<td>DP(3)</td>
<td>Vector for Sensed Velocity</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-12</td>
<td>DP(4)</td>
<td>Quaternion for Attitude</td>
</tr>
</tbody>
</table>

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:
3.5 EPHMERIS 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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-10</td>
<td>DP(3)</td>
<td>Position Vector</td>
</tr>
<tr>
<td>11-16</td>
<td>DP(3)</td>
<td>Velocity Vector</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-12</td>
<td>DP(4)</td>
<td>Quaternion IMU-1</td>
</tr>
<tr>
<td>13-20</td>
<td>DP(4)</td>
<td>Quaternion IMU-2</td>
</tr>
<tr>
<td>21-28</td>
<td>DP(4)</td>
<td>Quaternion IMU-3</td>
</tr>
<tr>
<td>29-36</td>
<td>DP(4)</td>
<td>Quaternion Selected</td>
</tr>
<tr>
<td>37-38</td>
<td>I(2)</td>
<td>Source of Selection (spare)</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-10</td>
<td>DP(3)</td>
<td>Vector IMU-1</td>
</tr>
<tr>
<td>11-16</td>
<td>DP(3)</td>
<td>Vector IMU-2</td>
</tr>
<tr>
<td>17-22</td>
<td>DP(3)</td>
<td>Vector IMU-3</td>
</tr>
<tr>
<td>23-28</td>
<td>DP(3)</td>
<td>Vector Selected</td>
</tr>
<tr>
<td>29-30</td>
<td>I(2)</td>
<td>Source of Selection (spare)</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-4+2S</td>
<td>D</td>
<td>Solution Vector</td>
</tr>
<tr>
<td>5+2S-4+2S+D</td>
<td>DP(C)</td>
<td>Lower Triangular Covariance by Rows</td>
</tr>
</tbody>
</table>

Where

\[
S = \text{Solution size} \\
D = \frac{S(S+1)}{2}.
\]
3.9 RELATIVE TRAJECTORY FILE

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

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-16</td>
<td>DP(6)</td>
<td>Fiducial State</td>
</tr>
<tr>
<td>17-28</td>
<td>DP(6)</td>
<td>Relative State</td>
</tr>
</tbody>
</table>

3.10 PLOT DISPLAY FILES

Table 3.10.1 summarizes the various options for output parameters from the program xqdisp. Here 0 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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-16</td>
<td>DP(6)</td>
<td>Coordinates</td>
</tr>
</tbody>
</table>
Table 3.10.1. Parameter Groups

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

In column 5,

A indicates orbiter attitude needed
O 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:

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-22</td>
<td>DP(9)</td>
<td>Components of Matrix by rows</td>
</tr>
</tbody>
</table>

3.12 COVARIANCE FILE

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

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-16</td>
<td>DP(6)</td>
<td>Sigmas for Base State</td>
</tr>
<tr>
<td>17-28</td>
<td>DP(6)</td>
<td>Sigmas for Relative State</td>
</tr>
</tbody>
</table>

3.13 BIAS FILE

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

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-A</td>
<td>DP(S-12)</td>
<td>Bias Solutions</td>
</tr>
<tr>
<td>A+1-B</td>
<td>DP(S-12)</td>
<td>Bias Solution Sigmas</td>
</tr>
</tbody>
</table>

where s is the solution size

\[
A = 4 + (S-12) \times 2 \\
B = A + (-12) \times 2
\]
3.14 RESIDUAL FILE

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

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-6</td>
<td>DP</td>
<td>Edit Status (-1 or 1)</td>
</tr>
<tr>
<td>7-8</td>
<td>DP</td>
<td>Observation Value</td>
</tr>
<tr>
<td>9-10</td>
<td>DP</td>
<td>Residual Value</td>
</tr>
<tr>
<td>11-12</td>
<td>DP</td>
<td>Residual Sigma</td>
</tr>
<tr>
<td>13-14</td>
<td>DP</td>
<td>Bias Value</td>
</tr>
<tr>
<td>15-16</td>
<td>DP</td>
<td>Bias Sigma</td>
</tr>
</tbody>
</table>

3.15 QA FILES

The following files are created by the various QA processors:

The Attitude Angular Acceleration File

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-10</td>
<td>DP(3)</td>
<td>Angular Acceleration</td>
</tr>
<tr>
<td>11-12</td>
<td>DP</td>
<td>Magnitude</td>
</tr>
</tbody>
</table>

The Sensed Acceleration File

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MXI</td>
<td>Header</td>
</tr>
<tr>
<td>5-10</td>
<td>DP(3)</td>
<td>Sensed Acceleration</td>
</tr>
<tr>
<td>11-12</td>
<td>DP</td>
<td>Magnitude</td>
</tr>
</tbody>
</table>
The Master Noise File

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-6</td>
<td>DP</td>
<td>End time of interval (Begin time is timetag)</td>
</tr>
<tr>
<td>7-8</td>
<td>DP</td>
<td>Average value of observation</td>
</tr>
<tr>
<td>9-28</td>
<td>DP(IO)</td>
<td>Value of noise associated with various order of divided difference</td>
</tr>
</tbody>
</table>

The Selected Noise File

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-6</td>
<td>DP</td>
<td>End time of intervals (Begin time is timetag)</td>
</tr>
<tr>
<td>7-8</td>
<td>DP</td>
<td>Average value of observation</td>
</tr>
<tr>
<td>9-10</td>
<td>DP</td>
<td>Selected value of noise for time interval</td>
</tr>
</tbody>
</table>

The Star File

<table>
<thead>
<tr>
<th>Integer Word</th>
<th>Type</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>MIX</td>
<td>Header</td>
</tr>
<tr>
<td>5-6</td>
<td>DP</td>
<td>Inertial Azimuth Angle</td>
</tr>
<tr>
<td>7-8</td>
<td>DP</td>
<td>Inertial Elevation Angle</td>
</tr>
<tr>
<td>9-10</td>
<td>DP</td>
<td>Angle between the previous pointing vector and current pointing vector</td>
</tr>
</tbody>
</table>

3.16 SENSOR FILE

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

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


APPENDIX II

CONTEXT FILES
<table>
<thead>
<tr>
<th></th>
<th>total 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-r-r-r--- 1 relbet RELBET 261 Dec 10 13:42 BaseTime.h</td>
</tr>
<tr>
<td>2</td>
<td>-r-r-r--- 1 relbet RELBET 2942 Dec 10 12:46 Gbfile.h</td>
</tr>
<tr>
<td>3</td>
<td>-r-r-r--- 1 relbet RELBET 1333 Dec 10 13:42 Time.h</td>
</tr>
<tr>
<td>4</td>
<td>-r-r-r--- 1 relbet RELBET 130 Dec 10 13:07 anlists.h</td>
</tr>
<tr>
<td>5</td>
<td>-r-r-r--- 1 relbet RELBET 1193 Dec 10 13:03 array.h</td>
</tr>
<tr>
<td>6</td>
<td>-r-r-r--- 1 relbet RELBET 339 Dec 10 12:59 cell.h</td>
</tr>
<tr>
<td>7</td>
<td>-r-r-r--- 1 relbet RELBET 122 Dec 10 12:42 fileio.h</td>
</tr>
<tr>
<td>8</td>
<td>-r-r-r--- 1 relbet RELBET 381 Dec 10 12:46 gbfile.x.h</td>
</tr>
<tr>
<td>9</td>
<td>-r-r-r--- 1 relbet RELBET 161 Dec 10 12:47 gflags.h</td>
</tr>
<tr>
<td>10</td>
<td>-r-r-r--- 1 relbet RELBET 1498 Dec 10 12:47 gbfsstruct.h</td>
</tr>
<tr>
<td>11</td>
<td>-r-r-r--- 1 relbet RELBET 1657 Dec 10 12:47 gbhdrrec.h</td>
</tr>
<tr>
<td>12</td>
<td>-r-r-r--- 1 relbet RELBET 427 Dec 10 12:47 gbmiscdef.h</td>
</tr>
<tr>
<td>13</td>
<td>-r-r-r--- 1 relbet RELBET 642 Dec 10 13:07 gnamlist.h</td>
</tr>
<tr>
<td>14</td>
<td>-r-r-r--- 1 relbet RELBET 214 Dec 10 13:40 gstack.h</td>
</tr>
<tr>
<td>15</td>
<td>-r-r-r--- 1 relbet RELBET 435 Dec 11 09:48 gstack.typ.h</td>
</tr>
<tr>
<td>16</td>
<td>-r-r-r--- 1 relbet RELBET 73 Dec 10 12:58 input.h</td>
</tr>
<tr>
<td>17</td>
<td>-r-r-r--- 1 relbet RELBET 625 Dec 10 12:59 lists.h</td>
</tr>
<tr>
<td>18</td>
<td>-r-r-r--- 1 relbet RELBET 500 Dec 10 13:07 message.h</td>
</tr>
<tr>
<td>19</td>
<td>-r-r-r--- 1 relbet RELBET 18 Dec 10 13:00 newcell.h</td>
</tr>
<tr>
<td>20</td>
<td>-r-r-r--- 1 relbet RELBET 968 Dec 10 13:07 ptrlist.h</td>
</tr>
<tr>
<td>21</td>
<td>-r-r-r--- 1 relbet RELBET 235 Dec 10 13:04 quat.h</td>
</tr>
<tr>
<td>22</td>
<td>-r-r-r--- 1 relbet RELBET 281 Dec 10 13:07 ratetable.h</td>
</tr>
<tr>
<td>23</td>
<td>-r-r-r--- 1 relbet RELBET 645 Dec 10 12:42 stop.h</td>
</tr>
<tr>
<td>24</td>
<td>-r-r-r--- 1 relbet RELBET 601 Dec 10 13:07 timeline.h</td>
</tr>
</tbody>
</table>
```c
#include <stdio.h>

extern CALTIME BaseCalTime;
extern double JD_BaseTime;
extern double GBeginTime;
extern double GEndTime;
extern double GDeltTime;
extern double LBeginTime;
extern double LEndTime;
extern double LDeltTime;

#define BASETIME_DOT_H 1
#define BASETIME_DOT_H 1
```

Dec 10 13:42 1986 BaseTime.h Page 1
typedef struct
{
  /* file record size in bytes */
  int size;
  /* time offsets for field */
  int off_time;
  /* time base */
  int base_time;
  /* id name */
  char *id;
  /* file mode */
  char *mode;
  /* caching mode */
  int cflags;
  /* file type */
  int type;
  /* file name */
  char *name;
} GFILE;

#define GFILE_FMT "GFILE_FMT\n"

/* file record structure */
typedef struct
{
  /* file record size in bytes */
  int size;
  /* time offsets for field */
  int off_time;
  /* time base */
  int base_time;
  /* id name */
  char *id;
  /* file mode */
  char *mode;
  /* caching mode */
  int cflags;
  /* file type */
  int type;
  /* file name */
  char *name;
} GFILE;

/* file record structure */
typedef struct
{
  /* file record size in bytes */
  int size;
  /* time offsets for field */
  int off_time;
  /* time base */
  int base_time;
  /* id name */
  char *id;
  /* file mode */
  char *mode;
  /* caching mode */
  int cflags;
  /* file type */
  int type;
  /* file name */
  char *name;
} GFILE;
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 GFILE_DOT_DEF_ONLY
66
67
68    extern gbtoff();
69    extern gbwhead();
70    extern gbrhead();
71
72    extern gbpos();
73
75    extern char *gbread();
76
77    extern gbwrite();
78
79    extern char *gbdread();
80
81    extern gbdwrite();
82
83    extern GFILE *makeGBF();
84    extern char *makeGBData();
85
86    extern GFILEPTR gbopen();
87
88    extern GFILEPTR 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 GFILE_DOT_H 1
104 #endif
```c
#define TIME_DOT_H

typedef struct { /*calendar date*/
   int year;
   int month; /*1=Jan, 2=Feb, ...*/
   int day;
   int weekday; /*O=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;

#else TIME_DOT_DEF_ONLY

extern CALTIME *GetCurTime();
extern void fprtCurTime();
extern void prtCurTime();
extern void 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 mnthsnum();
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    extern void fprinttime();
58    extern void sprinttime();
59    extern void prrtctime();
60    extern void fprintsec();
61    extern void sprintsec();
62    extern void prrsec();
63
64    #define TIME_DOT_H 1
65
66    #endif
67
68    #endif
#ifndef ANLISTS_DOT_H
extern int addANLitem();
extern int prtANLitem();
extern void prtANList();
#define ANLISTS_DOT_H 1
#endif
#ifndef ARRAY_DOT_H
    /* matrix.c */
extern double *mxm(); /* matrix.c */
extern double *mxmc(); /* matrix.c */
extern double *mxv(); /* matrix.c */
extern double *mtxv(); /* matrix.c */
extern double *mt(); /* matrix.c */
extern double *mtxm(); /* matrix.c */
extern double *mxmt(); /* matrix.c */
    /* prarray.c */
extern int prt3mat(); /* prarray.c */
extern int prt3vec(); /* prarray.c */
extern int prt3mat(); /* prarray.c */
extern int fparray(); /* prarray.c */
extern int fparray(); /* prarray.c */
    /* rmatrix.c */
extern double *rmmx(); /* rmatrix.c */
extern double *rmxmc(); /* rmatrix.c */
extern double *rmxv(); /* rmatrix.c */
    /* vector.c */
extern double *vadd(); /* vector.c */
extern double *vaddto(); /* vector.c */
extern double vdist(); /* vector.c */
extern double vdot(); /* vector.c */
extern double *vfadd(); /* vector.c */
extern double *vfaddto(); /* vector.c */
extern double *vsub(); /* vector.c */
extern double *vsub(); /* vector.c */
extern double vrss(); /* vector.c */
extern double *vsset(); /* vector.c */
extern double *vzero(); /* vector.c */
extern double vunit(); /* vector.c */
define ARRAY_DOT_H 1
#endif
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;
    }
} val;
}
CELL, *P_CELL;

typedef P_CELL (*P_FUNCTION)();
struct FILE_INFO {
    char *name;
    char *fraid;
    char *(function)();
    char *gbf;
    int option;
    struct FILE_INFO *next;
};
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>extern gloff():</td>
</tr>
<tr>
<td>2</td>
<td>extern gheader():</td>
</tr>
<tr>
<td>3</td>
<td>extern gpos():</td>
</tr>
<tr>
<td>4</td>
<td>extern char *gbrad():</td>
</tr>
<tr>
<td>5</td>
<td>extern char *gbread():</td>
</tr>
<tr>
<td>6</td>
<td>extern char *gbfile():</td>
</tr>
<tr>
<td>7</td>
<td>extern char *gbwrite():</td>
</tr>
<tr>
<td>8</td>
<td>extern char *gbhead():</td>
</tr>
<tr>
<td>9</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>10</td>
<td>extern char *gbkhead():</td>
</tr>
<tr>
<td>11</td>
<td>extern char *gbclose():</td>
</tr>
<tr>
<td>12</td>
<td>extern char *gbclose():</td>
</tr>
<tr>
<td>13</td>
<td>extern char *gbfree():</td>
</tr>
<tr>
<td>14</td>
<td>extern char *gbfree():</td>
</tr>
<tr>
<td>15</td>
<td>extern char *gbopen():</td>
</tr>
<tr>
<td>16</td>
<td>extern char *gopen():</td>
</tr>
<tr>
<td>17</td>
<td>extern char *gclose():</td>
</tr>
<tr>
<td>18</td>
<td>extern char *gclose():</td>
</tr>
<tr>
<td>19</td>
<td>extern char *gfree():</td>
</tr>
<tr>
<td>20</td>
<td>extern char *gfree():</td>
</tr>
<tr>
<td>21</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>22</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>23</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>24</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>25</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>26</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>27</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>28</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>29</td>
<td>extern char *ghead():</td>
</tr>
<tr>
<td>30</td>
<td>extern char *ghead():</td>
</tr>
</tbody>
</table>
1 /*flags and such for gbfile routines*/
2 #define ERRVAL -1
3 /*error occurred*/
4 #define NOERR 0
5 /*no error*/
6 #define NO_TBYTE -1
7 /*no time word in data record*/
/*names for gbs file fields and text buffer locations*/
/*To use short hand for fields:
All entries have the string "GBS" preceding the field names which
must be specified by a define or HS as the gbs 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.n_record
#define RSIZE  GBS.rs
#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 HDRS 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 HCOVER 16
#define HCDES 36
#define HUDATE 136
#define HUVER 144
#define HUDES 160
#define HBDATE 264
#define HBYR 264
#define HBM0 268
#define HBA0 272
#define HBHR 276
#define HEMIN 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 LHSTYPE sizeof(4)
#define LHCDATE sizeof(8)
#define LHCOVER sizeof(16)
#define LHCDES sizeof(104)
#define LHUDATE sizeof(8)
#define LHUVER sizeof(16)
#define LHUDES sizeof(104)
#define LHBDATE (5 * sizeof(int))
#define LHBYR sizeof(int)
#define HBM0 sizeof(int)
#define HBA0 sizeof(int)
#define HBHR sizeof(int)
#define HEMIN sizeof(int)
#define HBSEC sizeof(double)
#define HSTRT sizeof(double)
#define HSTOP sizeof(double)
#define LHNREC sizeof(int)
57  #define LHFRMSZ  sizeof(int)
58  #define LHSPEC   sizeof(int)
59  #define LHTBYTE  sizeof(int)
/*miscellaneous defines for header manipulation*/

#define HDR hdrbuf +
#define HDRBUF_SIZE 520

#define GET_CUR_FPOS (if((FPOS = lseek(FDES,0,1))< ORIG) FPOS = ORIG;)
#define RESTORE_FPOS (if(FPOS < ORIG) FPOS = ORIG; lseek(FDES,FPOS,0);)

#define WORDSIZE sizeof(int)
#define TSIZE sizeof(CALTIME)

#define CUR_YR (CurTime->date).year
#define CUR_MO (CurTime->date).month
#define CUR_DA (CurTime->date).day
```c
#define GNAMLIST_DOT_H

typedef struct {
    char *adress; /* address of named item*/
    char *name;  /* name of item*/
} GNL_ITEM, *GNL_ITEM_PTR;

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

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

#ifndef GNAMLIST_DOT_DEF_ONLY
extern void fprtnlist();
extern GNLIST *Make_GNLIST();
extern char *GNLgetName();
extern int Add_GNLIST();
#endif

#define GNAMLIST_DOT_H 1
#endif
```
extern char *q_pop();
extern char *q_push();
extern char *gspop();
extern char *gspush();
extern char *gspeek();
extern char *gspeak();
extern char *gempty();
extern char *gsfree();
extern P_GSTACK make_gstack();
typedef struct {
    char *base;
    char *front;
    char *next;
    int block_size;
    int max;
    int n_blocks;
} GSTACK, *P_GSTACK, QUEUE, *P_QUEUE;

/* A stack consists of blocks with fixed size, numbered from bottom to top starting with empty stack number 0. The pointer to next block in stack points to next available space. The pointer to front is basically a Queue application and indicates the front of line for queu pop*/
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;
    } 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(), isnumber(), 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 fpriStars();
extern int prtStars();
extern char *Stat_Msg;
extern int Error_Count;
extern int Warning_Count;
extern int StatErrExit;
extern void fpriFinish();
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
#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 addresses. 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*/

#ifdef PTRLIST_DOT_DEF_ONLY
extern int addptr();
extern int rmptt();
extern PTRLIST *makeplist();
extern void freeplist();
#endif

#define PTRLIST_DOT_H 1
#endif
```c
#define RATE_TABLE_DOT_H

typedef struct { double time;
  double base;     /* base value */
  double rate;     /* rate of flow per unit time */
} BASE_RATE_FLOW, *BASE_RATE_FLOW_PTR;

#define RATE_TABLE_DOT_DEF_ONLY

double get_rate_table_value();

#endif

#define RATE_TABLE_DOT_H

#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_dp_fj;
    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_thesh;
    float dv_disp[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 fprintfTimeLine();
#endif
#endif
#define TIMELINE_DOT_H 1
#endif
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a62sc</td>
<td>atmospheric curve fit parameter</td>
<td>DOUBLE PRECISION a62sh</td>
<td>atmospheric curve fit scale height</td>
<td>DOUBLE PRECISION a62hh</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a62r</td>
<td>atmospheric curve fit parameter</td>
<td>DOUBLE PRECISION a62rh</td>
<td>atmospheric curve fit scale height</td>
<td>DOUBLE PRECISION a62rr</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67a</td>
<td>1976 std atmos. empirical factor a</td>
<td>DOUBLE PRECISION a67b</td>
<td>1976 std atmos. empirical factor b</td>
<td>DOUBLE PRECISION a67c</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67f0fl</td>
<td>1976 std atmos. solar flux</td>
<td>DOUBLE PRECISION a67f0fx</td>
<td>1976 std atmos. solar flux</td>
<td>DOUBLE PRECISION a67c1</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67r</td>
<td>1976 std atmos. curve fit parameter a</td>
<td>DOUBLE PRECISION a67r1</td>
<td>1976 std atmos. curve fit parameter b</td>
<td>DOUBLE PRECISION a67r2</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67r3</td>
<td>1976 std atmos. curve fit parameter c</td>
<td>DOUBLE PRECISION a67r4</td>
<td>1976 std atmos. curve fit parameter d</td>
<td>DOUBLE PRECISION a67r5</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67r6</td>
<td>1976 std atmos. curve fit parameter e</td>
<td>DOUBLE PRECISION a67r7</td>
<td>1976 std atmos. curve fit parameter f</td>
<td>DOUBLE PRECISION a67r8</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a67r9</td>
<td>1976 std atmos. curve fit parameter g</td>
<td>DOUBLE PRECISION a67r10</td>
<td>1976 std atmos. curve fit parameter h</td>
<td>DOUBLE PRECISION a67r11</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76a</td>
<td>1976 std atmos. empirical factor a</td>
<td>DOUBLE PRECISION a76b</td>
<td>1976 std atmos. empirical factor b</td>
<td>DOUBLE PRECISION a76c</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76f0fl</td>
<td>1976 std atmos. solar flux</td>
<td>DOUBLE PRECISION a76f0fx</td>
<td>1976 std atmos. solar flux</td>
<td>DOUBLE PRECISION a76c1</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76r</td>
<td>1976 std atmos. curve fit parameter a</td>
<td>DOUBLE PRECISION a76r1</td>
<td>1976 std atmos. curve fit parameter b</td>
<td>DOUBLE PRECISION a76r2</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76r3</td>
<td>1976 std atmos. curve fit parameter c</td>
<td>DOUBLE PRECISION a76r4</td>
<td>1976 std atmos. curve fit parameter d</td>
<td>DOUBLE PRECISION a76r5</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76r6</td>
<td>1976 std atmos. curve fit parameter e</td>
<td>DOUBLE PRECISION a76r7</td>
<td>1976 std atmos. curve fit parameter f</td>
<td>DOUBLE PRECISION a76r8</td>
</tr>
<tr>
<td>a62c</td>
<td>DOUBLE PRECISION a76r9</td>
<td>1976 std atmos. curve fit parameter g</td>
<td>DOUBLE PRECISION a76r10</td>
<td>1976 std atmos. curve fit parameter h</td>
<td>DOUBLE PRECISION a76r11</td>
</tr>
</tbody>
</table>

**Note:** The table contains data for various models and parameters related to atmospheric and solar flux models.
DOUBLE PRECISION a62sc, a62sh, a62rr, a62rh, a76a, a76b, a76f10,
a76rh1, a76a1, a76a2, a76a3, abmc1g, abms1g, abmgde, abmalt, abm1
*, abm0, abmcd1, abmcb1, abmcb2, abmcm2, abmr1f

COMMON /xxatm/ a62sc, a62sh, a62rr, a62rh, a76a, a76b, a76f10,
c start file mtf.bias

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 end file mtf.bias

c
c start file mtf.cct

* *

**start common block xxcct**

**start common block xxcct**

INTEGER cct
c cct info
Indicates which tape drive the input cct is on

INTEGER curyr
c cct info
Current year indicator on cct data records

INTEGER datrec
c cct info
Number of words per data record

INTEGER datscn
c cct info
Number of data records per scan

INTEGER error ( 5 )
c cct info
Error flag

INTEGER licnum
c cct info
Number of samples per cycle to process cct

INTEGER nbuf ( 3200 )
c cct info
Buffer to accept cct hdr and data records

INTEGER numwrdd
c cct info
Number of words per data record

INTEGER numrec
c cct info
Number of records processed

INTEGER o4000
c cct info
The indicator of header record present on cct header records

INTEGER scnrec
c cct info
Number of scans per data record

COMMON /xxcct/ cct, curyr, datrec, datscn, error, licnum, nbuf,
* numwrdd, numrec, o4000, scnrec
c start file mtf.con

c

c

***************
c

11 c start common block xxcon
12 c

13 c

14 c

15 c

16 c

17 c

18 c

19 c

20 c

21 c

22 c

23 c

24 c

25 c

26 c

27 c

28 c

29 c

30 c

31 c

32 c

33 c

34 c
1 c
2 c start file mtf.data
3 c
4 c
5 c
6 c
7 c ***************************************************************
8 c start common block xxdata
9 c ***************************************************************
10 c
11 c INTEGER ldxtab
12 c length of data ex table
13 c
14 c
15 c INTEGER dtxclid( 25 )
16 c data x id table
17 c ids of data that are excluded from processing
18 c
19 c DOUBLE PRECISION gwtab( 2,25 )
20 c wieghts and bias table
21 c observation weights and bias information. on input
22 c
23 c gwtab(1,*)= obs sigma in internal units
24 c
25 c gwtab(2,*)= obs bias in internal units
26 c
27 c COMMON /xxdata/ ldxtab, dtxclid, gwtab
28 c
29 c end block xxdata
30 c
31 c
32 c
33 c end file mtf.data
34 c
c start file mtf.dprm

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 lvh

c 5. sshl vehicle 1 in vehicle 2 shell

c 6. txyz vehicle 2 m50 state

c 7. tel t vehicle 2 m50 elements (a,e,i,node,perigee,tra)

c 8. tuvw vehicle 2 in vehicle 1 uvw

c 9. ttvh vehicle 2 in vehicle 1 lvh

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 attitude)

c 15. auvw uvw inertial pitch-yaw-roll (321 angles of quaternion obtained by

c multiplying uvw \rightarrow m50quat via by the input attitude

c conjugate q, i.e. q1 * con(q) )

c 16. rpv range,veh1 and veh2 position magnitudes,

c range rate,veh1 and veh2 velocity magnitudes

Note: If a relative trajectories is used to generate any of the above

c options both the vehicle and the target option must be set.

COMMON /xxdprm/ spg

c end block xxdprm

c

c end file mtf.dprm
c start file mtf.dwnccct

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  end block dwnfmt_cct

c

c

c  end file mtf.dwnccct

c
c start file mtf.dwnfiles

*****

start common block dwnfmt_files

****************************************************

 CHARACTER*72 frame( 1 )

 output gff file options for downlist formatter,
 c for local interface between user and common block xxfiles in subroutine

dfnput

 CHARACTER*4 hdrid( 1 )

 output gff file options for downlist formatter,
 c for local interface between user and common block xxfiles in subroutine

dfnput

 CHARACTER*4 cfrmid( 10 )

 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, i.e., 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.

 INTEGER filid( 1 )

 frame ID,s for output gff RELBET file

 INTEGER frsize( 3 )

 frame size
 c the integer word size of the different portions of the frame

 (1) the number of integer words including the time-tag which
 c make up the header portion of the frame

 (2) the number of integer words in data portion of the frame

 (3) the number of dummy places to fill after the data portion
 c of the frame in integer words

 INTEGER idseq( 200 )

 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 - the first number must be the internal id of the msid
c representing a time value which is used as the
time-tag of the output frame, this value is further
negated to emphasize that this parameter is destined
for the time-tag position of the header portion of
the frame as mentioned above the actual frame id is
input in the list cfmid.

- the next number must be a number greater than 200
representing the frame id to be placed on
the frame id position of the header portion of the
output frame in the order in which they are
herein specified.

EX: idseq = -74, 1000, 56, 57, 58;

INTEGER dbflag(1)
process frames with bad data
when set non-zero indicates that file frames are to be output
even if some data bad (note: if time-tag is bad no data output)

INTEGER chgdef(1)
change the default values
non_zero indicates that the inputs for fsize, cfmid and idseq
are to be used instead of the defaults.
Note: if chgdef is set then all settings of fsize, cfmid
and idseq must be set.

INTEGER end(1)
end flag
indicates that current input block is the last files input block
and the next input block is an mspids input block

COMMON /dwnfmt_files/ fname, hdrid, cfmid, filid, fsize, idseq,
dflag, chgdef, end
end block dwnfmt_files
end file mtf.dwnfiles
c start file mtf.dwnmsids
3 c
4 c
5 c
6 c **********************************************************************
7 c start common block dwnfmt_msids
8 c **********************************************************************
9 c CHARACTER*10 msid(1)
11 c alphanumeric m/sid number associated with downlisted parameter on cc
13 c t
15 c INTEGER id_nav(2)
17 c the internal datum id and flag indicating nav buffer parameter
19 c (1), data id has range 1 to 200
20 c (2), =0, not part of nav buffer; =1, part of nav buffer
21 c DOUBLE PRECISION off_sc_conv_lo_hi(5)
23 c the double precision data for converting and checking m/sid values
25 c (1), offset term added to value
26 c (2), scale factor multiplied on value before validation check
27 c (3), conversion factor to achieve internal RELBET units
28 c (4), minimum expected value
29 c (5), maximum expected value
30 c default: = 0.0, 1.0, 1.0, -1.e30, 1.e30
31 c INTEGER end(1)
33 c indicates that current input block is the last msids input block
35 c and the next input block is a cc input block
37 c COMMON /dwnfmt_msids/ msid, id_nav, off_sc_conv_lo_hi, end
39 c end block dwnfmt_msids
41 c
42 c
43 c end file mtf.dwnmsids
44 c
c start file mtf.erth
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

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 flattening of earth, ie the ratio of the polar to equatorial radius
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 start file mtf.file

************

start common block xfile

************

CHARACTER*72 name1
att 1 name
name for first attitude file

INTEGER unta1
att 1 unit
unit for first attitude file

INTEGER uzea1
att 1 use
usage code for first attitude file

CHARACTER*72 name2
att2 name
name for second attitude file

INTEGER unta2
att2 unit
unit for second attitude file

INTEGER uzea2
att 2 use
usage code for second attitude file

CHARACTER*72 name1
eph 1 name
name for first ephemeris file

INTEGER unte1
eph 1 unit
unit for first ephemeris file

INTEGER uze1
eph 1 use
usage code for first ephemeris file

CHARACTER*72 name2
eph 2 name
name for second ephemeris file

INTEGER unte2
eph 2 unit
unit for second ephemeris file

INTEGER uze2
eph 2 use
c usage code for second ephemeris file
58 c
59 CHARACTER*72 namd1
60 c data 1 name
61 c name for data file 1
62 c
63 INTEGER untd1
64 c data 1 unit
65 c unit for data file 1
66 c
67 INTEGER uzed1
68 c data 1 use
69 c usage code for data file 1
70 c
71 CHARACTER*72 namd2
72 c data 2 name
73 c name for data file 2
74 c
75 INTEGER untd2
76 c data 2 unit
77 c unit for data file 2
78 c
79 INTEGER uzed2
80 c data 2 use
81 c usage code for data file 2
82 c
83 CHARACTER*72 namv1
84 c svel 1 name
85 c name for sensed velocity file 1
86 c
87 INTEGER untv1
88 c svel 1 unit
89 c unit for sensed velocity file 1
90 c
91 INTEGER uzev1
92 c svel 1 use
93 c usage code for sensed velocity file 1
94 c
95 CHARACTER*72 namv2
96 c svel 2 name
97 c name for sensed velocity file 2
98 c
99 INTEGER untv2
100 c svel 2 unit
101 c unit for sensed velocity file 2
102 c
103 INTEGER uzev2
104 c svel 2 use
105 c usage code for sensed velocity file 2
106 c
107 CHARACTER*72 namr
108 c rtrj name
109 c name for relative trajectory file
110 c
111 INTEGER untr
112 c rtrj unit
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 uns
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 1
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, uzeef1, 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 start file mtf.fils

c *********************************************
c start common block xxfils

c *********************************************
c INTEGER filsta(20)
c internal processing status for files

c INTEGER frnnum(20)
c number of frames processed for files

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, frnnum, frmsze, datseq, dbproc, lastid

c end block xxfils

c end file mtf.fils

c
c start file mtf.gnr1
3 c
4 c
5 c
6 c
7 c*****************************************************************************
8 c start common block xgxnr1
9 c*****************************************************************************
10 c
11 c INTEGER bugs
12 c debug file
13 c
14 c
15 c INTEGER in
16 c input file unit designation
17 c
18 c
19 c INTEGER term
20 c unit for short print
21 c
22 c
23 c INTEGER out
24 c unit for output print
25 c
26 c
27 c INTEGER pbug(19)
28 c debug flags
29 c
30 c
31 c INTEGER pfatal
32 c fatal error flag
33 c initialized to 0 in ingrrs, set negative on calls to gerror.
34 c -1 fatal error: program should stop
35 c 0 no error condition
36 c >0 program defined stop flag
37 c
38 c INTEGER interm
39 c standard input unit
40 c specifies unit for standard input. set to 5 in xgxnr1
41 c
42 c COMMON /xgxnr1/ bugs, in, term, out, pbug, pfatal, interm
43 c
44 c end block xgxnr1
45 c
46 c
47 c
48 c end file mtf.gnr1
49 c
c start file mtf.graf

c

c

c

c

c

 *********************************************************************
c

c

 start common block xndgraf

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 ttimul( 4 )
57 c title scale
58 c title scale: specifies the scale for title. A negative values cause
59 c the corresponding title to be underlined.
60 c
61 c CHARACTER*8 curves( 20 )
62 c curves to plot
63 c curves to plot. Up to 20 curves may be defined. Up to 7 characters
64 c can be used for curves name. The last character is reserved for dollar
65 c sign.
66 c
67 c INTEGER pframe
68 c frame flag
69 c Plot frame flag: specifies whether a frame is to be drawn around
70 c plot.
71 c > 0 Draw frame
72 c else No frame
73 c
74 c INTEGER xygrid( 3 )
75 c xy grid flags
76 c xygrid flag: specifies the grid options. The first two entries
77 c specify the grid frequency k for the x and y axes respectively.
78 c K > 0 k grid line per tic mark
79 c K = 0 No grid lines
80 c K < 0 k tic marks per grid line
81 c The third entry specifies the line type. Options are same as for
82 c curve line types.
83 c
84 c CHARACTER*4 xyaxs( 2 )
85 c xy axes option
86 c specifies the character version of axes options. The options are:
87 c S for linear display scale, L for log, I for absolute scale,
88 c P for Polar. Only the first two characters matter, e.g.,'IL', is
89 c linear in X and Log in Y.
90 c
91 c CHARACTER*4 t11wd
92 c title width option
93 c
94 c CHARACTER*4 l11wd
95 c label width option
96 c
97 c CHARACTER*4 x11ln( 3 )
98 c line options
99 c
100 c REAL xypage( 2 )
101 c page size
102 c the page size is the area which is taken up by the entire graphic
103 c display --both label and plot. It is specified in nominla inches,
104 c the first entry corresponding to the horizontal (x) dimension and
105 c the second to the vertical (y) dimension. The term nominal is used
106 c since the actual size depends upon the particular graphic device.
107 c the actual size may be smaller for hp. A typical page size is
c 11 by 8.5

REAL porgin( 2 )
c physical origin location
REAL xyarea( 2 )
c subplot area
c this is the area encompassed by the x and y axes. thus the value
specifies the length of the x axis and the second value specifies
the length of the y axis. it is specified in nominal inches. the
default value for subplot area is 6 by 6. a typical subplot area
is 8.5 by 6.0 for hp screen.

REAL frmtlk

REAL gmmrg

c margin
c specifies a distance in nominal inches from the plot
c area border within which points will be plotted. Positive values
allow for points being plotted outside the plot area. Negative values
c ensure that point will lies within the plot area.

REAL xhite

c axis label scale : specifies for axis labels in multiple of .14

c
REAL xyznh( 3 )
c axes scale per inch

REAL xyzsp( 3 )
c axes units per tic

c specifies absolute scale of axes in axis units per inch.

c and only if option = 0 is used(no scaling is done by program)

REAL xmmx( 2 )
c x axis min/max
c this parameter is specified if and only if option = 0 is
used. this implies no scaling is done by program.

REAL ymmx( 2 )
c y axis min/max
c see xmmx for detail

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

c *******************************************
c start common block xxgrav

c *******************************************

c DOUBLE PRECISION rgrav
c grav radius
c geopotential model earth radius
c

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 end block xxgrav
c end file mtf.grav
c start file mtf.init

DOUBLE PRECISION trvint( 7,2 )
init t,r,v
initial time and state. time is seconds since base time and
occurs in entry 1,i. state is m50 cartesian and occurs as
position, velocity in terms 2-7,i. here i refers to the vehicle

COMMON /xxinit/ trvint
end block xxinit
end file mtf.init
c  start file mtf.kal

c  

c  ****************************************

  start common block xxkal

  ****************************************

  DOUBLE PRECISION dtmax
  max step
  maximum allowed time step for kalman filter

  DOUBLE PRECISION edcrit(25)
  edit criterion
  if abs(resid) > sigma*edcrit for a given observation, then that
  observation is edited

  DOUBLE PRECISION var(20)
  observation variances
  These are the default observation sigmas reset to variances on
  initialization process.

  INTEGER lrmod1
  noise model option
  This option implements the lrbet3 range, range rate, and
  range bias noise models :=0, do not use; >0, use

  DOUBLE PRECISION ncons(10)
  lrbet3 noise model constants
  These are the noise model constants: (1-4), range
  ;(5-8), range bias; (9,10), range rate

  DOUBLE PRECISION undrwt(2)
  underweighting options
  undrwt(1) >0, implements the underweighting of the update
  by adjusting the computed residual variance by a factor of (1-undrwt).

  Note: undrwt(1) =0, no underweighting
  undrwt(2) , sets the criterion for underweighting . i.e., whenever
  RSS(relative position sigmas)**2 >undrwt(2)**2.

  INTEGER sxclvd(24)
  state exclusion flags
  if the sxclvd flag corresponding to an element in the filter state
  vector is 1, then that element is excluded from consideration in the
  filter processing

  COMMON /xxkal/ dtmax, edcrit, var, lrmod1, ncons, undrwt, sxclvd

  end block xxkal

  end file mtf.kal
c start file mtf.mas

c

c

***********************************************************************
c start common block xxmas

***********************************************************************
c

       INTEGER imstab

    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/ imstab, mastab

    c end block xxmas

    c

    c

    c end file mtf.mas

    c
c start file mtf.mast

INTEGER xbugs
  unit number of debug file

INTEGER hifile
  (unused) max unit allowed for internal files
  default=10

INTEGER hiunit
  (unused) highest unit
  the highest unit number used for assigned files

INTEGER maxfil
  maximum number of files
  the maximum number of files to be written to be used for dimension
  c and looping purposes

INTEGER maxtyp
  maximum number of m/sid's
  the maximum number of m/sid's to be processed on any file to be
  used for dimension and looping purposes

INTEGER msnum
  number of m/sid's to be processed

INTEGER xin
  input file

INTEGER xout
  output print file

INTEGER xbug( 20 )
  debug print flags

INTEGER tcorfl
  time current file

DOUBLE PRECISION timbeg
  begin time
  begin time for data retrieval for files under construction
DOUBLE PRECISION timend
end time for data retrieval
c
DOUBLE PRECISION timoff
time off
delta difference to check time-tags against the record clock times
c
COMMON /xxmast/ xbugs, hifile, hiunit, maxfil, maxtyp, msnum, xin
*, xout, xpbug, tcrflt, timbeg, timend, timoff
c
end block xxmast
c
c
end file mtf.mast
c
c start file mtf.max

* *

**start common block xxmax**

**integer maxveh**

**max number of vehicles**

**integer nveh**

**number of vehicles**

**common /xxmax/ maxveh, nveh**

**end block xxmax**

**end file mtf.max**

**c**
c start file mtf.misc

c

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

c
c start file mtf.moon

c

c

c

************************************************************
c start common block xxmoon

************************************************************

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 end file mtf.moon

c
c start file mtf.msid
c

************************************************************************
c start common block xmsid
c************************************************************************

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

COMMON /xmsid/, adrtbl, lmsbuf, frmloc, fmtim, xdatid, ddatum.
c
           start file mtf.name

**
           start common block xxname

**

            CHARACTER*56 prnam

            processor name

            INTEGER lprnam

            length of processor name

            INTEGER prbase

            processor baseline

            INTEGER prver

            processor version

            CHARACTER*12 prgver

            gff name/version

            COMMON /xxname/ prnam, lprnam, prbase, prver, prgver

            end block xxname

            end file mtf.name

            c
c start file mtf.nflz

********************************************
c start common block xxnflz
********************************************

INTEGER funit ( 5 )
c file units
c specifies unit number for files

CHARACTER*72 fname( 5 )
c file names
c specifies names of files used, uses are as follows
c for all

1  input first ephemeris or relative trajectory file
2  input second ephemeris or relative trajectory file
**note that posx determine which vehicle corresponds to which file
for xqdisp
3  input attitude file
4  output plot file
5  not used

for xcmpar
3  output ephemeris for vehicle 1
4  output ephemeris for vehicle 2
5  output relative trajectory of veh 2 with veh 1 base

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
time in the times input specifies a minimum number of records
c between consecutive output times.
c the following options are valid

1  first ephemeris (xqdisp,xcmpar)
2  second ephemeris (xqdisp,xcmpar)
3  attitude file (xqdisp)
else  error
INTEGER bfric
58  c  base file record
59  c
60  c
61  INTEGER curfl
62  c  current file index
63  c
64  c
65  INTEGER basefi
66  c  base file index
67  c
68  c
69  INTEGER fstat(5)
70  c  file status array
71  c  positive values indicate file is okay and open, negative open
72  c  and error, 0 file not open
73  c
74  INTEGER flx(5)
75  c  file data length
76  c  specifies the number of dp data words needed from the file
77  c
78  INTEGER posx(2)
79  c  location for data
80  c  absolute value give input file id, either file 1 or 2.
81  c  the sign specifies whether the state (+) or the relative
82  c  state (-)
83  c
84  INTEGER lrdx(2)
85  c  interpolation flag
86  c  specifies interpolation option for file i. options are
87  c
88  c  -1  from base file (set by program, not user input)
89  c
90  c  1   use two point position and velocity interpolation
91  c
92  c  2-10 use lagrangian interpolation with the specified number of p
93  c  oints
94  c
95  c
96  INTEGER xflg(2)
97  c  file used for state i
98  c  note that this is set to the file need by the vehicle in ndnfl and
99  c
100 c  used in xgwget as the file index for the vehicle state. furthermore
101 c  note that ndnfl uses xqneed to determine whether or not the vehicle
102 c  is considered. this is set in xqkmptf
103 c
104 INTEGER xqneed(3)
105 c  state/att flags
106 c  positive value indicates respectively vehicle 1,2, or attitude
107 c  is needed. set by xqkmptf and used by ndfnpt
108 c
109 INTEGER print
110 c  print flag
111 c  positive value to generate print to unit out
112 c
113 INTEGER plot
113 c plot flag
114 c positive value to save plot file to file 4
115 c
116    INTEGER trjout( 3 )
117 c traj flags
118 c positive value to generate the specified file, note
119 c that this is used by xcmpar and ignored by xqdsp
120 c
121 c  1    generate ephemeris for vehicle 1
122 c  2    generate ephemeris for vehicle 2
123 c  3    generate relative trajectory, base =veh1, rel = vehicle 2
124 c
125 c
126 c
127 c
128    COMMON /xrnflz/ funit, fname, bfopf, bfrec, curl, base!, f1stat
129    * , flx, posx, lordx, xfig, xqneed, print, plot, trjout
130 c
131 c end block xrnflz
132 c
133 c
134 c
135 c end file mtf.nflz
136 c
c start file mtf.ntrp

* *

* *

* *

* *

** start common block xxntrp **

** start common block xxntrp **

INTEGER rbuf( 12,2 )
record buffer

DOUBLE PRECISION xbuf( 120,2 )
traj intrp value table

DOUBLE PRECISION xvalz( 12,2 )
traj valu table

DOUBLE PRECISION fbuf( 120,2 )
intrp factor table

DOUBLE PRECISION tbuf( 10,2 )
time table

COMMON /xxntrp/ rbuf, xbuf, xvalz, fbuf, tbuf

end block xxntrp

end file mtf.ntrp
c start file mtf.obs

c

---

c COMMON /xxobs/ pcoas, pdgrv, pradar, ptrack, mibov, qibov, xi

---

end file mtf.obs

---

end block xxobs

---

end common block xxobs

---

start common block xxobs

---

c ************************************************************

---

c INTEGER pcoas

---

coas initialization

---

INTEGER pdgrv

---

partials flag

---

INTEGER pradar

---

radar initialization

---

INTEGER ptrack

---

star tracker initialization

---

DOUBLE PRECISION mibov( 3,3 )

---

attitude matrix for observor

---

DOUBLE PRECISION qibov( 4 )

---

observor attitude quaternion

---

DOUBLE PRECISION xi( 6,2 )

---

vehicle states

---

COMMON /xxobs/ pcoas, pdgrv, pradar, ptrack, mibov, qibov, xi
c start file mtf.pkt
c

----------------------------------------------------------------------------------------------------------------------------------
c start common block xxpkt
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
  file 1  input ephemeris 1 (traj or rel traj)
c
  file 2  input ephemeris 2 (traj or rel traj)
c
  file 3  attitude 1
c
  file 4  attitude 2
c
  file 5  sensed velocity 1
c
  file 6  sensed velocity 2
c
  file 7  data file 1
c
  others  as needed by particular program
c
  note that not all the files need be opened or used in a
c particular program
c
  COMMON /xxpkt/ packet
c
  end block xxpkt
c
  end file mtf.pkt
c
c start file mtf.pmmx

********************************************
c start common block xxpmmx
********************************************
     REAL pmmx( 2,21 )
c min/max values
13 c min/max values for y axis
c
15 COMMON /xxpmmx/ pmmx
16 c
cend block xxpmmx
18 c
19 c
20 c
c end file mtf.pmmx
22 c
c start file mtf.prnt

c ****************************************
c start common block xprnt

c ****************************************
c     INTEGER lpage
12 c lines/page
13 c specifies number of lines per page. If negative or
14 c zero, then no paging is provided.
15 c     INTEGER npage
16 c page number
17 c
18 c     INTEGER lnz
19 c line count
20 c
21 c     INTEGER dinz
22 c line count increment
23 c
24 c     INTEGER co180
25 c column width option
26 c positive value sets formats for 80 columns.
27 c otherwise, 130 columns.
28 c     INTEGER header
29 c header print option
30 c positive value causes a header of time and scale
31 c information to be printed for each output time
32 c     INTEGER nformat
33 c format id
34 c specifies output format. options are
35 c 1 floating point
36 c else fixed point
37 c
38 c COMMON /xprnt/ lpage, npage, lnz, dinz, co180, header, nformat
39 c
40 c end block xprnt
41 c
42 c end file mtf.prnt
INTEGER pvvent (2)

vent force

double precision dt, nom

specifies largest step size in integration step

COMMON /xxprop/ rvopt, pprpop, paero, pcb, pdrag, pharm, pmoon,

* prad, psvel, psun, pvvent, dt, nom

end block xxprop

end file mtf.prop

end
c start file mtf.qcrv

c

c

INTEGER ncrvs

c number of curves to plot

c

CHARACTER*8 kname(20)
c curve names

c user defined curve names. Up to 7 characters can be used for curve

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

INTEGER lmrk(20)
c line option of curves

c symbol frequency: Specify the frequency of plot symbols for the

c corresponding curve. For a value k:

  k > 0 symbol every kth point. Line through each point.

  k = 0 no symbols. Line through each point.

  k < 0 symbol every kth point. No line.

INTEGER nlabel(20)
c point label options

c label frequency: Specifies the frequency of integer labels for

c curve points.

  N < 0 label every Nth point in the sequence 1,2,3,...

  N > 0 label every Nth point in the sequence 1,N,2N,...

  N = 0 no labels

INTEGER ksymbol(20)
c curve plot symbol code

c Specifies the plotting symbols for corresponding curve.

For more detail on the different symbol codes, see DISSPLA manual

c for symbols and their corresponding sequence numbers.

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.

INTEGER kedit(20)
c edit plot options

c

INTEGER kline(20)
c line type code
58 c Specifies the line type option for curve. Option are:
59 c 0 connected
60 c 1 Dot
61 c 2 Dash
62 c 3 Chained dot
63 c 4 Chained dot dash
64 c
65 REAL kstep(20)
66 c step size
67 c curve interval: specifies the plot interval for curve i.e., begin, end, and step size. Times are specified as seconds from base time.
68 c If kstep > 0 then use in time seconds, else use counts
70 c
71 REAL kspan(2,20)
72 c begin,end
73 c begin and end time span (time or count depend)
74 c
75 REAL psize(20)
76 c plot symbol size
77 c specifies the the size of the plot symbols.
78 c
79 INTEGER crvz(20)
80 c plot curves ids
81 c indices of curves considered in current plot
82 c
83 INTEGER idxyz(3,20)
84 c parameter ids
85 c indices of parameters corresponding to each curve
86 c these reflect the labels kparams
87 c
88 COMMON /xxqcrv/ ncrvs, kname, kparms, imrk, niabel, ksmbol, kfile
89 *, kedit, kline, kstep, kspan, psize, crvz, idxyz
90 c
91 c end block xxqcrv
92 c
93 c
94 c
95 c end file mtf.qcrv
96 c
```c
1  c
2  c start file mtf.qcur
3  c
4  c
5  c
6  c **********************************************
7  c start common block xxqcur
8  c **********************************************
9  c
10  c
11  c  INTEGER cf1
12  c  current file id
13  c
14  c
15  c  CHARACTER*4 kfid(3)
16  c  parm frame ids
17  c
18  c  INTEGER kidat(3)
19  c  parm data indices
20  c
21  c  REAL kqmmx(2,3)
22  c  parm min/max
23  c
24  c  INTEGER kqpmx(3)
25  c  parm min/max flag
26  c
27  c  REAL kqoset(3)
28  c  parm offsets
29  c
30  c  REAL kqscal(3)
31  c  parm scale factors
32  c
33  c  CHARACTER*12 kqunit(3)
34  c  parm unit names
35  c
36  c  INTEGER kqdeln
37  c  count delta
38  c
39  c  INTEGER kqnbeg
40  c  start count
41  c
42  c  INTEGER kqend
43  c  end count
44  c
45  c  INTEGER kqnt
46  c  next count
47  c
48  c
49  c
50  c
51  c
52  c
53  c
54  c
55  c
56  c
```
INTEGER kqtoth
    c total count
INTEGER kqcur
    c current count
INTEGER kqtoti
    c total inclusions
INTEGER kqcuri
    c current inclusions
DOUBLE PRECISION kqdelta
    c time delta
DOUBLE PRECISION kqtime
    c start time
DOUBLE PRECISION kqtime
    c end time
DOUBLE PRECISION kqnext
    c next time
INTEGER kqrec
    c current record
INTEGER frmafl
    c frame id flag
    c positive value indicates that frame ids are not checked
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
INTEGER kqloq(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
COMMON /xxqcur/ cfl, kfid, kidat, kqmmx, kqpmx, kqoset, kqscal,
* kqunit, kqdeln, kqmbeg, kqenend, kqnxtn, kqtotn, kqcurm, kqtotl,
* kqcuri, kqdelr, kqmbeg, kqenend, kqnxtt, kqrec, ftmall, kqedit,
* kqlop

end block xxqcur

c

c end file mtf.qcur

c
c  start file mtf.qgen
3  c
4  c
5  c
6  c
7  c******************************************************************************
8  c start common block xxqgen
9  c******************************************************************************
10 c
11 c    CHARACTER*4 qpdev
12 c    device for plot
13 c
14 c
15 c    INTEGER qplevel
16 c    level of plot
17 c
18 c
19 c    INTEGER qdvup
20 c    device initialization
21 c
22 c
23 c    INTEGER qperu
24 c    error unit initialization
25 c
26 c
27 c    INTEGER popunt
28 c    dispose unit
29 c
30 c
31 c    INTEGER ndim
32 c    dimension of current curve
33 c    the default ndim is 2.
34 c
35 c    REAL shtie
36 c    standard symbol height
37 c
38 c
39 c    COMMON /xxqgen/ qpdev, qplevel, qdvup, qperu, popunt, ndim, sthtie
40 * sthtie
41 c
42 c  end block xxqgen
43 c
44 c
45 c
46 c  end file mtf.qgen
47 c
c start file mtf.qlbf

c

c******************************************************************************
c start common block xxqlbf
c******************************************************************************

INTEGER ipkleg( 210 )

c legend buffer

c
COMMON /xxqlbf/ ipkleg

c end block xxqlbf

c

c end file mtf.qlbf

c
c start file mtf.gprm

c

c

c

***************
c start common block xxgprm

c

***************
c

INTEGER pmxfig(21)
c

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 minimum, plot points below minimum at min values
c -3 omit points above maximum, plot points below maximum at max values
c

INTEGER pword(21)
c

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 pfld(21)
c

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 parameter.
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

c units names
c specifies the units names for parameters
c

CHARACTER*8 pname(21)
c

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

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
58 c value that is subtracted from parameter before display.
59 c specified in the units determined by pscale. note that min/max
60 c are checked after the offset is removed and thus should be relative
61 c to the offset
62 c COMMON /xxqprm/ pmxflg, pword, pfid, punits, pname, pscale, pofset
64 c
65 c end block xxqprm
66 c
67 c
68 c
69 c end file mtf.qprm
70 c
c start file mtf.rnpx

DOUBLE PRECISION trnp0

DOUBLE PRECISION rnpO( 3,3 )

c rnp matrix

c true of date to m50 transformation matrix at base time.
c computed by program if abs(rnpO(1,1))>1.2
c else input used

DOUBLE PRECISION cdetut

c et offset

c specifies offset between ephemeris time and universal time

c cdetut(1)= et - ut in seconds

c cdetut(2)= rate of et - ut increase in seconds/second

DOUBLE PRECISION alphah

c base hour angle

c base hour angle of earth. computed upon initialization

COMMON /xxrnpx/ trnp0, rnp0, cdetut, alphah

end block xxrnpx

end file mtf.rnpx
C
C start file mtf.rpst
C
C******************************************************************************
C start common block xrxpst
C******************************************************************************
C
       CHARACTER*72 infil
C input obs file name
C
       CHARACTER*72 outf il
C output obs file name
C
       DOUBLE PRECISION antang
C antenna angle
C antenna angle for radar entry in radians
C
       COMMON /xrxpst/ infil, outf il, antang
C
C end block xrxpst
C
C end file mtf.rpst
C
INTEGER opt

Input options on state covariances U, V, W

opt = 1 indicates that the input state covariances are referenced to the base state covariances in the U, V, W coordinate frame.

opt = 0 indicates that the input state covariances are referenced to the target's initial state (U, V, W) covariances (similar to user inputs).

DOUBLE PRECISION rvccov(2, 2)

Position, velocity covariance for the base state.

rvccov(1, 1) is the position, velocity covariance for the initial position, velocity of the base state.

rvccov(2, 2) is the position, velocity covariance for the relative state.

DOUBLE PRECISION sncon(2)

Constants in the state noise computation.

If you have a downrange in the base state (sncon = 1), the downrange corresponds to an uncertainty of 20 n.m. downrange in the relative state per orbit. To increase the downrange error by a factor of n, multiply the corresponding sncon by n**2.
c start file mtf.sen

C******************************************************************************
c start common block xxsen
C******************************************************************************
c INTEGER maxobs
12 c maximum number of obs
13 c
14 c INTEGER maxsen
15 c maximum number of sensors
16 c c
17 c c INTEGER icoas
18 c coas id
19 c
20 c INTEGER iradar
21 c rendezvous radar id
22 c
23 c INTEGER itrky
24 c y star tracker id
25 c
26 c INTEGER itrkz
27 c z star tracker id
28 c
29 c INTEGER isen1
30 c 1st extra sensor id
31 c
32 c INTEGER isen2
33 c 2nd extra sensor id
34 c
35 c INTEGER isen3
36 c 3rd extra sensor id
37 c
38 c DOUBLE PRECISION rsob( 3,7 )
39 c sensor offsets
40 c
41 c DOUBLE PRECISION qbs( 4,7 )
42 c sensor attitude quaternions m50 to sensor
43 c
44 c COMMON /xxsen/ maxobs, maxsen, icoas, iradar, itrky, itrkz, isen1
45 c * , isen2, isen3, rsob, qbs
c start file mtf.sprm

**start common block xxxprnm**

```c
  CHARACTER*40 gmsg
  generic message
  generic message in dpf header
  CHARACTER*8 seq
  file sequence id
  eight character identifier sequence unique to each product file.
  contents are 'fnqsfrv' where
  fn flight number
  sq starting sequence number for first file on tape
  sf sequence flag where
  00=sq only one sequence on file
  01=sq start of first sequence
  nn=sq start of sequence nn
  rv revision number

  INTEGER spg( 20 )
  display group flags
  positive value specifies parameter groups to display
  options are
  1 orbiter gmt
  2 mcc gmt
  3 ground elapsed time
  4 orbiter m50 state
  5 orbiter in target lvlh
  6 orbiter in target uvw
  7 orbiter euler angles to uvw
  8 orbiter m50 attitude matrix
  9 orbiter m50 quaternion
  10 orbiter attitude rate
  11 look angle and rates to target
  12 range and range rate
  13 simulation flag (used for data drop out info instead)
  1 = data in tracking intervals (default)
  0 = data not in tracking intervals
  14 target m50 state
  15 target m50 relative state
  16 target in orbiter uvw
  17 target in orbiter lvlh

  INTEGER szid
  header length
  index of record size in header
  INTEGER hlng
  length of header
```
57  c  number of integer words in header
58  c
59  c       REAL dict( 3 )
60  c  dictionary record
61  c  dictionary record format
62  c
63  c       REAL datbuf( 2 )
64  c  data buffer
65  c  data buffer used for creating Univac tape
66  c
67  c       COMMON /xxsprm/ gmsg, seq, spg, szid, hing, dict, datbuf
68  c
69  c  end block xxsprm
70  c
71  c
72  c
73  c  end file mtf.sprm
74  c
c start file mtf.sptom

c

c

********************************************
c start common block xxsptom

c********************************************

DOUBLE PRECISION pfreq

print frequency

c nominal print frequency

c

DOUBLE PRECISION sptol

tolerance

c tolerance within which special time is printed

c

DOUBLE PRECISION sptime( 20 )

special times

c table of special print times as seconds since base time

c

COMMON /xxsptom/ pfreq, sptol, sptime

c

c end block xxssptom

c

c end file mtf.sptom

c
c start file mtf.sun

c

c

c

******************************************************************************
c start common block xxsun

c******************************************************************************

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

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

*****
c start common block xxtimc

INTEGER cmptim( 200 )
indicates whether the time associated with the msid corresponding
with this array position is to be computed from the dow

INTEGER skewid( 200 )
indicates the data record location of the proper skew for this msid

REAL timdat( 200,2 )
time data from cct

REAL rbfdat( 16 )
time data from cct

INTEGER cyenum
the indicator of header record present on cct header records

COMMON /xxtimc/ cmptim, skewid, timdat, rbfdat, cyenum

c end block xxtimc

c end file mtf.timc
c start file mtf.time

c

c ***********************************************

c start common block xctime

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
c DOUBLE PRECISION tend
c end time
c end time as seconds since base time
c
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
c DOUBLE PRECISION tbase
c base time
c
COMMON /xctime/ date, dates, tbegin, tend, delta, endopt, endval, tbase
c
end block xctime
c
c end file mtf.time
c
c start file mtf.toff

c ******************************************************
c start common block xtoff

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
DOUBLE PRECISION bjdg
c ground base julian date
c
DOUBLE PRECISION dtsg
c s/g time factor
c
DOUBLE PRECISION tlift
c time of lift off
INTEGER drive
tape drive
INTEGER tape
tape option
if tape > 0 univac tape is produced
INTEGER hpb in
if hpb in > 0 then hpb binary file is produced
DOUBLE PRECISION dt a ble ( 50, 2 )
data dropout table
data dropout time intervals table
COMMON / x x toff / 1 date, d t d a te, dtbias, qa freq, t l a s e, bjds, bjd g
* , d t s g, t l i ft, drive, tape, hpb in, d t a b l e
end block x x toff
end file mtf.toff

c start file mtf.tols

**start common block xxtols**

c *******************************************************

c DOUBLE PRECISION cto1

c convergence tolerance
c

c DOUBLE PRECISION cto11

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/ cto1, cto11, invtol, tol
c

c end block xxtols
c

c end file mtf.tols
c
*start file mtf.usys

**start common block xxusys**

**INTEGER usysex( 5 )**

**INTEGER nsysex**

**INTEGER usysin**

**CHARACTER*4 usyst( 5 )**

**CHARACTER*4 nsclz( 10,5 )**

**DOUBLE PRECISION usclz( 10,5 )**

**COMMON /xxusys/ usysex, nsysex, usysin, usyst, nsclz, usclz**

**end block xxusys**

**end file mtf.usys**
c start file mtf.vcx

c double precision aeroc(2)
c drag numbers

c double precision cd(2)
c drag coefficient

c double precision chord(2)
c vehicle chord

c double precision c(3.2)
c cylindrical fit parameters

c double precision d(5.2)
c zutek fit parameters

c double precision kd(2)
c drag multiplier

c double precision drgfac(2)
c aerodynamic drag factor

c double precision rbarna(3.2)
c actual cg pos rel to nominal cg

c double precision sref(2)
c vehicle reference area

c double precision sarea(2)
c area for solar rad

c double precision srflec(2)
c vehicle reflectivity

c double precision solfac(2)
c solar radiation factor
INTEGER horder(2)
max order of harmonics

INTEGER hdgree(2)
maximum degree of harmonics

DOUBLE PRECISION sagate(2)
threshold for sensed accelerations to be included in
propagation. if magnitude of the acceleration does not exceed
this threshold, then the acceleration is omitted.

COMMON /xxvcx/ aeroc, cd, chord, c, d, kd, drgfac, rbarra, sref,
*sarea, srflac, solfac, horder, hdgree, sagate

end block xxvcx

c

c

c
c

c
c start file mtf.vnt

DOUBLE PRECISION vnttab(4, 10, 2)

vent timeline
two timeline entries based on last index, max of 10 entries per timeline. Each entry consists of the following
1 start time of vent
2 body x force
3 body y force
4 body z force

COMMON /xxvnt/ vnttab

c end block xxvnt

c end file mtf.vnt
c start file mtf.writ

**start common block xxwrit**

```
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 end block xxwrit
```

```
c end file mtf.writ
```
c
start file mtf.atm
c
c
******************************************************************************
c start common data block xxatm
c******************************************************************************
data a62sc / .2590478000d-03 /
data a62sh / .1206650000d+06 /
data a62rr / .1294000000d-07 /
data a62rh / .1225053493d+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 abmcl1g / .7986355100d+00 /
data abmslg / .6018150230d+00 /
data abmgd / .1375000000d+01 /
data abmalt / .2222395555d+06 /
data abm1 / -.2515818000d+02, -.116683668d-04, .9922058998d+04, * 232139100d+02, -.1625013451d-04, .7868452230d+06 / 
data abm0 / -.1986449600d+00, .4152999344d-05, -.3926329360d+05/ 
data abmsca / .8999972860d+05 /
data abmcb1 / .4604988777d-04 /
data abmcb2 / -.4500000000d-04 /
data abmcm2 / -.9886459900d+00 /
data abmrf / .1225004798d+01 /
c
cend block xxatm
c
cend file mtf.atm
c
c start file mtf.bias

c *******************************************************
 c start common data block xxbias
 c *******************************************************

    data sgbias / 50.d0,.01745d0,.01745d0,.1d0,.0002d0,.0002d0,
                      * .0002d0,.0002d0,.00074d0,.00074d0 /
    data timcon / 400.d0,10000.d0,10000.d0,400.d0, 10000.d0,10000.d0,
                      * 10000.d0,10000.d0, 400.d0,400.d0 /

c end block xxbias

c end file mtf.bias

c
c start file mtf.con

c

c

c

c

c

c

c

c

8 c start common data block xxcon

9 c

data clight / 0.2997925d9 /
data pi / .314159265358979324d1 /
c

c

c

c

c

c end block xxcon

18 c

19 c end file mtf.con

20 c
c start file mtf.erth

c

c

c

c
***********
c
start common data block xxerth
c
***********

data mxerth / .3986012d15 /
data req / .6378166d7 /
data rpol / .6356784283607107d7 /
data we1 / .729211514645921d-4 /

c

c end block xxerth
c
c
c
c end file mtf.erth
c
c start file mtf.file

# start common data block xxfile

# end block xxfile

# end file mtf.file
c start file mtf.gnr1

c *****************************************************
c start common data block xxgnrl

c *****************************************************
data bugs / 6 /
data in / 69 /
data term / 6 /
data out / 6 /
data pbug / 19*0 /
data pfatal / 0 /
data interm / 5 /
c end block xxgnrl
c end file mtf.gnr1
c
Dec 16 14:08 1986 ddgraf Page 1

1 c
2 c start file mtf.graf
3 c
4 c
5 c
6 c
7 c ********************************************
8 c start common data block xxgraf
9 c ********************************************
10 c
11 data option / 1 /
12 data ttimul / 2.0 . 1.5 . 0.0 . 0.0 /
13 data xygrid / 2*1.0 /
14 data xypage / 11.0 . 8.5 /
15 data porigin / 2*1.0 /
16 data xyarea / 8.5 . 6.0 /
17 data xyhite / 1.2 /
18 data xyang / 45.0 . 2*0.0 /
19 data plednt / 1.0 /
20 data lgnpos / 8.47 . 3.5 /
21 c
22 c end file mtf.graf
23 c
c start file mtf.grav
c
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 cterms / 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 end file mtf.grav
c
c start file mtf.kal

***********
c start common data block xkcal
***********
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 end block xkcal
c
c
c end file mtf.kal
c
c start file mtf.mas

start common data block xxmas

end block xxmas

c end file mtf.mas

c
c start file mtf.max

c

c

c

c

c

c

start common data block xxmax

c

data maxveh / 2 /
data nveh / 1 /
c
c

c end block xxmax
c
c
c

c end file mtf.max
c
c start file mtf.misc

c

c

*************
c start common data block xxmisc

*************

data jobdes / 2*"no job description given" /
data obsrvr / 1 /
data target / 2 /

c

c

c

c end block xxmisc


c

c

c

c end file mtf.misc


c
c start file mtf.nflz

c ****************************************
c start common data block xxnflz

c ****************************************
data funit / 21, 22, 23, 24, 25 /
c
c end block xxnflz
c
c end file mtf.nflz
c
c
1
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 start file mtf.qcrv

c

c

**************
c start common data block xxqcrv

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 /
data psise / 20*1. /

c

c end block xxqcrv

c

c end file mtf.qcrv

c
c start file mtf.qgen

c common data block xxqgen

data qpdev / 'hp' /
data sthtie / 0.14 /
end block xxqgen
end file mtf.qgen
Dec 16 14:08 1986 ddrnpx Page 1

2 0 C 0 0 0 0 0 0 start file mtf.rnpx

3 8 C 0 0 0 0 0 0 start common data block xnxnpx

4 11 data rmp0 / 1.20.0.8.0 8.0 0 0 0 0

5 12 data cdetut / 0.0 0 0 0 0

6 14 C 0 0 0 0 0 0 end block xnxnpx

7 16 C 0 0 0 0 0 0 end file mtf.rnpx
c start file mtf.rpst

c

c

c

c

******************************************************************************
c start common data block xxr pst

c******************************************************************************

data antang / 1.169370564 /

c

c end block xxr pst

c

c

c end file mtf.rpst
c start file mtf.scov

c ********************************************
c start common data block xxscov

c ********************************************

data lopt / 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
* .8*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
* .08d0, .6*0.d0, .08d0

data sncon / 1.603d-6, .6412d-6 /

c end block xxscov
c c end file mtf.scov

c
** start file mtf.sen **

** start common data block xxsen **

```c

data maxobs / 16 /
data maxsen / 4 /
data icoes / 4 /
data t3radar / 1 /
data irtky / 3 /
data irtkz / 2 /
data rsob / .139445122d2, .33933d1, -.176524d1,  
*.173104302d2, -.28194d0, -.124206d1, -.1649d2, -.525d1, .29d1,  
*.1386078d2, -.1408d1, -.322326d1, 3+0.0d0 /
data qbs / .833885825201d0, .0d0, .0d0, .5519369829422d0,  
*.0264059344345d0, .705011118937d0, .7072182848987d0,  
*.0010958282010d0, .0640725060379d0, -.0653106301822d0,  
*.7050333478320d0, -.7032476190239d0, .0d0, .0d0, 1d1, .0d0,  
* 1.0d0, 3+0.0d0 /
```

** end block xxsen **

** end file mtf.sen **
c start file mtf.sptm

***
c start common data block xxsptm
***
c
   data pfreq / 4.0d0 /
   data sptime / 20+1.0d38 /

c
   c end block xxsptm
   c
   c
   c end file mtf.sptm
   c
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 start file mtf.time

**-common data block xxtime**

data date / 1985, 1, 0, 0, 0 /
data date / 0, 0 /
data tbegin / -1.d30 /
data tend / 1.d30 /
data delta / 1 /
data endopt / 0 /
data endval / 1.d30 /

c end block xxtime
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 start file mtf.usys

c

c

c

c

***************************************************************
c start common data block xxusys

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

c

**************
c start common data block xxvcx

**************

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 end block xxvcx

c
c
c
c end file mtf.vcx

c
c start file mtf.writ

*-------------------------------------------------------------
c start common data block xxwrit
*-------------------------------------------------------------
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 end block xxwrit
c end file mtf.writ
c start file mtf.bias

c

c start block xxbias

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 end block xxbias

c

c end file mtf.bias

c
c start file mtf.cct

c start block xxcct

call setin( cct, 'cct'/char(0), 'i+1'/char(0) )
call setin( lcynum, 'lcynum'/char(0), 'i+1'/char(0) )
call setin( o4000, 'o4000'/char(0), 'i+1'/char(0) )
c end block xxcct

c end file mtf.cct

c
c start file mtf.con

c

c start block xxcon

c

call setin( clight, 'clight'//char(0), 'd'//char(0) )
call setin( pi, 'p'//char(0), 'd'//char(0) )

c end block xxcon

c

c end file mtf.con

c
c start file mtf.data

C start block xxdata

call setin( dtxcll, 'dtxclid'/char(0), 'i+25'/char(0) )
call setin( gwbtab, 'gwbtab'/char(0), 'd+50'/char(0) )

c end block xxdata

c end file mtf.data

c
c

c start file mtf.dot
3 c
4 c
5 c start block xxdot
6 c
7 c
8 c
9 c end block xxdot
10 c
11 c
12 c
13 c end file mtf.dot
14 c
c start file mtf.dprm
c start block xxdprm
c
    call setin( spg, 'spg'/char(0), 'i+20'/char(0) )
c end block xxdprm
c
c end file mtf.dprm
c
c start file mtf.erth

c start block xxerth

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 end block xxerth

c

c end file mtf.erth
call setin( nama1, 'nama1'//char(0), 's72*1'//char(0) )
call setin( uzea1, 'uzea1'//char(0), 'i*1'//char(0) )
call setin( unt1a, 'unt1a'//char(0), 'i*1'//char(0) )
call setin( name2, 'name2'//char(0), 's72*1'//char(0) )
call setin( unt2a, 'unt2a'//char(0), 'i*1'//char(0) )
call setin( uzea2, 'uzea2'//char(0), 'i*1'//char(0) )
call setin( unt2e, 'unt2e'//char(0), 'i*1'//char(0) )
call setin( name1, 'name1'//char(0), 's72*1'//char(0) )
call setin( uze2, 'uze2'//char(0), 'i*1'//char(0) )
call setin( unte1, 'unte1'//char(0), 'i*1'//char(0) )
call setin( uze1, 'uze1'//char(0), 'i*1'//char(0) )
call setin( unte2, 'unte2'//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( namd1, 'namd1'//char(0), 's72*1'//char(0) )
call setin( unt1e, 'unt1e'//char(0), 'i*1'//char(0) )
call setin( uzed1, 'uzed1'//char(0), 'i*1'//char(0) )
call setin( untd1, 'untd1'//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 start file mtf.fils
c
start block xxfils
c
call setin( filsta, "filsta"/char(0), 'i*20'/char(0) )
call setin( frmsze, "frmsze"/char(0), 'i*60'/char(0) )
call setin( datseq, "datseq"/char(0), 'i*4000'/char(0) )
call setin( dbproc, "dbproc"/char(0), 'i*20'/char(0) )
c end block xxfils
c
c end file mtf.fils
c
c start file mtf.gnrl

start block xxgmr

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) )

end block xxgmr

c end file mtf.gnrl

c
c start file mtf.graf

! start block xxgraf

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( tti1mu1, 'tti1mu1'//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( xyaxes, 'xyaxes'//char(0), 's4*2'//char(0) )
call setin( ttlw, 'ttlw'//char(0), 's4*1'//char(0) )
call setin( lblw, 'lblw'//char(0), 's4*1'//char(0) )
call setin( xyzin, 'xyzin'//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( frthk, 'frthk'//char(0), 'r+1'//char(0) )
call setin( mrgin, 'mrgin'//char(0), 'r+1'//char(0) )
call setin( xyhite, 'xyhite'//char(0), 'r+1'//char(0) )
call setin( xyzchn, 'xyzchn'//char(0), 'r+3'//char(0) )
call setin( xyzstp, 'xyzstp'//char(0), 'r+3'//char(0) )
call setin( ymmx, 'ymmx'//char(0), 'r+2'//char(0) )
call setin( zmxx, 'zmxx'//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) )

! end block xxgraf

c end file mtf.graf

c
c start file mtf.grav
c
c start block xxgrav
c
call setin( rgrav, 'rgrav'/char(0), 'd+1'/char(0) )
call setin( cterms, 'cterm'/char(0), 'd+35'/char(0) )
call setin( stterms, 'stterm'/char(0), 'd+35'/char(0) )
call setin( jterms, 'jterm'/char(0), 'd+7'/char(0) )
c end block xxgrav
c
c end file mtf.grav
c
c start file mtf.init

c start block xxinit

c
    call setin( trvint, 'trvint'//char(0), 'd+14'//char(0) )

c end block xxinit

c end file mtf.init

c
c
start file mtf.kal

start block xxkal

call setin( dtmax, 'dtmax'/char(0), 'd*1'/char(0) )
call setin( edcrit, 'edcrit'/char(0), 'd*25'/char(0) )
call setin( var, 'var'/char(0), 'd*20'/char(0) )
call setin( lrmol, 'lrmol'/char(0), 'i*1'/char(0) )
call setin( ncons, 'ncons'/char(0), 'd*10'/char(0) )
call setin( undrwt, 'undrwt'/char(0), 'd*2'/char(0) )
call setin( sxcl, 'sxcl'/char(0), 'i*24'/char(0) )

end block xxkal

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

c start block xxmast

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( tcrcfl, 'tcrcfl'/char(0), 'i*1'/char(0) )
    call setin( timoff, 'timoff'/char(0), 'd*1'/char(0) )

c end block xxmast

c

c end file mtf.mast

c
c start file mtf.max

c

c start block xxmax

c
    call setin( nveh, 'nveh'//char(0), 'i*i'//char(0) )

c
end block xxmax

c

c
end file mtf.max

c
1  c  c  start  file  mff.  misc  
2  c  c  start  block  xxmisc  
3  c  c  call  setin  (  jobdes.  
4  c  c  call  setin  (  obsrv.  
5  c  c  call  setin  (  target.  
6  c  c  end  block  xxmisc  
7  c  c  end  file  mff.  misc  
8  c  c  
9  c  c  
10  c  c  
11  c  c  
12  c  c  
13  c  c  
14  c  c  
15  c  c  
16  c  c  
17  c  c  
c start file mtf.msid
c

C start block xxmsid
c

C end block xxmsid
c

C end file mtf.msid
c
c
start file mtf.nflz
c

start block xxnflz
c
    call setin( fname, 'fname'//char(0), 's72*5'//char(0) )
call setin( bfopt, 'bfopt'//char(0), 'i+1'//char(0) )
call setin( posx, 'posx'//char(0), 'i+2'//char(0) )
call setin( lordx, 'lordx'//char(0), 'i+2'//char(0) )
call setin( print, 'print'//char(0), 'i+1'//char(0) )
call setin( plot, 'plot'//char(0), 'i+1'//char(0) )
call setin( trjout, 'trjout'//char(0), 'i+3'//char(0) )
c
end block xxnflz
c
c
end file mtf.nflz
c
c start file mtf.obs

C start block xxobs

C end block xxobs

C end file mtf.obs

C
1 c
2 ! start file mtf.pkt
3 c
4 ! start block xxpkt
5 c
6 c
7 c
8 c
9 c! end block xxpkt
10 c
11 c
12 c
13 c! end file mtf.pkt
14 c
c start file mtf.prnt

 c start block xxprnt

 c

 call setin( lpage, 'lpage'/char(0), 'i*1'/char(0) )
call setin( col80, 'col80'/char(0), 'i*1'/char(0) )
call setin( header, 'header'/char(0), 'i*1'/char(0) )
call setin( nfrmat, 'nfrmat'/char(0), 'i*1'/char(0) )

 c end block xxprnt

 c

 c end file mtf.prnt

 c


c start file mtf.prop

c
c
c start block xxprop

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( dtynom, 'dtynom'//char(0), 'd*2'//char(0) )
c end block xxprop

c
c
c end file mtf.prop

c
c start file mtf.qcrv

c

c start block xxqcrv

c

call setin( kname, 'kname'/char(0), 's8+20'/char(0) )
call setin( kparams, 'kparams'/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( ksymbol, 'ksymbol'/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 end block xxqcrv

c
c end file mtf.qcrv

c
1 c
2 c start file mtf.q1bf
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.q1bf
14 c
c start file mtf.qprm

start block xxqprm

call setin( pmyflg, 'pmyflg'//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 end block xxqprm

c end file mtf.qprm

c
c start file mtf.rnpx

c

c start block xxrnpx

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 end file mtf.rnpx

c
c start file mtf.rpst

8     call setin( infil, 'infil'/char(0), 's72*1'/char(0) )
9     call setin( outf1, 'outfil'/char(0), 's72*1'/char(0) )
10    call setin( antang, 'antang'/char(0), 'd*1'/char(0) )

12 c end block xxrpst

16 c end file mtf.rpst
c start file mtf.scov
c
c start block xxscov
c
call setin( lopt, 'lopt'//char(0), 'i*1'//char(0))
call setin( rvco, 'rvco'//char(0), 'd+72'//char(0))
call setin( snco, 'snco'//char(0), 'd+2'//char(0))
c
c end block xxscov
c
c
c end file mtf.scov
c
c start file mtf.sen

c

c start block xxsen

    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 end block xxsen


c end file mtf.sen

c
c start file mtf.sprm

c

start block xxsprm

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 end block xxsprm

c

c end file mtf.sprm

c
c start file mtf.sptm
c
start block xxspc

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 xxspc
c

end file mtf.sptm
c
c start file mtf.sun


c


c start block xxsun


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), '1*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 end block xxsun


c


c end file mtf.sun


c
c start file mtf.svbi
c

c start block xxsvb1
c
call setin( psvb, 'psvb'/'char(0), 'i*2'/'char(0) )
call setin( svbtb, 'svbtb'/'char(0), 'd*160'/'char(0) )
c
c end block xxsvb1
c
c
c end file mtf.svbi
c
c start file mtf.time

c start block xxtimex

c
    call setin( date,'date'//char(0), '1+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 end block xxtimex

c
end file mtf.time
c start file mtf.toff
c
5 c start block xxtoff
6 c
7 c
8 call setin( lddate, 'ldate'//char(0), 'i'//char(0) )
9 call setin( dtdate, 'dtdate'//char(0), 'i'//char(0) )
10 call setin( dtbias, 'dtbias'//char(0), 'd'//char(0) )
11 call setin( qafreq, 'qafreq'//char(0), 'i'//char(0) )
12 call setin( tlapse, 'tlapse'//char(0), 'd'//char(0) )
13 call setin( drive, 'drive'//char(0), 'i'//char(0) )
14 call setin( tape, 'tape'//char(0), 'i'//char(0) )
15 call setin( hpbin, 'hpbin'//char(0), 'i'//char(0) )
16 call setin( dtable, 'dtable'//char(0), 'd100'//char(0) )
17 c
18 c end block xxtoff
19 c
20 c
21 c
22 c end file mtf.toff
23 c
c start file mtf.usys
c
start block xxusys
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 end block xxusys
c
c end file mtf.usys
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( srflct, 'srflct'//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 start file mtf.wrt

c start block xxwrit

c
    call setin( files, 'files'//char(0), 's12*10'//char(0) )
call setin( recipnt, 'recipnt'//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 end block xxwrit

c

c end file mtf.wrt

c
c hdrparm proc
    c id header index parameters
    c hbdate = start of base date y,m,d,h,m
        integer hbdate
        parameter (hbdate=67)
    c hbsac = seconds portion of base date
        integer hbsac
        parameter (hbsac=72)
    c hcdate = date of creation yy/mm/dd c*8
        integer hcdate
        parameter (hcdate=3)
    c hcvr = version of program used for creation
    c stored as c16 (4 integer words)
        integer hcvr
        parameter (hcvr=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 hnrrec = number of records with data in it
        integer hnrrec
        parameter (hnrrec=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 hdate = date of creation yy/mm/dd c*8
        integer hdate
        parameter (hdate=35)
    c hdes = description of last update
    c 100 characters
        integer hdes
        parameter (hdes=41)
    c hspec = file spec block (extra 11 words)
        integer hspec
        parameter (hspec=80)
    c huer = version of update program (16characters)
        integer huer
        parameter (huer=37)
    c)
    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 PNREC 40
16 define PNDICT 41
17 define iosccs "e(#)lopdefs 4.1 last update 86/12/12 16:24:32"
c ioparm 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 pcrc = current record number
  integer pcrc
  parameter (pcrc=22)
  c pfrm= frame size in integer words
  integer pfrm
  parameter (pfrm=20)
  c phdr= header size in integer words
  integer phdr
  parameter (phdr=21)
  c pnhrc = number of header records
  integer pnhrc
  parameter (pnhrc=40)
  c pwc = start of file (32 characters)
  integer pwc
  parameter (pwc=1)
  c pnrec = total number of dictionary records
  integer pnrec
  parameter (pnrec=41)
  c prf = total number of records in file
  integer prf
  parameter (prf=24)
  c pwr = read/write (user) flag
  integer pwr
  parameter (pwr=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 pty = file type
  integer pty
  parameter (pty=28)
  c punit = unit number
  integer punit
  parameter (punit=19)

c d

c end
c pktparm proc
2 c(pktparm parameters
3 c pktparm provides parameters for access to the i/o packets
4 c array packet
5 c eph1 = first ephemeris file.
6 integer eph1
7 parameter (eph1=1)
8 c eph2 = second ephemeris file.
9 integer eph2
10 parameter (eph2=2)
11 c att1 = first attitude file.
12 integer att1
13 parameter (att1=3)
14 c att2 = second attitude file.
15 integer att2
16 parameter (att2=4)
17 c svl1 = first sensed velocity file.
18 integer svl1
19 parameter (svl1=5)
20 c svl2 = second sensed velocity file.
21 integer svl2
22 parameter (svl2=6)
23 c obs1 = first observation file.
24 integer obs1
25 parameter (obs1=7)
26 c obs2 = second observation file.
27 integer obs2
28 parameter (obs2=8)
29 c rel1 = first relative trajectory file.
30 integer rel1
31 parameter (rel1=9)
32 c)
33 c end
c slitparam proc

1 c slitparam parameters
2 c nsol = dimension of solution vector
3 4 c d parameter (nsol = 22)
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

<table>
<thead>
<tr>
<th>NAME</th>
<th>FILE</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add_GNList</td>
<td>&quot;gnamlist.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>BaseCalTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>BaseTime</td>
<td>&quot;fileio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>Begin</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>CurSysTime</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>End</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>Error_Count</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>File</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>FileBaseTime</td>
<td>&quot;gbhead.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>Frame3</td>
<td>&quot;frame3d.c&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>GBeginTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>GDelTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>GEndTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>GMTday</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>GMTsec</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>GNLgetName</td>
<td>&quot;gnamlist.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>GetCurTime</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>HBtoHS</td>
<td>&quot;gbHBufMove.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>HStoHB</td>
<td>&quot;gbHBufMove.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>Help</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>JD_BaseTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>LBeginTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>LDelTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>LEndTime</td>
<td>&quot;BaseTime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>MakeTimeLine</td>
<td>&quot;timelines.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>Make_GNLIST</td>
<td>&quot;gnamlist.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>NTBLCK</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTCLOSE</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTEOF</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTFILE</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTOPEN</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTRREAD</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTRITE</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NTRW</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td>NTRWRL</td>
<td>&quot;ntran.c&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>Prime_id</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>Radius</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>Save_Str_Buf</td>
<td>&quot;str_store.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>Second_id</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>StatErrExit</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>Stat_Msg</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>Time</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>UNVOUT</td>
<td>&quot;unvout.c&quot;</td>
<td>Mktape</td>
</tr>
<tr>
<td>UVW_Cart</td>
<td>&quot;uvw.c&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>Warning_Count</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>a121q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a123q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a131q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a132q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a212q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a213q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a231q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a232q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a312q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a313q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a321q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a323q</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>a32t36</td>
<td>&quot;a32t36.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>acnvrt</td>
<td>&quot;acnvrt.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>adbl12i</td>
<td>&quot;acnvrt.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>addANLitem</td>
<td>&quot;anlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>addfile</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>addptr</td>
<td>&quot;ptrlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>aero</td>
<td>&quot;aero.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>aerror</td>
<td>&quot;ingrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>a12db1</td>
<td>&quot;acnvrt.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>a12rl</td>
<td>&quot;acnvrt.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>aijq</td>
<td>&quot;aijkq.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>amenu</td>
<td>&quot;amenu.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ang2</td>
<td>&quot;ang2.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>angle_rate</td>
<td>&quot;qaatt.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>append</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>arctan</td>
<td>&quot;arctan.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ar12i</td>
<td>&quot;acnvrt.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>arshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ascale</td>
<td>&quot;ascale.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>automx</td>
<td>&quot;automx.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>awarn</td>
<td>&quot;ingrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>berror</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>binop</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>bldout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>bsearch_timeline</td>
<td>&quot;timelines.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>bwarn</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>c2shl</td>
<td>&quot;c2shl.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>car</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>cart</td>
<td>&quot;cart.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>cbbody</td>
<td>&quot;cbody.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>cbugs</td>
<td>&quot;xcoas.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>cdr</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>cdrag</td>
<td>&quot;cdrag.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>cdtojd</td>
<td>&quot;cmpdat.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>center</td>
<td>&quot;center.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>char2l</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>cinit</td>
<td>&quot;xcoas.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>clas</td>
<td>&quot;clas.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>clsfil</td>
<td>&quot;obtfil.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>clsout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>cmp2sg</td>
<td>&quot;cmp2sg.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>cmpdat</td>
<td>&quot;cmpdat.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>cmvbit</td>
<td>&quot;cmvbit.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>coash</td>
<td>&quot;xcoas.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>coasv</td>
<td>&quot;xcoas.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>combine</td>
<td>&quot;obsnois.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>comptw</td>
<td>&quot;comptw.b&quot;</td>
<td>Product</td>
</tr>
</tbody>
</table>
# ROUTINE CROSS REFERENCE (cont’d)

<table>
<thead>
<tr>
<th>NAME</th>
<th>FILE</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>cons</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>constm</td>
<td>&quot;constm.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>cover_file</td>
<td>&quot;qacover.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>cros</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>cti0</td>
<td>&quot;tio.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>cvprop</td>
<td>&quot;cvprop.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>cyldr</td>
<td>&quot;cyldr.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>datetime</td>
<td>&quot;datetime.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>days</td>
<td>&quot;days.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>days</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>days1bc</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>daysxx</td>
<td>&quot;daysxx.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>db12t</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>dbshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>dciph</td>
<td>&quot;dciph.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>dciphb</td>
<td>&quot;dciphr.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>dctprt</td>
<td>&quot;dctpnt.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>ddnas</td>
<td>&quot;ddna.b&quot;</td>
<td>Noisanal</td>
</tr>
<tr>
<td>ddnos</td>
<td>&quot;ddnois.b&quot;</td>
<td>Noisanal</td>
</tr>
<tr>
<td>delout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>dfdata</td>
<td>&quot;dfdata.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>dfinput</td>
<td>&quot;dfinput.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>dhms</td>
<td>&quot;dhms.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>div</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>dkal</td>
<td>&quot;dkal.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>dnumber</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>dollar</td>
<td>&quot;dollar.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>dpfmt</td>
<td>&quot;dpfmt.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>dplot</td>
<td>&quot;dplot.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>dprod</td>
<td>&quot;dprod.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>dprop</td>
<td>&quot;dprop.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>dprst</td>
<td>&quot;dprst.b&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>dsmth</td>
<td>&quot;dsmth.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>dspxq</td>
<td>&quot;dspxq.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>dtio</td>
<td>&quot;tio.b&quot;</td>
<td>Prompts</td>
</tr>
</tbody>
</table>
## ROUTINE CROSS REFERENCE (cont'd)

<table>
<thead>
<tr>
<th>NAME</th>
<th>FILE</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>dvalue</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>dvdt</td>
<td>&quot;dvdt.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>dwnfmt</td>
<td>&quot;dwnfmt.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>dxcmp</td>
<td>&quot;dxcmp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>dxqdsp</td>
<td>&quot;dxqdsp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>dxset</td>
<td>&quot;dxset.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>eb2asc</td>
<td>&quot;eb2asc.c&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>edit_frame</td>
<td>&quot;filedit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>egrss</td>
<td>&quot;ingrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>eph2_in</td>
<td>&quot;mkinit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>eph2rel</td>
<td>&quot;eph2rel.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>eq</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>err_code</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>err_hppcode</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>etsec</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>euler</td>
<td>&quot;euler.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>fetchbits</td>
<td>&quot;ibmcvt.c&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>fgprtd</td>
<td>&quot;gbprtd.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>file2io</td>
<td>&quot;file2io.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>file_size</td>
<td>&quot;sln2r1.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>fileinfo_io</td>
<td>&quot;fileinfo_io.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>fileio</td>
<td>&quot;fileio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>fprtCurTime</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>fprtFinish</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>fprtStars</td>
<td>&quot;prt_util.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>fprtTimeline</td>
<td>&quot;timelines.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>fprtaray</td>
<td>&quot;prtaray.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>fprttctime</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>fprtdate</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>fprtgnlist</td>
<td>&quot;gnamlist.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>fprthms</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>fprtxsec</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>freeGBF</td>
<td>&quot;gbopen.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>freeplist</td>
<td>&quot;ptrlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>fskip_lines</td>
<td>&quot;prt_util.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>------------</td>
</tr>
<tr>
<td>fvalue</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>gbclose</td>
<td>&quot;gbopen.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gdbread</td>
<td>&quot;gbio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gdbwrite</td>
<td>&quot;gbio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfc1s</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfc1m</td>
<td>&quot;gbfcom.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>gbfdc</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfnw</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfpn</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfps</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbpphead</td>
<td>&quot;gbprtd.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfree</td>
<td>&quot;gbopen.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbfrnt</td>
<td>&quot;gbfio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbnew</td>
<td>&quot;gbopen.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbopen</td>
<td>&quot;gbopen.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbpthead</td>
<td>&quot;gbptd.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbpos</td>
<td>&quot;gbio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbread</td>
<td>&quot;gbio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbrhead</td>
<td>&quot;gbhead.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbtime</td>
<td>&quot;gbtime.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbtoff</td>
<td>&quot;gbhead.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbwhead</td>
<td>&quot;gbhead.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gbwrite</td>
<td>&quot;gbio.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>gdisp</td>
<td>&quot;gdisp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>gdspop</td>
<td>&quot;gdspop.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>genout</td>
<td>&quot;genout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>gerror</td>
<td>&quot;legrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>get</td>
<td>&quot;get.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>get_next_frame</td>
<td>&quot;file2io.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>get_rate_table_value</td>
<td>&quot;ratetable.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>getangle</td>
<td>&quot;qastar.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>getbit</td>
<td>&quot;getbit.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>getbits</td>
<td>&quot;eb2asc.c&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>getbits</td>
<td>&quot;unvout.c&quot;</td>
<td>Mktape</td>
</tr>
<tr>
<td>getdat</td>
<td>&quot;getdat.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>gethdr</td>
<td>&quot;gethdr.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>getin</td>
<td>&quot;initinput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>getm</td>
<td>&quot;getm.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>getnxt</td>
<td>&quot;getnxt.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>getout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>getrnp</td>
<td>&quot;getrnp.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>getspg</td>
<td>&quot;getspg.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>gfcls</td>
<td>&quot;gfcls.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfdict</td>
<td>&quot;gfdict.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfemsg</td>
<td>&quot;gfemsg.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfend</td>
<td>&quot;gfend.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gff_in</td>
<td>&quot;filedit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>gffdsp</td>
<td>&quot;gffdsp.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfnew</td>
<td>&quot;gfnew.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfopen</td>
<td>&quot;gfopen.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfread</td>
<td>&quot;gfread.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfhdr</td>
<td>&quot;gfhdr.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gftime</td>
<td>&quot;gftime.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gftims</td>
<td>&quot;gftims.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfwhdr</td>
<td>&quot;gfwhdr.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gfwrit</td>
<td>&quot;gfwrit.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>gmenu</td>
<td>&quot;qmenu.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>gndsp</td>
<td>&quot;gndsp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>grfnpt</td>
<td>&quot;pltnpt.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>gempty</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gfree</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gspeek</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gspeek</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gspeek</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gspeek</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gspeek</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>gwarn</td>
<td>&quot;iegrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>harm</td>
<td>&quot;harm.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>hdrprt</td>
<td>&quot;hdrprt.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>hms2ds</td>
<td>&quot;hms2ds.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>hms2sec</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>hptou5</td>
<td>&quot;hptou5.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>12char</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>12dbl</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>12real</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ibmcvt</td>
<td>&quot;ibmcvt.c&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>ibshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>icshft</td>
<td>&quot;icshft.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>ident</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ident3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ilegrss</td>
<td>&quot;ilegrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>imatq</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>imatq</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>imove</td>
<td>&quot;imove.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>imzero</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>infmt</td>
<td>&quot;infmt.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>ingrss</td>
<td>&quot;ingrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>initin</td>
<td>&quot;initinput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>inumber</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>invrs</td>
<td>&quot;invs.r&quot;</td>
<td>RELBET/Filter</td>
</tr>
<tr>
<td>iopprt</td>
<td>&quot;iopprt.b&quot;</td>
<td>Gff</td>
</tr>
<tr>
<td>ioschk</td>
<td>&quot;ioschk.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>irot</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>isatom</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>isdnumber</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>isearch_timeline</td>
<td>&quot;timelines.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>isfunction</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>isinumber</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>issymbol</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>itio</td>
<td>&quot;tio.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>ivalue</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>ivshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ivzero</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ixatm</td>
<td>&quot;ixatm.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixbias</td>
<td>&quot;ixbias.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixcon</td>
<td>&quot;ixcon.b&quot;</td>
<td>Input</td>
</tr>
</tbody>
</table>
### ROUTINE CROSS REFERENCE (cont’d)

<table>
<thead>
<tr>
<th>NAME</th>
<th>FILE</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ixdata</td>
<td>&quot;ixdata.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixdprm</td>
<td>&quot;ixdprm.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixerth</td>
<td>&quot;ixerth.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixfile</td>
<td>&quot;ixfile.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixgnrl</td>
<td>&quot;ixgnrl.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixgraf</td>
<td>&quot;ixgraf.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixgrav</td>
<td>&quot;ixgrav.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixinit</td>
<td>&quot;ixinit.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixkal</td>
<td>&quot;ixkal.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixmas</td>
<td>&quot;ixmas.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixmax</td>
<td>&quot;ixmax.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixmisc</td>
<td>&quot;ixmisc.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixmoon</td>
<td>&quot;ixmoon.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixnflz</td>
<td>&quot;ixnflz.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixprnt</td>
<td>&quot;ixprnt.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixprop</td>
<td>&quot;ixprop.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixqcrv</td>
<td>&quot;ixqcrv.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixqgen</td>
<td>&quot;ixqgen.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixqprm</td>
<td>&quot;ixqprm.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixrpst</td>
<td>&quot;ixrpst.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixscov</td>
<td>&quot;ixscov.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixsen</td>
<td>&quot;ixsen.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixsprm</td>
<td>&quot;ixsprm.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixsptm</td>
<td>&quot;ixsptm.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixsun</td>
<td>&quot;ixsun.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixsvbi</td>
<td>&quot;ixsvbi.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixtime</td>
<td>&quot;ixtime.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixtoff</td>
<td>&quot;ixtoff.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>ixusys</td>
<td>&quot;ixusys.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>lxvcx</td>
<td>&quot;lxvcx.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>lxvnt</td>
<td>&quot;lxvnt.b&quot;</td>
<td>Input</td>
</tr>
<tr>
<td>j2c</td>
<td>&quot;j2c.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>j2ymd</td>
<td>&quot;j2ymd.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>jdtocd</td>
<td>&quot;cmpdat.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>jmpcomp</td>
<td>&quot;qaranjmp.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>jul2cal</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>juldate</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>jultime</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>kalman</td>
<td>&quot;kalman.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>krelo</td>
<td>&quot;krelo.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>kreloc</td>
<td>&quot;krelo.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>lax</td>
<td>&quot;lax.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>length</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>lgfac</td>
<td>&quot;lgint.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>lgint</td>
<td>&quot;lgint.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>linit</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>linput</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>lint</td>
<td>&quot;lint.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>lintin</td>
<td>&quot;lintin.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>locate</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>lrntrp</td>
<td>&quot;lrntrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>lrntrp</td>
<td>&quot;reltrp.c&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>lsearch_timeline</td>
<td>&quot;timelines.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>lsmin</td>
<td>&quot;lsmin.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>lst</td>
<td>&quot;lst.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>lstfnd</td>
<td>&quot;lstfnd.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>m2qsub</td>
<td>&quot;m2qsub.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>main</td>
<td>&quot;cmp2sg.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;eph2rel.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;fiche.c&quot;</td>
<td>Fich</td>
</tr>
<tr>
<td>main</td>
<td>&quot;filedit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;gbfcom.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;mkinit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;obsnois.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qaatt.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qacover.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qanois.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qaranjmp.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qastar.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;qasv.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>main</td>
<td>&quot;rdwt.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;read_set.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;read_sit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;rlvrsrl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;sin2rl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>main</td>
<td>&quot;stop.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>makeDate</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>makeGBData</td>
<td>&quot;gbmake.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>makeGBF</td>
<td>&quot;gbmake.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>makeHMS</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>makeTime</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>make_gstack</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>makeplist</td>
<td>&quot;ptrlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>match</td>
<td>&quot;match.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>mtq</td>
<td>&quot;qtnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>matq</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>member</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>mkinit</td>
<td>&quot;mkinit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>mktape</td>
<td>&quot;mktape.b&quot;</td>
<td>Mktape</td>
</tr>
<tr>
<td>mmxchk</td>
<td>&quot;mmxchk.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>mntnum</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>moonup</td>
<td>&quot;moopos.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>moopos</td>
<td>&quot;moopos.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>msgdsp</td>
<td>&quot;msgdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>mshift3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mshift</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mt</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtran</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtran3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtxm</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtxm3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtxv</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtxv</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mtxv3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>mult</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>mvmul</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mvmul3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mx3ops</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxadd</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxm</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxmc</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxmt</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxmt3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxmul</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxmul3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxops</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mxv</td>
<td>&quot;matrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mzero</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>mzero3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ndfnit</td>
<td>&quot;ndnflz.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>ndfnpt</td>
<td>&quot;ndnflz.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>ndhead</td>
<td>&quot;ndhead.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>ndnflz</td>
<td>&quot;ndnflz.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>ndpage</td>
<td>&quot;ndpage.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>newGBF</td>
<td>&quot;gbhead.c&quot;</td>
<td>Gbfile</td>
</tr>
<tr>
<td>newcell</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>newcell</td>
<td>&quot;newcell.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>newpg</td>
<td>&quot;msgdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>ngss</td>
<td>&quot;ingrss.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>nil</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>nitout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>nkal</td>
<td>&quot;nkal.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>nlist</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>nnois</td>
<td>&quot;nnois.b&quot;</td>
<td>Noisanal</td>
</tr>
<tr>
<td>normq</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>nplot</td>
<td>&quot;nplot.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>nprod</td>
<td>&quot;nprod.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>nprop</td>
<td>&quot;nprop.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>nrmlzq</td>
<td>&quot;nrmlzq.b&quot;</td>
<td>Math</td>
</tr>
</tbody>
</table>
### ROUTINE CROSS REFERENCE (cont’d)

<table>
<thead>
<tr>
<th>NAME</th>
<th>FILE</th>
<th>DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>nrpst</td>
<td>&quot;nrpst.b&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>nsmth</td>
<td>&quot;nsmth.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>nstrng</td>
<td>&quot;nstrng.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>nxcmp</td>
<td>&quot;nxcmp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>nxqdisp</td>
<td>&quot;nxqdisp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>nxtd</td>
<td>&quot;nxtd.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>nxtd</td>
<td>&quot;nxtd.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>obcls</td>
<td>&quot;obout.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>obcode</td>
<td>&quot;obcode.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>obest</td>
<td>&quot;obest.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>obout</td>
<td>&quot;obout.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>obread</td>
<td>&quot;obread.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>obtfil</td>
<td>&quot;obtfil.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>obwrit</td>
<td>&quot;obwrit.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>opnout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>opnprp</td>
<td>&quot;opnprp.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>out_bias</td>
<td>&quot;sln2rl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>out_cov</td>
<td>&quot;sln2rl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>out_rel</td>
<td>&quot;sln2rl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>pchrat</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>pcpt</td>
<td>&quot;pcpt.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>pdsp</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>pitch</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>plotx</td>
<td>&quot;plotx.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>pltnpt</td>
<td>&quot;pltnpt.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>pmenu</td>
<td>&quot;qxmenu.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>pred</td>
<td>&quot;pred.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>print_record</td>
<td>&quot;read_set.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>print_record</td>
<td>&quot;read_sit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>prodx</td>
<td>&quot;prodx.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>prpnpt</td>
<td>&quot;prpnpt.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>prt3mat</td>
<td>&quot;prtarray.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>prt3tmat</td>
<td>&quot;prtarray.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>prt3vec</td>
<td>&quot;prtarray.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>prtANList</td>
<td>&quot;anlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>prtANLitem</td>
<td>&quot;anlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>prtCurTime</td>
<td>&quot;systime.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>prtFinish</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>prtStars</td>
<td>&quot;prt_util.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>prtarray</td>
<td>&quot;prtarray.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>prtctime</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>prtdate</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>prthms</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>prtin</td>
<td>&quot;prt_input.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>prtsec</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>psqls</td>
<td>&quot;psout.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>psout</td>
<td>&quot;psout.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>psstrz</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>ptbl</td>
<td>&quot;ptbl.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>put72</td>
<td>&quot;unvout.c&quot;</td>
<td>Mktnpe</td>
</tr>
<tr>
<td>putchr</td>
<td>&quot;unvout.c&quot;</td>
<td>Mktnpe</td>
</tr>
<tr>
<td>putexp</td>
<td>&quot;putexp.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>pwr</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>q121a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q123a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q131a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q132a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q212a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q213a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q231a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q232a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q2crv</td>
<td>&quot;q2crvz.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2crvz</td>
<td>&quot;q2crvz.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2draw</td>
<td>&quot;q2draw.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2ecrv</td>
<td>&quot;q2crvz.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2fnsf</td>
<td>&quot;q2grph.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2grph</td>
<td>&quot;q2grph.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2lgn</td>
<td>&quot;q2crvz.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2lsto</td>
<td>&quot;q2crvz.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>q2msub</td>
<td>&quot;q2msub.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>q312a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q313a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q321a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q323a</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>q_pop</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>q_push</td>
<td>&quot;gstack.c&quot;</td>
<td>Stacks</td>
</tr>
<tr>
<td>qanois</td>
<td>&quot;qanois.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>qatape</td>
<td>&quot;qatape.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>qcxq</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qcxq</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qdevin</td>
<td>&quot;qdevin.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qdevrl</td>
<td>&quot;qdevin.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qdot</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qijka</td>
<td>&quot;qijka.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>qintp</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qintrp</td>
<td>&quot;qintrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>qjntx</td>
<td>&quot;qjntx.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qplot</td>
<td>&quot;qplot.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qplt2d</td>
<td>&quot;qplt2d.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qpxget</td>
<td>&quot;qpxget.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qrspltn</td>
<td>&quot;qrspltn.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qrot</td>
<td>&quot;qrot.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qrspltn</td>
<td>&quot;qrspltn.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>qtnops</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qtoim</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qtom</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qvirot</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qvrot</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qwntrp</td>
<td>&quot;qwntrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>qxmenu</td>
<td>&quot;qxmenu.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>qxq</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qxq</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qxqc</td>
<td>&quot;qnops.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qxqc</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>qxstuf</td>
<td>&quot;qxstuf.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>qxvalu</td>
<td>&quot;qxvalu.b&quot;</td>
<td>Charutil1</td>
</tr>
<tr>
<td>range</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>ranrat</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>read_set</td>
<td>&quot;read_set.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>read_sit</td>
<td>&quot;read_sit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>real2i</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>ref_in</td>
<td>&quot;filedit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>rel_in</td>
<td>&quot;mkinit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>rl2beg</td>
<td>&quot;rlfill.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rl2end</td>
<td>&quot;rlfill.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlback</td>
<td>&quot;rlfill.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlbwds</td>
<td>&quot;rledit.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlcls</td>
<td>&quot;rlcls.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlcntr</td>
<td>&quot;rlcntr.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rldsp</td>
<td>&quot;rldsp.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rledit</td>
<td>&quot;rlledit.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlfill</td>
<td>&quot;rlfill.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlfnt</td>
<td>&quot;rlfill.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlfwds</td>
<td>&quot;rlledit.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rldsp</td>
<td>&quot;rldsp.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlhdsp</td>
<td>&quot;rlhdsp.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlmid1</td>
<td>&quot;rlmid1.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlnew</td>
<td>&quot;rlnew.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlintrp</td>
<td>&quot;rlintrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>rlopen</td>
<td>&quot;rlopen.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rpdsp</td>
<td>&quot;rldsp.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>riread</td>
<td>&quot;rlread.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rltime</td>
<td>&quot;rltime.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rlvsrl</td>
<td>&quot;rlvsrl.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>r1writ</td>
<td>&quot;rlwrit.b&quot;</td>
<td>Files</td>
</tr>
<tr>
<td>rmefile</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>rmptr</td>
<td>&quot;ptrlists.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>rmxm</td>
<td>&quot;rmatrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>rmxmc</td>
<td>&quot;rmatrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>rmxv</td>
<td>&quot;rmatrix.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>rzero</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>roll</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>rolrat</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>rot</td>
<td>&quot;qtnops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>rotate</td>
<td>&quot;rotate.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>rptost</td>
<td>&quot;rptost.b&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>rrf file</td>
<td>&quot;qaranjmp.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>rst</td>
<td>&quot;rst.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>rtangl</td>
<td>&quot;rtangl.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>rtio</td>
<td>&quot;tio.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>rtntrp</td>
<td>&quot;relntrp.c&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>rtntrp</td>
<td>&quot;rtntrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>runkut</td>
<td>&quot;runkut.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>rvprop</td>
<td>&quot;rvprop.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>rvshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>rvup</td>
<td>&quot;rvup.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>rvzero</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>sclset</td>
<td>&quot;sclset.b&quot;</td>
<td>Plot</td>
</tr>
<tr>
<td>search</td>
<td>&quot;search.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>sec2hms</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>secsnds</td>
<td>&quot;secnds.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>sensed vel</td>
<td>&quot;qasv.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>setDate</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>setHMS</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>setMaxErr</td>
<td>&quot;stat_msg.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>setTime</td>
<td>&quot;timemake.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>setin</td>
<td>&quot;initlinput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>setq</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>sfilt</td>
<td>&quot;sfilt.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>sfinit</td>
<td>&quot;sfinit.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>sfout</td>
<td>&quot;sfout.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>sftrat</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>shaft</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>shoxq</td>
<td>&quot;dspxq.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>sign</td>
<td>&quot;sign.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>skip_lines</td>
<td>&quot;prt_util.c&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------</td>
</tr>
<tr>
<td>smmul3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>smooth</td>
<td>&quot;smooth.r&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>snoise</td>
<td>&quot;snoise.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>solrad</td>
<td>&quot;s1rad.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>spfmt</td>
<td>&quot;spfmt.b&quot;</td>
<td>Qatape</td>
</tr>
<tr>
<td>sphdr</td>
<td>&quot;sphdr.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>sprtctime</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>sprtdate</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>sprthms</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>sprtsec</td>
<td>&quot;timeprint.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>star_track</td>
<td>&quot;qastar.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>starz</td>
<td>&quot;msgdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>std_time</td>
<td>&quot;timedate.c&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>stinit</td>
<td>&quot;xtrack.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>stm</td>
<td>&quot;stm.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>stmrv</td>
<td>&quot;stmrv.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>stop</td>
<td>&quot;stop.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>stprop</td>
<td>&quot;stprop.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>strsave</td>
<td>&quot;strsave.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>strstore</td>
<td>&quot;strstore.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>sub</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>sum</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>sunpos</td>
<td>&quot;sunpos.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>sunup</td>
<td>&quot;sunpos.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>superg</td>
<td>&quot;superg.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>svalue</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>svelbs</td>
<td>&quot;svelbs.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>svftch</td>
<td>&quot;svftch.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>svmul</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>svmul3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>svntrp</td>
<td>&quot;svntrp.b&quot;</td>
<td>Interpolate</td>
</tr>
<tr>
<td>symbol</td>
<td>&quot;lists.c&quot;</td>
<td>Lists</td>
</tr>
<tr>
<td>symxv</td>
<td>&quot;symxv.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>tangle</td>
<td>&quot;tangle.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>taperd</td>
<td>&quot;taperd.c&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>tblchk</td>
<td>&quot;tblchk.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>tblkck</td>
<td>&quot;tblkck.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>tdsp</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>tenrnd</td>
<td>&quot;tenrnd.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>test_print</td>
<td>&quot;qastar.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>text_in</td>
<td>&quot;filedit.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>tio</td>
<td>&quot;tio.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>tpdsp</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>tpstrz</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>trackh</td>
<td>&quot;xtrack.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>trackv</td>
<td>&quot;xtrack.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>trnion</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>trnrat</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>tstep</td>
<td>&quot;tstep.b&quot;</td>
<td>Propagate</td>
</tr>
<tr>
<td>tstrz</td>
<td>&quot;tpdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>twobod</td>
<td>&quot;twobod.b&quot;</td>
<td>Celestial</td>
</tr>
<tr>
<td>uax</td>
<td>&quot;uax.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>udpfmt</td>
<td>&quot;udpfmt.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>uinfmt</td>
<td>&quot;uinfmt.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>unop</td>
<td>&quot;linput.c&quot;</td>
<td>Lininput</td>
</tr>
<tr>
<td>uspfmt</td>
<td>&quot;uspfmt.b&quot;</td>
<td>Product</td>
</tr>
<tr>
<td>ustr</td>
<td>&quot;ust.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>usub</td>
<td>&quot;linput.c&quot;</td>
<td>Lininput</td>
</tr>
<tr>
<td>uvw2m</td>
<td>&quot;uvw2m.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>uvwlvh</td>
<td>&quot;uvw1vh.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>uvwmat</td>
<td>&quot;uvwmat.b&quot;</td>
<td>Coordinate</td>
</tr>
<tr>
<td>vadd</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vadd</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vadd3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vaddto</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>valdat</td>
<td>&quot;valdat.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>vdist</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vdot</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vdot</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vdot3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>vfadd</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vfaddto</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vfmul</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vlist</td>
<td>&quot;input.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>vlrest</td>
<td>&quot;vlrest.b&quot;</td>
<td>Filter</td>
</tr>
<tr>
<td>vnort</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vnorm3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vntfch</td>
<td>&quot;vntfch.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>vrss</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vrunit</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vset</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vshft3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vshft</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vsub</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vsubb</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vsub3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vunit</td>
<td>&quot;mxops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vunit3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vzero</td>
<td>&quot;arshft.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vzero2</td>
<td>&quot;vector.c&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>vzero3</td>
<td>&quot;mx3ops.b&quot;</td>
<td>Math</td>
</tr>
<tr>
<td>wndang</td>
<td>&quot;wndang.b&quot;</td>
<td>Force</td>
</tr>
<tr>
<td>write_fi_e</td>
<td>&quot;rdw.t.c&quot;</td>
<td>Fman</td>
</tr>
<tr>
<td>wrout</td>
<td>&quot;getout.b&quot;</td>
<td>Downfor</td>
</tr>
<tr>
<td>xibnk</td>
<td>&quot;qxtstuf.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>xblank</td>
<td>&quot;qxtstuf.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>xcmpar</td>
<td>&quot;xcmpar.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xoas</td>
<td>&quot;xoas.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>xdsp</td>
<td>&quot;msgdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>xdsp2</td>
<td>&quot;msgdsp.b&quot;</td>
<td>Message</td>
</tr>
<tr>
<td>xf8dmp</td>
<td>&quot;xf8dmp.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>xi2chr</td>
<td>&quot;xi2chr.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>xnth</td>
<td>&quot;xnth.b&quot;</td>
<td>Charutil</td>
</tr>
<tr>
<td>xqdatz</td>
<td>&quot;xqwtmz.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>xqdsp</td>
<td>&quot;xqdsp.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>xqdspl</td>
<td>&quot;dspxq.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xqdsph</td>
<td>&quot;dspxq.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xqkmp1</td>
<td>&quot;xqkmpt.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xqkmp2</td>
<td>&quot;xqkmpt.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xqkmpt</td>
<td>&quot;xqkmpt.b&quot;</td>
<td>Numdis</td>
</tr>
<tr>
<td>xqtime</td>
<td>&quot;xqwtmz.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>xqwget</td>
<td>&quot;xqwget.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>xqwint</td>
<td>&quot;xqwget.b&quot;</td>
<td>Dsputil</td>
</tr>
<tr>
<td>xqwtmz</td>
<td>&quot;xqwtmz.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>xradar</td>
<td>&quot;xradar.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>xtrack</td>
<td>&quot;xtrack.b&quot;</td>
<td>Obs</td>
</tr>
<tr>
<td>yesno</td>
<td>&quot;yesno.b&quot;</td>
<td>Prompts</td>
</tr>
<tr>
<td>ymd2j</td>
<td>&quot;ymd2j.b&quot;</td>
<td>Time</td>
</tr>
<tr>
<td>yyback</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylbin</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylcrank</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyerror</td>
<td>&quot;linput.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylstate</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyextra</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyfnd</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylin</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyinput</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyleng</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylex</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylineno</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yylook</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1sp</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1state</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yymatch</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yymorfg</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1spsp</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1out</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1output</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yy1previous</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yysbuf</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>NAME</td>
<td>FILE</td>
<td>DIRECTORY</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>yysptr</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yysvec</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yytchar</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yytext</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yytop</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyunput</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>yyvstop</td>
<td>&quot;lex.c&quot;</td>
<td>Linput</td>
</tr>
<tr>
<td>zutek</td>
<td>&quot;zutek.b&quot;</td>
<td>Force</td>
</tr>
</tbody>
</table>
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 ephemeris 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.

\[ t \quad \text{current time (input)} \]
\[ r \quad \text{moon position at current time (output)} \]

moonup
Creates moon ephemeris based on xxmoon parameters.
rotate.b(3)  (Celestial)  rotate.b(3)

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  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.

\[ t \quad \text{current time (input)} \]
\[ r \quad \text{sun position at current time (output)} \]

sunup
Creates sun ephemeris based on xxsun parameters.
NAME  
utbod

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 (ix, lx)
   integer ix
   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*1 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 (numDer,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, x1blank

SYNOPSIS
subroutine qxstuf (i,p,i1)
character*1 p (i)
integer i1
entry xblank (i,p)
entry x1blank (i,p,i1)

DESCRIPTION
qxstuf
Sets character array to blanks and finds first nonblank in string.
  i  length of character string
  p  character string to be blanked
  i1 index to non blank
xblank
Sets character array to blanks.

x1blank
Finds 1st non blank in string.
NAME
_xvalu

SYNOPSIS

subroutine _xvalu (i,p,i1)
    integer i
    character*1 p (i)
    integer i1

DESCRIPTION

_xvalu
    Converts string to integer.
    i     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 (_,ntm)
    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
\text{Converts euler angles to quaternion elements.}
\text{alpha euler angles in sequence.} 1=\text{roll}, 2=\text{pitch}, 3=\text{yaw (input) }
\text{q quaternion elements generated from the euler sequence (output) }

a121q
\text{Performs a roll-pitch-roll to quaternion operation.}

a123q
\text{Performs a roll-pitch-yaw to quaternion operation.}

a131q
\text{Performs a roll-yaw-roll to quaternion operation.}

a132q
\text{Performs a roll-yaw-pitch to quaternion operation.}

a212q
\text{Performs a pitch-roll-pitch to quaternion operation.}

a213q
\text{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,xot,yot,zot
     (output)
NAME
cart

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

DESCRIPTION
cart
Converts classical 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. argument 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. ascending node
  5. argument of perigee
  6. true anomaly (output)
x cartesian state (input)
NAME
euler

SYNOPSIS
subroutine euler (kop, n, alpha, q, rmat)
integer kop
integer r
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.

ekop 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. '321' = 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 orthogonal frame from 3 dimensional vectors x,y,z.

SYNOPSIS

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

DESCRIPTION
Frame3
forms orthogonal 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 (Ve^i 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 lvih state given inertial state and inertial to lvih matrix.
  xi  input inertial state
  xo  output lvih state where
      Pos0 = lm \times Posi
      Vel0 = lm \times (Vel1 + Posi \times omega)
  lm  transformation matrix from inertial to lvih 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.
Performs a pitch-yaw-roll operation.

Performs a pitch-yaw-pitch operation.

Performs a yaw-roll-pitch operation.

Performs a yaw-roll-yaw operation.

Performs a yaw-pitch-roll operation.

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 cartesian 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 lvln coordinates from uvw info.

xu    uvw coordinates (input)

w     angular rate of local vertical frame (input)

xl    lvln 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 Vel)
r radius of vehicle (input)
w angular rate of local vertical frame norm(Pos x Vel)/(r • r) (output)
NAME
dfdata

SYNOPSIS
data dfdata

DESCRIPTION
dfdata
   Initializes common blocks.
NAME
defnput

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 downlist 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(string1,string2,m)
int *string1;
int *string2;
int *m;

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

DESCRIPTION
eb2asc
C routine to convert ebcdic to ascii

string1 string of characters to be convert (input)

string2 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 position (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.
getdat.b(3) (Downfor) getdat.b(3)

NAME
getdat

SYNOPSIS
subroutine getdat (recfnd)
integer recfnd

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

recfnd flag indicating 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
gout, 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 nc. entries in array
type type of conversion 0 - sp : t - dp
fetchbits function to get k-bits from word(right justified)
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 msid's

TRW Houston System Services - 1 - (printed 12/18/86)
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 CSE in either 36 bit or 72 bit word lengths.
3. Only data for which the CSE 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
Initializes 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
xawget, xqwint

SYNOPSIS
subroutine xawget (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
xawget
Obtains trajectory and attitude information from files. Entry xqwint
initializes 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 fich can generate a multi-reel file. However, the fich processor will treat each tape as an individual file. It requires three user inputs which are put on the command line.

usage: fich file run_id tape_drive where
    file - name of print file to put on the tape
    run_id - Univac type runid 660???
    tape_drive - C or 1
NAME
center

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

fiintrp(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
RELNET 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
double precision ed
dooble precision obias
dooble precision obs
dooble precision res
   integer status

DESCRIPTION
obwrit
   RELBE7 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 (n)
double precision tm
integer n
integer rec (n)
double precision t (n)
double precision e (n)
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 rlmidl.

iop  /c 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
rdsp, rlpdsp, rldsp

SYNOPSIS
subroutine rdsp (p,u)
    integer p (45)
    integer u
    entry rlpdsp (p,u)
    entry rldsp (p,u)

DESCRIPTION
rdsp
    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.

rldsp
    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
e file +1 edit value for okay < -10 for error
iop  i/c 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 beginning.
NAME
rlfill, r12end, rlback, r12beg, rlfrnt

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

entry r12end (iop,n,iep,fnd,rec,t,e,stat)

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

entry r12beg (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 
r12end attempt to fill entries in array after specified index. If
unsuccessful r1fnt and rlback error exit. If unsuccessful r12end and
r12beg 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 r12end:

ielo=start of data, filling starts at iep+1; r12beg: iep=current 1st
index, filling starts at 1 to iep; r12end: iep=current 1st index, filled
from iep to n r12beg 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)

r12end
Attempts to fill entries in array from iep to n.

rlback
Attempts to fill entries in array at fnd+1.

r12beg
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 (lop, tm, n, fnd, rec, t, e, stat)
  integer lop (*)
  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.

lop  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
riopen

SYNOPSIS
subroutine riopen (iop,name,unt,uze,status)
  integer iop
  character name
  integer unt
  integer uze
  integer status

DESCRIPTION
riopen 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
RELBE 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 15
   integer ed
double precision data (*)
   integer status

DESCRIPTION

rlwrit

RELBEET 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

Diffs 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.

TRW Houston System Services - 1 - (printed 12/18/86)
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
dsmth

DESCRIPTION
This block data routine initializes the common used by this program.
NAME
getnxt

SYNOPSIS
subroutine getnxt (tnxt, 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 if 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, ededit, 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 ededit
  double precision uwterm

DESCRIPTION
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 underweighting 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
invrs.r(3)  (RELNET/Filter)  invrs.r(3)

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.
nsmth.b(3) (Filter) nsmth.b(3)

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 ODeSt (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 value

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 ph1*c*phitranspose.
p state transition matrix (input).
c covariance (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. Sentinel 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).

ORDER PAGE IS OF POOR QUALITY
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 (nsc1,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
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 sigmas 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 sigmas file 2 <= time of frame1
frame2b the frame from relative sigmas file 2 > time of frame1

TRW Houston System Services - 1 -

(printed 12/18/86)
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
criteria.
Multiple frame ids can be specified to indicate the frames
to be edited.

USAGE:
filedit <-a__><-c><-t__><-c__><-p__><-f__><-o__><-h__><-i__><-j__>

where
-a__ Provides the edit value to be applied.
Zero is the same as default.
Negative input indicates edited frames are to be unedited.
Positive input indicates unedited frames are to be edited.
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 edited 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 edited.

\(-f\) 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 interpreted as ignored if \(-t\) option not used.

\(-t\) 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 edited 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 \(-t\) 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 edited.
filedit.c(3) (Fman)

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 edited
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 i.e., 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 input 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

```c
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 repeatedly 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 f8obsObs_ddnois Na
```

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

```
obsnois f8obs_ON Obs_ddnoisNd
```

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

```c
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:
```
geat <e> <b> > file1 ....
```
where `-e` is the maximum acceptable angular acc magnitude (default 10)

`-b` is the minimum angular acc to determine an event (default 0.0005)

The options are parsed 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:
```
quacover <-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 danois. 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 fttn number)
_q_ is the quantization value
_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 .02 90 .9; 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 repeatedly 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
                     > 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 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 Rd. 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

TRW Houston System Services  - 1 - (printed 12/18/86)
NAME
main, star_track, getangle, test_print - Star Tracker obs history

SYNOPSIS

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

char *star_track(frame1,frame2)
char *frame1,_frame2;

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

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

DESCRIPTION

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 quaternion 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
same name 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)
-t__ is the minimum 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 interv
ptr pointer, not used
frame current data for file

TRW Houston System Services - 1 - (printed 12/18/86)
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
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 10
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
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
-t_ indicates begin time in gmt seconds
-e_ indicates last time in gmt seconds
-c_ 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 reads two relative trajectory files and determines the
difference 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 relset da 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 -r - - - - - - - - - - - - h
where:
-n - 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

TRW Houston System Services  - 1 -  (printed 12/18/86)
NAME
main, out_rel, out_coy, out_bias, file_size

SYNOPSIS
int main(argc, argv)
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
sin2rl 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 UVx 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
sin2rl <-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 aerodynamic 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 acceleration (output)
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 acceleration (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
cylar

SYNOPSIS

subroutine cylar (wb, rho, fb)
double precision fb (3)
double precision rho
double precision wb (3)

description
cylar
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 in 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 unbiased of sensed acceleration.
  v    vehicle id (input)
  t    desired time (input)
  m    body to inertial matrix (input)
  as   sensed acceleration (input/output)
NAME
  svfch

SYNOPSIS
  subroutine svfch (ivloc, inc, tc, tnew)
  integer ivloc
  integer inc
  double precision tc
  double precision tnew

DESCRIPTION
  svfch
  Computes sensed acceleration.
  ivloc  vehicle id (input).
  inc    step direction (input).
  tc     current time (input).
  tnew   next time (output).
NAME  
	blook

SYNOPSIS  

subroutine tblook (imx, ltb, tb, t, i)  
integer imx  
integer ltb  
double precision tb (ltb, imx)  
double precision t  
integer i

DESCRIPTION  
	blook  
Finds first entry in a time increasing table that is at or before a given  
time. The table is double precision 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)
ccb cosine of side slip angle(output)
sa sine of angle attack(output)
sa 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 previously read gff
file_info file information of input gff

declaration function sends the data frames which have times bracketing 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 response 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

TRW Houston System Services - 1 - (printed 12/18/86)
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        current frame from driver file
frame

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 concatenated 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.

TRW Houston System Services - 1 - (printed 12/18/86)
NAME
HStoHB, HBtoHS - move gbf header info to and from buffer

SYNOPSIS

void HStoHB(_drbuf, gbf)
GBFILE *gbf;
char *ndrbuf;

void HBtoHS(gof, 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 gbfio C routines

SYNOPSIS

void gbfnew(file, fname, hrid, jobdes, wrdfrm, date, dates, status)
    int *file;
    char *fname;
    char *hrid;
    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
    hrid     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

gbfcis
  general close

gbfdfc
  close function for use with downlist processor to close files then reopen and display header and return begin/end times

gbfoDs
  open the file name for output display in place of stderr used if accessing the function gbdfdc above
NAME

FileBaseTime, newGBF, gbtsoff, 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(gof)
GBFILEPTR gbf;

DESCRIPTION

newGBF
forms new gbsstructure and sets input fields. Returns null pointer is error encountered.

gbtsoff
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

gbos, goread, gowrite, gbdread, gbdwrite - i/o of gbfiles

SYNOPSIS

\[ \text{gbpos(gbf,rec)} \]
\[ \text{GBFILE *gbf;} \]
\[ \text{unsigned rec;} \]
\[ \text{char *gbread(gbf)} \]
\[ \text{GBFILE *gbf;} \]
\[ \text{int gowrite(gbf, data)} \]
\[ \text{GBFILE *gbf;} \]
\[ \text{char *data;} \]
\[ \text{char *gbdread(gbf, rec)} \]
\[ \text{GBFILE *gbf;} \]
\[ \text{unsigned rec;} \]
\[ \text{int gbdwrite(gbf,rec,data)} \]
\[ \text{GBFILE *gbf;} \]
\[ \text{unsigned rec;} \]
\[ \text{char *data;} \]

DESCRIPTION

gbos
positions file pointer to desired record of referenced gbfile

goread
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)

gowrite
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 record type. errors
result in invalid pointer value (0)

gbdwrite
like gowrite 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 size = sizeof(int)*int_size. Errors result in null pointer being
returned. Fields are not set if pointers are null.
NAME
gbclosc, freeGBF, gbfree, gbopen, gbnew - open, close and free of gbfiles

SYNOPSIS

void gbclose(gbf)
    GFILEPTR gbf;

void freeGBF(gbf)
    GFILEPTR gbf;

void gbfree(gbf)
    GFILEPTR gbf;

GFILE *gbopen(name,mode)
    char *name;
    char *mode;

GFILEPTR gbnew(name, size, tbyte, tbase, story, format, mode)
    char *name;
    char *mode;
    int size;
    int tbyte;
    char *story, *format;
    CALTIME *tbase;

DESCRIPTION

gbclosc
    closes file and updates header if file had a write key set.

freeGBF
    frees all space allocated to gbfile structure. 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

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  

`gbopen` 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 correctly 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

```c
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 should 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 obtained 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
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
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 (input)
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 \texttt{xend} frame to the GFF file.

\begin{itemize}
\item \texttt{iop} \hspace{1cm} i/o packet(input/output)
\item \texttt{stdout} \hspace{1cm} standard out unit number(input)
\item \texttt{filout} \hspace{1cm} print file unit number(input)
\item \texttt{frame} \hspace{1cm} frame buffer(input)
\item \texttt{status} \hspace{1cm} i/o status(output)
\end{itemize}
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, spcbk, 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 spcbk (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)
  spcbk    block of information particular to the file type (input)
  dict     input dictionary (input)
  status   I/O status (output)

TRW Houston System Services - 1 - (printed 12/18/86)
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

TRW Houston System Services - 1 - (printed 12/18/86)
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 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
   timtag     timetag
   recnum     current record number
NAME

gfrhdr

SYNOPSIS

subroutine gfrhdr (hdr, iop, ndrsz, frame)
  integer ndrsz
  integer hdr (ndrsz)
  integer iop (*)
  integer frame (*)

DESCRIPTION

gfrhdr

  Reads the header from the GFF file.

  ndrsz  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, time1, timeo, index, status)
  integer iop (-)
  integer filout
  integer stdout
  double precision time1
  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  /o packet
filout print file unit number
stdout standard output unit number
time1 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 gfwhar (hdrsz, iop, hdr)
   integer hdrsz
   integer hdr (harsz)
   integer iop (*)

DESCRIPTION
gfwhar
   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
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
icshft.b(3) (Gff) icshft.b(3)

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.
ixbias.b(3) (Input) ixbias.b(3)

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 xxaprm.
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
ixgrr1

SYNOPSIS
subroutine ixgrr1

DESCRIPTION
ixgrr1
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.
ixinit.b(3) (Input) ixinit.b(3)

NAME
ixinit

SYNOPSIS
subroutine ixinit

DESCRIPTION
ixinit
Input of common block xxinit.
NAME

ixkal

SYNOPSIS

subroutine ixkal

DESCRIPTION

ixkal:
Input of common block ixkal.
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 xvmisc.
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.
ixprop.b(3) (Input) ixprop.b(3)

NAME
ixproD

SYNOPSIS
subroutine ixprop

DESCRIPTION
ixproD
Input of common block xxprop.
NAME
ixqcrv

SYNOPSIS
subroutine ixqcrv

DESCRIPTION
ixqcrv
Input of common block xxqcrv.
NAME
ixagen

SYNOPSIS
subroutine ixagen

DESCRIPTION
ixagen
Input of common block xxagen.
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
  \texttt{ixscov}

SYNOPSIS
  subroutine \texttt{ixscov}

DESCRIPTION
  \texttt{ixscov}
  Input of common block \texttt{xxscov}.

NAME
ixsen

SYNOPSIS
subroutine ixsen

DESCRIPTION
ixsen
Input of common block data.
NAME
  ixsprm

SYNOPSIS
  subroutine ixsprm

DESCRIPTION
  ixsprm
  Input of common block xxspm.
NAME
ixsptm

SYNOPSIS
subroutine ixsptm

DESCRIPTION
ixsptm
Input of common block ixsptm.
NAME
ixsun

SYNOPSIS
subroutine ixsun

DESCRIPTION
ixsun
Input of common block xxsun.
NAME
ixsvbi

SYNOPSIS
subroutine ixsvbi

DESCRIPTION
ixsvbi
Input of common block xixsvbi.
NAME
ixtime

SYNOPSIS
subroutine ixtime

DESCRIPTION
ixtime
Input of common block xxtime.
ixtoff.b(3) (Input) ixtoff.b(3)

NAME
ixtoff

SYNOPSIS
subroutine ixtoff

DESCRIPTION
ixtoff
Input of common block xxtoff.
NAME
ixusys

SYNOPSIS
subroutine ixusys

DESCRIPTION
ixusys
Input of common block xxusys.
ixvcx.b(3) (Input) ixvcx.b(3)

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).

x2 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
dooble precision q1 (4)
dooble precision t2
dooble precision q2 (4)
dooble precision t
dooble 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 desireo 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

```c
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 necessary
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 input file
NAME
yyleng, yymorf9, yytchar, yyin, yyout, yyestate, yylex, yyvstop, yycrank,
ysvec, yytoD, yybgin, yymatch, yyextra, yylineno, yytext, yylstate, yyisp,
yyolsp, yysbuf, yystrr, yyfnd, yyprevious, yylook, yyback, yyinput,
yyoutput, yyunput

SYNOPSIS

int yyleng;
int yymorf9;
int yytchar;
FILE *yyin, *yyout;
struct yysv *yyestate;
yylex()
int yyvstop[];
struct yywork yycrank[];
struct yysv *ysvec[];
struct yywork *yytoD;
struct yysv *yybgin;
uchar yymatch[];
uchar yyextra[];
int yylineno;
uchar yytext[];
struct yysv *yylstate[], **yyisp, **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

TRW Houston System Services - 1 - (printed 12/18/86)
int
  end of yylex

yyextra
  UNISRC_ID @(#)27.2  8E/02/26

yyolsp
  char yysbuf[YYLMAX];
  * char *yysptr = yysbuf;
  * ----- nils8 ******

yyinput
  the following are only used in the lex library
linput.c(3)                     (Linput)                     linput.c(3)

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
(currently "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
consistent 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

dwr
power
NAME
newcell

SYNOPSIS

P_CELL newcell()

DESCRIPTION
newcell
allocate space
NAME

prtln - print the contents of the input blocks one variable at a time

SYNOPSIS

void prtln(loc,name,dimtype)
char *loc;
char *name;
char *dimtype;

DESCRIPTION

prtln
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, y)
lists.c(3)  (Lists)  lists.c(3)

P_CELL x, l;
int length(l)
P_CELL l;

P_CELL locate(x, l, m)
P_CELL x, l, m;

DESCRIPTION
limit initialization
cons
  cons cell constructor
inumber
  integer number cell constructor
dnumber
  double number cell constructor
symbol
  symbol cell constructor
isatom
  predicate: is cell atom?
isnumber
  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
tvalue
  return tvalue 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, ai2db1, arl2i, adbl2i

SYNOPSIS
subroutine acnvrt (i,v,ivl,rvl)
integer i
integer ivl (i)
real rv1 (i)
double precision v (i)
entry ai2rl (i,ivl,rvl)
entry ai2db1 (i,ivl,v)
entry arl2i (i,rv1,ivl)
entry adbl2i (i,v,ivl)

DESCRIPTION
acnvrt
Converts vector arrays of one type to another type.
i dimension indexes (input)
ivl integer vectors (input/output)
rvl real vector (input/output)
v double precision vectors (output)
ai2rl
Shifts integer vector to real vector.
ai2db1
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, dbsnft, t2char, t2real, t2dcl, char2i, real2d, db2i

SYNOPSIS
subroutine arshft (i,v,v2,v3,iv1,iv2,rv1,xv,rv2)
integer i
oouPle precision v (1)
double precision v2 (1)
double precision v3 (1)
integer iv1 (1)
integer iv2 (1)
real rv1 (1)
character*4 xv (1)
real rv2 (1)

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 dbsnft (i,v2,v3)
entry t2char (i,iv1,xv)
entry t2real (i,iv1,rv1)
entry t2dcl (i,iv1,v)
entry char2i (i,xv,iv1)
entry real2i (i,rv1,iv1)
entry db2i (i,v,iv1)

DESCRIPTION
arshft
Moves data between arrays of different types (no conversion).
1 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.

ivshift
  Shifts the integer vector iv1 to iv2.

rvshift
  Shifts the real vector rv1 into rv2.

itshift
  Shifts an integer array starting at last element.

dbsshift
  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 symmetric 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 *txm(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

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 mvmu13 (m1,v1,v2)
entry mtxv3 (m1,v1,v2)
entry mshft3 (m1,m2)
entry mzero3 (m1)
entry ident3 (m1)
entry smmu13 (x,m1,m2)
entry mxmu13 (m1,m2,m3)
entry mtxm3 (m1,m2,m3)
entry mxms3 (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 an 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
   Normalizes 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, i1, m1, m2, m3, m, mt, v1, v2, v3, x, v, mm)
integer i
integer j
integer p
real rmi (i, j)
integer im1 (i, j)
double precision m1 (i, j)
double precision m2 (i, j)
double precision m3 (i, j)
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 mm (j, i)
entry mxmul (i, j, p, m1, m2, m3)
entry mxmul (i, j, m1, v1, v2)
entry mtxv (i, j, m, v2, v1)
entry mxadd (i, j, m, v1, v2, v3, m, mm)
entry mshift (i, j, m1, m)
entry mtran (i, j, m1, mt)
entry mzero (i, j, m1)
entry imzero (i, j, m1)
entry rmzero (i, j, m1)
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, v, v2)

DESCRIPTION
mxops
Performs various matrix and vector operations.
i dimension index (input)
j dimension index (input)
mxops (Math)

- **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.

**mxxv**
- 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 fprtarray(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 preceded 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
qx, qcxq, qxqc, qintp, qvrot, qvirot, adot, imatq, matq, normq - quaternion operations

SYNOPSIS

```c
double qx(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 t5, t1, t2;

double qvrot(vout, q, v)
double vout[], v[];
QUATERNION q;

double qvirot(vout, q, v)
double vout[], v[];
QUATERNION q;

double adot(aadot, q, v)
double aadot[];
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

**qx**

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 = q2 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 quaternion 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
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 quaternion (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 acq (q1,q2,q2)
  entry axq (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.

  acxq
Sets q3 to q1 conjugate times q2.

  axq
Sets \( q_3 \) equal to \( q_1 \) times \( q_2 \) conjugate.

\texttt{gtom}

Converts \( q \) to matrix \( m \).

\texttt{qtoim}

Computes matrix of inverse rotation represented by \( q \).

\texttt{rot}

Uses \( q \) to transform \( v \) to \( x \).

\texttt{irot}

Uses conjugate of \( q \) to transform \( v \) to \( x \).

\texttt{imatq}

Converts matrix \( m \) to inverse quaternion \( q \).

\texttt{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, 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 a 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 triangle by rows times a
vector.

dimension of array containing symmetric matrix (input)
dimension of vector (input)
array containing symmetric matrix: lower triangular portion stored
by rows (input)
vector to be multiplied (input)
product vector (output)
tenrnd

DESCRIPTION
	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, vset, vfmul, vrss, 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(_, 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.
vector.c(3) (Math) vector.c(3)

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

vzeroc
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 - Address and Name lists

SYNOPSIS

```c
int addANLitem(address, name)
char *address, *name;

int prtANLitem(address, string)
char *address, *string;

void prtANList()
```

DESCRIPTION

addANLitem

Add entry to address/name list. Returns item number (starting at 1) if
okay. Returns 0 if list is full.

prtANLitem

Searches address/name lists and displays name and address 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 address/name list to stdout

TRW Houston System Services - 1 - (printed 12/18/86)
NAME
datetime

SYNOPSIS
subroutine datetime (unit)
integer unit

DESCRIPTION
datetime
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

```c
char *GNLgetName(aoress,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_GNLList(adress,dsop,list)
char *adress, *dsop;
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 preceded by description dscp and consists of octal adress, name followed by newline

**Make_GNLIST**

initialize and set up a namelist of addresses 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.

**Add_GNList**

add entry to Namelist. Returns current number of items in list. If list is full, -1 is returned.
iegrss.b(3) (Message) iegrss.b(3)

NAME
iegrss, gwarn, gerror

SYNOPSIS
subroutine iegrss (p, stdout, filout)
caller 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 accumulates counts.
gerror Displays error message and accumulates counts.

TRW Houston System Services - 1 - (printed 12/18/86)
NAME

ingress, ngrss, awarn, aerror, egress

SYNOPSIS

subroutine ingress
character p

entry ngrss
entry awarn (p)
entry aerror (p)
entry egress

DESCRIPTION

ingress
Contains entry points that provide for the start and stop of all RELBET processors. Ingress 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. Egress is a generic stopping point for relbet routines. It displays a termination message and schedules the new execution 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 to terminal.

awarn
Displays warning and accumulates counts.

aerror
Displays error message and accumulates counts.

egress
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,ul,u2)
  character x
  integer ul
  integer u2
  entry xdsp (x,ul)
  entry xdsp2 (x,ul,u2)
  entry starz (ul)
  entry newpg (ul)

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 asterisks and newpg flips a page.

  x    string to display (input)
  ul   1st display unit (input)
  u2   2nd display unit (input)

xdsp
Displays string x to unit ul.

xdsp2
Displays string x to unit ul,u2.

starz
Displays a line of asterisks.

newpg
Flips a page in display file.
NAME

fskip_lines, skip_lines, fprtStars, prtStars - display utilities

SYNOPSIS

fskip_lines(fd, lines)
FILE *fd;
int lines;

skip_lines(lines)
int lines;

fprtStars(fd, nl)
FILE *fd;
int nl;

prtStars(nl)
int nl;

DESCRIPTION

fskip_lines
  Skips specified number of lines by print newlines to fd.

skip_lines
  Sends specified number of newlines to stdout.

fprtStars
  Prints nl lines of 60 stars followed by a cr to specified stream.

prtStars
  Prints nl lines of 60 stars followed by a cr to stdout.
NAME
addptr, rmptr, makeplist, freeplist - maintain pointer lists

SYNOPSIS

```c
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 occurrence 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 length 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. Dir 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_hcode, 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_hcode(f)
FILE *f;

void err_code(f)
FILE *f;

void berror(msg,code)
cchar *msg;
int code;

void bwarn(msg,code)
cchar *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 first 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 ':' then the
HP code errinfo is written as 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 necessary. 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

TRW Houston System Services - 1 - (printed 12/18/86)
NAME
  tpdsp, tdp, pdp, tpstrz, tsusz, pstrz

SYNOPSIS
  subroutine tpdsp (msg)
    character msg
  entry tdp (msg)
  entry pdp (msg)
  entry tpstrz
  entry tsusz
  entry pstrz

DESCRIPTION
  tdp
    Displays messages to print and terminal units.
    msg    message string (input/output)

  pdp
    Displays message to terminal unit.

  tdp
    Displays message to nominal print unit out.

  tpstrz
    Displays string of asterisks to terminal and print file.

  tsusz
    Displays string of asterisks to terminal.

  pstrz
    Displays string of asterisks 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
    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
    p 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
dnois

SYNOPSIS
  program dnois

DESCRIPTION
dnois
   Driver for noise analysis using variate divided difference.

TRW Houston System Services - 1 - (printed 12/18/86)
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

dsqx, xsqsp, xqdsph, shoxq

SYNOPSIS

subroutine dsqx(t,is,if,d,stat)
double precision t
integer is
integer if
double precision d(6)
integer stat
	entry xqdsph(stat)
tenry xqdsph(t)
tenry shoxq(t,xpg,if,d,is)

DESCRIPTION

dsqx
Entry points for initialization and performing xqdsph output.
time tag (input/output)
is scale option: 1=lilvvv,2=lvaaa3=lvaww,4=aaawww
where 1=eng,v=vel,a=ang,n=n.c ,w=ang rate (input)
if parameter index(input/output)
da data buffer(input/output)
stat status flag (output)

xqdsph
Initializes compute flags and check for errors.

xqdsph
Initializes display heading for parameter group.

shoxq
Performs output for xokmpnt.
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
Solicit 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 gnsp (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)
   array of interpolation factors (input/output)
 t   array of time tags for entries
 td   array of time tags for entries
 z   desired time (input)

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

TRW Houston System Services - 1 -
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
  xcmpr

SYNOPSIS
  program xcmpr

DESCRIPTION
  xcmpr
  driver for comparison of trajectories. Output may be 2 trajectories or relative trajectory
NAME
mosp

SYNOPSIS
program mosp

DESCRIPTION
mosp
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)
ccharacter*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 exclusion flag (output)
rtangl.b(3) (Obs) rtangl.b(3)

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 rendezvous radar roll and shaft, and coas and startracker horizontal
and vertical angles. \( \tan(g) = \frac{r_{ua}}{r_{ub}} \)
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 necessarily
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 rendezvous radar roll and shaft, and cos and startracker horizontal and vertical angles. \(
tan(\theta) = \frac{r \cdot ua}{r \cdot ub}
\)

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 necessarily 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)
  ccash
  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, pohrat

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 pohrat (g,gr,gv)

DESCRIPTION
xradar
Computes rendezvous radar observations and partials.

\begin{itemize}
\item \textbf{g} \quad \text{Computed observation (output)}
\item \textbf{gr} \quad \text{Partial of observation with respect to target vehicle inertial position (output)}
\item \textbf{gv} \quad \text{Partial of observation with respect to target vehicle inertial velocity (output)}
\end{itemize}

range
Computes range.

ranrat
Computes range rate.

shaft
Computes shaft.

sftrat
Computes shaft rate.

roll
Computes roll.

rolrat
Computes roll rate.
xradar.b(3) (Obs) xradar.b(3)

trnnion
Computes trunnion.

trnrat
Computes trunnion rate.

pitch
Computes pitch.

pcnrat
Computes pitch rate.

COMMENTS
Assumes rates have been compensated for observing frame angular motion.
xtrack.b(3) (Obs) xtrack.b(3)

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 independent 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, a2crv, a2ecrv, q21sto, q21egn

SYNOPSIS
subroutine q2crvz (ikurve, l, j)
integer ikurve
integer j
integer i

entry q2crv (ikurve)
entry q2ecrv (ikurve)
entry q21sto (ikurve, i, j)
entry q21egn (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
Finnishes drawing curve.

q21sto
Stores legend information.

q21egn
Displays legend.
q2draw.b(3)  (Plot)  q2draw.b(3)

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)

TRW Houston System Services  - 1 -  (printed 12/18/86)
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 device.
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
qs1ptn, qr1ptn

SYNOPSIS
subroutine qs1ptn (dotdsn)
  integer dotdsn
  entry qr1ptn (dotdsn)

DESCRIPTION
qs1ptn
  Entry points to set and reset line patterns.
    dotdsn line type (input/output)

qr1ptn
  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, iw (3)
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 another. This program extracts 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 the array itemp. n can be any number of bits, or 18, 36, or 72 for univac created records. itemp is treated as 32 bit word for current applications.

n  number of bits (input)
in  initial array (input)
iw  initial word (input)
ib  initial bits (input)
itemp temporary storage (output)
ow  output word (output)
ob  output bits (output)

COMMENTS
Must not be compiled with 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 the 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 RELBE1 product parameters.
count  countflag (output)
dindex  index for tracking intervals (input)
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(chan1,status)
int *chan1;
int *status;

void NTEOF(chan1,status)
int *chan1;
int *status;

void NTFILE(chan1,fcount,status)
int *chan1;
int *fcount;
int *status;

void NTOPEN(chan1,drive,status)
int *chan1;
int *drive;
long int *status;

void NTREAD(chan1,nobytes,buffer,status,overflow)
int *chan1;
int *nobytes;
char *buffer;
long int *status;
long *overflow;

void NTRITE(chan1,nobytes,buffer,status)
int *chan1;
int *nobytes;
char *buffer;
int *status;

void NTRW(chan1,status)
int *chan1;
int *status;

void NTRWRL(chan1,status)
int *chan1;
int *status;

void NTBLCK(chan1,fcount,status)
int *chan1;
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(chan1,status)
where all the arguments are integers. The chan1 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 occurs, 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(chan1,fcount,status)
call NTOPEN(chanl, drive, status)
call NTREAD(chanl, noabytes, buffer, status, overflow)
call NTRITE(chanl, noabytes, buffer, status)
call NTBLCK(chanl, fcount, status)

For detailed 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/rmt0 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;
  - 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 physical 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 noabytes, 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 noabytes 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, av)
character prompt
class type character
integer nv
integer iv (*)
real rv (*)
double precision av (*)

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)
av double precision value array (input/output)
NAME
dciphnr

SYNOPSIS
subroutine dciphnr (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

dciphnr
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, av, rv, str)
character p
integer l
integer iv (1)
double precision dv (1)
real rv (1)
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.
nprop.b(3)  (Propagate)  nprop.b(3)

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(fatal)
integer fatal

DESCRIPTION

driver for opening the files needed for propagation

opnprp

fatal 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 (nstep, 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
dpfm

SYNOPSIS
subroutine dpfmt (temp, dpnew)
integer temp (3)
integer dpnew (2)

DESCRIPTION
dpfm
convert 72 bit unvisc 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 (inbut,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 user's program can read Univac data using Univac subscripting.

ibit1  bit position in input(iword1) for start of move (internal & output,input,input & updated)
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

```c
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.
gstack.c(3) (Stacks) gstack.c(3)

>=0, displacement from top
< 0, displacement from bottom.
Return null pointer if out-of-range, non null otherwise.

gempty
Empty contents of stack. Set top of stack to zero. Return null pointer if
error else return begin of stack.

gfree
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
input 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 calendar 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.
etime 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 (datel,aate2,secs)
double precision datel (6)
double precision date2 (6)
double precision secs

DESCRIPTION
secnds
Determines the difference in seconds between two calendar times.
dateln 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,
seconds since 1970 and days since 1970. These variables are set upon
calling 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 current 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

TRW Houston System Services - 1 - (printed 12/18/86)
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 Jan 1BC 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, i.e., 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, 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;
in 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, prtrdate, fprrthms, sprthms, prrthms, fprrctime,
sprtctime, prrtcime, fprsec, sprtsec, prrsec - display formatted time
information

SYNOPSIS

void fprtdate(file, date)
FILE *file;
CALDATE *date;

void sprtdate(s, date)
char *s;
CALDATE *date;

void prtrdate(date)
CALDATE *date;

void fprrthms(file, hms)
FILE *file;
HMSTIME *hms;

void sprthms(s, hms)
char *s;
HMSTIME *hms;

void prrthms(hms)
HMSTIME *hms;

void fprrtctime(file, time)
FILE *file;
CALTIME *time;

void sprrtctime(s, time)
char *s;
CALTIME *time;

void prrtctime(time)
CALTIME *time;

void fprrtsec(file, sec)
FILE *file;
double sec;

void sprrtsec(s, sec)
char *s;
double sec;

void prrtsec(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 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.
timeprint.c(3)

prtdate
  Formatted print of date to stdout

fprthms
  Formatted print of hms to file in hr:min:sec format.

sprrthms
  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.

sprrtctime
  Formatted print of calendar time to specified string.

prtctime
  Formatted print of calendar time to stdout

fprtsec
  Formatted print of nms to file in hr:min:sec format.

sprrtsec
  Formatted print of nms to string in hr:min:sec format.

prrtsec
  Formatted print of nms 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)