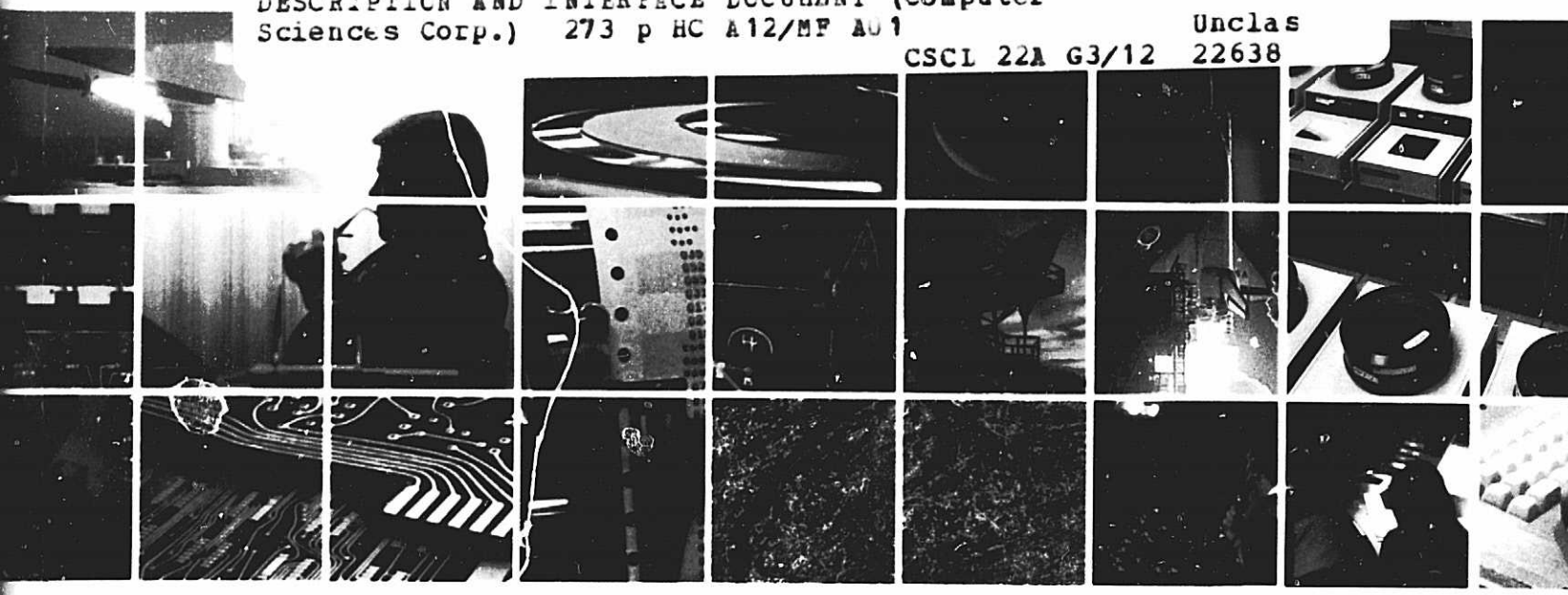
CSC/SD-84/6055

FLIGHT EXPERIMENT DEMONSTRATION SYSTEM (FEDS) FUNCTIONAL DESCRIPTION AND INTERFACE DOCUMENT

Prepared for
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Goddard Space Flight Center
Greenbelt, Maryland

CONTRACT NAS 5-27888
Task Assignment 420

DECEMBER 1984



CSC

COMPUTER SCIENCES CORPORATION

NASA CR 175290

CSC/SD-84/6055

FLIGHT EXPERIMENT DEMONSTRATION SYSTEM (FEDS)

FUNCTIONAL DESCRIPTION AND

INTERFACE DOCUMENT

Prepared for

GODDARD SPACE FLIGHT CENTER

By

COMPUTER SCIENCES CORPORATION

Under

Contract NAS 5-27888
Task Assignment 420

Prepared by:

Roger C. Belcher 12/18/84
R. Belcher Date

Dwight E. Shank 12/18/84
D. Shank Date

Approved by:

Sharon R. Waligora 12/18/84
S. Waligora Date
Technical Supervisor

Donald Page 12-19-84
G. Page Date
Functional Area Manager

ABSTRACT

This document presents a functional description of the Flight Experiment Demonstration System (FEDS) and of interfaces between FEDS and external hardware and software. FEDS, developed at the Goddard Space Flight Center (GSFC) System Technology Laboratory (STL), Code 550, is a modification of the Automated Orbit Determination System (AODS) developed at the STL during 1981 and 1982. FEDS has been developed to support a ground demonstration of microprocessor-based onboard orbit determination.

This document provides an overview of the structure and logic of FEDS and details the various operational procedures to build and execute FEDS. It also documents a microprocessor interface between FEDS and a TDRSS user transponder and describes a software simulator of the interface used in the development and system testing of FEDS.

TABLE OF CONTENTS

Section 1 - Introduction and System Overview. 1-1

1.1 Overview 1-1

1.1.1 Hardware Configuration. 1-2
1.1.2 Software Configuration. 1-2
1.1.3 Data Flow 1-6
1.1.4 Time Systems in FEDS. 1-11
1.1.5 Data Collection 1-13

1.2 Description of This Document 1-14

Section 2 - FEDS Executive (EXEC) Task. 2-1

2.1 Basic Executive Control Techniques 2-3

2.1.1 Use of RSX-11M(S) System Priorities . . . 2-3
2.1.2 Timeslicing 2-5
2.1.3 Use of Global System Event Flags. 2-7

2.2 Functional Flow of the Executive 2-7

2.2.1 FEDS Initialization 2-12
2.2.2 Control Command Processing. 2-13
2.2.3 Task Scheduling 2-18
2.2.4 End-of-Task Processing. 2-24
2.2.5 Activity Log Generation 2-27

2.3 Error Handling 2-28
2.4 Fast-Timing Feature. 2-29

Section 3 - Information Processing Tasks. 3-1

3.1 Data Capture (DATCAP) Task 3-1
3.2 Input Processor (INPPRO) Task. 3-5
3.3 Data Preprocessor (PREPRO) Task. 3-12

3.3.1 TDRS Orbit File Generation. 3-14
3.3.2 TDRS Orbit File Update. 3-17
3.3.3 TDRS Maneuver Recovery. 3-18
3.3.4 Observation Data Preprocessing. 3-18

3.4 Data Manager (DATMGR) Task 3-19

3.4.1 TDRS Orbit File Management. 3-23
3.4.2 Observations File Management. 3-24

3.5 Output Processor (OUTPRO) Task 3-26

TABLE OF CONTENTS (Cont'd)

Section 4 - Computational Tasks 4-1

4.1 Orbit Propagator (ORBIT) Task. 4-2
4.2 State Predictor (STATE) Task. 4-9
4.3 Doppler Predictor (DOPPLE) Task. 4-13
4.4 Estimator (ESTIM) Task 4-18
4.5 Observation Modeling (OBSMDL) Task 4-22

Section 5 - Communications Box. 5-1

5.1 Communications Box Hardware. 5-1
5.2 Communications Box Interface Functions 5-1
5.3 Communications Box Operation 5-4
5.4 Communications Box Simulator 5-6

Section 6 - System Construction and Operation Guidelines

6.1 Operational Configurations 6-1
6.2 System Construction. 6-4
6.3 System Operation 6-5

Appendix A - External Interface

Appendix B - Output Message Formats

Appendix C - Data Packet Descriptions

Appendix D - FEDS Update Procedures and Command Files

Appendix E - Summary of FEDS Requirements

References

LIST OF ILLUSTRATIONS

Figure

1-1	Hierarchy of FEDS Tasks.	1-3
1-2	FEDS Data Flow	1-8
2-1	Baseline Diagram of Exec	2-2
2-2	EXEC Data Flow	2-9
3-1	Baseline Diagram of DATCAP	3-2
3-2	DATCAP Data Flow	3-3
3-3	Baseline Diagram of INPPRO	3-6
3-4	INPPRO Data Flow	3-9
3-5	Baseline Diagram of PREPRO	3-15
3-6	PREPRO Data Flow	3-16
3-7	Baseline Diagram of DATMGR	3-20
3-8	DATMGR Data Flow	3-22
3-9	Baseline Diagram of OUTPRO	3-27
3-10	OUTPRO Data Flow	3-28
4-1	Baseline Diagram of ORBIT.	4-5
4-2	Functional Block Diagram of ORBIT.	4-6
4-3	ORBIT Data Flow.	4-7
4-4	Baseline Diagram of STAPRE	4-10
4-5	STAPRE Data Flow	4-11
4-6	Baseline Diagram of DOPPRE	4-16
4-7	DOPPRE Data Flow	4-17
4-8	ESTIM Data Flow.	4-20
4-9	Baseline Diagram of ESTIM.	4-21
4-10	Baseline Diagram of OBSMDL	4-24
4-11	OBSMDL Data Flow	4-26
5-1	Communications Box Block Diagram	5-2
5-2	Transponder Interface Menu	5-5
5-3	SIMCB Data Flow.	5-10
5-4	Baseline Diagram for SIMCB	5-11
6-1	FEDS Demonstration Configuration	6-3
6-2	FEDS (on LSI) Communications Line Configuration, Communications Box Used.	6-7
6-3	FEDS (on LSI) Communications Line Configuration, Communications Box Simulator Used. .	6-8
6-4	Command File To Install FEDS on the PDP. . . .	6-10
6-5	FEDS (on PDP) Communications Line Configuration, Communications Box Simulator Used. .	6-11
6-6	ADSINS.CMD	6-12
6-7	ADSLSI.CMD	6-13
6-8	REMOVE.CMD	6-15
6-9	ABOFEDS.CMD.	6-15
6-10	ADSABO.CMD	6-15
6-11	REMFEDS.CMD.	6-16
6-12	ADSABO.CMD	6-16

LIST OF TABLES

Table

2-1	System Priorities of FEDS Tasks.	2-4
2-2	AODS System Event Flags.	2-8
2-3	Service Control Commands and Results	2-14
5-1	Transponder Interface Commands	5-7

SECTION 1 - INTRODUCTION AND SYSTEM OVERVIEW

This document is a functional design and interface description of the prototype version of the Flight Experiment Demonstration System (FEDS) developed at Goddard Space Flight Center's (GSFC's) Systems Technology Laboratory (STL), Code 550. The prototype FEDS demonstrates, in a laboratory environment, the feasibility of using microprocessors to perform onboard orbit determination in an automated manner with limited ground support. FEDS is supported in the laboratory environment by the FEDS Environment Simulator for Prototype Testing (ADEPT), which provides all external information required for FEDS operation and monitors FEDS performance.

1.1 OVERVIEW

FEDS fulfills the requirements specified in Appendix E, which is an updated version of those given in Reference 1. FEDS captures all data and control commands uplinked by the simulator. Based on the required time schedule, it processes the uplinked data, predicts one-way Doppler data, predicts state vector tables, and estimates and corrects the user spacecraft state using simulated observations data. Least-squares estimation is performed by a sliding batch process that uses real-time accumulated Tracking and Data Relay Satellite System (TDRSS) observations data provided by the transponder. The Communications Box serves as an interface between the transponder and FEDS. Predicted one-way Doppler data are output from FEDS through the Communications Box to the transponder. Predicted state vector tables and estimator reports are downlinked to ADEPT after they are generated. FEDS also records all status messages and error messages in an activity log that is downlinked to ADEPT either at regular time intervals or when the log is full.

Critical error messages are immediately downlinked to ADEPT to inform the user.

1.1.1 HARDWARE CONFIGURATION

FEDS is implemented on both the STL PDP-11/70 minicomputer under the RSX-11M operating system and the STL PDP-11/23 microcomputer under the RSX-11S operating system. The PDP-11/23, which is the target computer, has 256K bytes of random access memory (RAM) and no peripherals. FEDS communicates with ADEPT, which resides on the PDP-11/70, through communications lines (EIA RS-232C asynchronous interface) that are connected to terminal ports on the PDP-11/70. Two communications lines are used to communicate with ADEPT: one incoming line for receiving uplinked messages from ADEPT and one outgoing line for downlinking messages to ADEPT. The LSI version of FEDS communicates with the transponder through the Communications Box. One communications line (EIA RS-232C asynchronous interface) is used in the communication between the Communications Box and the LSI version of FEDS. Many communications lines (detailed in Appendix A.3) are used in the communication between the transponder and the Communications Box.

1.1.2 SOFTWARE CONFIGURATION

Since no peripherals are available on the PDP-11/23, all data must be stored in RAM. In addition, overlaying of tasks is impossible. Because of these factors, FEDS is composed of 11 separate tasks and 4 global COMMON areas that are installed and fixed in memory during execution. The FEDS software configuration is shown in Figure 1-1. One executive task controls the execution of the other FEDS tasks, which are divided into primary and secondary tasks.

The executive task controls the execution of FEDS. It acts as a minioperating system that allocates time slices to the primary tasks based on the data received, uplinked schedules,

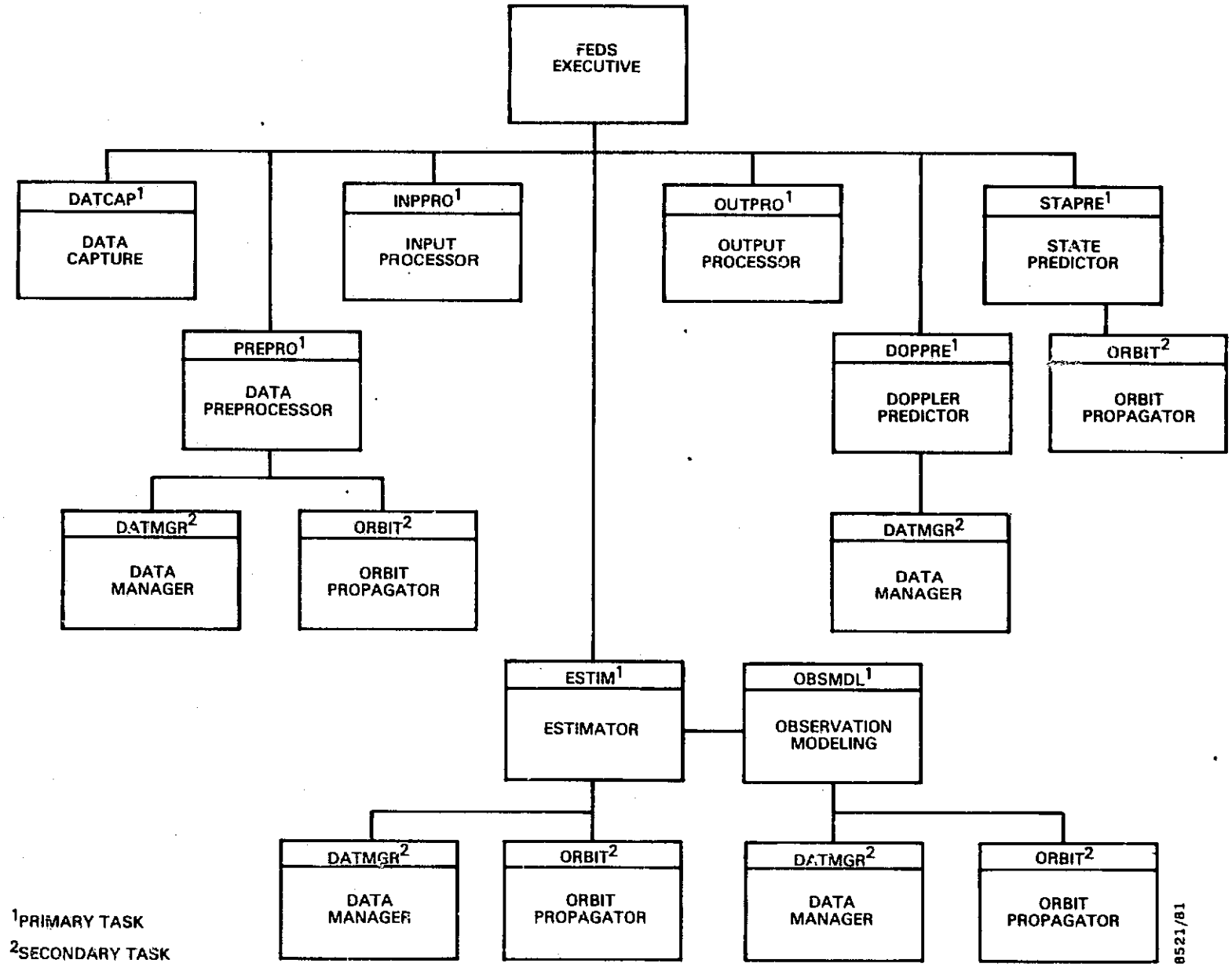


Figure 1-1. Hierarchy of FEDS Tasks

the current status of the FEDS tasks, and a predetermined set of priorities. The executive task also generates an activity log based on system status messages for the other FEDS tasks. All error messages received by the executive are loaded in the activity log. When the error is considered severe, the message is also scheduled for immediate downlink to ground control. In addition, the executive task processes all control commands received from ground control.

1.1.2.1 Primary Tasks

Primary tasks perform specific functions scheduled by the executive. Each primary task is completely controlled by the executive; the executive decides when a primary task is to be executed and determines which function the task is to perform. All communication between the executive and the primary tasks and all communication among the primary tasks are performed through global COMMON blocks. FEDS contains the following eight primary tasks:

1. Data Capture (DATCAP). This task captures all incoming messages, identifies uplinked control commands and notifies the executive, performs limited message validation, loads data and command messages into the input queue for later processing by the input processor, and loads observation messages into the observation buffer.

2. Input Processor (INPPRO). This task checks input messages for validity and stores input data in the appropriate global COMMON blocks.

3. Data Preprocessor (PREPRO). This task validates raw observation data and converts observation data to internal units, generates the Tracking and Data Relay Satellite (TDRS) orbit files, updates the TDRS orbit files based on uplinked new TDRS vectors, and performs TDRS maneuver recovery.

4. Doppler Predictor (DOPPRE). This task predicts (simulates) one-way Doppler data for a specified time interval.

5. State Predictor (STAPRE). This task generates a predicted state vector table over a specified time interval based on the current best estimate of the user spacecraft state.

6. Estimator (ESTIM). This task performs least-squares estimation by means of a sliding batch process to estimate the six components of the user spacecraft state vector and, optionally, one atmospheric drag coefficient and three coefficients of the frequency model for the one-way Doppler data.

7. Observation Modeling (OBSMDL). This task computes one-way, averaged TDRSS Doppler observations and partial derivatives as requested by the estimator. OBSMDL is an extension of the estimator because of memory restrictions and is, therefore, mainly controlled by the estimator.

8. Output Processor (OUTPRO). This task prepares the messages to be downlinked, performs the actual downlinking of the messages to ADEPT, and outputs messages to the Communications Box.

1.1.2.2 Secondary Tasks

Secondary tasks perform functions that several of the primary tasks require to perform their duties. Because of this arrangement, a secondary task is controlled by the primary task that is currently using it. Communication between a secondary task and the primary task using it is performed by SEND and RECEEV system directives. A secondary task will, however, access global COMMON blocks for uplinked constants

and control parameters. The two FEDS secondary tasks are as follows:

1. Data Manager (DATMGR). This task contains the observations file and two TDRS orbit files and performs all storage (writing) and retrieval (reading) of observation data and TDRS state vectors. It is used by the PREPRO, ESTIM, OBSMDL, and DOPPRE primary tasks.

2. Orbit Propagator (ORBIT). This task propagates the TDRS and user spacecraft state vectors using multistep integration and interpolation methods. It is used by the PREPRO, ESTIM, OBSMDL, and STAPRE tasks.

1.1.3 DATA FLOW

The 11 tasks that compose FEDS communicate with each other through the use of global COMMON blocks that are grouped by usage into four major global COMMON areas:

1. GLB1. This area contains all control information, the activity log and all information required to generate it, all global constants, the initialization table, and estimation control parameters.

2. GLB2. This area contains the observations queue, the new TDRS vectors, and the tracking and maneuver schedules.

3. GLB3. This area contains the predicted state vectors table, predicted one-way Doppler data, the differential correction (DC) summary and statistics report, and the DC residuals report to be downlinked. It also contains the global COMMON blocks that allow communication between the estimator and the observation model.

4. GLB4. This area contains the input queue.

All communication and data flow among primary tasks are performed using these global COMMON areas, and the executive

communicates with the FEDS tasks through the global COMMON only. Figure 1-2 shows the interfaces of the FEDS tasks with the global COMMON areas and with each other.

The following information is input to FEDS (see Appendix A):

- Input data uplinked by ADEPT
  - New TDRS vectors
  - Maneuver schedule
  - Tracking schedule
  - Initialization table
  - Miscellaneous constants
  - Estimation control parameters
  - Station parameters
  - Geopotential tables
  - Atmospheric density table
  - Timing coefficients
  - Experiment parameters
- Input data transmitted by the Communications Box
  - Time-tagged Doppler Observation
  - External clock time
- Control commands from ADEPT
  - START
  - STOP
  - REBOOT
  - ABORT
  - SUSPEND
  - CONTINUE
  - MARK TIME
  - RESUME
  - BEGIN FAST TIMING
  - STOP FAST TIMING
  - SET CLOCK
  - STATUS REQUEST

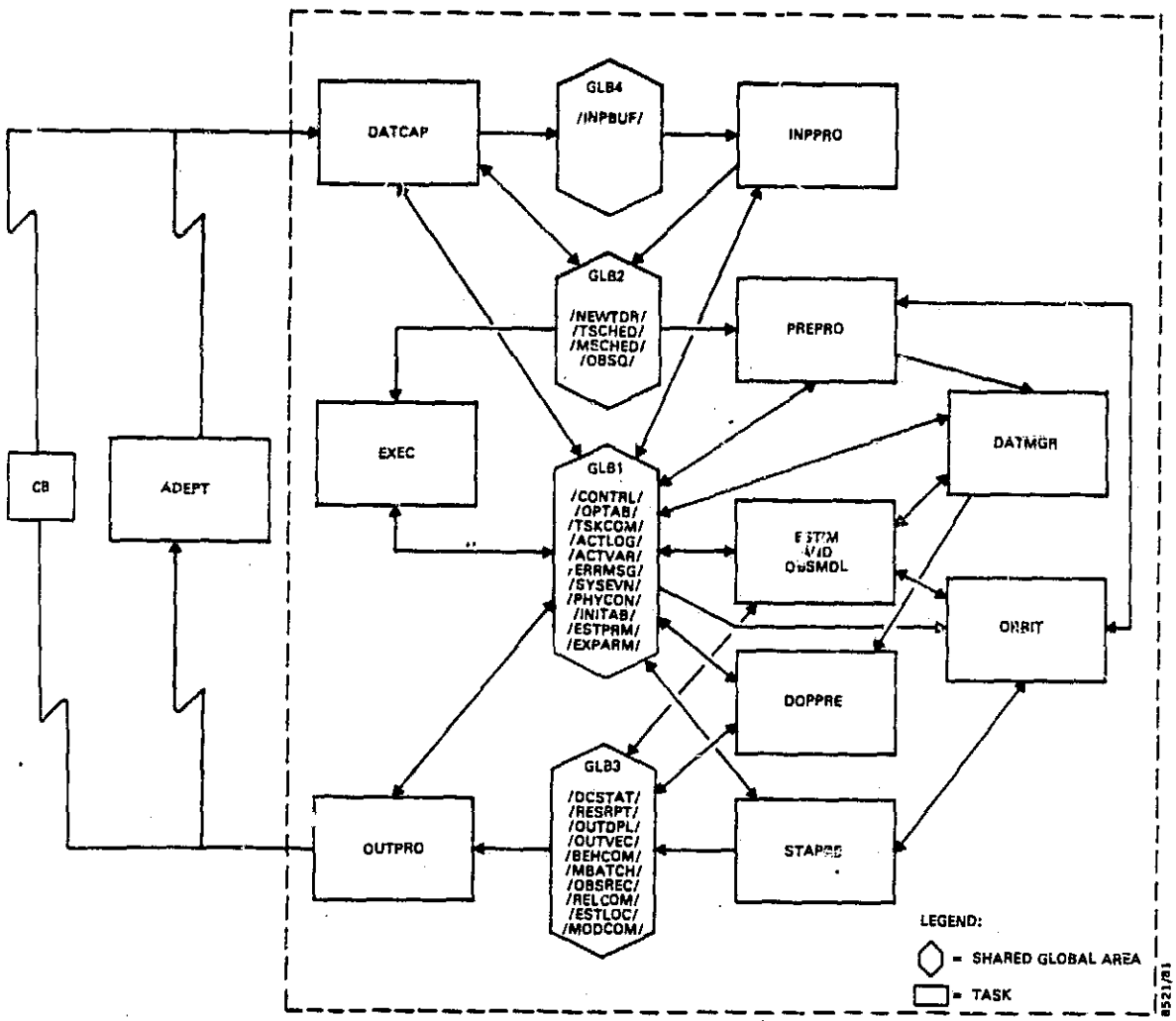


Figure 1-2. FEDS Data Flow

- Control Flags from the Communications Box
  - Carrier lock signal
  - Communications established signal

The following information is output from FEDS (see Appendix A)

- Output data downlinked to ADEPT
  - Activity log
  - Priority messages (critical error messages and idle time messages)
  - Predicted state vector tables
  - Predicted one-way Doppler frequency shift
  - DC residuals report from the estimator
  - DC summary and statistics report from the estimator
- Predicted one-way Doppler frequency shift data message output to the Communications Box
- Control messages output to the Communications Box
  - Communication initialization
  - Time request
  - Reset Doppler accumulator
  - Doppler measurement request

The flow of data through the FEDS tasks is as follows:

- FEDS Executive. EXEC uses task status information, tracking and maneuver schedules, and the system time along with knowledge of recently received uplinked data and control commands to assign functions to and schedule FEDS tasks for execution. It also maintains an activity log that is periodically downlinked to ground control. In addition, it generates critical error messages for downlink when necessary.

- Data Capture. DATCAP captures all uplinked messages on demand and loads them in the input queue for later processing. It also extracts all control commands and passes them to the FEDS executive for immediate processing. For messages from the transponder, DATCAP sets flags for the FEDS executive and loads observations into the observation buffer in /OBSQ/.

- Input Processor. INPPRO identifies all data in the input queue and loads all valid data into the appropriate global COMMON blocks where it will be used by the other tasks.

- Data Preprocessor. PREPRO preprocesses the observations data in the observations buffer in /OBSQ/, and sends it to the data manager in chronological order to be written in the observations file. PREPRO also generates and updates the TDRS orbit files based on uplinked TDRS vectors in /NEWTDR/.

- Data Manager. DATMGR reads or writes data in the observations file or in the TDRS orbit files as requested by the primary tasks. These files are stored internally in DATMGR memory.

- Estimator and Observation Modeling. ESTIM estimates the user spacecraft state and other solve-for parameters as specified in the initialization table in /INITAB/. The sliding batch estimation process is controlled by the estimation control parameters in /ESTPRM/. During estimation, ESTIM requests OBSMDL to compute observations, based on the current best estimate of the state (propagated by ORBIT), that correspond to the observations retrieved from the observations file by DATMGR. A state update is then computed and applied based on a comparison of the observed and computed values of the observations data. This process produces two output reports: a DC summary and statistics

report, /DCSTAT/, and a DC residuals report, /RESRPT/, both of which are later downlinked to ground control.

- State Predictor. Using ORBIT, STAPRE generates the predicted state vector tables based on the current best estimate of the user spacecraft state. This information is stored in /OUTVEC/ for use by DOPPRE and for downlink to ground control.

- Doppler Predictor. DOPPRE predicts one-way Doppler data based on the user spacecraft vectors in the predicted state vector table in /OUTVEC/ and on the TDRS vectors retrieved from the TDRS orbit file through DATMGR. The predicted Doppler data is stored in /OUTDPL/ for downlink to ground control and output to the Communications Box.

- Output Processor. OUTPRO downlinks the output information to ground control and the Communications Box.

- Orbit Propagator. ORBIT propagates a given state vector; optionally computes the associated partial derivatives using a multistep integrator and interpolator; and sends the results to the requesting primary task.

1.1.4 TIME SYSTEMS IN FEDS

It is important to understand the time systems used in FEDS. To reduce the number of time conversions required in FEDS, all data time tags are converted on input to an internal time system in which most computations will be performed. Time tags on data to be output are then converted back to the external time system before downlink.

All incoming data is time tagged with a Universal Time Coordinated (UTC) time in one form or another. Observation data times are in Parallel Grouped Binary time code 5 (PB5) format consisting of the last four digits of the Julian day, seconds, and milliseconds. All time tags of state vectors

and the times in uplinked schedules are input in YYMMDDHHMMSS.SS format. During input processing, all these times are converted to seconds from reference in atomic (A.1) time using the timing coefficients table. Before information is downlinked, it is returned to UTC time in YYMMDDHHMMSS.SS format.

The advantage of A.1 time is that time advances at a constant rate; that is, no discontinuities occur periodically as in the UTC time system. The advantage of keeping all times in seconds from reference is that the system (computer's) clock can be used to measure real time or simulation time as well as execution time.

Two reference times are used throughout FEDS. The simulation reference time is the time that is uplinked in the START command in YYMMDDHHMMSS.SS format, synchronized to within several seconds of the PB5 generator. The system reference time is the system clock time (YYMMDDHHMMSS.SS) when the START command is received by FEDS. These two times, which actually represent the same time in two different ways, are used to synchronize the system clock time and the simulation time. After the simulation reference time and the system reference time have been established, an offset is computed to bring the simulation time into agreement with the PB5 generator. During FEDS demonstration, the PB5 generator will be an external clock synchronized to within 1 millisecond of current UTC obtained from the Ground Spaceflight Tracking and Data Network (GSTDN). In this manner, FEDS can schedule simulation events based on the system clock.

At certain places in FEDS, times must be converted to a modified Julian date (modified by 2430000). This is made simple by computing and saving the modified Julian date of the simulation reference time. A time in seconds from

reference can be converted to a modified Julian date by simply converting it to days and adding it to the reference Julian date.

Ephemeris time (ET) is also used in the orbit propagator to compute the position of the Sun and the Moon. When necessary, the orbit propagator performs this conversion.

1.1.5 DATA COLLECTION

FEDS collects observation data to perform orbit estimation so that more observation data can be collected. For a flight system, a tracking signal would be transmitted at a constant frequency from a ground station and collected on board. The onboard system would then use the Doppler-shifted frequency record to estimate location. For a demonstration system, since the receiving transponder is stationary, the frequency transmitted will be shifted to simulate data that would be received by a satellite in a given orbit. These data come in nominal 10-minute passes. The transponder will form an observation by adding the received frequency to a constant bias and accumulating data in a nondestruct mode in a 40-bit accumulator.

The flow of control of FEDS begins with the extension of the file of predicted Doppler frequency shift 5 minutes before the beginning of a pass. Twenty seconds before the beginning of a pass the transponder accumulator is reset to zero. To accomplish this, FEDS sends a message to the Communications Box to reset the accumulator, the Communications Box sends a message to the transponder to reset the accumulator, and the transponder resets the accumulator to zero. FEDS then requests a time message and uses the subsequent reply to update the current simulation time. The Communications Box accesses the PB5 generator and sends the current time to FEDS. FEDS then begins to output predicted Doppler frequency offset. When FEDS sends a predicted Doppler

message, containing the predicted offset in the form of a frequency control word, the Communications Box passes the frequency control word to the transponder for use in signal acquisition. FEDS outputs a predicted Doppler message at a user-specified frequency.

When signal lock occurs, the Communications Box sends a signal lock message indicating that FEDS should stop transmitting predicted Doppler messages and that observation data is being collected. FEDS responds to the signal lock message with a request for a Doppler observation. When the Communications Box has received a Doppler observation request from FEDS and an accumulator reading from the transponder, it accesses the PB5 generator to obtain the current time and transmits an observation message. FEDS again responds by transmitting a request for a Doppler observation. This process will continue until the tracking signal is lost. FEDS will try to reacquire signal lock by resuming output of predicted Doppler messages until the end of the scheduled tracking pass. FEDS will then perform end-of-pass processing to prepare for the next tracking pass.

The Doppler file is initially generated by the first execution for each tracking pass of the Doppler predictor wherein 60 records of data are written to the file. The Doppler file is extended throughout the pass in a wraparound manner so that at least half of the file (30 records) is in the future. This procedure maintains the immediate availability of the predicted frequency shift for output when the tracking signal is lost.

1.2 DESCRIPTION OF THIS DOCUMENT

Sections 2, 3, and 4 of this document describe the FEDS executive, the FEDS information processing tasks, and the FEDS computational tasks, respectively. These sections also contain an overview, baseline diagrams, and data flow diagrams

grouped by function for each of the FEDS tasks. Section 5 describes the Communications Box used in FEDS. Section 6 discusses the construction and operation of FEDS.

There are five appendixes: Appendix A contains descriptions of external interfaces in FEDS. Appendix B contains output message descriptions. Appendix C contains detailed descriptions of the data packets used to send information to and receive information from secondary tasks. Appendix D contains the command files that are used for updating the system. Appendix E contains a summary of the FEDS requirements.

SECTION 2 - FEDS EXECUTIVE (EXEC) TASK

The FEDS executive task (EXEC) controls FEDS execution using the RSX-11M(S) system services. The executive controls what each FEDS primary task is doing and when each task is executing. Since FEDS is a real-time system, the FEDS executive must ensure that all schedules are met and that all time-critical functions are performed. The executive must monitor all FEDS queues, anticipate problems, and take action to avoid backlogs. The executive must also ensure that all incoming data are processed as quickly as possible by the time-consuming computational tasks. In addition, the execution must service uplinked control commands soon as they are received.

Processing priorities can change rapidly in FEDS because of changing system status and uplinked schedules and data. This rapid changing of priorities requires that the executive be able to switch quickly from one primary task to another to ensure that the highest priority function is being performed at any given time. To accomplish this, the executive uses a timeslicing technique that allows a task to execute for only a specified length of time before the executive resumes control, reevaluates priorities, and allows the same or another task to execute during the next time slice, and so on.

Because the executive is executed at the end of each time slice, it is important that it be time efficient; for this reason, all sequential executive functions are included in one large routine called EXEC. Only time conversion routines and certain activity log generation subroutines that are used repeatedly throughout the executive are called by EXEC. A baseline diagram of the FEDS executive task is shown in Figure 2-1.

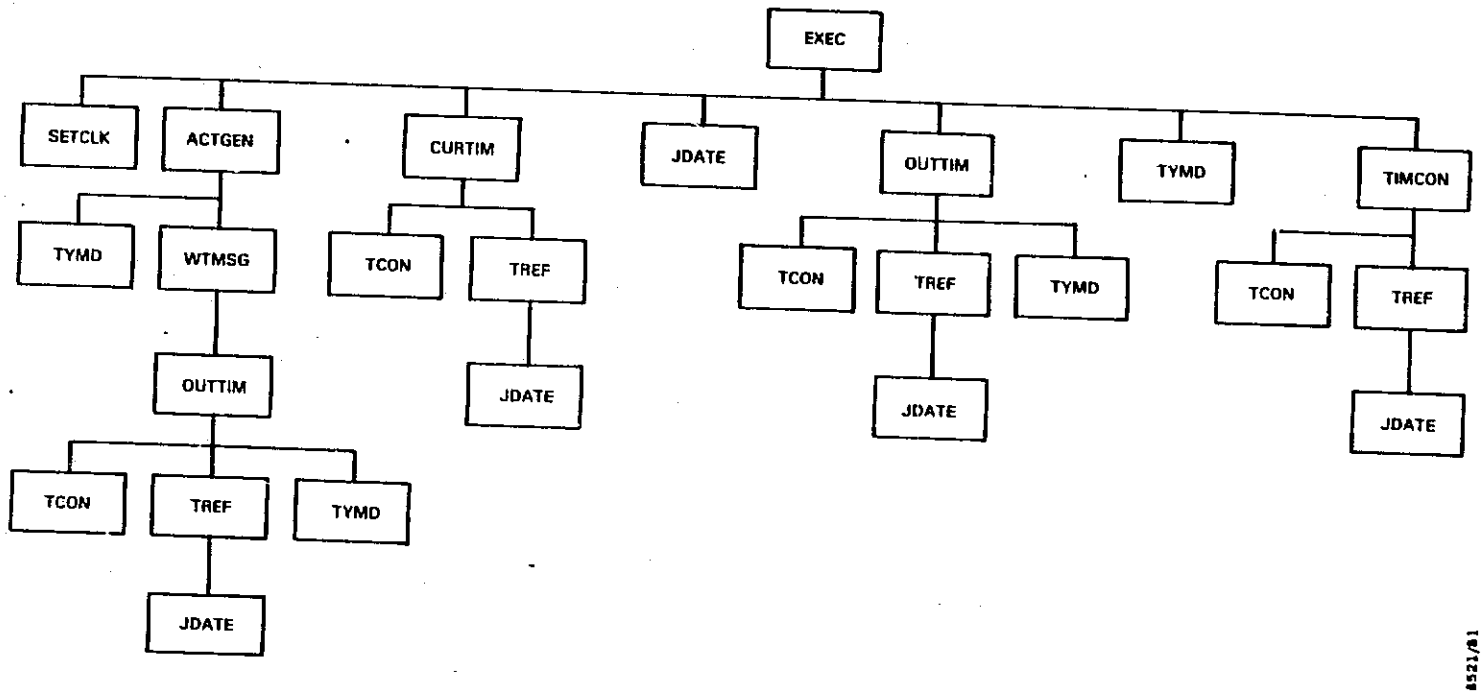


Figure 2-1. Baseline Diagram of EXEC

2.1 BASIC EXECUTIVE CONTROL TECHNIQUES

Because many FEDS functions must be performed simultaneously, the executive uses a combination of RSX-11M(S) system priorities, a basic timeslicing technique, and global system event flags to control the execution of FEDS primary tasks. Use of these techniques and the system services available under the RSX-11M(S) operating system enables the FEDS executive to give the central processing unit (CPU) to the primary task performing the highest priority function at any given time. These control techniques are described in the following subsections.

2.1.1 USE OF RSX-11M(S) SYSTEM PRIORITIES

The FEDS tasks are assigned different RSX-11M(S) system priorities as shown in Table 2-1. With the knowledge of each task's priority and the relative priorities among the tasks, the executive can change the task that is executing rather easily. Based on the priorities given in Table 2-1 and on a fundamental understanding of the RSX-11M(S) operating system, FEDS will perform in the following ways:

- The data capture (DATCAP) task, which has the highest system priority, will interrupt any other task that is executing, including the executive, when it receives an message (one that satisfies a queue input/output directive (QIO) issued by DATCAP). This assures the executive that data will be captured on demand and without any direct supervision by the executive. After receiving the message, DATCAP issues another QIO and goes into a wait to the message source state, thereby removing itself from contention for the CPU until the next message is received.

- The executive, which has the second highest system priority (70), will gain control any time one of its wait conditions (WAITFR and WFLOR directives) is satisfied as long as DATCAP is not executing at the time. If DATCAP is

Table 2-1. System Priorities of FEDS Tasks

TASK NAME	HIGH-PRIORITY LEVEL	LOW-PRIORITY LEVEL
EXECUTIVE — EXEC	70	70
PRIMARY TASKS		
DATA CAPTURE — DATCAP	80	80
INPUT PROCESSOR — INPPRO	50	1
DATA PREPROCESSOR — PREPRO	50	1
ESTIMATOR — ESTIM	50	1
OBSERVATION MODELING — OBSMDL	55	1
DOPPLER PREDICTOR — DOPPRE	50	1
STATE PREDICTOR — STAPRE	50	1
OUTPUT PROCESSOR — OUTPUT	50 (ADEPT) 65 (CB)	1
SECONDARY TASKS		
DATA MANAGER — DATMGR	60	60
ORBIT PROPAGATOR — ORBIT	60	60

9808-(83)-84

[1]THE HIGH PRIORITY IS ASSIGNED TO THE TASKS DURING TASK BUILDING.

executing, EXEC will gain control after DATCAP goes into a wait state.

- Secondary tasks (DATMGR and ORBIT) have a priority (60) between the primary tasks and the executive. They will be executed immediately whenever they are requested by a primary task and can be interrupted by either DATCAP or EXEC.

- Primary tasks other than DATCAP will execute only when other active FEDS tasks with higher priority are waiting or are suspended. If one primary task has a system priority of 50 and the others have a priority of 1, the task with priority 50 will be executed. Unlike the priorities assigned to other primary tasks, the high system priority assigned to OBSMDL is 55 rather than 50, which allows the operating system to complete housekeeping functions when OBSMDL exits before allowing the ESTIM task to continue.

Due to the time-critical nature of the information transmitted from FEDS to the Communications Box, OUTPRO will have a higher priority (65) than that of the secondary tasks when outputting to the Communications Box.

2.1.2 TIMESLICING

The FEDS time-slicing scheme is based on the rules just cited. The tasks that are time sliced are the primary tasks other than DATCAP. After these tasks are initialized, their system priority is set to 1. Then, whenever one of these tasks is to be executed, its system priority is raised to the high-priority level, allowing it to be the primary task that will execute when the higher priority tasks give up the CPU. Thus, when the executive selects a primary task to execute during the next time slice, it simply raises the system priority of that task. It then issues a system mark time (MARK TIME) directive and waits either for the primary task to complete or until the end of the time slice, whichever comes first. This allows the selected primary task to

execute. When control returns to the executive, the system priority of that primary task is lowered to 1. It should be noted that the priority of OBSMDL is raised and lowered based on the priority of ESTIM when the estimator is scheduled.

This scheme is somewhat complicated when a primary task has requested (called) a secondary task that has not yet completed when the time slice ends. For example, primary task A at priority 50 is waiting for an event flag to be set by the secondary task running at priority 60. In this situation, the same procedure is followed when the executive takes control from the secondary task. Primary task A's priority is lowered to 1. When a new primary task, B (other than OUTPRO sending data to the Communications Box), is selected for the next time slice, its priority is raised to 50, and the executive gives up control by performing a MARK TIME. This time, however, the secondary task continues executing since its priority (60) is higher than that of the selected primary task B. When the secondary task completes and sets the event flag for which primary task A was waiting, task A does not gain control because its priority is 1. The system then selects task B, which has the highest priority (50) of the tasks contending for the CPU. This procedure ensures that execution of primary tasks will not be blocked by a request for a secondary task that is already in use by another primary task. When the primary task B is OUTPRO sending data to the Communications Box, the executive will raise OUTPRO's priority to 65. OUTPRO will then gain control of the CPU and execute the completion. Upon completion of OUTPRO, the executive will regain control to schedule the next primary task.

The length of the time slice is an EXEC parameter that may be set before compilation and task building are performed. This allows the executive to be tuned to use the optimum

time slice. However, the time slice may not be changed during FEDS execution.

2.1.3 USE OF GLOBAL SYSTEM EVENT FLAGS

The RSX-11M(S) operating system has a set of global event flags available to all active tasks. A global event flag signals the occurrence of a specific event during execution. Each event flag is identified by a unique number. Global event flags allow one task to detect and control, if necessary, events occurring in other active tasks. They may be set and/or cleared by either active tasks or system services.

The FEDS executive uses these global event flags to monitor events occurring in other FEDS tasks. A list of the global event flags used by the FEDS executive and their functions is given in Table 2-2. In most cases, the executive uses these event flags as a means of regaining control after it gives up the CPU to a lower priority task.

2.2 FUNCTIONAL FLOW OF THE EXECUTIVE

The communication and the data flow between EXEC and the other FEDS tasks are shown in Figure 2-2. FEDS execution begins when the FEDS executive is started. The executive first performs an initialization procedure that includes initializing local variables that will be used to perform task scheduling and the startup and initialization of all other FEDS tasks except DATCAP (see Section 2.2.1). After each primary task is initialized, its system priority is lowered. The executive then starts DATCAP and directs it to perform initialization and to accept only the START command from ground control and Communications Box messages. The executive then directs OUTPRO to send the Communications Box an initialization message and waits for DATCAP to set event flag IFLAG7, indicating that communication with the Communications Box has been established. The executive will again

Table 2-2. **FEDS** System Event Flags

LOGICAL NAME	ACTUAL EVENT FLAG NUMBER	FUNCTION	SET BY	CLEARED BY
IFLAG5	43	WHEN SET, IT SIGNALS THAT THE EXECUTING PRIMARY TASK HAS COMPLETED ITS ASSIGNED FUNCTION.	PRIMARY TASK	EXEC
IFLAG6	44	WHEN SET, IT SIGNALS THAT THE TIME SLICE HAS EXPIRED.	RSX-11M(S)	EXEC
IFLAG7	45	WHEN SET, IT SIGNALS THAT A CONTROL COMMAND OR MESSAGE FROM THE COMMUNICATIONS BOX HAS BEEN RECEIVED.	DATCAP	EXEC
IFLG10	46	WHEN SET, IT INDICATES THAT THE DATA MANAGER IS NOT ACTIVE; WHEN CLEAR, IT INDICATES THAT THE DATA MANAGER IS ACTIVE.	DATMGR	DATMGR
IFLG11	47	WHEN SET, IT INDICATES THAT THE ORBIT PROPAGATOR IS NOT ACTIVE; WHEN CLEAR, IT INDICATES THAT THE ORBIT PROPAGATOR IS ACTIVE.	ORBIT	ORBIT

8521/81

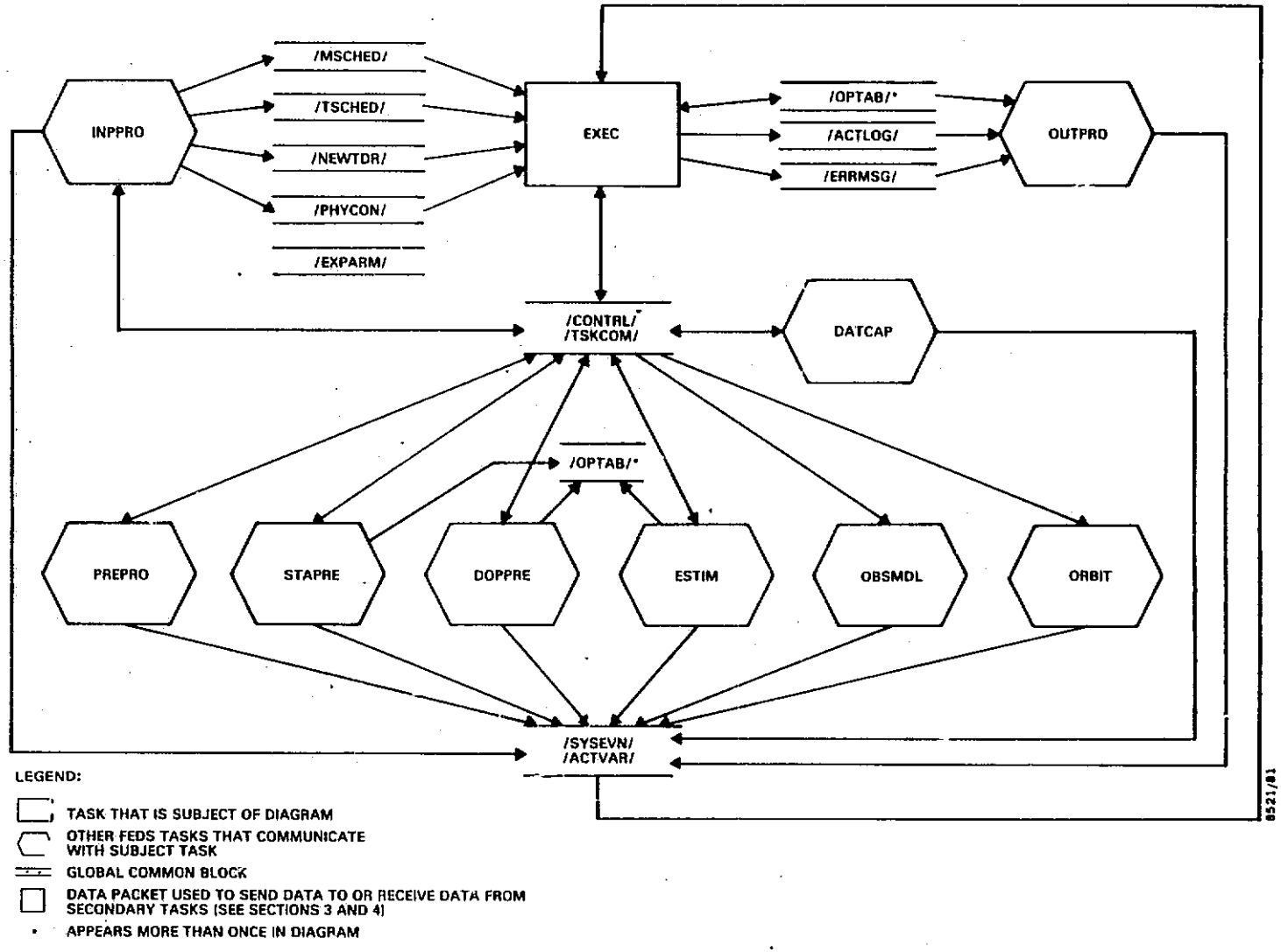


Figure 2-2. EXEC Data Flow

wait until DATCAP sets IFLAG7, indicating that a control command (in this case, the START COMMAND) has been received. At this point, if a command has been received, the executive performs the functions dictated by the control command (see Section 2.2.2). After control command processing has been completed, the executive calls ACTGEN to enter a message in the activity log (see Section 2.2.5) about the control command processed. From this point on, DATCAP will execute asynchronously, taking control when a message is received, storing it in the input queue, and then waiting for another uplinked message.

Next, EXEC calls CURTIM to obtain the current time in seconds from reference. It then schedules tasks based on the current time, the uplinked tracking and maneuver schedules, FEDS control flags and parameters, and the FEDS output table. When the primary task that is to execute during the next time slice and the function it will perform have been determined (see Section 2.2.3), EXEC checks to see whether a command has been received. If so, EXEC goes back to command processing, responds to the command, and performs task scheduling as described above.

If no control command is present and if a primary task has been selected, the executive proceeds to transfer control to the primary task. EXEC does this by raising the system priority of the selected primary task as described in Section 2.1.2. EXEC then clears event flag IFLAG5 and issues a MARK TIME system directive. This effectively sets a timer for the time slice, whose length is selected from whichever is the larger: the default time slice or the time until output to the Communications Box is scheduled. Next, EXEC gives up the CPU by waiting until one of three event flags is set. IFLAG5 will be set by the primary task if it completes its function before the time slice is over; IFLAG6

will be set by the RSX-11M(S) system whenever the time slice has expired; and IFLAG7 will be set by DATCAP if a control command or Communications Box message is received. The executive will regain control when at least one of these event flags is set.

When EXEC regains control, it tests all three event flags to see which condition(s) caused it to regain control. If the time slice has not expired, it is canceled. At this point, EXEC checks to see whether the primary task that was executing was the input processor (INPPRO). If so, EXEC checks whether INPPRO was interrupted in the middle of processing a block of data (BLKFLG is true). If this is the case, EXEC directs INPPRO to complete processing that block of data and waits for it to return control to EXEC (IFLAG5 is set). This prevents a mixing of old and new data in global COMMON blocks. At this point, the system priority of the primary task is lowered. Next, EXEC calls ACTGEN to record status and error messages from the primary task in the activity log. At this time, any severe error messages that are to be entered in the activity log (see Section 2.2.5) are also downlinked to ground control.

EXEC then continues to determine why it regained control. If a control command or Communications Box message reception occurred (IFLAG7 is set) and if the primary task did not complete its function (IFLAG5 is clear), EXEC transfers control to the command processing section (after clearing IFLAG7) and proceeds as described above.

If, however, the primary task completed its assigned function or if an error occurred in the primary task (IFLAG5 is set), EXEC performs end-of-task processing (see Section 2.2.4). This includes performing FEDS housekeeping functions, clearing the primary task's directive (IDIR(I))

if the primary task has removed itself from the task scheduling list (IACT(I)=0), and setting the primary task's return flag (IRET(I)) to zero. EXEC then goes to the command processing section and begins the cycle again.

If no primary task is selected for execution during the next time slice, EXEC checks for idle time or a stop condition. If a STOP command has been received and if there is no more data to process, EXEC directs the output processor (OUTPRO) to downlink the activity log and then to downlink the end-of-simulation message. EXEC then waits until DATCAP receives a START command at which time processing will resume with command processing. However, if a STOP command has not been received, EXEC finds the time of its next scheduled event and computes the amount of idle time until that event. When the fast-timing option is on and an idle time message has not already been sent, EXEC creates an idle time message, directs OUTPRO to downlink it immediately, and waits until it has been completed. EXEC then transfers control to the command processing section and the cycle begins again.

2.2.1 FEDS INITIALIZATION

On initiation, EXEC performs an initialization procedure, which initializes all local variables used in the executive. Event flags IFLG10 and IFLG11 are set to indicate that the data manager task, DATMGR, and the orbit propagator task, ORBIT, are not executing. Each primary task (except DATCAP) is then started up and directed to perform initialization. To do this, the executive clears event flag IFLAG5, requests the primary task by name (REQUEST directive), and waits for IFLAG5 to be set by the primary task to indicate that it has finished initialization. This effectively suspends the executive and allows the primary task to execute.

When control returns to EXEC, the primary task's system priority is lowered. This is repeated for each primary task.

Next, ORBIT is requested and directed to perform initialization in the same manner as primary tasks. Since ORBIT is a secondary task, its unique event flag, IFLG10, is used to indicate that ORBIT has finished initialization. Unlike the primary tasks, ORBIT will exit after performing initialization. This is the only direct interface that the executive has with a secondary task.

At this point, DATCAP is requested and directed to perform initialization and to accept only a START command from ground control. The executive then directs establishment of communication with the Communications Box and waits for DATCAP to set event flag IFLAG7 to indicate that a START command has been received.

2.2.2 CONTROL COMMAND PROCESSING

When a control command is received by DATCAP, the executive immediately gains control through IFLAG7. The control commands are processed according to the FEDS requirements given in Reference 1. The function of each control command is shown in Table 2-3. The executive responds to each specific control command as follows:

1. START command
    a. Sets the data capture directive to accept all valid uplinked messages
    b. Clears local flag ISTOP to allow FEDS processing to begin
    c. Sets the simulation reference times from the uplinked simulation reference time in the command

Table 2-3. Service Control Commands and Results

COMMAND \ ACTION	OVERALL ACTION	INPUT	OUTPUT	COMPUTATIONAL TASKS	DATA	CODE
REBOOT	CONTROL GIVEN TO BOOT ROUTINE	LOST	LOST	LOST	LOST	LOST
ABORT	AODS ABORTED	ONLY START COMMAND ACCEPTED	ACTIVITY LOG DOWNLINKED	STOPPED ABRUPTLY	LOST	KEPT
STOP	DATA PROCESSING TERMINATED	ONLY START COMMAND ACCEPTED	OUTPUT COMPLETED NORMALLY	COMPLETED NORMALLY	LOST	KEPT
START	DATA PROCESSING INITIATED	ALL DATA TYPES ACCEPTED	STANDARD OUTPUT INITIATED	REGULAR PROCESSING INITIATED	COLLECTED	NOT AFFECTED
SUSPEND	CURRENT COMPUTATIONAL PROCESSES SUSPENDED	ALL DATA TYPES ACCEPTED	STANDARD OUTPUT CONTINUES	SUSPENDED	KEPT	KEPT
CONTINUE	SUSPENDED COMPUTATIONAL PROCESSES RESUMED	ALL DATA TYPES ACCEPTED	STANDARD OUTPUT CONTINUES	RESUMED IF NOT AFFECTED BY INPUT; RESTARTED IF AFFECTED BY INPUT	KEPT	KEPT
STATUS REQUEST	ACTIVITY LOG OBTAINED	NOT AFFECTED	ACTIVITY LOG DOWNLINKED	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED
SET CLOCK	CLOCK ADJUSTED FOR FAST TIMING	NOT AFFECTED	ADJUSTED FOR TIME CHANGE	ADJUSTED FOR TIME CHANGE	NOT AFFECTED	NOT AFFECTED
MARK TIME	ALL PROCESSING STOPPED	ONLY RESUME COMMAND ACCEPTED	STOPPED	STOPPED	KEPT	KEPT
RESUME	ALL PROCESSING RESUMED	ALL DATA TYPES ACCEPTED	RESUMED	RESUMED	KEPT	KEPT
BEGIN FAST TIMING	ALL IDLE TIME WILL BE COMPRESSED OUT	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED
STOP FAST TIMING	AODS WILL RUN AT ITS REAL-TIME PACE	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED	NOT AFFECTED

8521/81

d. Establishes the FEDS system reference time by accessing the system clock

e. Synchronizes the simulation time with the PB5 generator

f. Computes and stores the Julian date of the simulation reference time

g. Gets the current time (from reference) and defaults the first activity log downlink time

h. Begins task scheduling

2. STOP command

a. Sets the data capture directive to accept only a START command

b. Sets local flag ISTOP that will cause FEDS processing to stop (to wait for START command) after all available data is processed and currently scheduled activities have been completed

3. REBOOT command (useful in flight system only)

a. Aborts all active FEDS tasks (primary and secondary)

b. Aborts EXEC

c. Requests the system boot routine (only a dummy boot routine is available at this time)

4. ABORT command

a. Directs OUTPRO to downlink the activity log and waits until OUTPRO has completed

b. Aborts all active FEDS tasks

c. Sets local flag INITLZ to cause the executive to reinitialize and to start over the next cycle

5. SUSPEND command. Suspends computational tasks to allow uplink of constants, tables, and/or control parameters that may affect them

   a. Temporarily suspends the Doppler predictor, state predictor, and estimator (unless the estimator is performing maneuver recovery bookkeeping) by removing the appropriate tasks from the scheduling list

   b. Sets local flag SUSPEN to keep the DOPPRE, STAPRE, and ESTIM tasks from being scheduled

6. CONTINUE command

   a. Directs INPPRO to process all input in the input queue up to the CONTINUE command and waits until INPPRO is finished.

   b. If an initialization table, the estimation control parameters, a geopotential table, an atmospheric density table, and/or the station parameters have been received since suspension, EXEC aborts the estimator, sets the appropriate restart flag, and requests ESTIM. This causes ESTIM to restart the function it was performing when the SUSPEND control command was received; otherwise, EXEC allows ESTIM to continue by putting it back in the scheduling list.

   c. If geopotential tables or atmospheric density tables have been received since suspension, EXEC aborts both DOPPRE and STAPRE, sets the appropriate restart flags, and requests these tasks. This allows them to restart the functions they were performing the next time they are scheduled for execution; otherwise, EXEC

allows them to continue as before by inserting them in the scheduling list.

d. Clears local flag SUSPEN to indicate that suspension is over.

7. MARK TIME command

a. Directs DATCAP to accept only a RESUME command

b. Stores the time that the FEDS mark time began (current time from reference)

c. Sets local flag MRKTIM to indicate that FEDS is marking time

d. Waits for event flag IFLAG7 to be set to indicate that a RESUME command was received

8. RESUME command

a. If the system is not marking time, EXEC rejects the command.

b. Computes the time pad necessary to make the timespan of the mark time transparent to FEDS tasks.

c. Clears local flag MRKTIM to indicate that the mark time is over.

9. BEGIN FAST TIMING command

a. Sets local flag FAST to indicate that the fast-timing option is on

b. Sets the minimum idle time allowed in FEDS from the time in the command

10. STOP FAST TIMING command: Clears local flag FAST to indicate that the fast-timing option is off

11. SET CLOCK command (used only when fast timing is on)

    a. Increments the system time pad by the number of seconds in the command, which effectively compresses out the specified amount of idle time

    b. Gets the new current time

    c. Adjusts the activity log output time by the number of seconds in the command

12. STATUS REQUEST command: Directs OUTPRO to downlink the activity log and wait until it has completed

2.2.3 TASK SCHEDULING

The executive schedules FEDS primary tasks for execution based on a series of logical tests performed by the executive. These tests were derived from the FEDS functional requirements included in Appendix E. Although each task can perform more than one function, only one function may be scheduled for one task at one time. However, all tasks may be scheduled simultaneously.

To reduce the execution time of the executive, the scheduling logic for each task is coded so that the smallest number of logical tests is executed to determine the highest priority function that the task is to perform. In most cases, this means scheduling the lowest priority function first so that it can be overridden by a higher priority function later, when necessary.

Tasks are scheduled by setting task directive IDIR(I) (where I is the task number or ID) to the proper function number. When no function is scheduled to be performed by task I, then IDIR(I) = 0. When a task is currently being executed, only a limited set of tests will be performed to see whether the schedule should be altered for that task.

After the highest priority function for each task has been identified and the directives have been set accordingly, the corresponding IACT(I) flag is set to 1 for each task to be scheduled. The task scheduled to perform the highest priority function is then identified by using a preset table of priorities in EXEC called IPRIOR. This table contains a FEDS priority (one that has nothing to do with RSX-11M(S) system priorities) for each function that can be performed by each task. When there is more than one task with the highest priority, a round-robin scheduling technique is used by which each task is given a time slice in turn.

Once the task is identified, the task's RSX-11M(S) system priority is raised (set to the primary task execution priority). A MARK TIME system directive is then set up for the length of a time slice, and the primary task is allowed to begin or to continue executing until it finishes its assigned function (IFLAG5 is set), until the time slice has expired (IFLAG6 is set), or until a control command or Communications Box message is received (IFLAG7 is set).

The task scheduling tests performed by the executive for each primary task are described in the following paragraphs. The tests are performed in the order given. Each successive positive decision overrides the previous one for a specific task. The executive sets the system directive for the primary tasks based on the following conditions:

1. DATCAP is not scheduled by the executive in this fashion because of its asynchronous I/O function.

2. INPPRO

    a. If there is data in the input queue, EXEC directs INPPRO to process input data (IDIR(2)=1).

b. If the input queue is almost full, EXEC directs INPPRO to process input data at a higher priority (IDIR(2)=2).

c. If there is no more data in the input queue and if the directive was set otherwise, EXEC does not direct INPPRO to process input data (IDIR(2)=0).

3. PREPRO

a. If the estimator is not running and if the preprocessor is not already scheduled, the executive does the following:

(1) If an observation buffer is full, it directs PREPRO to preprocess a buffer of observation data (IDIR(3)=1).

(2) If new observations have recently been added to the observations file, it directs PREPRO to extend the TDRS orbit files to cover the next scheduled tracking pass (IDIR(3)=6).

(3) If a new TDRS vector has been received and if the corresponding TDRS file has been created and is not currently busy, it directs PREPRO to update the entire corresponding TDRS orbit file (IDIR(3)=3).

(4) If a new TDRS maneuver update vector has been received and if the corresponding TDRS orbit file has been created and is not currently busy, it directs PREPRO to update the portion of the corresponding TDRS orbit file since the last maneuver (IDIR(3)=5).

(5) If a new TDRS vector has been received and the corresponding TDRS orbit file has not been created and if a tracking schedule for Doppler prediction has been received, it directs PREPRO to generate the corresponding TDRS orbit file over a current timespan (IDIR(3)=7).

(6) If the transponder is not currently locked onto the tracking signal and the current time is more than 30 seconds past the scheduled end of the tracking pass, it directs PREPRO to preprocess all observations data in the buffer and to perform end-of-pass processing (IDIR(3)=2).

b. If the preprocessor has not already been scheduled and if it is time for a TDRS maneuver and the corresponding TDRS orbit file has been created and is not currently busy, EXEC directs PREPRO to perform the maneuver for the specified TDRS (IDIR(3)=4 and ITDRSS = IDMAN).

4. DOPPRE: If DOPPRE is not already scheduled, the executive does the following:

a. If the current time is past the scheduled Doppler prediction time and an initialization table has been received, EXEC directs DOPPRE to extend the current table of predicted one-way Doppler data (IDIR(6)=2).

b. If the above tests have been passed and DOPPRE has not been requested to predict data for the current pass, EXEC directs DOPPRE to generate a table of predicted Doppler data (IDIR(6)=1).

5. STAPRE: If STAPRE is not already scheduled, the executive does the following:

   a. If it is time to generate a state predict table (done at regular scheduled intervals) and if an initialization table is present, EXEC directs STAPRE to extend the current state vector table (IDIR(8)=1).

   b. If a new state solution has been obtained (by the estimator), EXEC directs STAPRE to generate a new state predict table based on the new state solution (IDIR(8)=2).

   c. If a new initialization table has recently been received, EXEC directs STAPRE to generate a new state predict table based on the new a priori state vector given in the initialization table (IDIR(8)=3).

   d. If it is time for a user spacecraft maneuver, EXEC directs STAPRE to generate a new state predict table based on the estimated state after the maneuver and to perform maneuver recovery housekeeping functions (IDIR(8)=4).

6. ESTIM

   a. If the estimator is not currently scheduled, the executive does the following:

      (1) If this is the first batch and the observation timespan is equal to or larger than the requested batch timespan, EXEC directs ESTIM to perform a complete estimation (IDIR(5)=1).

      (2) When there is new data in the observations file, when the timespan of the observations file is adequate for a batch,

and when estimation precomputation has been performed, EXEC directs ESTIM to finish the estimation process using the new data (IDIR(5)=3).

(3) If the estimator has not been directed to do anything else in tests 1 and 2 above and if this is not the first batch, EXEC directs ESTIM to perform estimation precomputation (IDIR(5)=2).

b. If a user spacecraft maneuver has been identified by the preprocessor during observation preprocessing and the estimator is not currently executing, EXEC directs ESTIM to perform maneuver recovery (IDIR(5)=4).

c. When the estimator is currently scheduled to perform estimation precomputation but has not started yet, if new data has been added to the observations file, and if the observation timespan is adequate, EXEC directs ESTIM to perform complete estimation (IDIR(5)=1).

7. OUTPRO

a. If any of the following tests are passed, EXEC directs OUTPRO to output the specified function code (ICODE) to the Communications Box (IDIR(7)=10).

(1) If the current time is less than 20 but more than 10 seconds before the scheduled start time of the current pass and the accumulator in the transponder has not been reset since the last pass, EXEC directs OUTPUT to send a reset accumulator message (ICODE = 1).

(2) If the current time is less than 10 seconds before the scheduled start time of the current pass and the PB5 generator has not been accessed to obtain an accurate estimate of the current time, EXEC directs OUTPRO to request a clock time message from the Communications Box (ICODE = 2).

(3) If the current time is later than the scheduled time to output a predicted Doppler frequency shift, EXEC directs OUTPRO to form and transmit a frequency control word (ICODE = 3).

(4) If the transponder is locked onto the tracking signal and all requests for observation messages have been filled, EXEC directs OUTPRO to request an observation message.

b. EXEC sets the lock flag in the output table to indicate that it is time for regular activity log downlink.

c. EXEC loops through the output table, /OPTAB/, to locate the highest priority output requested by other tasks through the table; if any are found, EXEC directs OUTPRO to downlink output tables starting with the highest priority entry (IDIR = 1, 2, 3, 4, 5, or 6).

2.2.4 END-OF-TASK PROCESSING

After a task returns control to the executive by setting IFLAG5 to indicate that it has completed its assigned task, EXEC performs end-of-task processing. This consists of a series of housekeeping functions based on the particular

primary task that executed during the time slice. The end-of-task housekeeping functions performed for each primary task are as follows:

1. DATCAP: No housekeeping is required

2. INPPRO: No housekeeping is required

3. PREPRO

   a. If PREPRO has completed end-of-pass processing (IDIR(3)=2), EXEC searches through the remainder of the tracking schedule for the next scheduled tracking interval and sets the corresponding executive scheduling parameters.

   b. If PREPRO performed TDRS Maneuver recovery (IDIR(3)=4), EXEC searches through the remainder of the maneuver schedule until it finds the next scheduled TDRS maneuver; it then updates the corresponding executive scheduling parameters.

   c. If PREPRO has just finished extending the TDRS orbit files to cover the new observation timespan (IDIR(3).EQ.6), EXEC sets local scheduling flags to indicate that all observation preprocessing has been completed and that estimation may be performed (NEWOBS = false and NEWDAT = true).

4. ESTIM

   a. If ESTIM has not yet finished (IACT(5).GT.0) but has returned only to record an error message in the activity log, no housekeeping is performed.

b. If ESTIM has just finished a complete estimation cycle (IDIR(5).EQ.1), EXEC sets the first-time estimation flag (FIRST) to false.

c. If ESTIM has just finished estimation (IDIR(5).EQ.3 or 1), EXEC sets the data ready for estimation flag (NEWDAT) to false.

d. If ESTIM has just finished processing a user spacecraft maneuver (IDIR(5).EQ.4), EXEC sets the time of the last maneuver processed by PREPRO past the end time of the simulation.

5. DOPPRE: If DOPPRE has just finished generating the first 60 predicted frequency shift records, EXEC sets the time to transmit a predicted Doppler shift to the transponder to the time of the first record in the file

6. OUTPRO: If OUTPRO has just sent a message to the transponder, EXEC does one of the following:

a. If OUTPRO sent a request for clock time, EXEC sets a flag to indicate that the clock has been accessed for this pass.

b. If OUTPRO sent a command to reset the accumulator in the transponder, EXEC sets flags indicating that the accumulator has been reset before the upcoming pass and that the transponder is not currently locked onto the tracking signal.

c. If OUTPRO has just transmitted a predicted Doppler shift to the transponder, EXEC schedules the next time to transmit a predicted Doppler shift and sets the time slice to whichever is larger: time until output of the

next predicted Doppler shift or the default time slice.

d. If OUTPRO has sent a request for an observation, EXEC sets a flag indicating that a request for data is pending and sets the time to output the next predicted frequency shift to the current time.

7. STAPRE: If STAPRE has just performed user spacecraft maneuver recovery (IDIR(8).EQ.4), EXEC searches through the remainder of the maneuver schedule to locate the next scheduled user spacecraft maneuver and sets the corresponding executive scheduling parameters

After task-specific, end-of-task processing has been performed, EXEC clears the task directive (IDIR(I)=0) for the primary task that was executing if the task has removed itself from the scheduling list (IACT(I).EQ.0). In all cases, EXEC also clears the return status (error) flag (IRET(I)=0) for the primary task.

2.2.5 ACTIVITY LOG GENERATION

Activity log generation is performed by subroutine ACTGEN. Each time messages are to be inserted in the activity log, EXEC calls ACTGEN specifying the task that has recently executed. ACTGEN then checks the status flags and return status flags for the specified task in global COMMONs /SYSEVN/ and /TSKCOM/, respectively, to identify those messages that are to be entered in the activity log.

For each message to be generated, ACTGEN stores the message number and contents that it retrieves from global COMMONs /ACTVAR/ and /CONTROL/ in local COMMON /ALMESG/. For messages that contain times as part of their contents, ACTGEN converts the times to YYMMDDHHMMSS.SS format when necessary.

Once each message has been created, ACTGEN calls WTMSG to enter the message in the activity log. WTMSG first time tags the activity log message; then the message is inserted in the next available location in the activity log, /ACTLOG/. If the activity log is full, WTMSG directs OUTPRO to downlink it immediately and waits until OUTPRO has completed the downlink.

If the critical error flag is set by ACTGEN, the message is also downlinked as a critical error message. To do this, WTMSG loads the message into global COMMON /ERRMSG/ and directs OUTPRO to downlink it immediately. WTMSG waits until OUTPRO has completed the downlink and then automatically loads another message in the activity log stating that a critical error message was downlinked.

When all indicated messages for the specified task have been generated and entered in the activity log, ACTGEN returns control to EXEC.

2.3 ERROR HANDLING

Error handling is a combined effort between the executive and the primary task where the error occurred. Errors occurring in secondary tasks are reported through the primary task that called them.

When an error occurs in a primary task, the primary task evaluates the seriousness of the error. If the task can continue but the error should be recorded, return status flag IRET(I) is set. When the executive regains control, the error message will be entered in the activity log. If the task can continue but the error should be recorded immediately or, optionally, a critical error message should be downlinked to ground control, the return status flag is set, and the primary task immediately gives up control by setting IFLAG5. However, since the primary task can continue, it

does not remove itself from the scheduling list. This guarantees that it will get another time slice to continue processing.

When the primary task cannot continue processing due to a critical error, the primary task sets the return status flag appropriately, removes itself from the scheduling list (IACT(I)=0), performs other housekeeping functions necessitated by the error, and gives up control by setting IFLAG5. When the executive regains control, it enters the error message in the activity log and, optionally, based on preset indicators, downlinks the critical error message to ground control.

2.4 FAST-TIMING FEATURE

The fast-timing feature allows a simulation case using the Communications Box simulator to run faster than real time by compressing idle time out of the simulation. When a BEGIN FAST TIMING control command is received by FEDS, the fast-timing option is turned on.

From this point until a STOP FAST TIMING control command is received by FEDS, the following procedure is performed. Whenever FEDS has nothing scheduled immediately, it finds the time of its next scheduled event. When the amount of idle time (i.e., the time between the current time and the time of the next scheduled event) in FEDS is greater than the maximum amount of idle time permitted during fast timing (uplinked in the BEGIN FAST TIMING command), FEDS downlinks the time of the next scheduled event in an idle time message to ADEPT.

When the simulator receives this message, it checks its list of scheduled events. When an urgent command or data retransmission is required, the simulator ignores the idle time message. Otherwise, if the amount of idle time (i.e., the time between the current time and the next scheduled

uplink time) in the simulator is greater than the maximum amount of idle time permitted during fast timing, the simulator sets the amount of idle time to be compressed out (Δt) to the smaller of the FEDS idle time and the simulator idle time. The simulator then moves its current time ahead to compress out the idle time and uplinks to FEDS the amount of idle time to be compressed out (Δt) in a SET CLOCK control command. When FEDS receives the SET CLOCK control command, it adjusts the onboard current time by the uplinked Δt. At this point, the idle time has been compressed out of both systems. This process is repeated each time idle time is discovered in FEDS.

SECTION 3 - INFORMATION PROCESSING TASKS

Five FEDS tasks are mainly responsible for data movement, manipulation, and/or conversion:

- Data Capture (DATCAP)
- Input Processor (INPPRO)
- Data Preprocessor (PREPRO)
- Data Manager (DATMGR)
- Output Processor (OUTPRO)

The following subsections provide a functional description of each task, including baseline diagrams and data flow diagrams.

3.1 DATA CAPTURE (DATCAP) TASK

The primary responsibility of DATCAP is to receive uplinked data from ADEPT and observations data from the Communications Box and to dispatch them from the uplink buffer to the input message queue in the FEDS system. It also screens the uplinked message for high-priority data types. If the uplinked message is a high-priority type, DATCAP dispatches it to the executive for immediate use. DATCAP has the highest system priority of all the tasks in FEDS to allow it to capture data immediately after they are uplinked. Figure 3-1 is a baseline diagram of DATCAP; Figure 3-2 shows the communication and data flow among DATCAP and other FEDS tasks.

During initialization (INIT(1)=1), the input message queue pointers and counters are initialized. DATCAP clears all global system flags to indicate that it has not yet received a command message, a sentinel record, or a data message. DATCAP also clears the input message queue flag to indicate that no messages have been received.

DATCAP issues a "ready" message to ground control by using WTQIO to signal that it is ready to receive the next uplinked

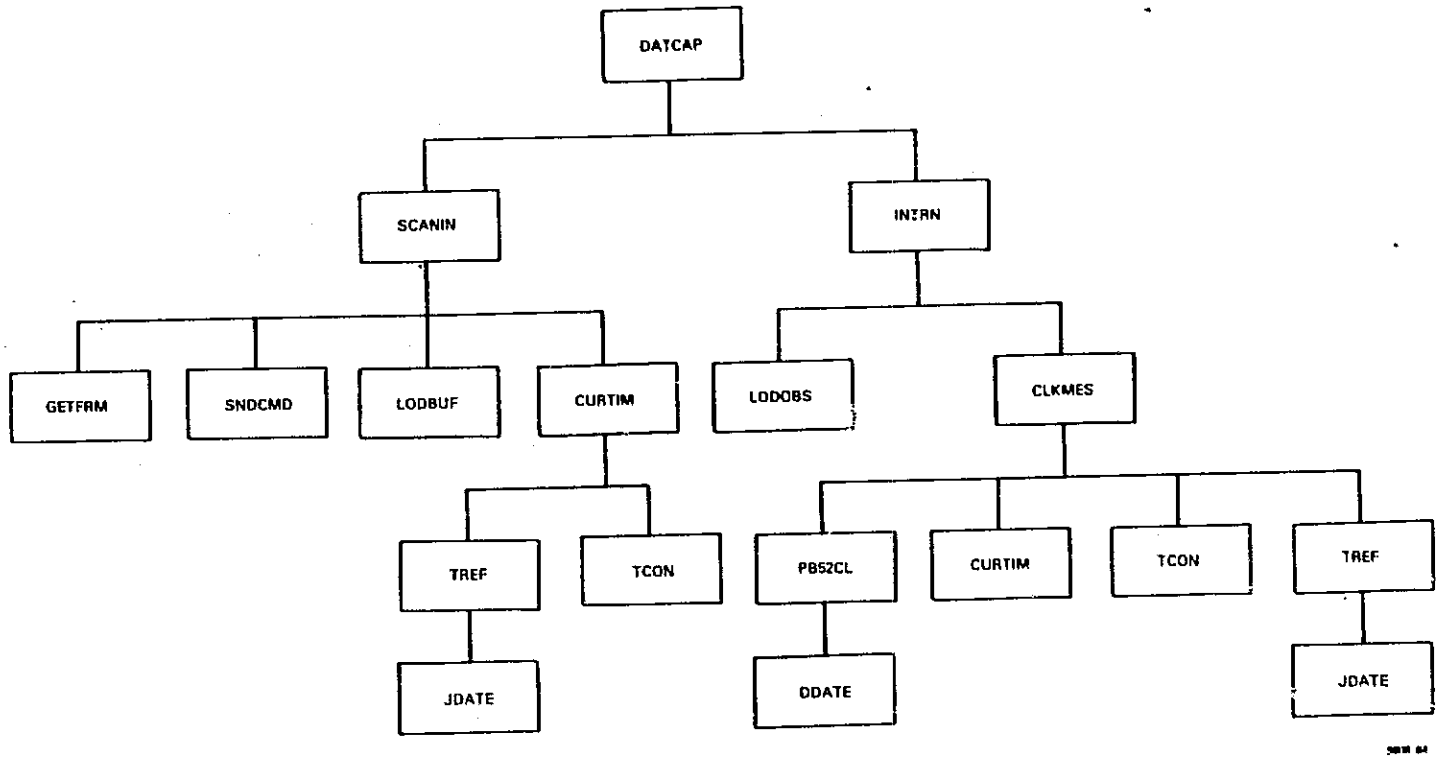


Figure 3-1. Baseline Diagram of DATCAP

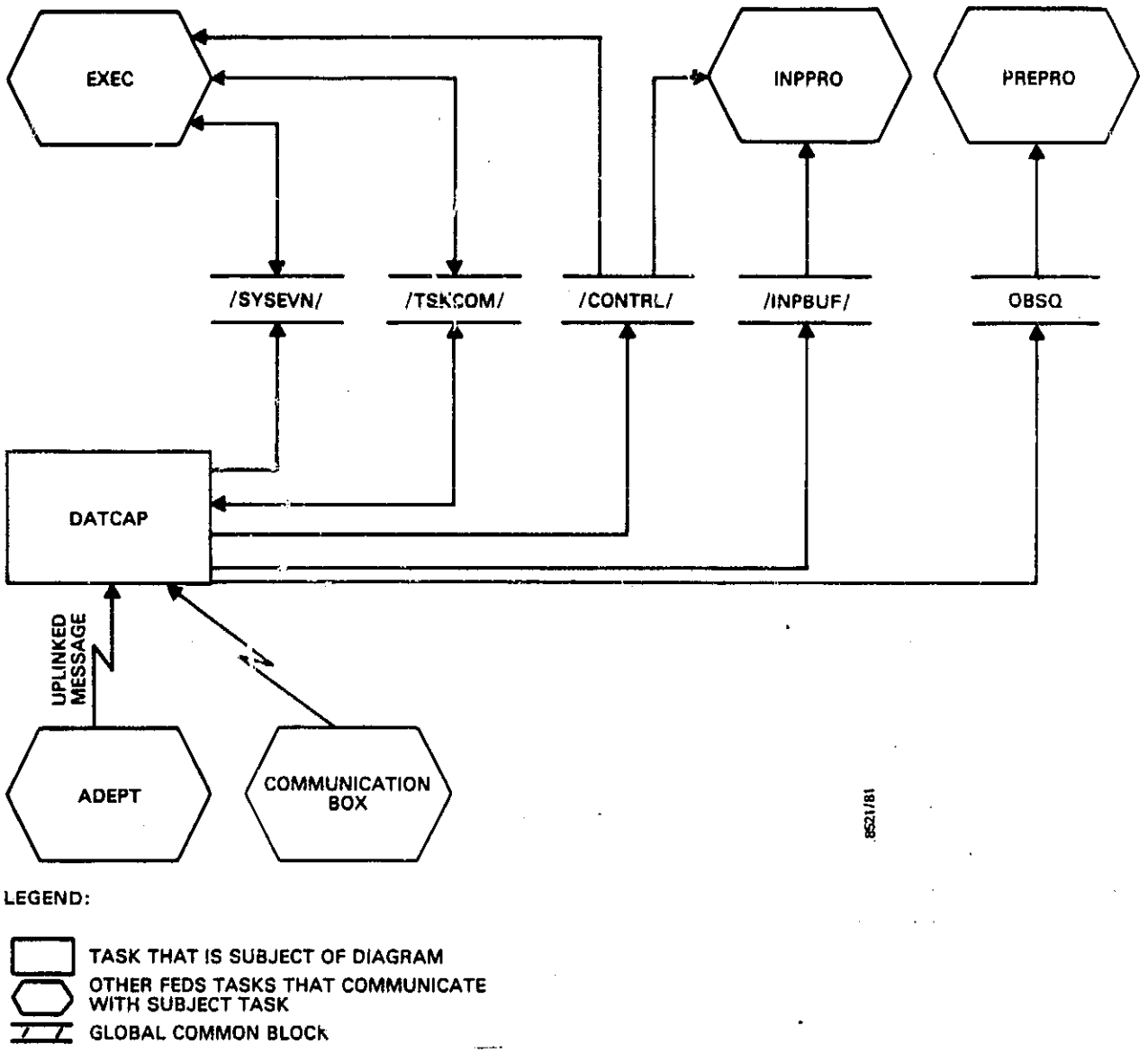


Figure 3-2. DATCAP Data Flow

message. It then issues QIOs to read the line to receive data from ADEPT and the line connecting FEDS to the Communications Box. At this point, DATCAP goes into a "wait" state and surrenders the CPU to allow other tasks to execute while it is waiting for a message. When a message arrives, DATCAP immediately takes control and receives the message because of its status as the highest priority task.

DATCAP first identifies the source of the message. If the message is from the Communications Box, DATCAP performs all processing and data storage before informing the FEDS executive that a message has been received from the Communications Box. If the message is from ADEPT, DATCAP performs preliminary validity checks on the uplinked message and sets the appropriate return status flag to inform the FEDS executive of the status of the QIO. DATCAP then proceeds to scan the uplinked message.

There are three types of messages uplinked from ADEPT: input data messages, high-priority control command messages, and the sentinel (end-of-transmission) record (see Appendix D of Reference 2). If the message is a sentinel record, subroutine SCANIN sets the global status flags to inform the FEDS executive and subroutine LODBUF loads it into the global COMMON /INPBUF/. If the synchronization characters in the uplinked message record header are bad, LODBUF loads the entire corrupted message into the input queue to keep the pointers consistent. The input processor will handle the corrupted message later. SCANIN next checks to see whether the message is acceptable.

If DATCAP has been directed by the executive to accept all incoming messages (IDIR(1)=1), it checks the type of message. If it is a control command, SCANIN first calls GETFRM to extract the command frame and then calls SNDCMD to transfer the command to the executive by means of global COMMON

/CONTRL/ and to set global event flag IFLAG7 to notify the executive that a control command was received. SCANIN next calls LODBUF to store the uplinked message (data or command) in the input queue.

If DATCAP has been directed by the executive to accept only a specific control command (e.g., a START command (IDIR(1)=2), a RESUME command (IDIR(1)=3), or a CONTINUE command (IDIR(1)=4)) and if the message contains the specified command, SCANIN calls SNDCMD to transfer the command to the executive by means of global COMMON /CONTRL/ and to set IFLAG7 to inform the executive that the command for which it is waiting has been received and is ready for processing. SCANIN then calls LODBUF to load the command in the input queue. If the message did not contain the specified command, the message is ignored, except when DATCAP is looking for a CONTINUE command, when all data messages are to be accepted.

After the message process, if the message was from ground control, DATCAP issues another "ready" message to the ground to indicate that it is ready to receive the next message. After issuing the QIO to read to the source of the previous message, DATCAP waits for the next message and the process begins again with the reception of that message. This process continues until DATCAP is aborted by the FEDS executive.

3.2 INPUT PROCESSOR (INPPRO) TASK

INPPRO is a primary task in FEDS. Its main function is to empty the input queue, /INPBUF/, and to store the input data in the appropriate global COMMON blocks. Figure 3-3 is a baseline diagram of INPPRO; Figure 3-4 shows the communication and data flow among INPPRO and other FEDS tasks.

During initialization (INIT(2)=1), INPPRO calls subroutine IPINIT to initialize all local flags and the observations queue link-list pointers. INPPRO then sets global event

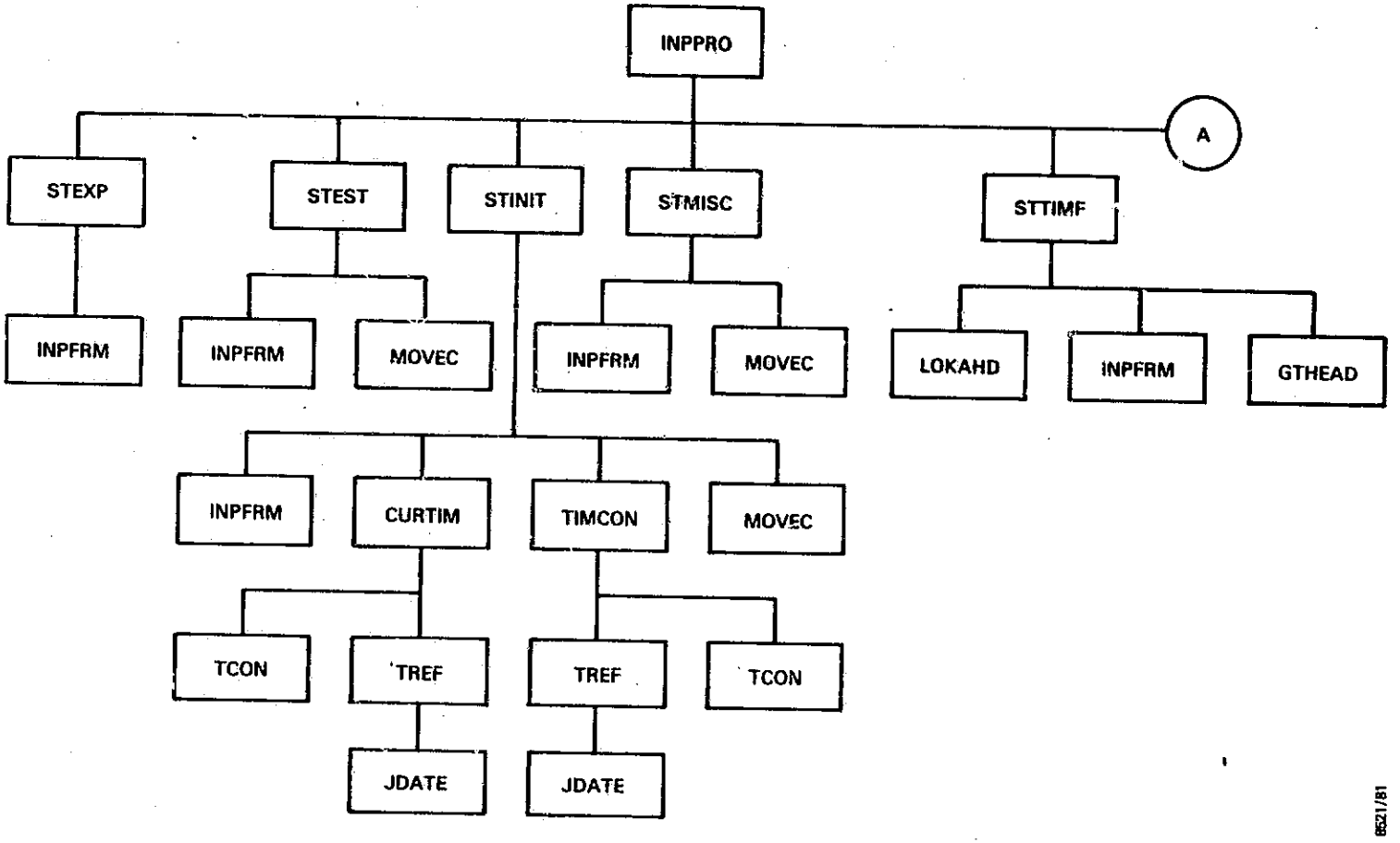


Figure 3-3. Baseline Diagram of INPPRO (1 of 3)

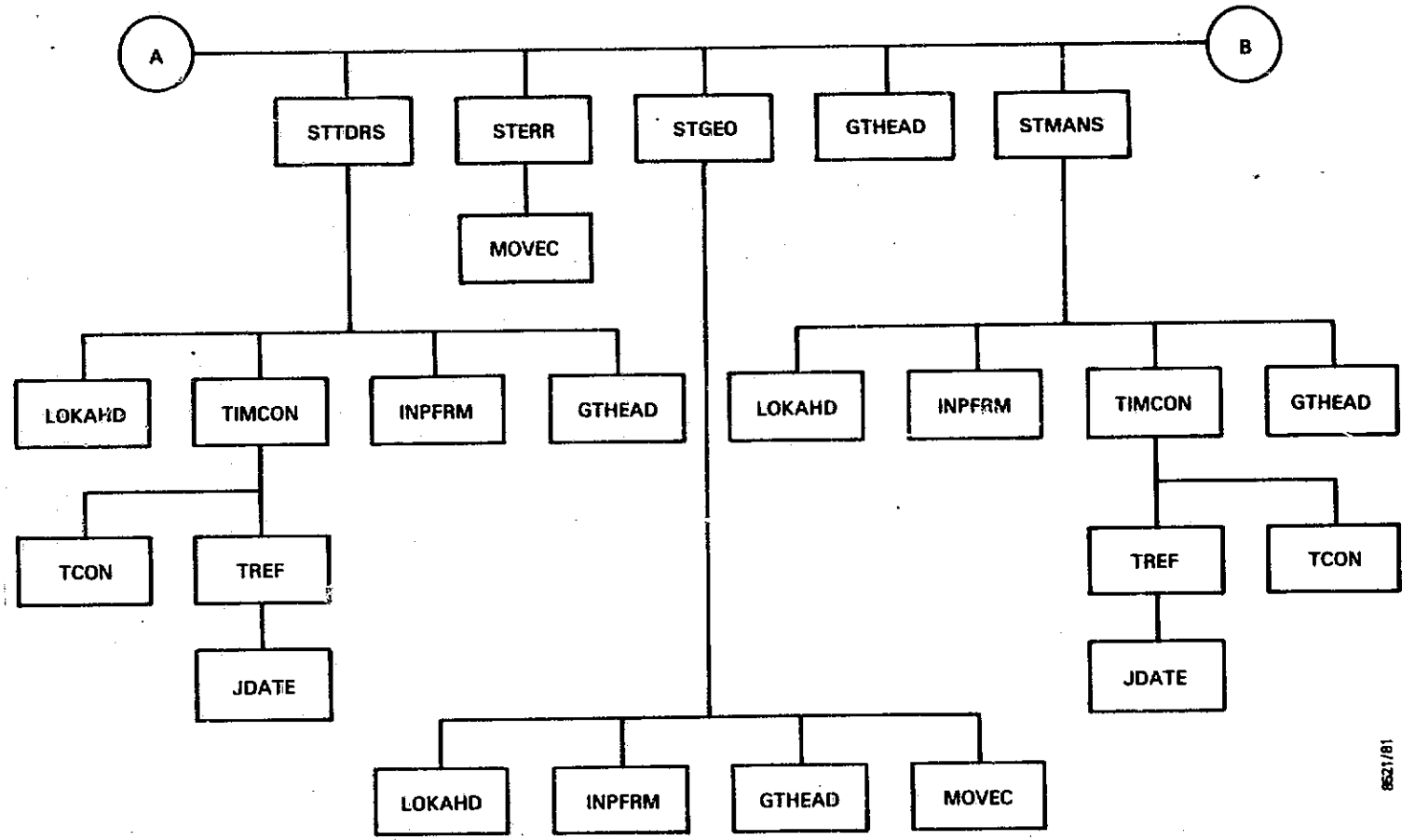


Figure 3-3. Baseline Diagram of INPPRO (2 of 3)

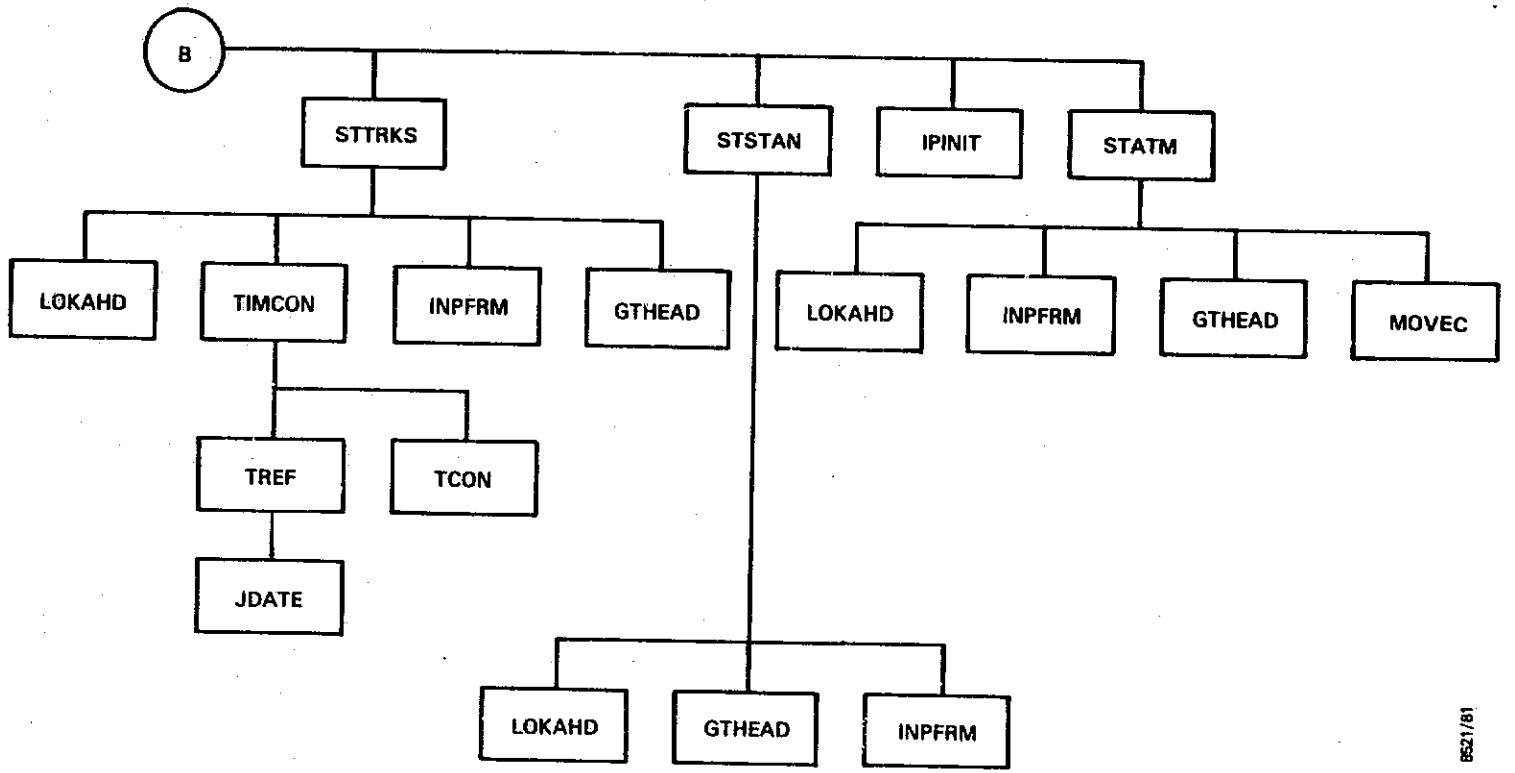


Figure 3-3. Baseline Diagram of INPPRO (3 of 3)

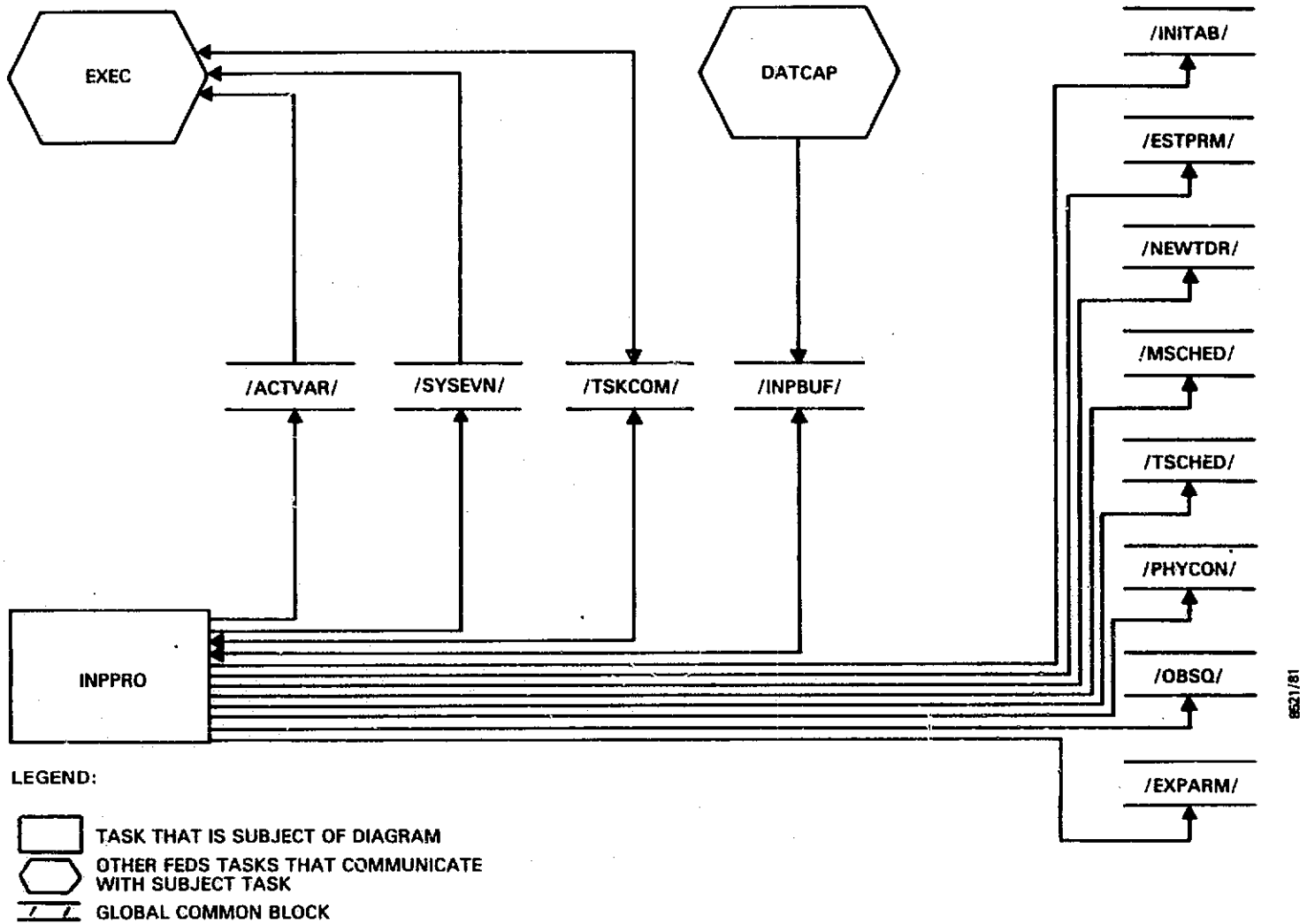


Figure 3-4. INPPRO Data Flow

flag IFLAG5 to return control to the executive. When INPPRO is given control again by the executive, it calls subroutine GTHEAD to extract the record header of the first record in the block and to perform a set of quick validity checks on the record header, including data corruption, end of transmission, validity of input type and input data indicators, transmission number, and block ID number. If an error is found by GTHEAD, INPPRO calls STERR to perform error recovery and sets IFLAG5 to give up control. Otherwise, INPPRO waits until the input queue has a complete block of data and then calls the appropriate subroutine to process the input message block. A list of the subroutines and the type of data they process follows.

Subroutine	Item Processed
STINIT	Initialization table
STEST	Estimation control parameters
STTDRS	New TDRSS vectors
STMANS	Maneuver schedule
STTRKS	Tracking schedule
STMISC	Miscellaneous constants
STSTAN	Station parameters
STGEO	Geopotential tables
STATM	Atmospheric density tables
STTIMF	Timing coefficients or constants
STEXP	Experiment parameters

The input data is stored in the proper global COMMON block where it will be used later by other FEDS tasks.

The subroutine that processes and stores the particular type of input message also validates each block of data before storing any part of it in the global COMMON block. The data block is checked for completeness and for corrupted messages. If the message block is valid, it is then checked to see whether it is acceptable input at this time, based on

the data previously received. For instance, miscellaneous constants can never be accepted after the first initialization table has been received. Geopotential tables, station parameters, and atmospheric density tables may be accepted only between a SUSPEND command and a CONTINUE command once data processing has begun. Estimation control parameters may be accepted only when the estimator/observation model is not executing or between SUSPEND and CONTINUE commands.

Once input processing has begun for a particular block of data, the processing subroutine sets BLKFLG to true to indicate that the INPPRO task is in the middle of processing a contiguous block of input messages. This is a safeguard to guarantee that an entire block of input data will be stored in a COMMON block at the same time. If an INPPRO time slice ends and if BLKFLG is true, the executive will allow INPPRO to finish processing the current block of messages before resuming control. Each time input processing of a message block is completed, BLKFLG is set to false.

If an error or an unacceptable message block is discovered during input processing of a specific message block, INPPRO calls subroutine STERR to store the information in global COMMON /ACTVAR/; this information will be used by the executive to record the error in the activity log and, optionally, to downlink a critical error message. When data corruption or an incomplete message block is detected, the record header information is supplied for the critical error message to allow ground control to identify and retransmit the erroneous message block. After STERR has stored this information, INPPRO sets IFLAG5 to return control to the executive so that it may report the error.

If no errors were found during input processing of the particular message block, INPPRO updates the input queue

pointers and counters. It then continues to the next message block and begins processing it with the call to GTHEAD described earlier.

The means by which the input processor task voluntarily gives up control depends on the input processor directive (IDIR(2)) set by the executive. If INPPRO has been directed to process all messages available in the input queue (IDIR(2)=1 or 2), INPPRO will continue processing message blocks until the input queue is empty or until an end-of-transmission record is detected. At this point, INPPRO will reinitialize the input queue pointers and counters and will report to the executive, through global parameters, on the condition that caused it to stop. It then removes itself from the scheduling list (IACT(2)=0) and sets IFLAG5 to return control to the executive.

If INPPRO has been directed to process all messages up to the CONTINUE command (IDIR(2)=3), it will continue processing until it detects the CONTINUE command. At this point, it sets the appropriate return status flag, removes itself from the scheduling list, and sets IFLAG5 to return control to the executive.

If INPPRO has been directed to finish processing the current message block only (IDIR(2)=4), INPPRO stops when it has completed the current block. INPPRO is directed to do this when it has not completed processing a message block when the time slice expires. At this point, INPPRO removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

3.3 DATA PREPROCESSOR (PREPRO) TASK

PREPRO is an independent primary task in FEDS. Each time it is given control, it is directed by the executive to perform a specified function.

During initialization (INIT(3)=1), PREPRO calls PPINIT to clear the local variables and to request the (secondary) DATMGR task and directs it to purge the observations file and the TDRS orbit files. PREPRO then sets IFLAG5 to return control to the executive.

When PREPRO regains control, it examines task directive IDIR(3) set by the executive to determine which of the following functions it is to perform:

- Preprocess the observation data, buffer full of data (IDIR(3)=1)
- Preprocess the observation data until end of data encountered, set end-of-processing flags, pass has ended (IDIR(3)=2)
- Generate new TDRS orbit files (IDIR(3)=7)
- Extend the TDRS orbit files (IDIR(3)=6)
- Update orbit file (IDIR(3)=3 or 5)
- Perform TDRS maneuver recovery (IDIR(3)=4)

To perform these functions, PREPRO uses the DATMGR and/or ORBIT secondary tasks. The communication between PREPRO and these secondary tasks is performed using the global task directive, return flag and global variables, utility subroutines VSEND and VRCEVE, and global event flags. Detailed descriptions of the methods for requesting DATMGR and ORBIT are given in Sections 3.4 and 4.1, respectively.

When PREPRO finishes its assigned function, it reports its activities to the executive by setting the global flags and by updating the global variables that will be used to generate activity log messages. PREPRO then removes itself from the scheduling list (IACT(3)=0) and sets IFLAG5 to return control to the executive.

Descriptions of the four major functions performed by PREPRO are given in the following subsections. Figure 3-5 is a baseline diagram of PREPRO; Figure 3-6 shows the communication and data flow among PREPRO and other FEDS tasks. An accompanying description of the send/receive data packets is given in Appendix C.

3.3.1 TDRS ORBIT FILE GENERATION

Two separate directives cause PREPRO to generate TDRS orbit files. These directives cause the same basic function to be performed; however, the method for computing the start and end times of the orbit files is different.

When PREPRO is directed to create the TDRS orbit files (IDIR(3)=7), the start time of the file is set to the simulation reference time and the end time is set to 10 minutes past the end of the first scheduled tracking pass. This directive is used only at the beginning of the simulation.

When PREPRO is directed to extend the TDRS orbit files (IDIR(3)=6), PREPRO sets the end time of the orbit files to 10 minutes after the end time of the next scheduled tracking pass. This directive is used after a tracking pass has ended or, possibly, when a new tracking schedule is uplinked.

When PREPRO is directed to perform either of these functions, new TDRS vectors for the requested timespan are added to all existing TDRS orbit files. If the orbit files are being created for the first time, one file is created for each unique new TDRS vector received (up to two TDRSs). Thus, all TDRS orbit files will have the same start and end times when PREPRO has completed its assigned function.

After PREPRO has determined the start and end times of the orbit files, it calls subroutine TDRORB to generate the specified TDRS orbit file. PREPRO provides the start time, end

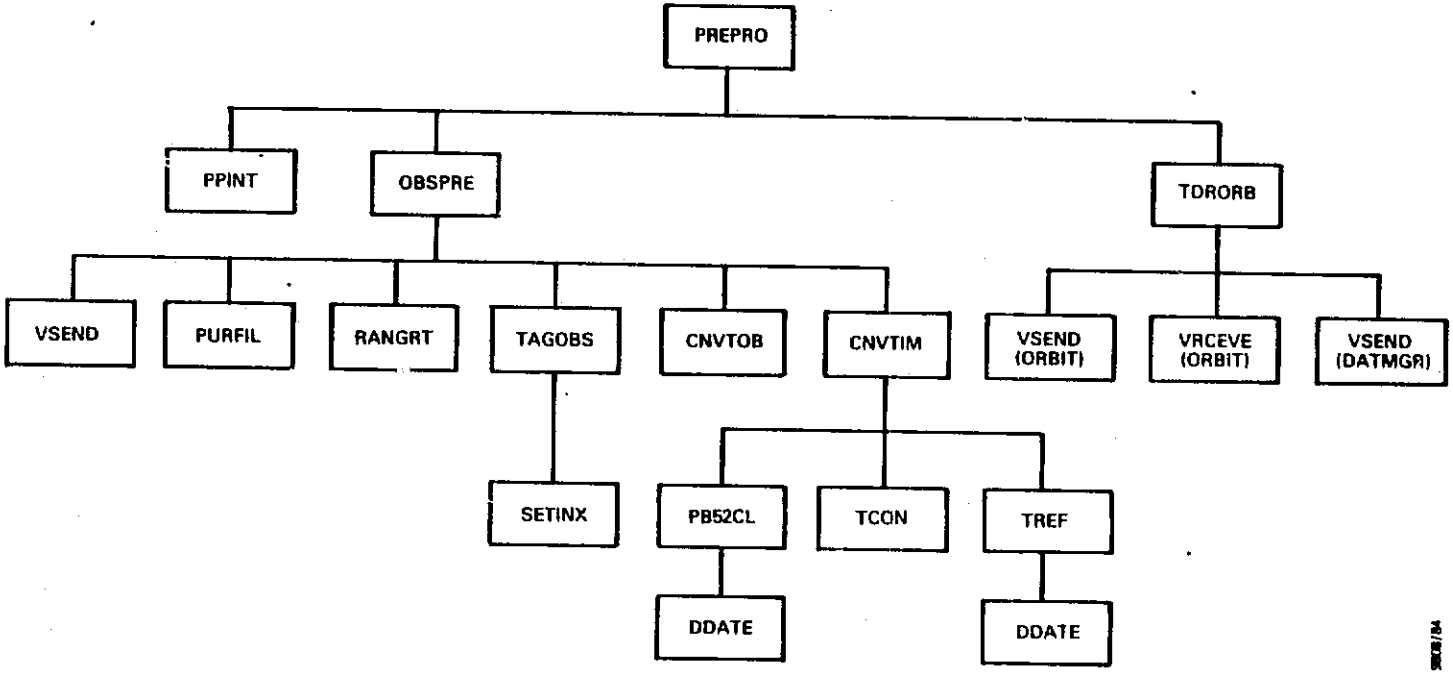


Figure 3-5. Baseline Diagram of PREPRO

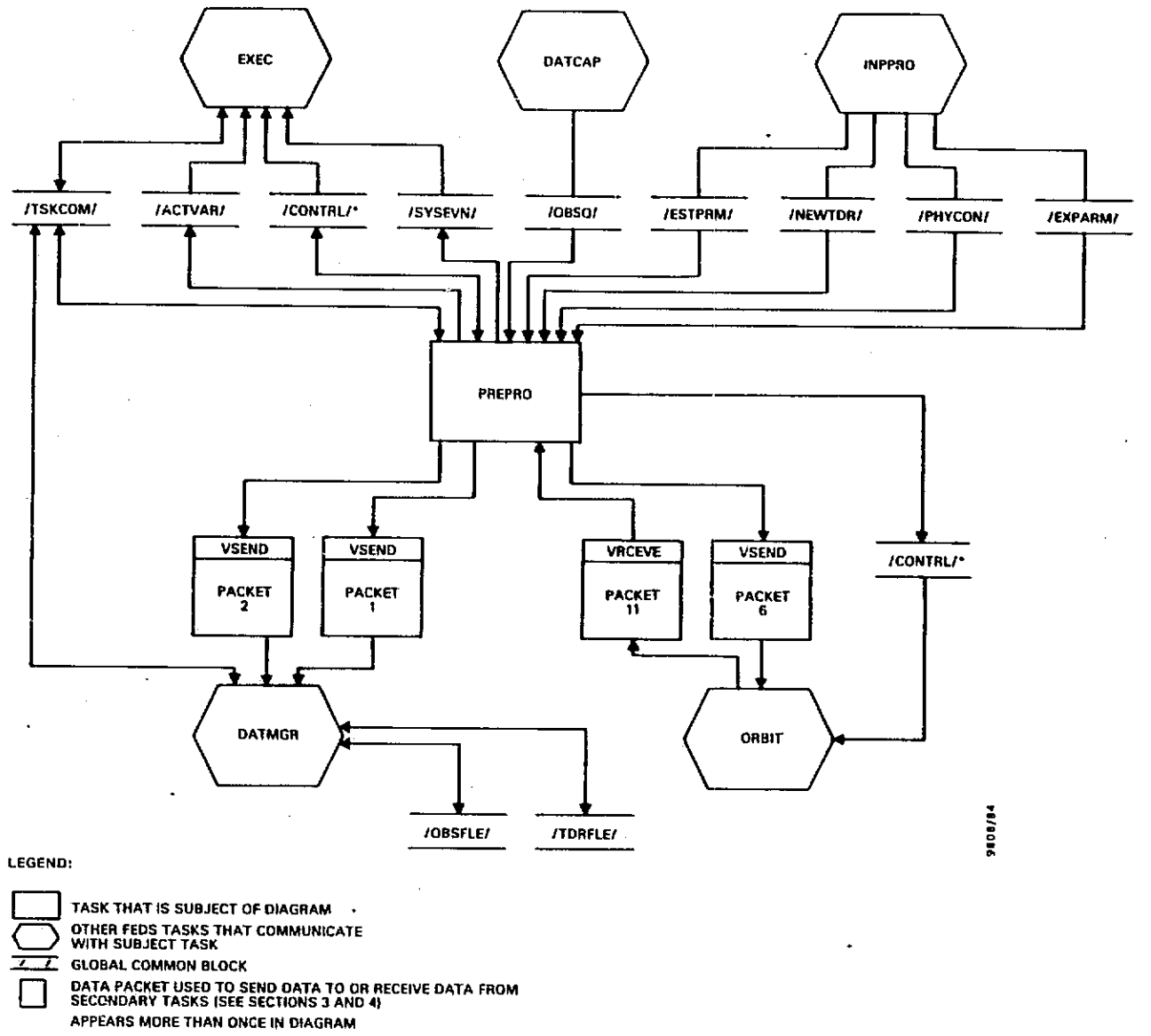


Figure 3-6. PREPRO Data Flow

time, and initial vector for the specified TDRS (specified by the internal TDRS ID). The initial vectors for generating new TDRS files are taken from global COMMON /NEWTDR/; the initial vectors for extending the TDRS orbit files are the vectors associated with the last entry in the current files. Subroutine TDRORB simply requests ORBIT to propagate the vector one step size at a time and then requests DATMGR to store the new TDRS vector in the next location in the orbit file. TDRORB also sets the start flag (ISTART) in the input parameters that are sent to the ORBIT task to indicate whether ORBIT should restart based on the input vector or whether to use its local table of backpoints to generate the requested vector.

While a TDRS orbit file is being generated, TDRORB blocks all other tasks from using the TDRS orbit file by setting global control flag TDRBSY(I) to indicate that the orbit file for TDRS I is busy and then clears the flag when the process is completed. TDRRDY(I) is set when a new orbit file has been generated for TDRS I. TDRORB returns control to PREPRO when the specified TDRS orbit file has been created or extended to the specified end time. PREPRO then calls TDRORB to create or extend the orbit file for the other TDRS. After this, PREPRO removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

3.3.2 TDRS ORBIT FILE UPDATE

Each time new TDRS vectors are received, the executive will direct the data preprocessor to update an entire TDRS orbit file or the portion of the specified orbit file that was affected by the most recent maneuver. If it is a simple update request (IDIR(3)=3), PREPRO sets the timespan of the update from the start time to the end time of the file. It then calls subroutine TDRORB to perform the update and

provides it with the start time, end time, and new TDRS vector taken from global COMMON /NEWTDR/ for the specified TDRS. TDRORB performs the update by requesting ORBIT and DATMGR as it does during TDRS orbit file generation (see Section 3.3.1), except that DATMGR is directed to replace the corresponding old vector in the TDRS orbit file with each new vector.

IF PREPRO is directed to update only the portion of the orbit file following the last maneuver (IDIR(3)=5), it sets the timespan of the update from the time of the last maneuver to the end time of the orbit file. PREPRO then proceeds as described above for a standard update.

When PREPRO has completed the assigned update for the specified TDRS, PREPRO removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

3.3.3 TDRS MANEUVER RECOVERY

When the current time matches the scheduled time of a TDRS maneuver, the executive schedules PREPRO to perform the TDRS orbit file maneuver recovery (IDIR(3)=4). Basically, PREPRO obtains the predicted vector after the maneuver from global COMMON /CONTRL/ and sets the timespan of the update from the maneuver time to the end time of the file. TDRORB is then called to update the file as described in Section 3.3.2. The current maneuver vector is saved for later use in local storage. PREPRO then sets the global flags, updates activity log variables, and surrenders control to the executive by removing itself from the scheduling list and setting IFLAG5.

3.3.4 OBSERVATION DATA PREPROCESSING

The last major function of PREPRO is to preprocess observation data (IDIR(3)=1 or 2). Subroutine OBSPRE is called to preprocess the observation data. If a user spacecraft

maneuver has occurred recently, OBSPRE checks the observation to check whether or not it is the first observation following the maneuver by comparing the observation time tag with the maneuver time. If it is the first observation following the maneuver, OBSPRE calls the DATMGR task to purge the observations file and the TDRS orbit files before the user spacecraft maneuver time. OBSPRE then proceeds by checking the validity or acceptability of the observation record based on the observation data time tag and validity flags for the observation data. If the data record is acceptable, OBSPRE converts the time (input in PB5 format) to the FEDS internal time format (A.1 seconds from reference); associates the current pass station ID, TDRS ID, and access method with the data record; and converts the Doppler observations to the proper engineering units. No smoothing of the observation data is done by PREPRO; however, observations are selected at the requested sample frequency. The preprocessed observation data record is then sent to DATMGR to be added to the observations file. This process is repeated for each observation record until the observations buffer is empty. At this point, the observation pass statistics and global flags are updated or set according to the directive and OBSPRE returns to PREPRO. PREPRO then removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

3.4 DATA MANAGER (DATMGR) TASK

As a secondary task in FEDS, DATMGR's main functions are to manage one observations file and up to two TDRS orbit files. These files are stored in DATMGR's local memory since no peripherals are available on the PDP-11/23. All access to these files by any FEDS task is controlled by the data manager that locates, reads, writes, and purges data when directed. Figure 3-7 is a baseline diagram of DATMGR;

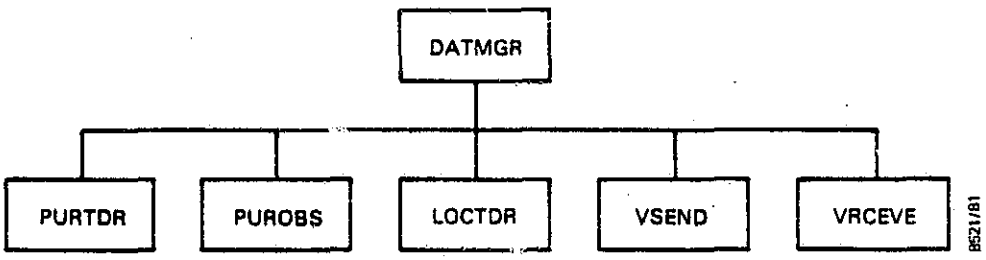


Figure 3-7. Baseline Diagram of DATMGR

Figure 3-8 shows the communication and data flow among DATMGR and other FEDS tasks. Appendix C describes the data packets.

Because DATMGR is a secondary task, it is controlled completely by the primary tasks PREPRO, ESTIM, OBSMDL, and DOPPRE. As shown in Figure 3-8, communication among DATMGR and the primary tasks is performed by means of task directive IDIR(4) and return status flag IRET(4) in global COMMON /TSKCOM/ and data packets that are sent to and from DATMGR. Global event flag IFLG10 is used to ensure that only one primary task at a time may use DATMGR. While the data manager is busy, IFLG10 is clear and when DATMGR is idle, IFLG10 is set. This allows a primary task to check whether DATMGR is busy before requesting it.

The procedure used by a primary task to request DATMGR is as follows. If the function performed by DATMGR requires input data from the primary task, the primary task sends the proper data packet to DATMGR via VSEND. The primary task then waits until DATMGR is free (IFLG10 is set). When DATMGR is free, the primary task sets task directive IDIR(4) to indicate which function DATMGR is to perform, requests DATMGR, and waits for IFLG10 to be reset by DATMGR to indicate that it has finished. If the function performed by DATMGR caused data to be output to the primary task, the primary task then receives the appropriate data packets using VRCEVE and checks whether an error occurred in DATMGR. The primary task must perform error recovery for any errors that occur in DATMGR.

During initialization (INIT(4)=1), DATMGR is called by PREPRO and is directed to purge all files. At this time, all local pointers and counters are cleared. When initialization is completed, DATMGR sets IFLG10 to notify PREPRO that it is finished and then exits.

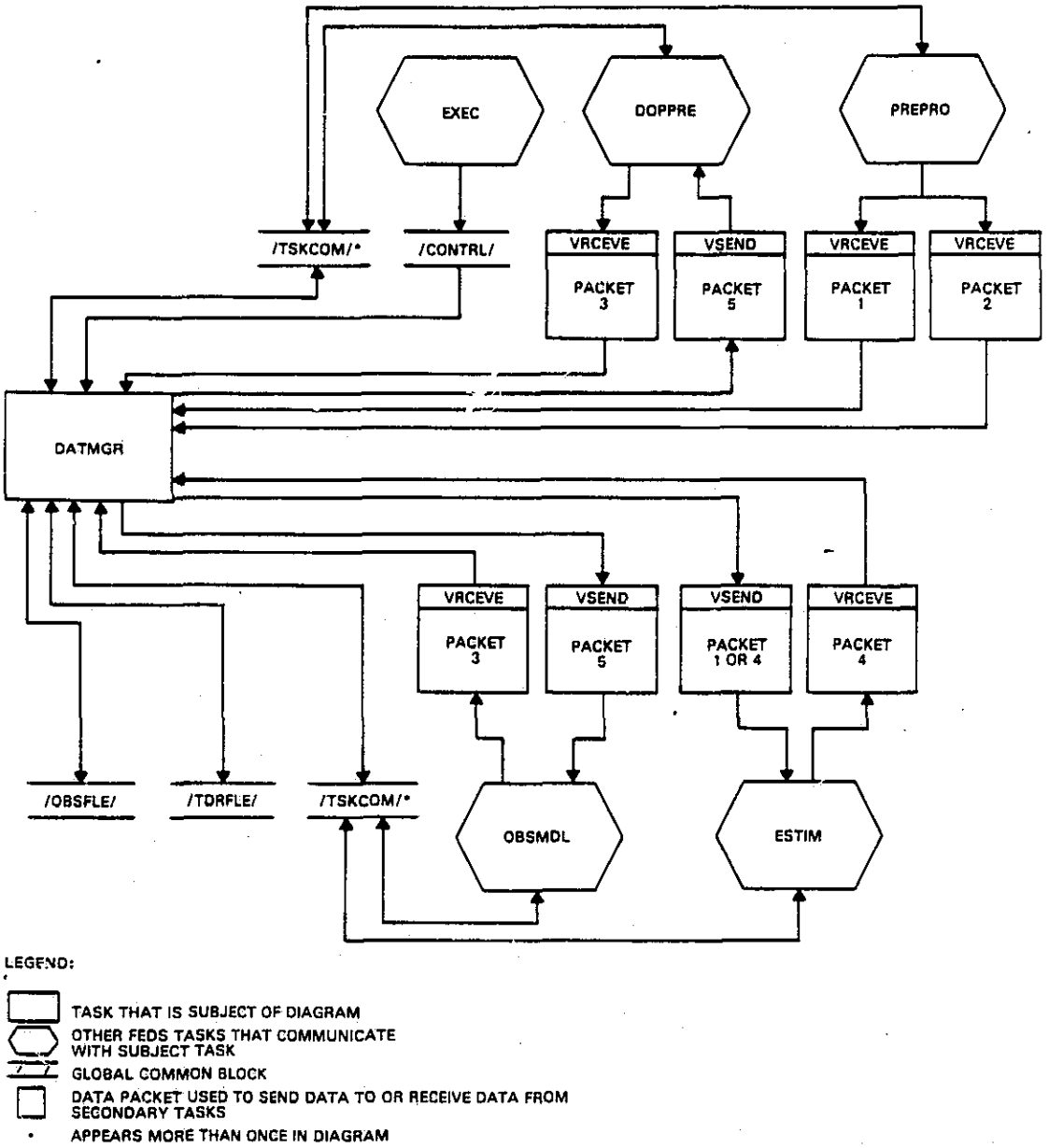


Figure 3-8. DATMGR Data Flow

Other functions that DATMGR performs relate specifically to TDRS orbit file management and observations file management, both of which are discussed in the following subsections.

3.4.1 TDRS ORBIT FILE MANAGEMENT

TDRS orbit files are stored in a wraparound fashion within a fixed-length storage area in local COMMON /TDRFLE/. The files are extended in a sequential record order starting in the first physical record. When all physical records in the orbit file are full, the next record is written in the first physical record of the file, thereby destroying the data previously stored there. The start pointer (first logical record) of the file is then moved to the second physical record. This process continues as new records are added to the file. The data manager uses the start and end pointers for each TDRS orbit file to find the location to write the next record. The time tags of the first and last logical records in each orbit file are also maintained. These times are used to locate a TDRS vector by time tag in the orbit files. When the TDRS orbit files are purged, these pointers and times are cleared, effectively emptying the files.

When DATMGR is directed to add a new TDRS vector to an orbit file (IDIR(4)=6), DATMGR receives the data packet containing the TDRS vector, its time tag, and the associated TDRS ID. DATMGR then stores the vector in the next logical record in the orbit file for the specified TDRS, and the start and end times and pointers for that orbit file are updated. DATMGR then sets IFLG10 and exits.

When DATMGR is directed to update a TDRS record according to its time tag (IDIR(4)=7), DATMGR receives the data packet containing the updated TDRS vector, its time tag, and the associated TDRS ID. DATMGR then calls LOCTDR to locate the current record in the specified orbit file that contains the TDRS vector with the same time tag. DATMGR replaces the old

vector in that record with the new (input) TDRS vector, sets IFLG10, and exits.

When DATMGR is directed to return a set of TDRS vectors surrounding a specified time to OBSMDL (IDIR(4)=8) or to DOPPRE (IDIR(4)=9), it receives the data packet containing the specified time tag and TDRS ID. DATMGR then calls LOCTDR to locate the record containing a time tag closest to the input time tag. Next, DATMGR loads into the output data packet 10 vectors that surround the input time tag and are retrieved from the specified orbit file. When possible, the vectors are chosen so that the input time falls in the middle of the timespan of the 10 TDRS vectors. If the input time tag is too close to the start time of the orbit file, the first 10 vectors in the file will be loaded into the output data packet; if the time tag is too close to the end time of the orbit file, the last 10 vectors will be loaded. The output data packet is then sent to the appropriate task and DATMGR sets IFLG10 and exits.

When DATMGR is directed to purge the portion of an orbit file before the time of a maneuver (IDIR(4)=10), it receives the data packet containing the input time tag and TDRS ID. Then, DATMGR resets the start pointer to point to the first record in the orbit file with a time tag equal to or greater than the input time tag and sets the start time of the orbit file accordingly. DATMGR sets IFLG10 and exits.

3.4.2 OBSERVATIONS FILE MANAGEMENT

The observations file is also a wraparound file that is stored in a fixed-length storage area in local COMMON /OBSFLE/. A set of start and end times is maintained for each observation pass in the observations file. These times are updated each time a new record is added to the file. As the file wraps around itself, the earliest pass in the file

gets shorter as the start time moves closer to the end time until the pass is eventually eliminated from the file. The observations file holds up to 125 observation records.

When DATMGR is directed to write a new observation record in the observations file (IDIR(4)=1), it receives the data packet containing the observation record. DATMGR writes the observation in the next record in the observations file and updates the pointers and pass timespans accordingly. DATMGR then sets IFLG10 to notify the calling task that it has finished and exits.

When DATMGR is directed to retrieve an observation record and send it to requesting task ESTIM (IDIR(4)=2), DATMGR retrieves the previous record and loads it in the output data packet. It then sends the data packet to the primary task, sets IFLG10, and exits.

When DATMGR is directed to reset the observation read pointer (IDIR(4)=3), it sets the observation read pointer to point to the last observation (the observation with the latest time tag) in the observations file. It then sets IFLG10 and exits.

When DATMGR is directed to update the end-of-pass indicator in the last observation written in the file (IDIR(4)=4), DATMGR changes the end-of-pass indicator in the last record in the file to 1, sets IFLG10, and exits.

When DATMGR is directed to update the last record that was read from the observations file (IDIR(4)=5), it receives the data packet containing the updated observation values. The information is then written into the record to which the observation read pointer is pointing. DATMGR sets IFLG10 and exits.

3.5 OUTPUT PROCESSOR (OUTPRO) TASK

OUTPRO is an independent primary task in FEDS that is responsible for downlinking output messages to ground control and for sending messages to the Communications Box. Six types of messages are downlinked from FEDS: predicted state vector tables, predicted one-way Doppler data, priority messages (critical errors and idle time messages), activity logs, DC summary and statistics reports, and DC residuals reports. In addition, OUTPRO downlinks a special end-of-simulation message when directed by the executive. Four types of messages are output to the Communications Box: current time request messages, reset accumulator command messages, predicted frequency shift messages, and Doppler measurement request messages. In addition, an initialization message is sent at the beginning of a simulation to verify communication between FEDS and the Communications Box.

Figure 3-9 is a baseline diagram of OUTPRO; Figure 3-10 shows the communication and data flow among OUTPRO and other FEDS tasks.

Like other primary tasks, OUTPRO is controlled by the FEDS executive. Most of this control is performed by the output control table, /OPTAB/. Each task that generates FEDS output information for downlink to ground control requests output of this information through the output table in the following manner. When the responsible task has completed generating the information and has stored it in the appropriate global COMMON block, it sets a lock flag (LCKFLG(I), where I indicates the type of data) that prevents any task from writing over and thus destroying the information before it is downlinked. The output control table also contains an output priority and the number of frames to be output (NFRAME(I)) for each type of output. Each time the executive gains control, it checks the output table for output

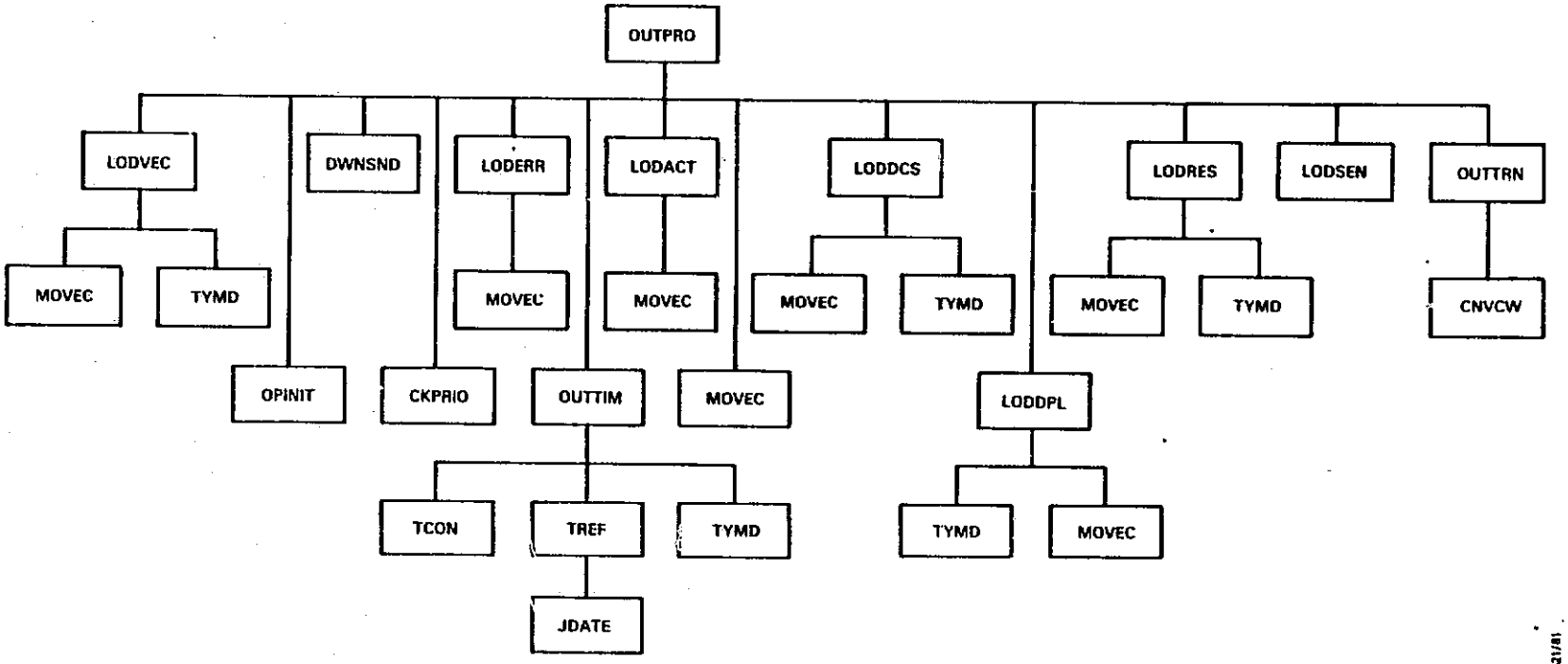


Figure 3-9. Baseline Diagram of OUTPRO

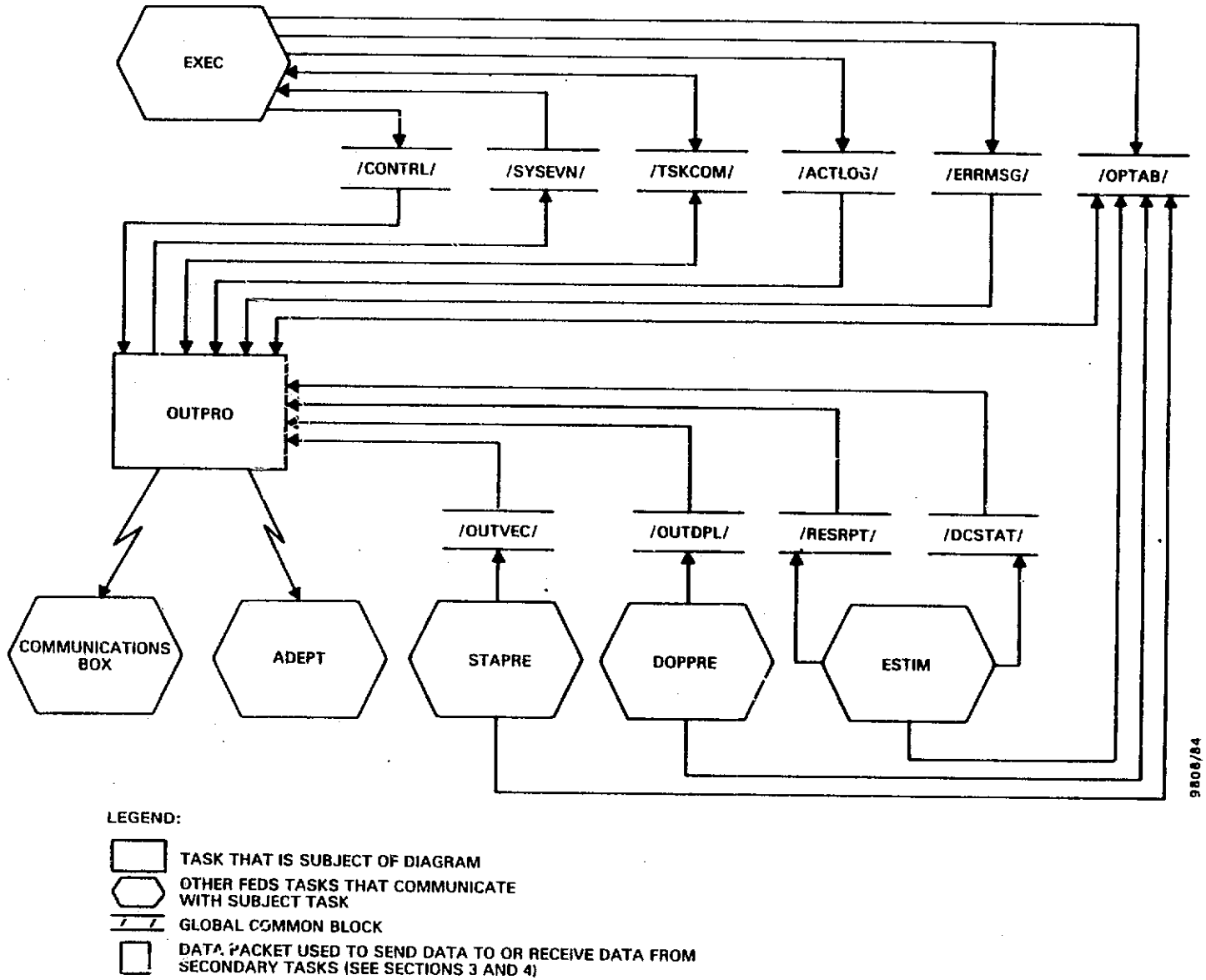


Figure 3-10. OUTPRO Data Flow

requests (lock flag is on). It then finds the highest priority output request in the output table. Directive IDIR(7) is set to the highest priority type of output to be downlinked, and OUTPRO is then scheduled as described in Section 2. When OUTPRO gains control it performs standard output. OUTPRO can also be directed to perform priority output; that is, it will downlink the specified information immediately and return control to the executive. Standard and priority output functions are described in the following paragraphs.

During initialization (INIT(7)=1), OUTPRO calls OPINIT to initialize local variables and pointers. OUTPRO then sets IFLAG5 to return control to the executive. When OUTPRO regains control, it examines task directive IDIR(7) to determine whether it is to perform standard or priority output.

If OUTPRO has been directed to perform standard output (IDIR(7) = 1, 2, 3, 4, 5, or 6), OUTPRO calls CKPRIO to scan through the output control table, /OPTAB/, to search for the highest priority information to be output. OUTPRO then calls one of the following subroutines to load the information into the output buffer:

- LODVEC (load next message of the state vector table)
- LODDPL (load one-way Doppler data)
- LODERR (load priority messages)
- LODACT (load activity log)
- LODDCS (load DC summary and statistics report)
- LODRES (load DC residuals report)

If OUTPRO has been directed to perform priority output (IDIR(7)=7 or 8), it calls either LODACT to load the next activity log message (IDIR(7)=7) or LODERR to load the priority message (IDIR(7)=8) in the output buffer. If OUTPRO has been directed to downlink an end-of-simulation message

(IDIR(7)=9), it calls LODSEN to load the sentinel record into the output buffer.

Each time a message is loaded into the output buffer, the number of frames is decremented by the number of frames loaded in the message. After the message has been loaded, OUTPRO calls OUTTIM to obtain the current simulation time in YYMMDDHHMMSS.SS format and inserts it into the record header of the downlinked message. OUTPRO then calls DWNSND to downlink the message to ground control. To do this, DWNSND issues a QIO directive to read the ready message sent by ground control. When a message is received, DWNSND checks whether it is a ready message or a retransmission request. If it is a retransmission request, DWNSND issues a QIO to downlink the previously downlinked message that was saved, issues a QIO for a ready message, and the output process described in the preceding paragraphs is performed. When a ready message is received, DWNSND issues a QIO to downlink the current output message in the output buffer. The message is then transferred to the save buffer and DWNSND returns to OUTPRO.

If there are more frames of this type, processing returns to the point at which OUTPRO calls the appropriate subroutine to load the message and processing continues as before.

If all frames of this type have been output (NFRAME(I)=0), output processing of this type of data has been completed. At this time, the output information storage area is unlocked (LCKFLG(I) is turned off). If standard output has been requested, OUTPRO searches the output control table by priority for the next type of output to be downlinked. If one is found, standard processing of that type of data is performed as described above. When no more types of requested output are left in the output control table, OUTPRO

removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

If all frames of the specified type of priority output have been output, OUTPRO simply removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

If OUTPRO has been directed to perform output to the Communications Box (IDIR(7)=10), it calls OUTTRN to form and transmit the requested message. OUTTRN examines ICODE in /OPTAB/ to determine which message is to be transmitted, forms the 11-byte message and immediately issues a QIO to transmit the message. After OUTTRN has returned, OUTPRO unconditionally removes itself from the scheduling list and sets IFLAG5 to return control to the executive.

SECTION 4 - COMPUTATIONAL TASKS

Five FEDS tasks are primarily responsible for performing computations in FEDS. These tasks generate data to be downlinked to ground control or produce intermediate quantities that must be used by other FEDS computational tasks to generate the output data. They are as follows:

- Orbit Propagator (ORBIT) propagates a given spacecraft state (and, optionally, its partial derivatives) and sends it to a specified task for output or for use in another computational model.

- State Predictor (STAPRE) generates tables of predicted user spacecraft state vector data for downlink to ground control and use in the Doppler predictor.

- Doppler Predictor (DOPPRE) generates predicted one-way Doppler data for downlink to ground control and output to the Communications Box.

- Estimator (ESTIM) estimates and corrects the current user spacecraft state based on the differences in observed and computed TDRSS observations. The output state vector is used for both state prediction and Doppler prediction.

- Observation Modeling (OBSMDL) computes TDRSS Doppler observations and partial derivatives that correspond to the data in the observations file based on the current best estimate of the user spacecraft state and the given TDRS position at each observation time.

The following subsections provide a functional description of each of these tasks including baseline diagrams and data flow diagrams. The mathematics for the computational models used in these tasks is given in the FEDS mathematical specification (Reference 2). Appendix C contains detailed

descriptions of the data packets, shown in the data flow diagrams, that are used for intertask communications.

4.1 ORBIT PROPAGATOR (ORBIT) TASK

ORBIT is a secondary task in FEDS. It is used by four of the FEDS primary tasks that require orbit propagation: PREPRO, STAPRE, ESTIM, and OBSMDL. Because of the FEDS time-slicing control scheme, the orbit propagator must be able to service a wide variety of back-to-back requests from the primary tasks. In addition, the application demands that ORBIT must have sufficient force-modeling capabilities to propagate both the high-altitude geosynchronous TDRS orbits and the low-altitude drag-perturbed user spacecraft orbit.

To fulfill the needs of other FEDS tasks, ORBIT has the capability of propagating four separate orbits simultaneously. This means that it can switch from propagating one orbit to propagating another without starting the integrator each time. Each time ORBIT is requested (called) by a primary task, it propagates the requested orbit only. It must finish executing one request before it can be called to do another; it is not reentrant.

The four orbits are identified by two FEDS spacecraft IDs. ISCID, included in the ORBIT input data packets (see Appendix C), is used by the primary task to indicate which orbit is to be propagated, and IDSC is used internally as an index in the ORBIT task. The four orbits are as follows:

ISCID	IDSC	Spacecraft Orbit
1	1	Orbit for TDRS 1; used by PREPRO
2	2	Orbit for TDRS 2; use by PREPRO
4	3	Past-time orbit for user spacecraft that includes associated variational equations; used by ESTIM and OBSMDL

ISCID	IDSC	Spacecraft Orbit
5	4	Real-time orbit for user spacecraft; used by STAPRE

ORBIT uses the multistep method of numerical integration when possible. Two pairs of predictor-corrector integrators from the Adams and Cowell groups of integrators are used. These methods are derived by integrating polynomials that interpolate (or, for the predictors, extrapolate) based on a table of backpoints. A set of previously computed accelerations forms the table of backpoints for a satellite state, and the partial derivatives of accelerations with respect to the initial (epoch) state form the backpoints for the variational equations. Each of the four spacecraft states has its own table of backpoints independent of the other spacecraft trajectories, and the past-time user spacecraft state (ISCID=4) has the variational equations table of backpoints associated with it.

Because the predictor-corrector methods have inherent limitations in startup capability, a separate procedure for initially filling the table of backpoints is required. In ORBIT, integration startup is performed by a Runge-Kutta-Fehlberg (RKF) integrator. However, this procedure is performed only when necessary for the following reasons: the RKF integrator is slow since it performs five derivative evaluations per step and is of low order (six RKF steps must be performed to approximate one step of the multistep integrator in accuracy).

These types of numerical integration alone are not entirely satisfactory for meeting the requirements placed on the orbit propagator. ORBIT must be able to frequently produce a satellite state and state transition matrix for the observation model and the estimator. The request times for these quantities will almost always be off-grid compared to a

fixed step-size ephemeris prediction. Single-step integration (using RKF integration) would enable ORBIT to produce results at the requested times; however, the extreme slowness of the single-step integration compared to multistep integration makes this method unacceptable here, especially for integrating the variational equations (a 6-by-6 or a 6-by-7 matrix).

The solution to this problem is the multistep method of interpolation, which is a generalization of the predictor-corrector methods. This interpolation method takes advantage of the existence of the tables of backpoints already present for multistep integration. Thus, multistep interpolation is the primary method used by ORBIT to produce the state and, optionally, the state transition matrix at the time requested by the calling task. Multistep integration is used only to extend the table(s) of backpoints (forward or backward) in time, and single-step integration is used only to fill the table(s) of backpoints initially.

Figures 4-1 and 4-2 present baseline diagrams of ORBIT. Figure 4-1 shows the hierarchy of the subroutines in ORBIT, whereas Figure 4-2 shows the subroutines grouped by function. The external communication and data flow among ORBIT and other FEDS tasks and the internal data flow in ORBIT are shown in Figure 4-3. Appendix C contains descriptions of the data packets. Further description of orbit integration and interpolation can be found in Reference 2. Following is a description of the data flow in ORBIT.

During initialization (INIORB=1), ORBIT calls ORBINI to initialize local pointers and flags. ORBIT then sets IFLG11 to indicate that it has finished and exits.

Each time ORBIT is requested and initialization is not to be done, the following procedure is performed. ORBIT first checks array ORCALL to determine which primary task has

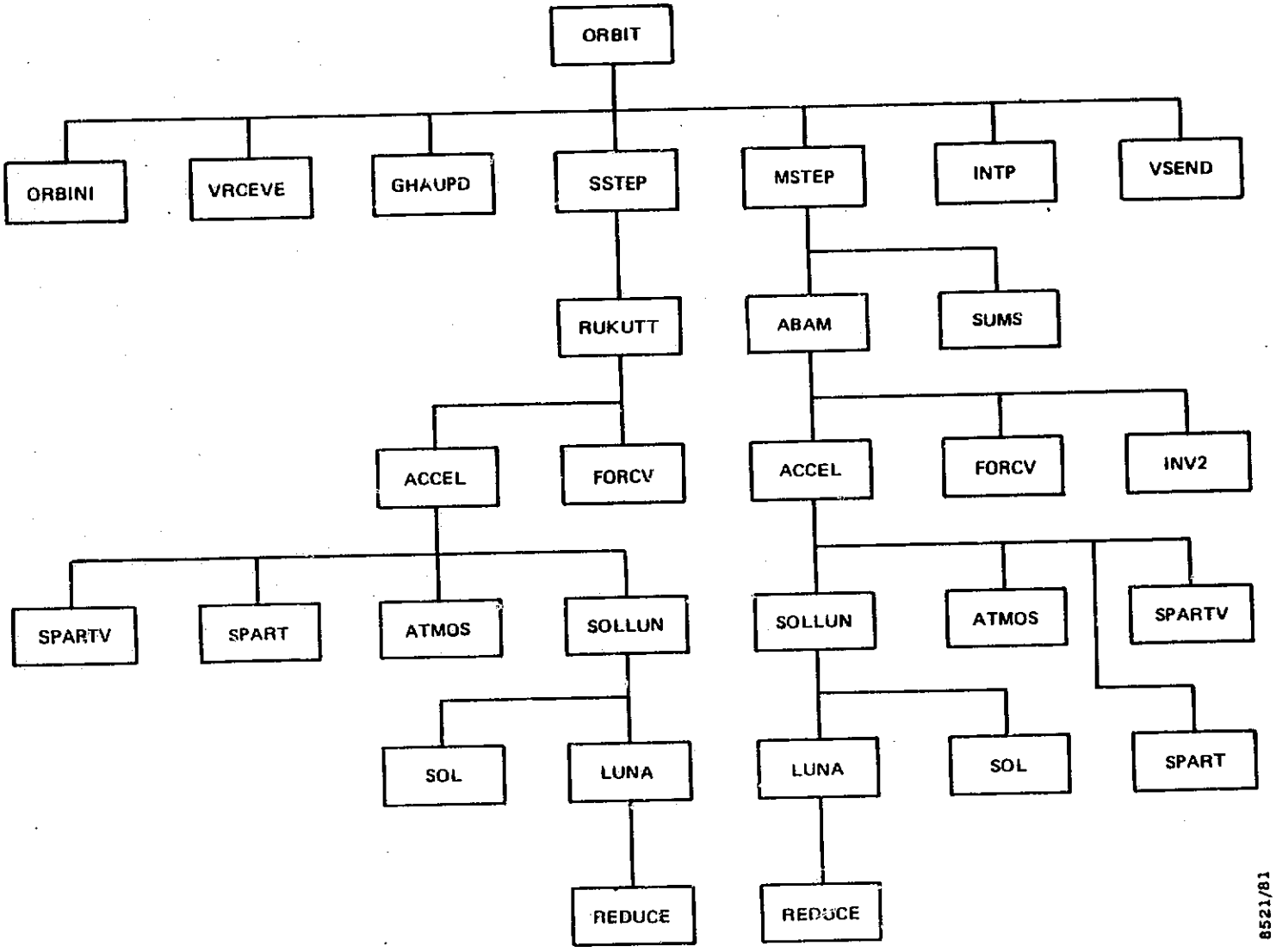
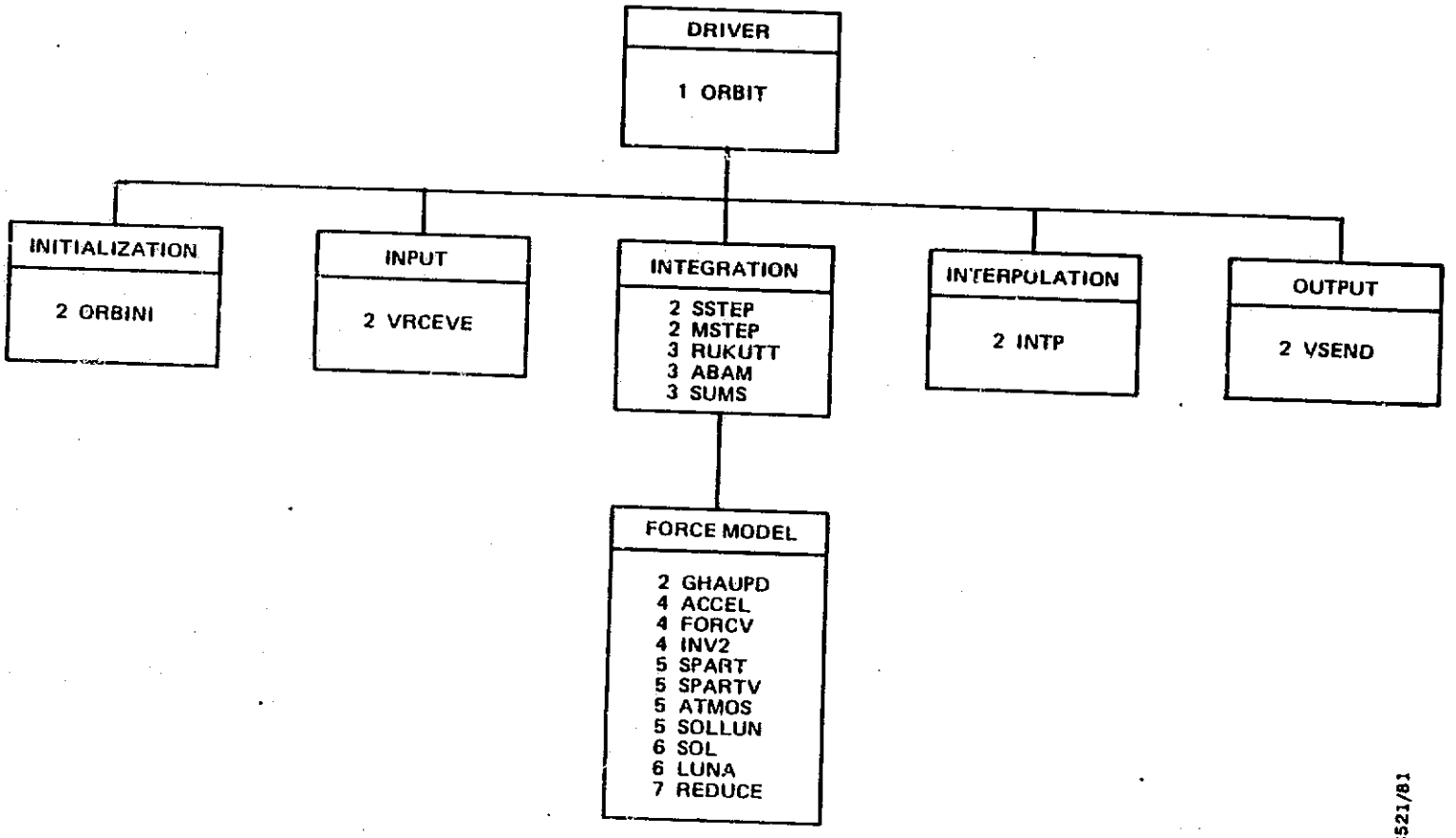


Figure 4-1. Baseline Diagram of ORBIT



NOTE: THE NUMBER PRECEDING THE SUBROUTINE NAME INDICATES THE HIERARCHY.

Figure 4-2. Functional Block Diagram of ORBIT

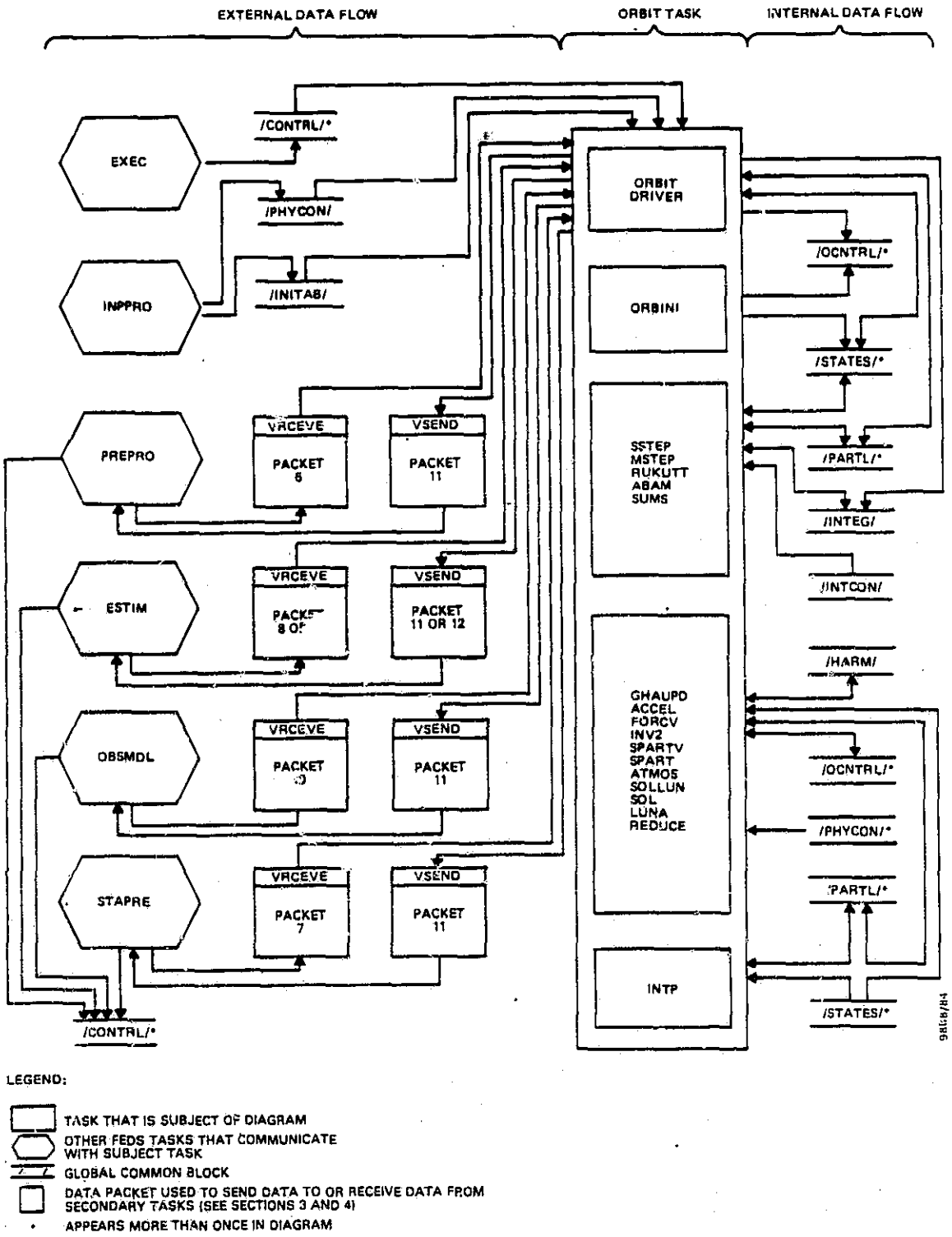


Figure 4-3. ORBIT Data Flow

requested it. The input data packet (see Appendix C) is then received from the specified primary task. ORBIT converts the input start and end times of integration from seconds from reference to modified Julian dates.

If partial derivatives are requested (IPART=1 or 2), the number of variational equations (NEQ) is set based on the solve for drag coefficient indicator in the input data packet. Next, ORBIT checks the input value of ISTART to determine whether integration startup is required (ISTART=1) or whether an output vector is to be produced based on the current backpoints table.

If integration startup is requested, ORBIT verifies that a reasonable starting vector was input. If not, an error flag is set, IFLG11 is set, and ORBIT exits. ORBIT also verifies that if partial derivatives are requested, the spacecraft ID (ISCID) is 4 (the only orbit for which partial derivatives can be computed). ORBIT then sets the integration step size based on the ISCID and the step sizes given in global COMMON /PHYCON/. If integration startup is requested, the direction pointer is set to indicate forward (IFWD(IDSC)=1) or backward (IFWD(IDSC)= -1) propagation, and the input vector is moved into the internal start vector array. For all cases, the Greenwich Hour Angle (GHA) is computed at the input start time, and the spacecraft area and mass and the Sun and the Moon force model indicators are set, based on the IDSC and the values given in global COMMON /PHYCON/.

If startup is requested, ORBIT proceeds to call SSTEP for each of 10 steps required to fill the table of backpoints for the specified IDSC. MSTEP is then called to compute and insert the 11th set of accelerations in the table of backpoints and to compute the second sums required for multistep integration and interpolation. Having filled the backpoints table, ORBIT then proceeds as if ISTART were zero.

When the table of backpoints is already full (ISTART=0), ORBIT checks the requested propagation end time (the requested time tag of the output vector). If the end time is within the timespan of the current table of backpoints, ORBIT simply calls INTP to obtain the output vector at the end time through multistep interpolation. Otherwise, ORBIT calls MSTEP to extend the table one step at a time in the direction indicated by IFWD(IDSC). This is done until the requested end time is within the timespan of the table, at which time INTP is called to produce the output state vector at the requested end time. The partial derivatives, if requested, are computed in conjunction with the state during the calls to SSTEP, MSTEP, and INTP.

The output state vector and, optionally, the partial derivatives (state transition matrix) are loaded into the output data packet along with flags that indicate which operations were performed. ORBIT then sends the data packet to the primary task that requested it, IFLG11 is set, and ORBIT exits.

4.2 STATE PREDICTOR (STAPRE) TASK

STAPRE is a primary task responsible for generating or extending the predicted state vector table for the user spacecraft. Figure 4-4 is a baseline diagram of STAPRE; Figure 4-5 shows the communication and data flow among STAPRE and other FEDS tasks. Appendix C describes the data packets.

There are two constants, SPINT and SPFREQ, in global COMMON /PHYCON/ that determine the size of the state predict table and how often it is generated. Since this table contains "predicted" vectors, it always contains a future timespan. The requirement for the state predict table is that the table must contain data that covers at least a specified amount of time in the future. To satisfy this requirement,

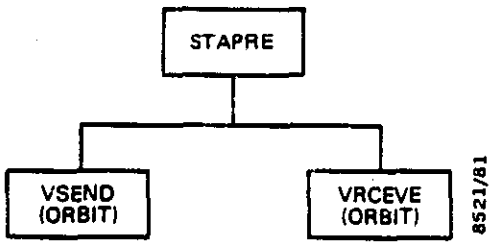


Figure 4-4. Baseline Diagram of STAPRE

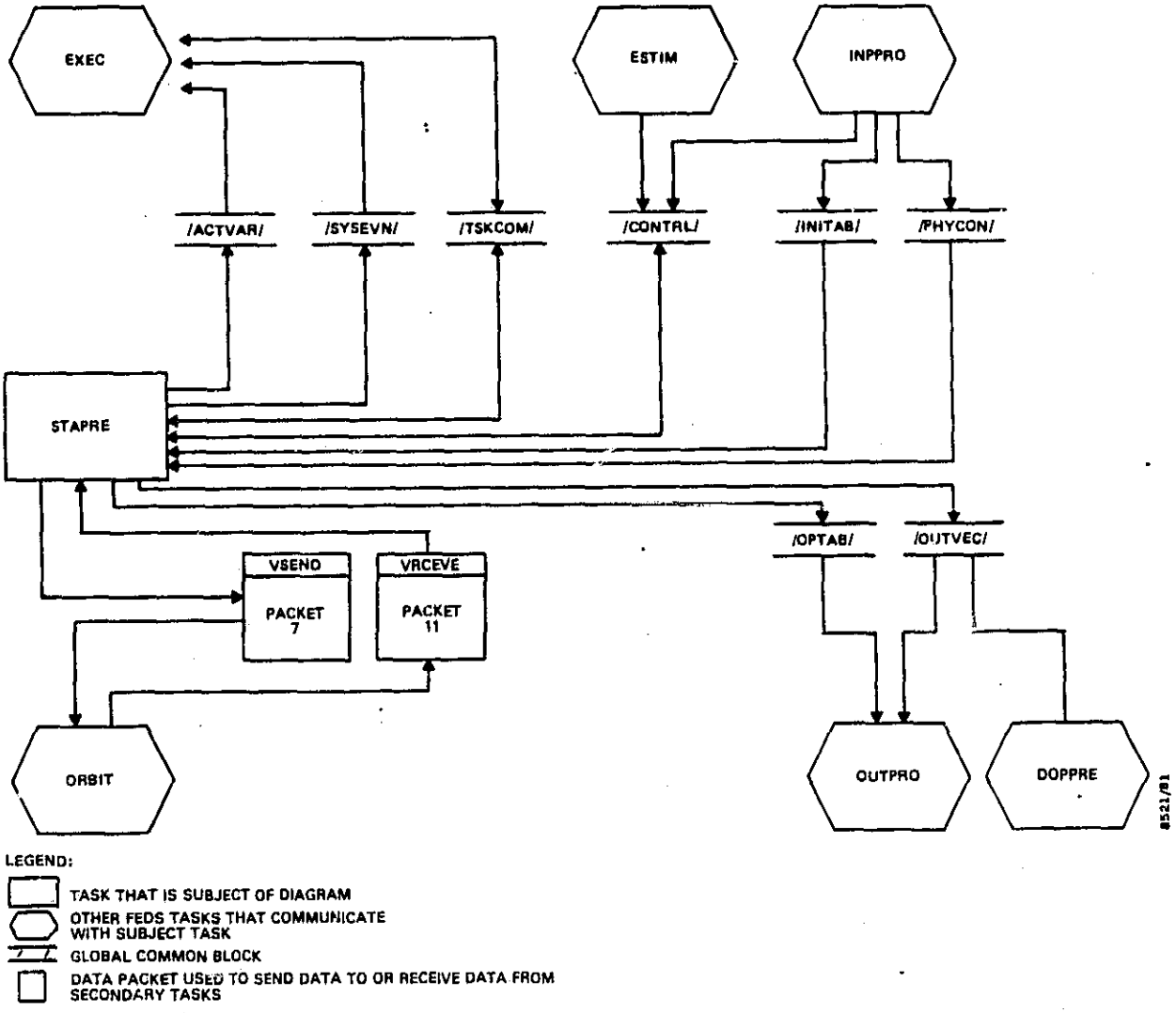


Figure 4-5. STAPRE Data Flow

a state vector table twice the specified size (in time units) is generated from the current time forward each time a new table is to be generated. Then, when half of the state vectors are out of date, the table is extended into the future by the table size. For instance, if the state predict table must contain at least 30 minutes of future data at 1-minute intervals, the table is generated to cover the next 60 minutes. Then, after 30 minutes when only 30 minutes of future data remain in the table, it is extended by 30 minutes, replacing the out-of-date data with the new data. After this extension, the last vector in the table should be time tagged 60 minutes after the current time.

Each time a new best estimate of the user spacecraft state is obtained from a new initialization table, from a new state solution from the estimator, or from a user spacecraft maneuver, a new state predict table is generated. The executive determines when the state predict table needs to be extended or generated based on a new vector and directs STAPRE to perform the appropriate function.

During initialization (INIT(8)=1), STAPRE initializes all local variables and flags and sets IFLAG5 to return control to the executive.

When STAPRE regains control, it examines task directive IDIR(8) to determine the function it is to perform. If STAPRE has been directed to generate a new predicted state vector table based on a new state solution (IDIR(8)=2), a new initialization table (IDIR(8)=3), or a user spacecraft maneuver (IDIR(8)=4), it retrieves the initial state vector from the specified source, saves it in /OUTVEC/ as the official FEDS reference state (for use by the Doppler predictor), sends the new vector to ORBIT, and requests ORBIT to restart orbit number 5 (ISCID=5) with the new vector and to

propagate it to the current time. STAPRE waits for ORBIT to complete and receives the output state vector at the current time. STAPRE then sets the start time of the table to the current time, sets the end time to the current time plus twice the specified size of the table (SPINT), and reinitializes the pointers. For each time interval, STAPRE then requests ORBIT to obtain the state vector and to store it in the next location of the state predict table.

When the table has been completed, STAPRE sets lock flag LCKFLG(1) in the output control table, /OPTAB/, to indicate that the state predict table is ready to be downlinked. It then saves the last vector in the table as the start vector for the next extension and sets IFLAG5 to return control to the executive.

When STAPRE is called to extend the state predict table (IDIR(8)=1), STAPRE sets the new end time of the table to the previous end time plus the specified size of the table. For each time interval to be added to the table, STAPRE requests ORBIT to obtain the state vector and to store it in the next location of the state predict table in a wraparound fashion. When the table has been extended, the start time is updated, and output pointers are updated to point only to the part of the table that contains the extension. The table is then locked as described earlier to indicate that the extension of the table is ready to be downlinked. The last vector in the table is saved as the start vector for the next extension, and STAPRE sets IFLAG5 to return control to the executive.

4.3 DOPPLER PREDICTOR (DOPPRE) TASK

DOPPRE is a primary task in FEDS that predicts one-way TDRSS Doppler observations over a specified tracking interval. The tracking intervals for one-way Doppler prediction are contained in the uplinked tracking schedule. Each tracking

interval is defined by a start time and an end time, which are specified in global COMMON /CONTRL/, and an observation frequency, which is specified in global and the COMMON /EXPARM/ TDRS ID to be used, which is specified in global COMMON /TSCHED/.

The requirement for the Doppler predict table is that the table must contain data covering at least a specified amount of time in the future. To satisfy this requirement, a Doppler frequency shift predict table, twice the specified size (in time units), is generated from the current time forward each time a new table is to be generated. Then, when half of the Doppler shift records are out of date, the table is extended into the future by the table size. For instance, if the Doppler predict table must contain at least 5 minutes of future data at 10-second intervals, the table is generated to cover the next 10 minutes. Then, after 5 minutes, when only 5 minutes of future data remain in the table, it is extended by 5 minutes, replacing the out-of-date data with the new data. After this extension, the last predicted frequency shift record in the table should be time tagged nearly 10 minutes after the current time.

The executive requests DOPPRE during task initialization and each time Doppler prediction is scheduled based on the tracking schedule. To ensure that the predicted Doppler data will be downlinked close to the start time of the tracking interval, DOPPRE is scheduled with a specified amount of pad time (TPAD) before the start time of each interval. This gives DOPPRE ample time to generate the data. DOPPRE uses both TDRS vectors retrieved by DATMGR for the specified TDRS and the predicted state vector table for the user spacecraft to perform the one-way Doppler prediction. DOPPRE does not interface with ORBIT. The mathematics for

the one-way TDRSS Doppler observation model is given in Reference 2.

Figure 4-6 is a baseline diagram of DOPPRE; Figure 4-7 shows the communication and data flow among DOPPRE and other FEDS tasks. Appendix C contains descriptions of the data packets.

During initialization (INIT(6)=1), DOPPRE calls DPINIT to initialize the local variables used in the Doppler prediction. When initialization is complete, DOPPRE sets IFLAG5 to return control to the executive.

When the executive directs DOPPRE to generate a Doppler predict table (IDIR(6)=1), the requested tracking interval is passed through global COMMON /CONTRL/. DOPPRE then predicts one-way Doppler data over this tracking interval in the following manner. DOPPRE locates the specific ground station associated with the specified TDRS ID. DOPPRE calls GHAUPN to compute the GHA update for the ground station at the start time of the tracking interval. After the GHA update is completed, DOPPRE calls DOPMDL to compute the tracking range at the starting time. This is done because a Doppler observation cannot be computed without the initial range at the start time of the tracking interval. DOPPRE then adds the specified observation frequency to the start time to obtain the first observation time tag. DOPPRE then calls DOPMDL to compute a Doppler observation at the observation time and loads the observation into global COMMON /OUTDPL/. The next observation time tag is computed, and DOPMDL is called to compute the associated observation as described above until the table is full or the pass end time is past.

When the executive directs DOPPRE to extend the Doppler predict table (IDIR(6)=2), DOPPRE computes the number of records to produce based on the time of the last entry in the table and the scheduled pass end time. DOPPRE then continues filling the Doppler predict table in wraparound fashion until the proper number of records has been computed.

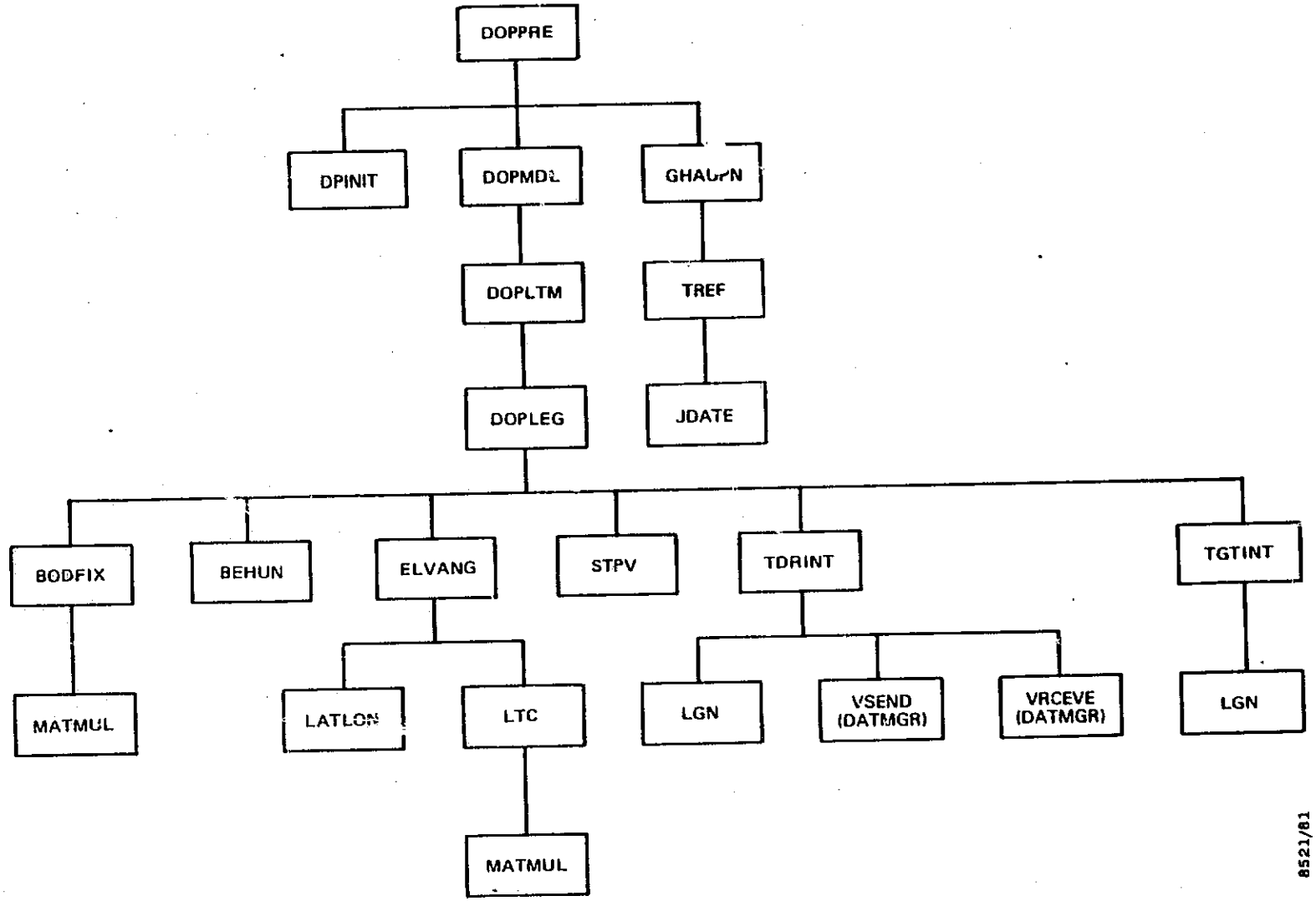


Figure 4-6. Baseline Diagram of DOPPRE

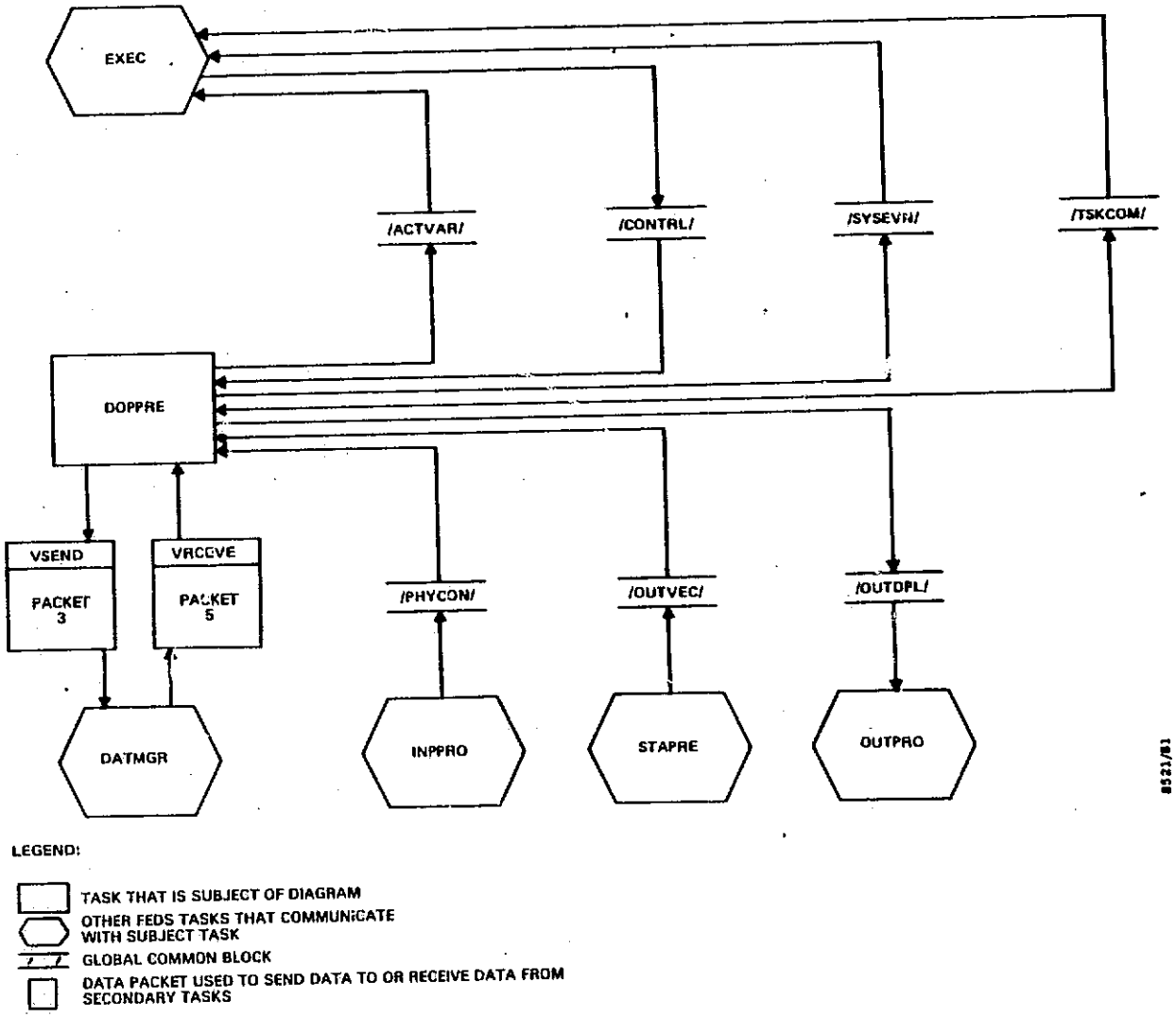


Figure 4-7. DOPPRE Data Flow

When all the requested Doppler observations have been computed and placed in COMMON /OUTDPL/, DOPPRE locks the Doppler predict table by setting LCKFLG(2) to .TRUE. in the output control table, which indicates that Doppler data is ready to be downlinked. DOPPRE then removes itself from the active task list (IACT(6)=0) and sets IFLAG5 to return control to the executive.

4.4 ESTIMATOR (ESTIM) TASK

ESTIM is a primary task that performs one of the major computational functions in FEDS. Its purpose is to estimate the user spacecraft (target) state using the most recent batch of observation data. A batch least-squares estimator is used to perform differential correction on the target's state parameters in a sliding batch mode in which the previous DC epoch is moved forward to encompass a fixed-length span of observation data. The state parameter set consists of a minimal set of six Cartesian state (position and velocity) components to which four optional parameters can be added--a drag term and three user spacecraft clock terms.

To limit the computational load, the estimation algorithm uses an editing scheme and a measurement partial derivatives computation that are nominally done only once per DC slide. The algorithm also allows a partial precomputation of the next DC slide before all the observations data for that slide are available.

ESTIM was originally designed to include both estimation logic and observation modeling. However, due to task memory limitations, the observation modeling has been separated into another task, OBSMDL. OBSMDL is completely controlled by ESTIM. Communication between the two tasks is accomplished through global COMMON blocks, and an event flag is used for task synchronization.

Figure 4-8 shows the data flow between ESTIM and the other FEDS tasks. Figure 4-9 provides a baseline diagram of ESTIM. Appendix C contains descriptions of the data packets.

Four FEDS executive function directives are processed under the control of ESTIM:

1. Perform full estimation (slide forward) over the latest fixed-length batch of observation data (IDIR(5)=1).
2. Partially precompute the next slide using the previous observation span (IDIR(5)=2).
3. Complete estimation (slide forward) after precomputation is finished (IDIR(5)=3).
4. Update a priori state parameters with the predicted user state after a maneuver (IDIR(5)= 4).

When ESTIM is first requested by the executive, the internal COMMON blocks used by ESTIM and OBSMDL are initialized to their default values in ESINIT, a global event flag is set, and ESTIM suspends itself. Upon resumption by the executive, ESTIM transfers control to either ESLIDE or ESMNVR, depending on the function directive to be performed. The first three function directives listed previously are performed by ESLIDE, and the last function is performed by ESMNVR. ESLIDE controls the slide advance and estimation process that consists of initialization (SLINIT), a state correction loop (SLITER), and status return and slide termination (SLEND).

The estimation algorithm consists of one or more iterations of corrections to the solve-for state based upon the most recent batch of multipass data: SLITER controls each iteration and returns updated state and estimation status (convergence/divergence) to ESLIDE. This sequence is performed

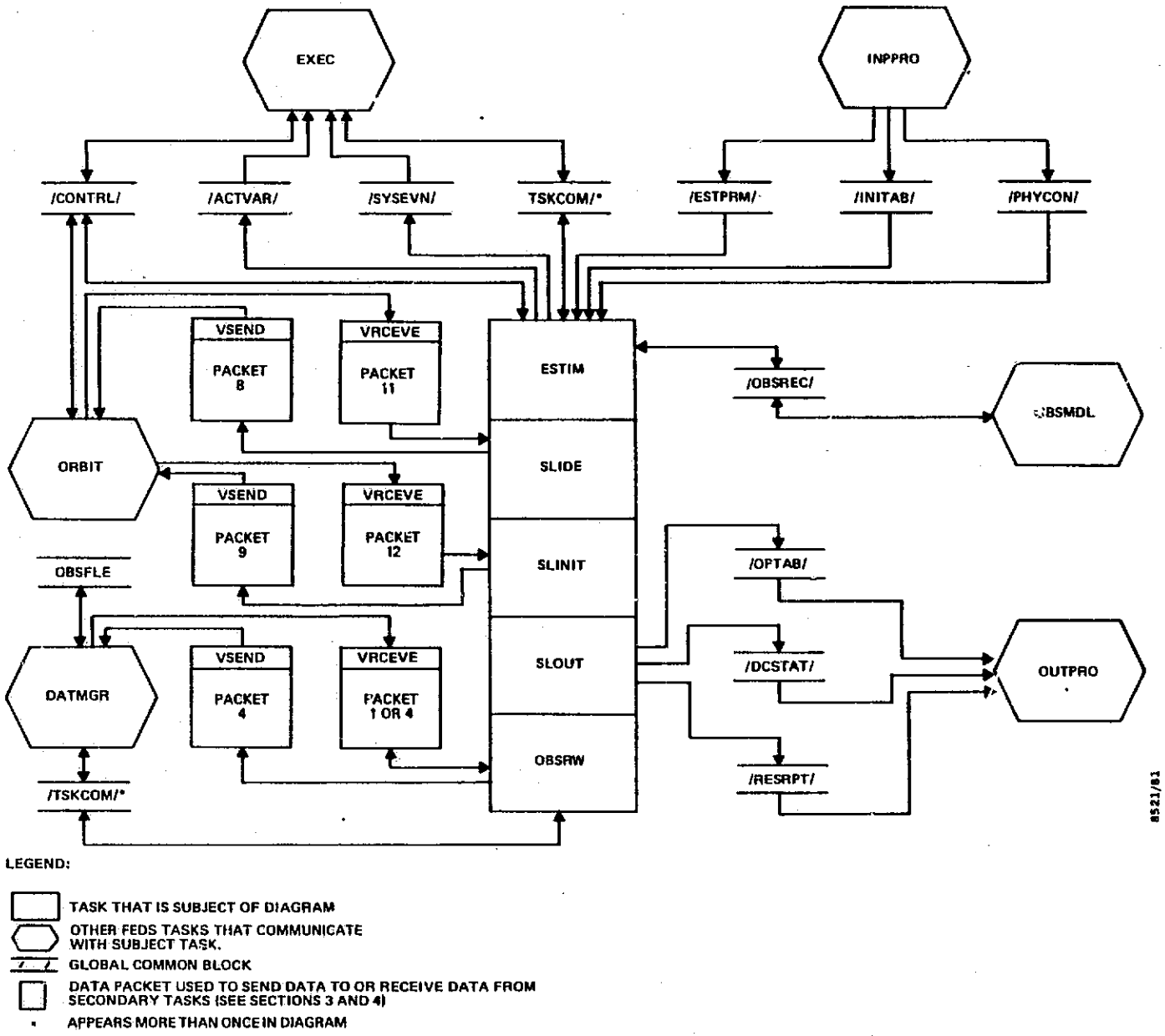


Figure 4-8. ESTIM Data Flow

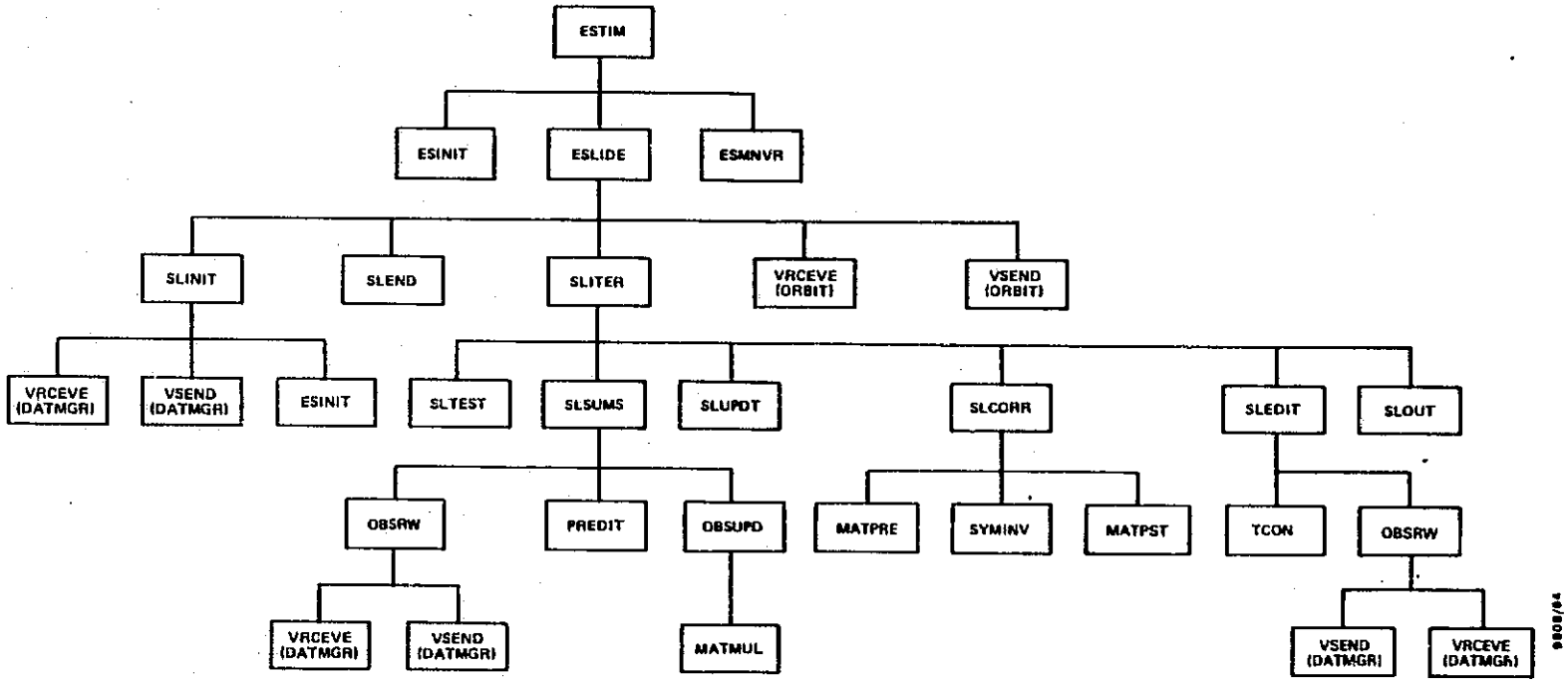


Figure 4-9. Baseline Diagram of ESTIM

for each of the different executive function directives: the internal logic that implements each directive is embedded in the modules.

Each iteration consists of a minimum of one pass through all of the available observation data within the timespan of the current slide. During this processing loop, the normal matrix and other batch estimation statistics are accumulated (SLSUMS); the state correction is computed (SLCORR); the solve state is updated (SLUPDT); and estimation reports are queued for downlink (SLOUT). Various types of observation editing are performed during this cycle. The first time that ESTIM uses each observation data record, PREDIT is called to detect grossly out-of-bounds measurements by checking TDRS-target-station geometry. During the first iteration of each slide and if linearity constraints are violated on subsequent slides, the observed-minus-computed residuals are edited if they are larger than acceptable, and one or more edit loops are performed to remove measurements from the accumulated sums based upon the current estimation statistics.

4.5 OBSERVATION MODELING (OBSMDL) TASK

OBSMDL is a primary task in FEDS that computes TDRSS observations based on given TDRS orbits and the current best estimate of the user spacecraft orbit. Unlike other primary tasks, OBSMDL is not controlled by the executive. OBSMDL is simply an extension of the estimator and is therefore controlled directly by ESTIM.

OBSMDL models one-way TDRSS Doppler observations and, optionally, computes the partial derivatives required in the estimation algorithm for these observations.

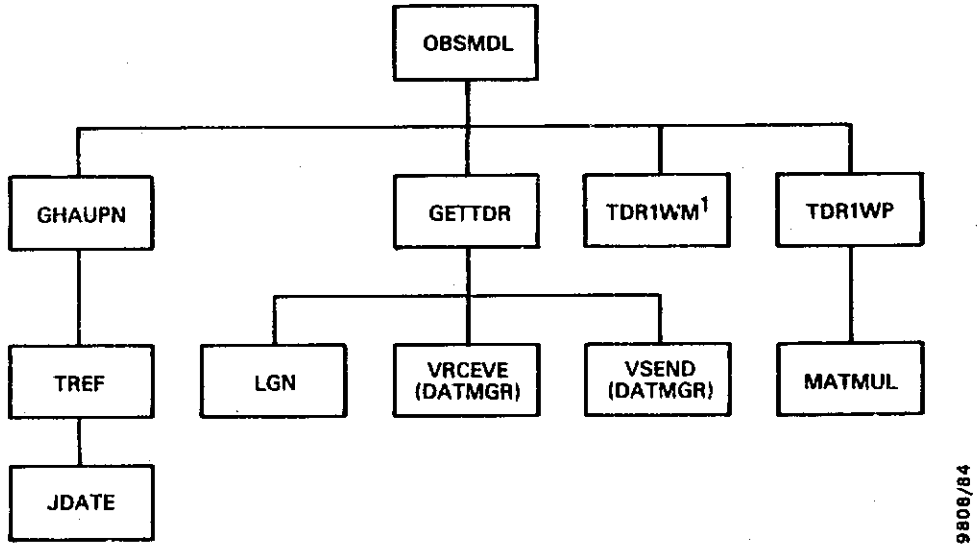
OBSMDL performs the following corrections for the specified types of observations:

- Backward light-time
- Tropospheric refraction
- Transponder delay
- User frequency offset
- User frequency drift
- User frequency drift rate

The computational models are described in Reference 2. Figure 4-10 is a baseline diagram of OBSMDL. The communication and data flow among OBSMDL and ESTIM and other FEDS tasks are shown in Figure 4-11. Appendix C presents descriptions of the data packets.

If ESTIM has requested OBSMDL to compute an observation residual, OBSMDL begins by retrieving the observation time tag from global COMMON and calling GHAUPN to compute the GHA for the ground station at the time tag. Next, OBSMDL calls TDR1WM to model one-way TDRSS Doppler observations. During observation modeling, subroutine SORBIT is used to propagate the target satellite orbit for short periods of time using a second-order Euler method. This avoids numerous calls to the ORBIT task during light-time correction computation.

If OBSMDL has been requested to compute partial derivatives, it calls TDR1WP to compute partial derivatives of TDRSS one-way Doppler observations at the given observation time.



[1]BASELINE DIAGRAM OF TDR1WM IS ON THE NEXT PAGE

Figure 4-10. Baseline Diagram of OBSMDL (1 of 2)

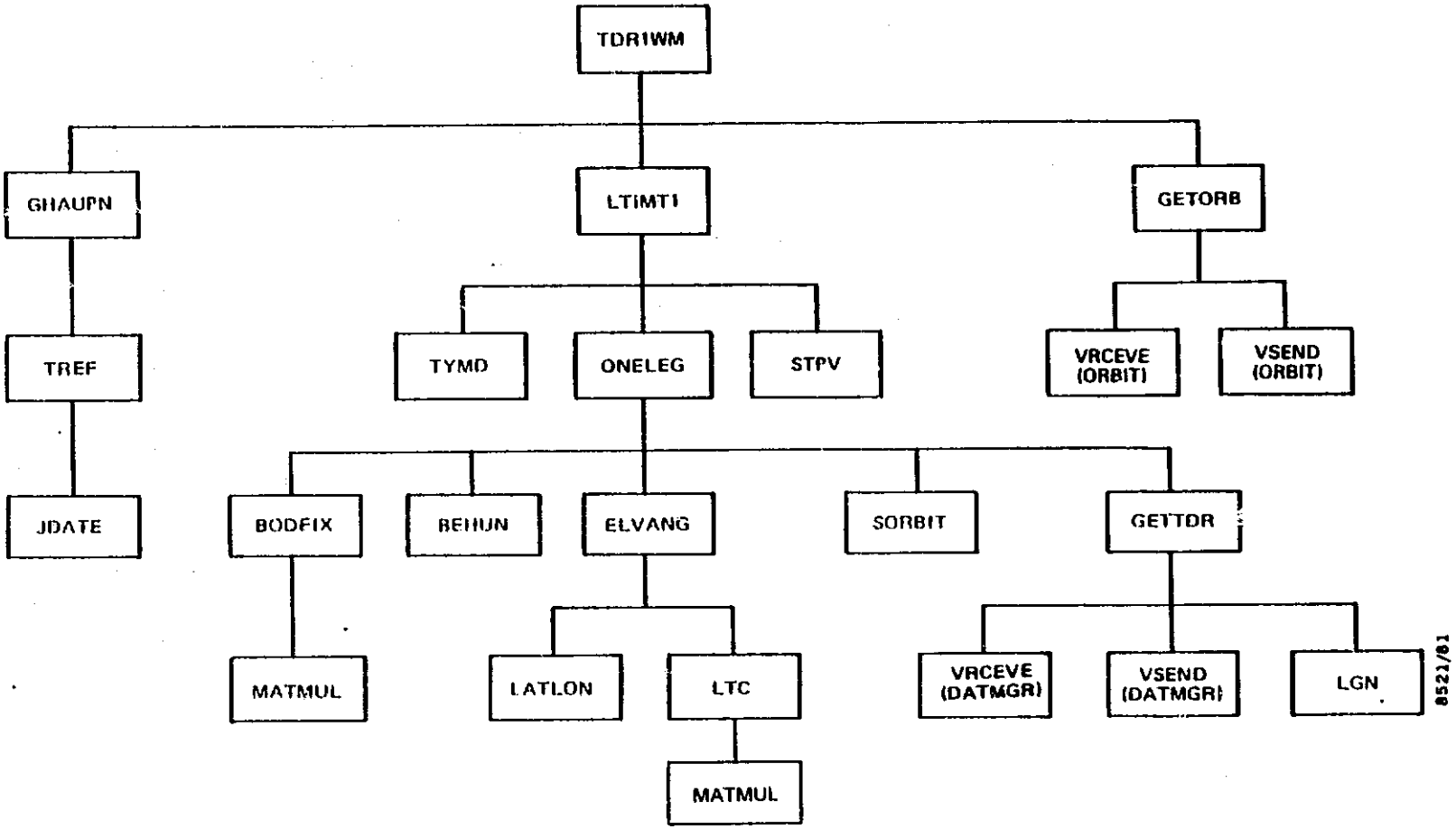


Figure 4-10. Baseline Diagram of OBSMDL (2 of 2)

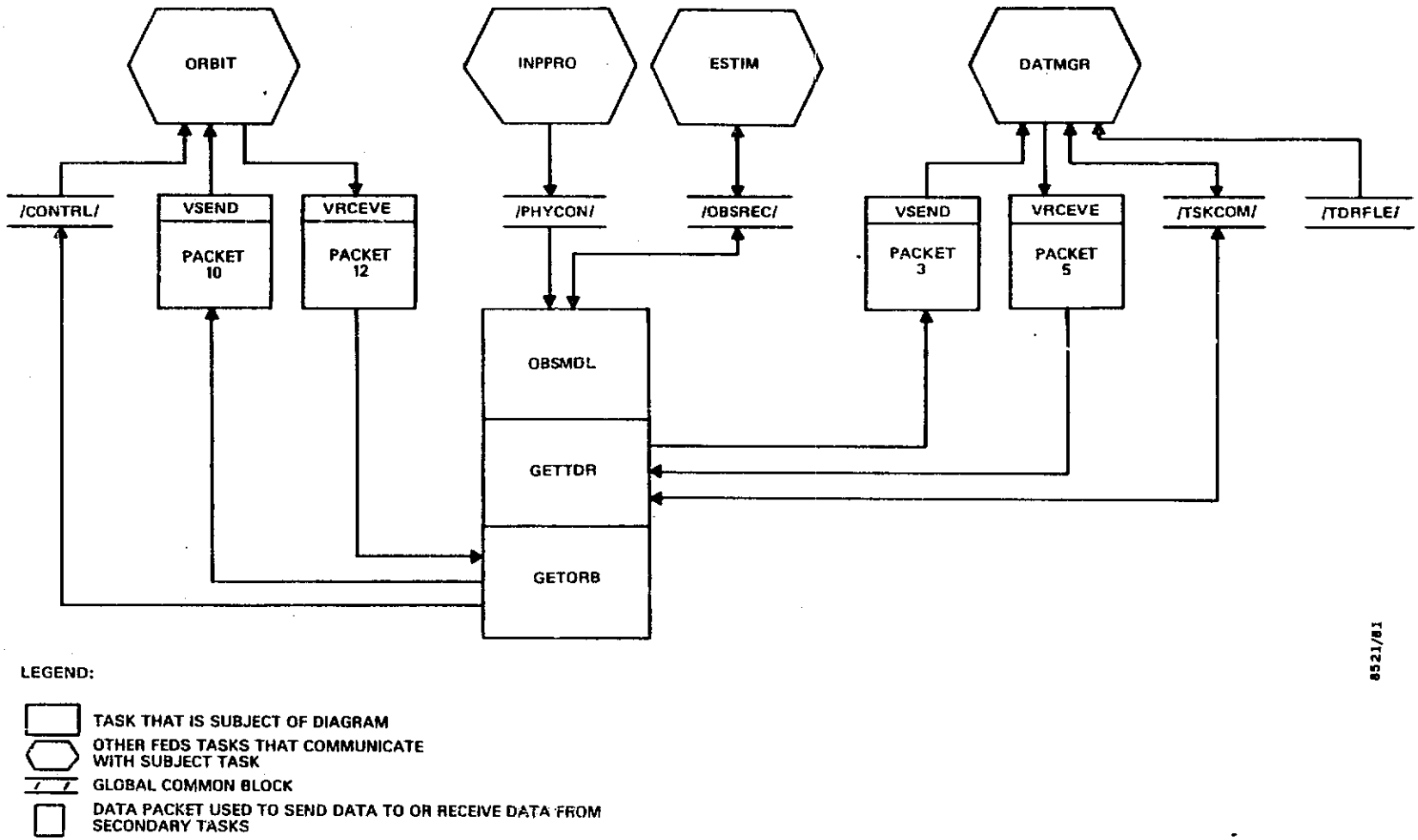


Figure 4-11. OBSMDL Data Flow

SECTION 5 - COMMUNICATIONS BOX

The Communications Box is a microprocessor-based piece of communications hardware that handles interfaces between FEDS on the PDP-11/23, a PB5 time code generator, and a second-generation transponder. Section 5.1 describes the Communications Box hardware. Section 5.2 discusses the interface functions of the Communications Box. Section 5.3 describes the operation of the Communications Box. Section 5.4 discusses a FEDS software testing program that simulates the functions of the Communications Box and transponder.

5.1 COMMUNICATIONS BOX HARDWARE

The Communications Box consists of a microprocessor and chassis, a CRT, a clock module, and a pulse generator. Together the clock module and pulse generator function as the PB5 time code generator. The CRT provides a means for a user to communicate with the Communications Box. The microprocessor controls the actions of the Communications Box. A block diagram for the Communications Box is given in Figure 5-1.

The microprocessor, an Intel 8085, resides in a single board computer (SBC-80/30) containing 8K of erasable, programmable read only memory (EPROM). The chassis containing the microprocessor and clock module also contains an interface board and terminal ports which provide input and output capabilities for data to and from FEDS and the transponder. The microprocessor can also access the clock module memory to obtain the PB5 time code.

5.2 COMMUNICATIONS BOX INTERFACE FUNCTIONS

The main functions of the Communications Box are to provide an interface between the transponder and FEDS and to access the PB5 time code generator. All actions taken by the Communications Box are in response to messages sent from FEDS or from the transponder. All message formats are provided in Appendix A.

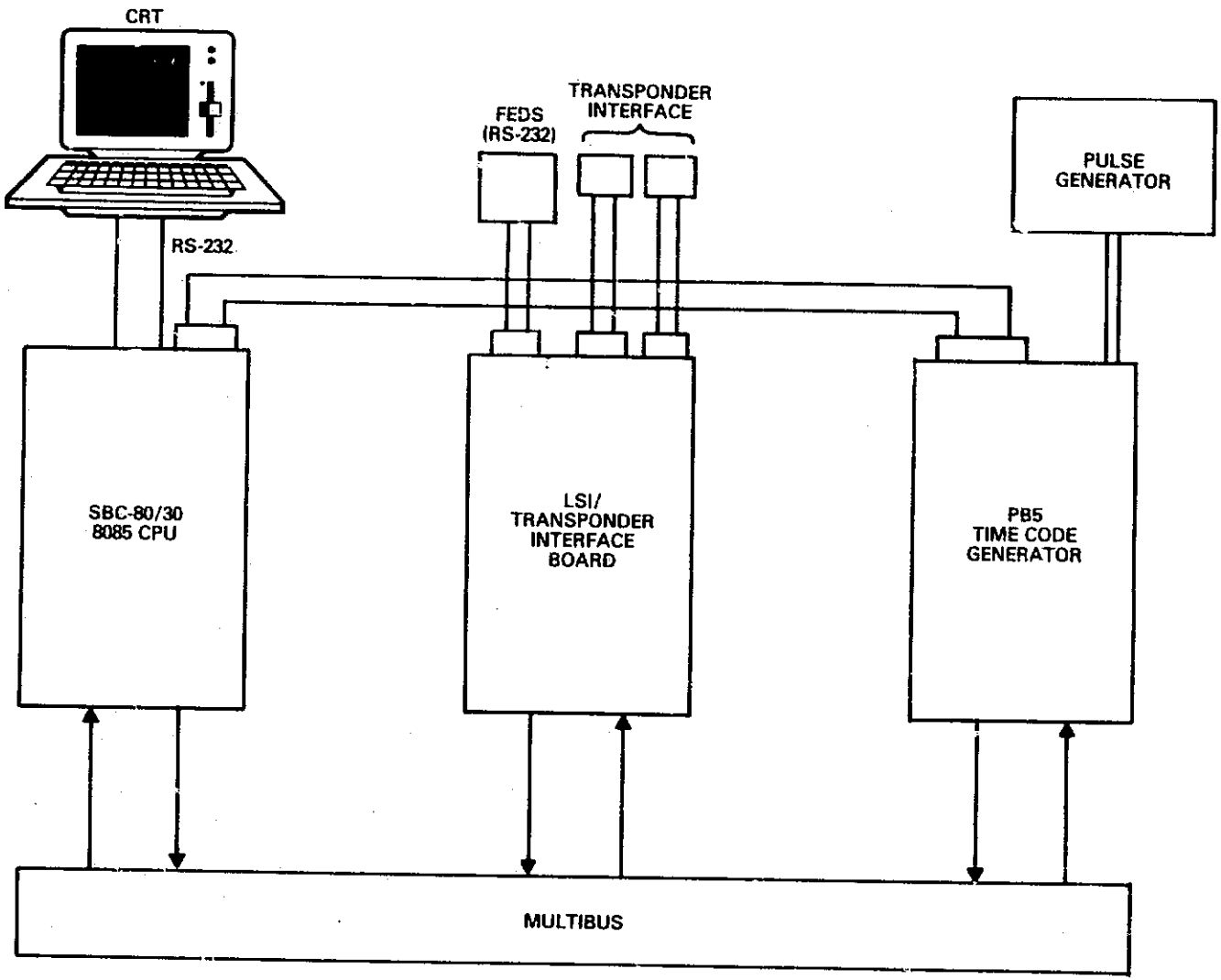


Figure 5-1. Communications Box Block Diagram

During initialization of the FEDS/Communications Box interface, FEDS transmits an initialization message to the Communications Box. The Communications Box responds by transmitting an initialization receipt message to FEDS.

At the beginning of the simulation and prior to each scheduled tracking pass, the Communications Box will receive a time request message from FEDS. The Communications Box will then obtain the PB5 time code from the time code generator and form and transmit the time message to FEDS. The Communications Box will also receive a reset Doppler accumulator message prior to each scheduled tracking pass. In response to this message, the Communications Box will set the Doppler reset input into the transponder.

When the transponder does not have a lock on the tracking signal during a scheduled tracking pass, the Communications Box will receive predicted Doppler messages at a preset frequency, nominally once every 6 seconds. The Communications Box will take two actions in response to this message. First, the frequency control word will be extracted from the predicted Doppler message and transmitted to the transponder. Then, a message indicating that a predicted Doppler message has been successfully processed will be transmitted to FEDS.

As soon as signal acquisition has occurred, the Communications Box receives a signal from the synchronization detected (sync detect) output of the transponder. The Communications Box responds by transmitting a signal acquisition message to FEDS.

During the tracking pass, the Communications Box collects observation data from the transponder and associated time tags from the PB5 generator and transmits this information to FEDS. After accumulating data over the Doppler averaging interval, the transponder will output a time strobe to the

Communications Box. The Communications Box immediately obtains the current PB5 time code and then clocks the 40-bit Doppler accumulator from the transponder over a serial port. If FEDS has transmitted a Doppler observation message, indicating FEDS is ready to receive an observation, the Communications Box immediately transmits the observation data message containing the Doppler accumulator and the PB5 time code to FEDs. Otherwise, the Communications Box will await the message from FEDS before transmitting the observation data. As soon as the transponder loses the lock on the tracking signal, the Communications Box receives a signal from the carrier lock port of the transponder and responds by transmitting a signal loss message to FEDS.

5.3 COMMUNICATIONS BOX OPERATION

The procedure to prepare the Communication Box for operation consists of five steps:

1. Connect terminal port TT1: on the PDP-11/23 to the designated terminal port on the Intel chassis with the cable provided.
2. Connect the Communications Box ports to the corresponding ports on the transponder.
3. Turn on the power switch on the Intel chassis front panel.
4. Turn on the power switch on the CRT.
5. After the cursor appears on the CRT, depress the "U" key repeatedly until the transponder interface menu appears (Figure 5-2). If the menu fails to appear, depress the reset button on the Intel chassis front panel and repeat this step.

Following this procedure, the Communications Box is prepared to initialize the FEDS interface and begin operation. The PB5 time code must now be initialized. For testing purposes,

```
D*  Display Memory (Dx, y)
G*  Go Command
Xr* Display/Set Register
M*  Move Memory
S*  Substitute Memory
N*  Single Step
I*  Insert Memory
T   Set Up Time
C   Resume Xponder Program
R   Initialize Xponder Routines
A   Display Time
H   Help Menu
Fx  Send Function Code to LSI
L   Send to CRT/LSI (C or L)

*Debug commands
```

Figure 5-2. Transponder Interface Menu

the "T" command (see Table 5-1) can be used. During the demonstration, a more accurate method of synchronizing the PB5 time code generator with Universal Time Coordinated (UTC) will be used. The "R" command (see Table 5-1) is then used to instruct the Communications Box to expect an initialization message from FEDS. If FEDS does not respond in the given time, the "R" command must again be entered before operation will begin. After FEDS has responded properly, the Communications Box will provide all interface functions described in Section 5.2.

Following completion of an execution, the Communications Box should be powered down by turning off the power switches on the Intel chassis front panel and the CRT.

Figure 5-2 shows the transponder interface menu. The use and function of each command is described in Table 5-1.

5.4 COMMUNICATIONS BOX SIMULATOR

The primary responsibility of the Communications Box Simulator (SIMCB) is to simulate the actions of the Communications Box and transponder that are pertinent to FEDS. SIMCB is requested by ADSIM when the Communications Box is not available for testing.

SIMCB receives, identifies, and transmits messages to and from FEDS. Valid messages that can be received from FEDS are

1. Initialization (function code 0)
2. Accumulator reset (function code 5)
3. Predicted Doppler commanded offset (function code 3)
4. Time request (function code 2)
5. Doppler request (function code 1)

Valid messages that can be transmitted to FEDS are

1. Carrier signal lock (function code 4)
2. Carrier signal lock lost (function code 6)
3. Initialization bit received (function code 0)

Table 5-1. Transponder Interface Commands (1 of 2)

COMMAND	DESCRIPTION	COMMAND SYNTAX	EXAMPLE
D Display Memory	Allows user to display any portion of memory on the CRT.	D<low address>,<high address> Carriage Return (cr)	[1]See example below; display lines too long for column.
Xr Display/Set Register	Allows user to display or set the 8085 CPU registers, program counter (p), or stack pointer(s).	X cr or Xr where r (in Xr) = register (a,b,c,d,e,h,l,m,p,s)	[2]See example below; display lines too long for column.
S Substitute Memory	Allows user to examine and optionally modify memory locations individually.	S<address> (space)	S4000 00-xx 11-yy 22-zz where xx, yy, zz are hex values
I Insert Memory	Allows user to insert large amounts of code into memory.	I<address> cr	[3]See example below; display lines too long for column.
G Go	Allows user to start program in RAM and set optional breakpoints. User must set up the instruction pointer to the start address before executing this command.	G[<start address>] [,-<breakpoint address>] [,-<breakpoint address>] cr	G cr Go from instruction pointer G4000 Go fro 4000 G4000--4010 Go fro 4000 til 4010 executed
M Move Memory	Allows user to move blocks of memory to the area of RAM, beginning at destination.	M<low address>,<high address>, <destination>	M4000, 41000, 6000
N Single Step	Allows user to single step through program one instruction at a time. PC register must be set to the address user wishes to step from.	N cr	—

0808-(02)/84

[1]D0,10 cr
0000 F3 C3 4F 00 F3 CD E8 05 C3 9C 03 37 0A 12 03
0010 F3

[2]X cr
A=01 B=02 C=03 D=04 E=05 F=16 H=FF L=FF M=FFFF P=1234 S=7F80
XA-01 cr

[3]I4000 cr
1234567890ABCDEF01234567890000012 (escape)
D4000,4010 cr
4000 12 34 56 78 90 AB CD EF 01 23 45 67 89 00 00 00
4010 12

Table 5-1. Transponder Interface Commands (2 of 2)

COMMAND	DESCRIPTION	COMMAND SYNTAX	EXAMPLE
T Set up PBS Time	Allows user to set time code generator.	Tcr	[4]See example below; display lines too long for column.
A Display PBS Time	Allows user to display time on the CRT.	Acr	A Julian day 1224 Seconds of day 83372 Ms of day 674 Time of day 23:56:30
R Initialize Xponder Interface	Initializes Transponder Interface Board and requests time delay to wait for response from FEDS.	Rcr	[5]See example below; display lines too long for column.
C Resume Xponder Program	Allows user to resume executing the transponder program used to simulate the transponder in integration testing. If user had interrupted the program for some reason, it can be restarted by using this command.	C	C 'Resume Xponder Program'
H Menu Display	Displays the Help menu.	H	—
F Send Function Code to LSI	Allows user to send function codes to the LSI.	Fx	Fzzzzzzzzzzzzzzzzzzzz x = 0 to 8 z's = function code value
L Send to CRT/LSI (C or L)	Allows user to send a function code to the LSI or to a CRT for debug purposes.	Lcr	[6]See example below; display lines too long for column.

[4]T
'Type in number of days' (message from interface)
1224 cr
'Type in hours & minutes'
2359
'Type in seconds'
30
Time is now entered into Timer Interface Board

[5]R
Initialize Xponder Interface
Enter delay in seconds for LSI's response (max 60 sec)
10

If FEDS responds correctly, then the message
'LSI SENT WAKEUP MESSAGE' is displayed on the CRT,
ELSE
'No Response from the LSI' is displayed

[6]L
'Enter C to test with CRT, L to test with LSI'
L
'Send to LSI'

The formats of all messages transmitted between SIMCB and FEDS are those used between FEDS and the Communications Box and are given in Appendix A.2.

Figure 5-3 illustrates the SIMCB data flow and Figure 5-4 presents a baseline diagram.

When SIMCB is initiated, it accesses a file of observations generated from Doppler predictor output during a previous FEDS execution. SIMCB reads the start and stop times of the first pass on the file, issues a QIO to read from the line connected to FEDS and places itself into a dormant state. SIMCB remains in a dormant state until a message is received. Upon receipt of the message, SIMCB identifies the message type and follows the appropriate path through the program.

If the received message was identified as an initialization message, SIMCB transmits an "initialization received" message and returns to the dormant state to wait for the next message. If the message is identified as an accumulator reset, SIMCB resets the accumulator to zero and returns to the dormant state to wait for another message. If the message is identified as a time request, SIMCB accesses the system clock; computes a simulation time in PB5 format based on the current system clock time, a reference system clock time, and a simulation reference time; transmits the PB5 time code to FEDS and returns to the dormant state to wait for another message.

If the message is identified as a predicted Doppler-commanded offset, SIMCB searches for the tracking schedule pass that contains the current simulation time. If no pass is found containing the current time, an error message is written to the terminal and SIMCB returns to the dormant state awaiting a message. If a pass is found, SIMCB will call LCKSET to estimate when signal lock will occur, based

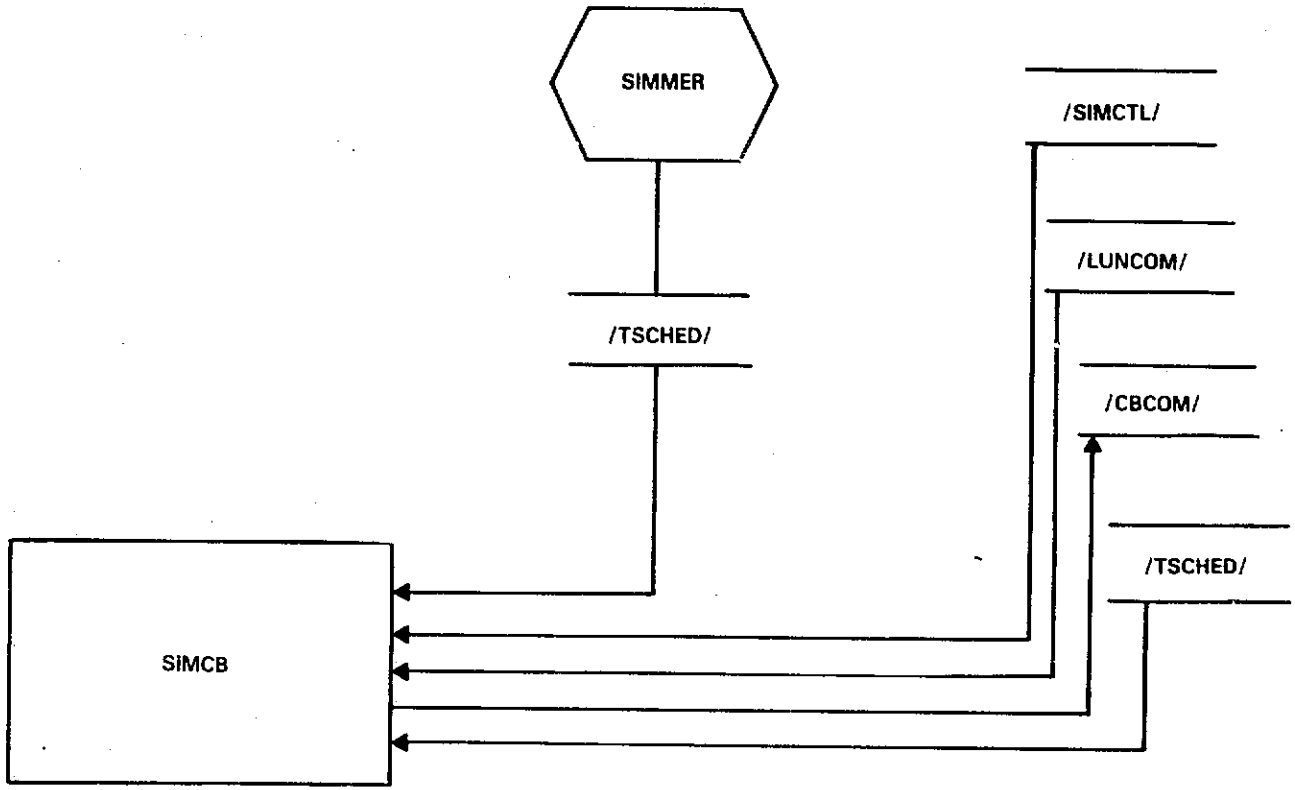


Figure 5-3. SIMCB Data Flow

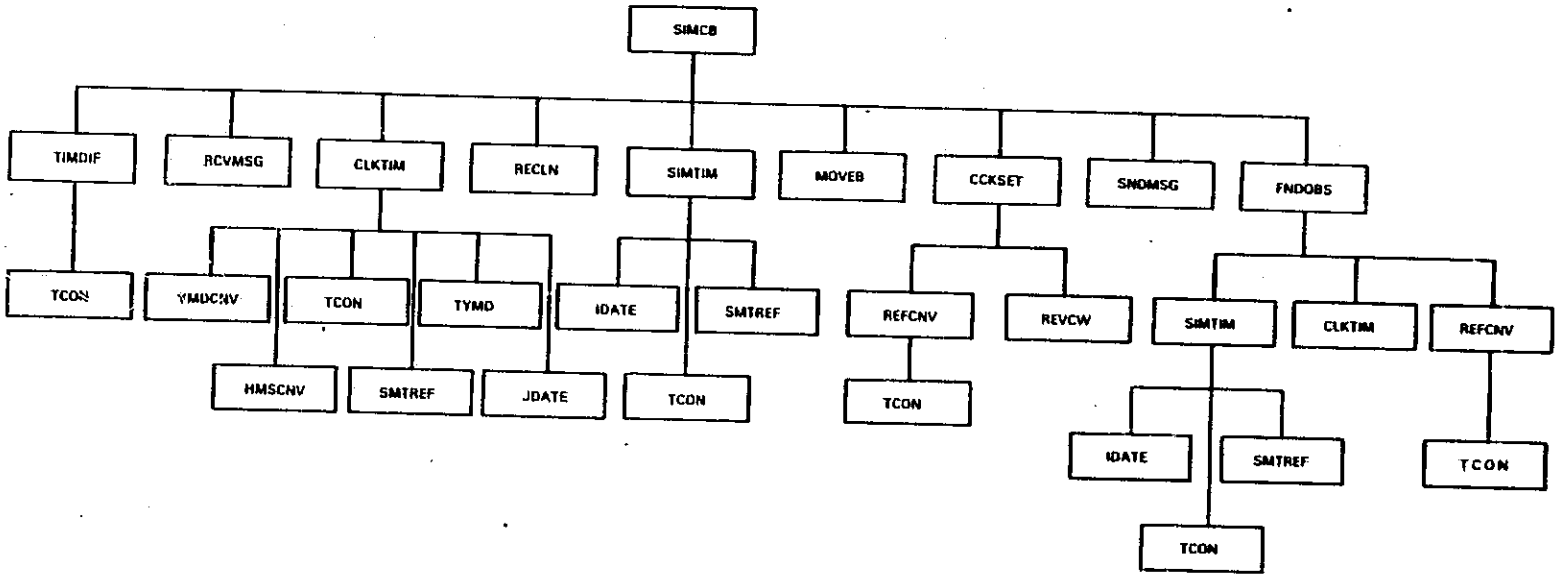


Figure 5-4. Baseline Diagram for SIMCB

on the commanded offset, the current time and data in the observations file. SIMCB then waits for either another message from FEDS or the simulation lock time.

If a message is received, SIMCB proceeds as indicated in the preceding discussion. If the signal lock time has passed, SIMCB will transmit a signal lock message to FEDS simulating acquisition of the tracking signal. SIMCB then waits for the end of the scheduled pass or a Doppler request. If SIMCB receives a message other than a Doppler request, it will send a lock message to FEDS and continue waiting. When a Doppler request message is received, SIMCB finds the first observation in the observations file that is time tagged after the current simulation time, scales and biases it, and adds it to a summation, modeling the accumulator. SIMCB then forms an observation message consisting of the observation time tag in PB5 format, the summation of observations in a 5-byte integer, and the proper function code and waits until the observation time tag or the scheduled end of the pass. When the observation time tag is reached, SIMCB transmits the observation message and waits to receive another Doppler request. Whenever the end-of-pass time is past, SIMCB sends a signal lock lost message to FEDS and waits for any message.

SECTION 6 - SYSTEM CONSTRUCTION AND OPERATION GUIDELINES

FEDS has been designed to support a demonstration of onboard orbit determination using observation data collected autonomously. Section 6.1 describes the expected configuration for the demonstration and testing. Section 6.2 details construction of FEDS, both for test systems and the operational system. Section 6.3 describes FEDS execution. (In this section the PDP-11/23 microcomputer is referred to by the name of its microprocessor, LSI-11/23.)

6.1 OPERATIONAL CONFIGURATIONS

During the FEDS demonstration, FEDS executing on the LSI-11/23 will be communicating with ADEPT by telecommunication lines and with a transponder through the Communications Box. FEDS will receive data messages and commands from ADEPT and will send output reports and messages to ADEPT. These messages are described in Appendix A.1. FEDS will transmit commands and predicted Doppler shift data to the Communications Box and receive data messages and flags from the Communications Box. These messages are described in Appendix A.2. In addition, FEDS will output status messages to a terminal to allow for simulation monitoring. These messages will be a compressed form of activity log messages, messages sent to or received from the Communications Box, and messages indicating task activity. These messages are described in Appendix B.

During the demonstration, observation data will be collected in a manner that simulates data collection on a satellite. A tracking signal will be transmitted from the White Sands Ground Terminal (WSGT). The tracking signal will be compensated at transmission time to offset the Doppler shift that would be observed by a satellite in the simulated orbit. The transponder will attempt to acquire the tracking signal

relayed by the TDRSS satellite, based on the frequency offset commanded by FEDS through the Communications Box. Both the LSI-11/23 running FEDS and the transponder will be located at GSFC.

Several data items will have to be consistent throughout the demonstration. A schedule of TDRS tracking passes has to be consistent between WSGT and FEDS. The schedule will be obtained by submitting a request for TDRS access at a TDRSS scheduling meeting and receiving a final schedule. The simulated user elements transmitted to FEDS by ADEPT at the start time of the demonstration must correspond to the user elements used by WSGT for Doppler compensation. It is expected that the user elements will be extracted from a Goddard Trajectory Determination System (GTDS) ephemeris tape provided by the Operations Support Computing Facility (OSCF). FEDS will therefore need to have a spacecraft model that is consistent with the model used in the generation of the tape.

TDRS elements must also be consistent. It is expected that ADEPT will transmit to FEDS a TDRS vector provided by the OSCF, and WSGT will use OSCF-determined TDRS elements transmitted over the NASA Communication Network. For timing consistency, the PB5 generator will have to be synchronized with UTC. Figure 6-1 presents the configuration for the demonstration.

Prior to the demonstration, a test will be executed to verify FEDS operational interfaces. Under this configuration, WSGT will transmit an uncompensated signal through the TDRSS satellite to the transponder. This test will verify the ability of FEDS to control signal acquisition, data collection, and signal loss under near demonstration conditions. Since Doppler compensation is not being performed, the user elements will not be needed at WSGT.

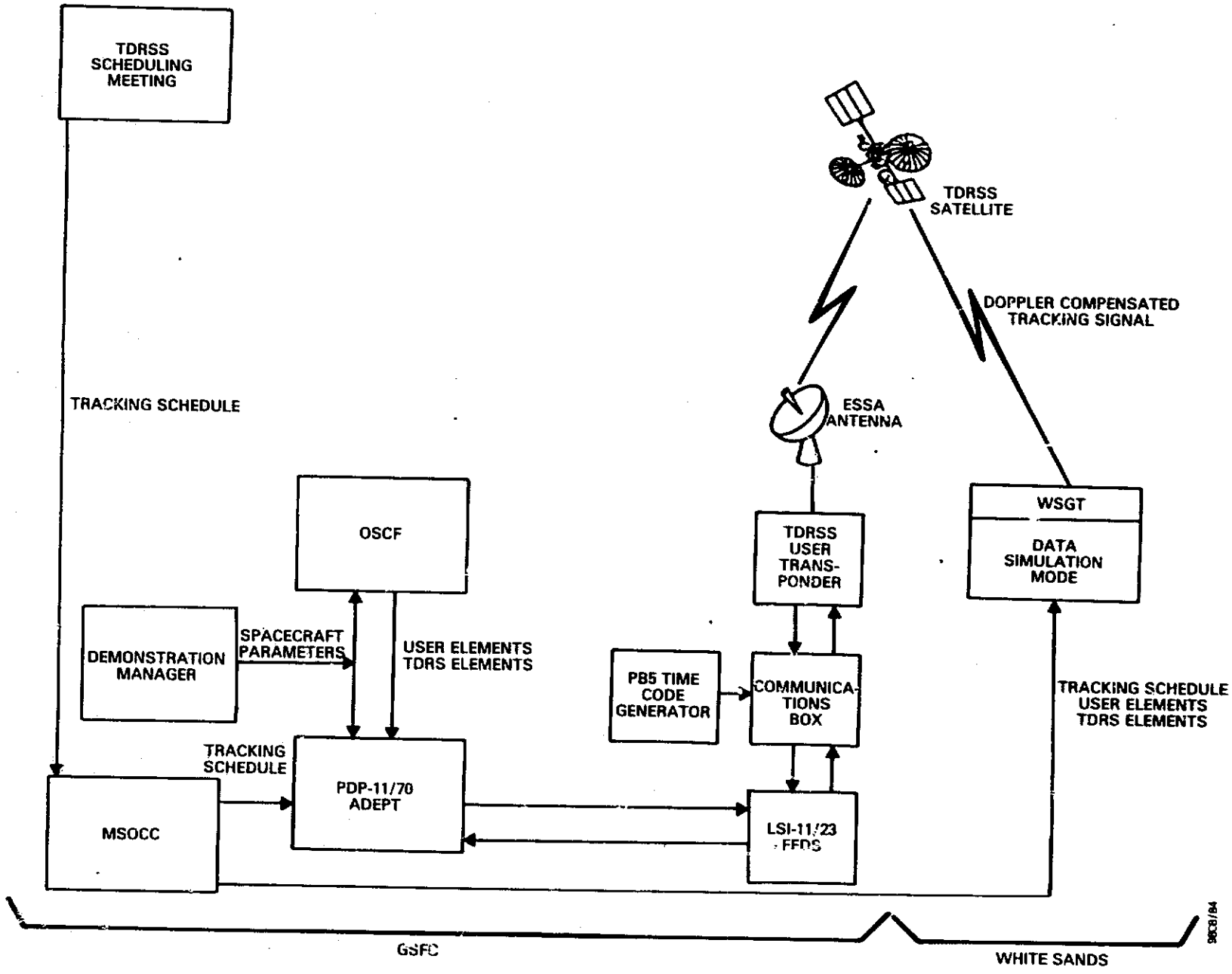


Figure 6-1. FEDS Demonstration Configuration

For testing during the development process and local testing of system modifications, two configurations are supported, one with FEDS executing on the LSI-11/23 and one with FEDS executing on the PDP. Under either of these configurations, all facilities and data required for FEDS execution reside in the System Technology Laboratory (STL). SIMCB provides all required messages from the Communications Box and observation data. Terminals are connected using cables. ADEPT will perform the same functions as in the demonstration configuration.

6.2 SYSTEM CONSTRUCTION

Two versions of FEDS are maintained. One version is to be executed on the LSI and the other executes along with ADEPT on the PDP-11/70. The LSI version of FEDS uses all four terminal ports on the LSI: two for communication with ADEPT, one for communication with the Communications Box or SIMCB, and one for a terminal. Because of a lack of available terminal ports, the PDP version simulates one of the lines between FEDS and ADEPT by using a system global event flag, VSEND in OUTPRO, and a VRCEVE in the ADEPT task RECEEV.

Command files are maintained to generate both versions of FEDS. A command file exists to act as the executive for compiling FEDS. The file, [224,1]COMPILE.CMD, prompts for the subsystem to be built, opens a corresponding file containing tne name of each subroutine to be compiled, and compiles them one by one until an error occurs in the compilation or an end of file is encountered. If an error is encountered, COMPILE will prompt for user input.

There are two ways to build the FEDS tasks for each version (LSI or PDP): one task at a time or all FEDS tasks with one command. To build tasks separately, task command files are available, each containing the task name followed by .CMD for the PDP version, or task name followed by 23.CMD for the

LSI version. The command file TKB.CMD will build all tasks for the PDP version and TKB23.CMD will build all tasks for the LSI version.

For the LSI version, a system image must also be built. To accomplish this, a user must log on under a privileged user identification code (UIC) and set the UIC to [1,64]. A system image is then initialized by executing the command file COP11S.CMD. The system is then generated by running VMR32, which will prompt for a command file. This prompt must be answered as follows:

ENTER FILENAME: @[224;1]FEDS23.CMD

Appendix D presents the command files referred to in this section.

6.3 SYSTEM OPERATION

Execution of the FEDS/ADEPT system is a three-step procedure. FEDS must first be loaded on the LSI or installed on the PDP. The simulator portion of ADEPT (ADSIM) must be installed and then system execution must be initiated by an operator.

Before FEDS is executed on the LSI, a system image must be created as described in Section 6.2. The user should then perform the following steps to downline load FEDS to the LSI-11/23:

1. Set the operator console (Hewlett Packard terminal) in the STL computer room for 300-baud rate and flip the corresponding switch on the terminal box to 300-baud rate.

2. Power up the LSI-11/23 using the POWER ON switch in the back of the computer cabinet, near the terminal ports. The LSI-11/23 will respond on the operator console as follows:

```
28
START?
```

To start the LSI-11/23, the user should enter "Y" followed by a carriage return.

3. Next, the user should verify that the communications lines between the PDP-11/70 and LSI-11/23 are connected properly. The proper connections for FEDS in demonstration configuration are detailed in Figure 6-2. Figure 6-3 shows the terminal port connections, using SIMCB.

4. At this point, the hardware is configured properly and the user is ready to download the system image. To do this, the user should enter the following commands under UIC[1,64]:

```
>REA LSI 2 TT32:
>LSI
LSI>BOOT RSX11S.SYS
```

At this time, the system image is downloaded into the LSI-11/23. This will take approximately 10 minutes, using a 9600-baud line. When loading is complete, FEDS will attempt to establish communication with the Communications Box or SIMCB and then wait for a start command from ADEPT.

Procedures that should be followed to minimize operational difficulties on the LSI are discussed in the following paragraphs. After the LSI power switch is turned on, the lines connecting the two computers must be cleared. This is done by the command

```
RUN [224,1] FLUSH
```

The LSI booting task must be terminated immediately after downloading of the system has been completed to prevent it from receiving data intended for ADEPT.

Installation of FEDS on the PDP is accomplished by executing the command file INSFEDS.CMD. This command file installs global common areas, terminal handlers, and FEDS task and fixes the tasks to correspond with the FEDS system image

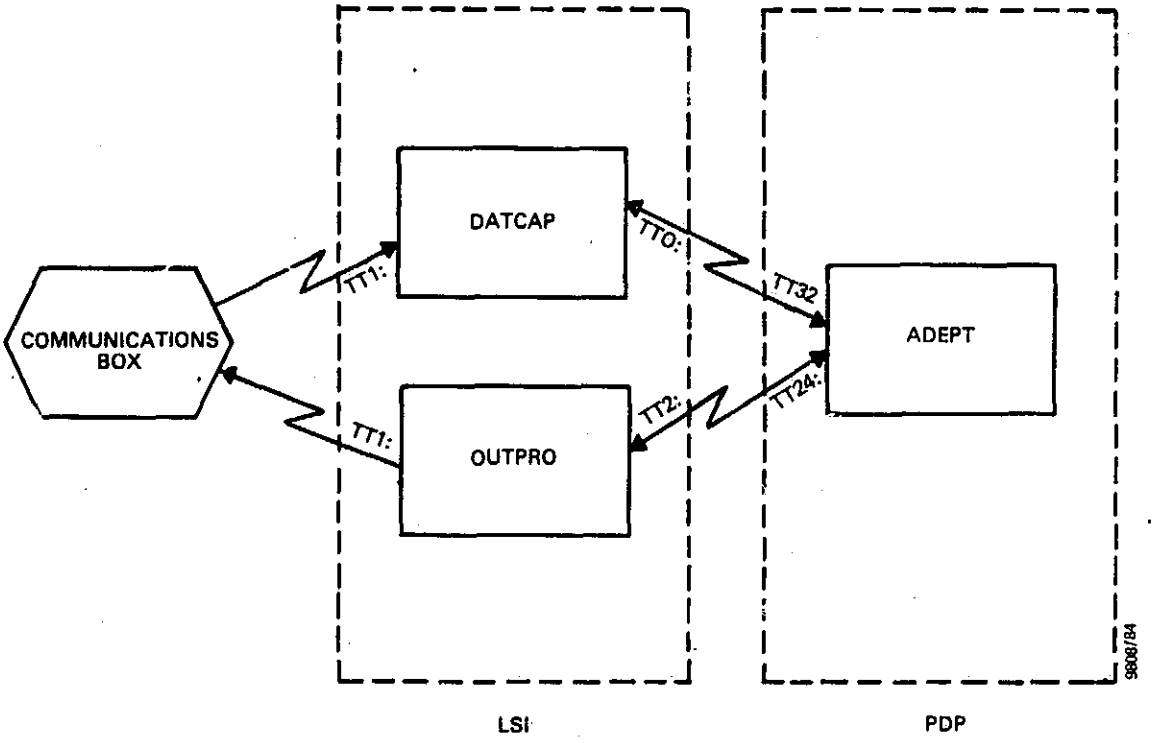


Figure 6-2. FEDS (on LSI) Communications Line Configuration, Communications Box Used

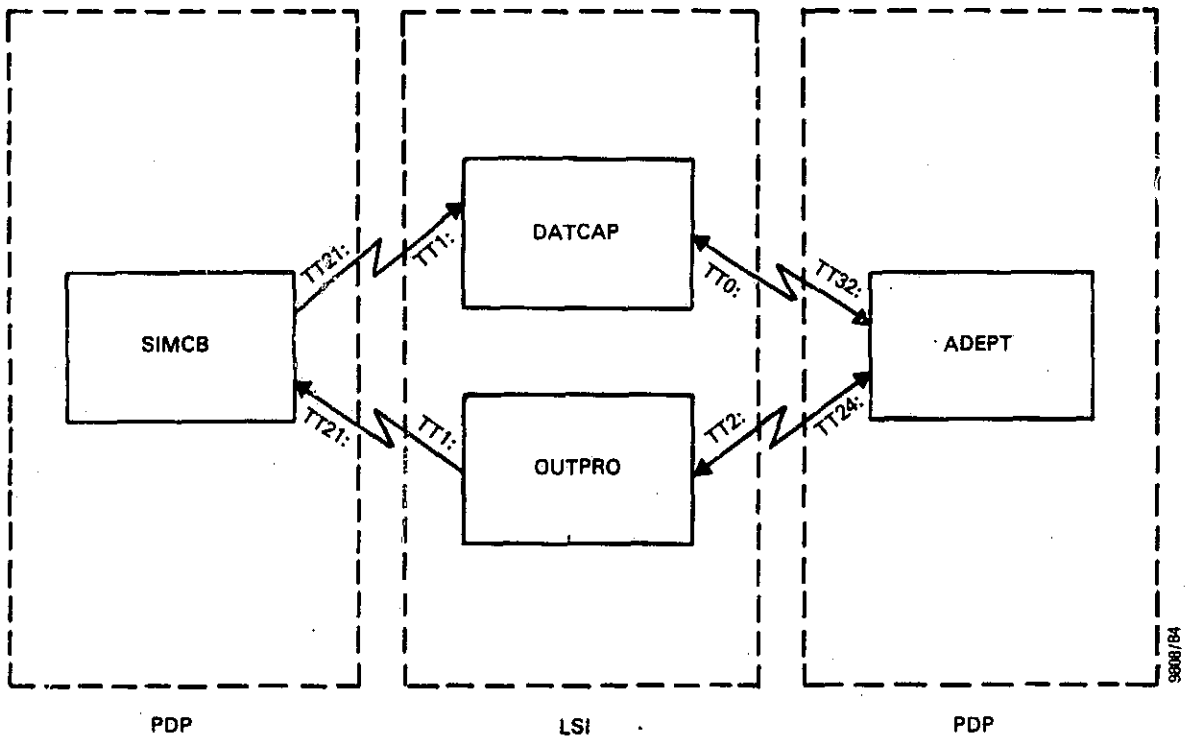


Figure 6-3. FEDS (on LSI) Communications Line Configuration, Communications Box Simulator Used

used on the LSI-11/23. The installation command file for FEDS on the PDP-11/70 is shown in Figure 6-4. Terminal port communications are shown in Figure 5-5.

Installation of ADEPT is also done by command file. For either FEDS configuration on the LSI-11/23, ADEPT is installed by the following command under UIC [224,2].

```
> @ADSLSI
```

The command file to install ADEPT on the PDP-11/70, ADSINS.CMD, is accessed by INSFEDS.CMD. These command files differ only in the task that is installed to capture data from OUTPRO. The command files to install the simulator portion of ADEPT are shown in Figures 6-6 and 6-7.

Nominally, operator interaction with ADEPT consists of entering a simulation ID, a verification of simulation configuration, and the instruction to begin. Data preparation for FEDS and optional operator interaction with ADEPT at run time are detailed in Reference 3.

To begin a run, the operator enters the following command:

```
> RUN ADSIM
```

ADSIM prompts the user to enter a simulation ID code for the simulation run as follows:

```
PLEASE INPUT A TWO CHARACTER SIMULATION IDENTIFICATION
CODE.  DO NOT INPUT BLANKS OR SPECIAL CHARACTERS>
```

The user then enters a two-character alphanumeric code for the simulation run, followed by a carriage return. The simulation code should not contain blanks or special characters. The simulation ID code should be the same as the simulation ID code used in data preparation.

```
INS [1,1]11SRES
RUN FLUSH
SET NOPAR:USRGB1
SET NOPAR:USRGB2
SET NOPAR:USRGB4
SET PAR:GLB3/BASE:3576/SIZE:400/COM
SET PAR:GLB4/BASE:4176/SIZE:600/COM
SET PAR:GLB1/BASE:4776/SIZE:200/COM
SET PAR:GLB2/BASE:6176/SIZE:200/COM
INS [1,1]GLB1
INS [1,1]GLB2
INS [1,1]GLB4
INS [1,1]GLB3
;
INS INPPRO.TSK/TASK=INPPRO/PRI=45.
INS OUTPRO.TSK/TASK=OUTPRO/PRI=45.
INS DATCAP.TSK/TASK=DATCAP/PRI=80.
INS EXEC.TSK/TASK=EXEC/PRI=53.
INS STAPRE.TSK/TASK=STAPRE/PRI=45.
INS ORBIT.TSK/TASK=ORBIT/PRI=52.
INS DATMGR/TASK=DATMGR/PRI=52.
INS DOPPRE/TASK=DOPPRE/PRI=45.
INS PREPRO/TASK=PREPRO/PRI=45.
INS ESTIM/TASK=ESTIM/PRI=45.
INS OBSMDL/TASK=OBSMDL/PRI=51.
;
;
FIX INPPRO
FIX OUTPRO
FIX DATCAP
FIX EXEC
FIX STAPRE
FIX ORBIT
FIX DATMGR
FIX DOPPRE
FIX PREPRO
FIX ESTIM
FIX OBSMDL
UIC 224 2
@ADSINS
;
RUN EXEC
>
```

Figure 6-4. Command File To Install FEDS on the PDP

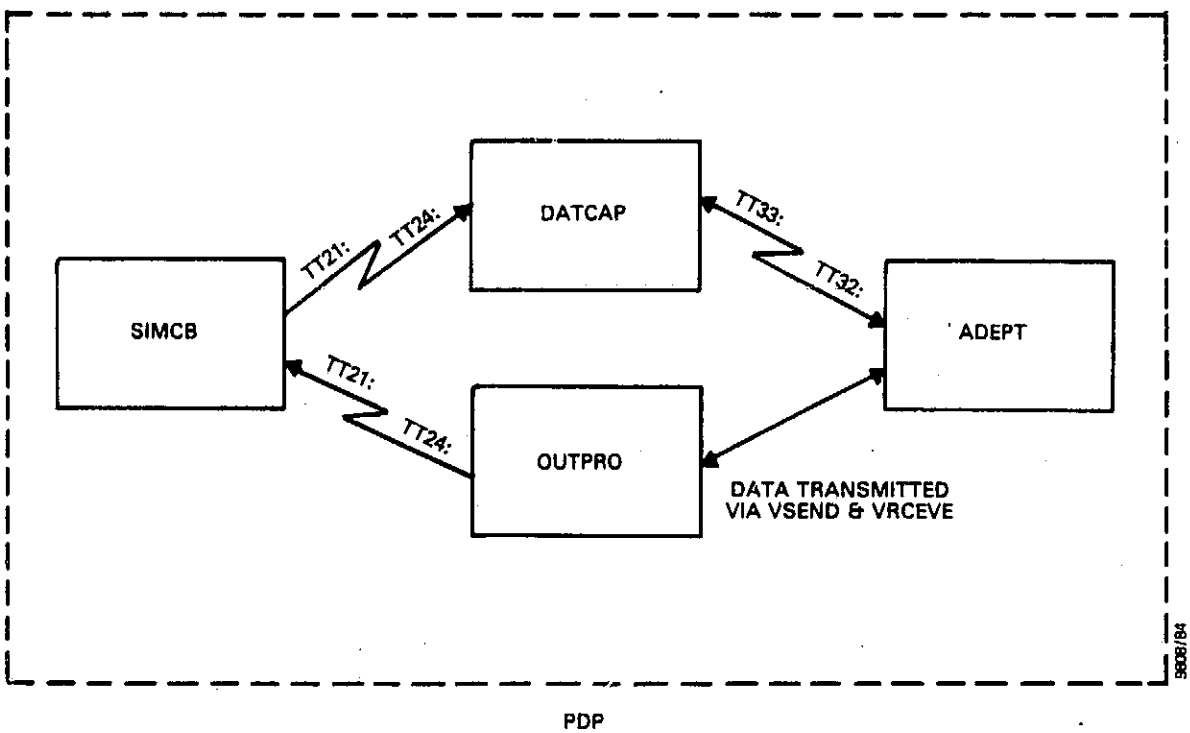


Figure 6-5. FEDS (on PDP) Communications Line Configuration, Communications Box Simulator Used

```
;COMMAND FILE TO INSTALL ADSIM TASKS
;   ADSINS.CMD
SET NOPAR:USRGB5
SET PAR:ADSGBL/BASE:6376/SIZE:200/COM
INS [1,1]ADSGBL/NOCHECK
INS ADSIM
INS SIMMER/NOCHECK/PRI=53.
INS DNLINK/NOCHECK
INS DNHIST/NOCHECK
INS SCREEN/NOCHECK
INS RECEEV70/NOCHECK/TASK=RECEEV
INS SIMCB/NOCHECK/PRI=54.
FIX SIMMER
FIX DNHIST
FIX RECEEV
FIX DNLINK
^
```

Figure 6-6. ADSINS.CMD (on PDP-11/70)

```
;COMMAND FILE TO INSTALL ADSIM TASKS
;    ADSINS.CMD
SET NOPAR:USRGB5
SET PAR:ADSGBL/BASE:6376/SIZE:200/COM
INS [1,1]ADSGBL/NOCHECK
INS ADSIM
INS SIMMER/NOCHECK/PRI=51.
INS DNLINK/NOCHECK
INS DNHIST/NOCHECK
INS SCREEN/NOCHECK
INS RECEEV/NOCHECK
INS SIMCB/NOCHECK/PRI=52.
FIX SIMMER
FIX DNHIST
FIX RECEEV
FIX DNLINK
^
```

Figure 6-7. ADSLSI.CMD (on LSI-11/23)

ADSIM then prompts for verification of simulation configuration as follows:

```
ADSIM - DATA OBSERVATION SIMULATION MODE IS COMMUNICATIONS BOX SIMULATOR (1)
DO YOU WISH TO CHANGE CURRENT DATA OBSERVATION ACCESS METHOD [Y/N]?
```

Three sources for observation data exist:

1. Communications Box simulator
2. Actual Communications Box
3. Observation file used to support AODS

Under normal situations, the operator will not wish to change the source of observation data. The simulator main menu will then be displayed as follows:

```
SIMULATOR MAIN MENU

1  INITIALIZE SIMULATION
2  BEGIN SIMULATION
3  RESUME A PREVIOUS SIMULATION
4  TERMINATE SIMULATION

INPUT COMMAND >
```

The operator will enter a 2 followed by a carriage return and simulation will begin. If the operator wishes to change the source of observation data, FEDS will prompt for the new data source before displaying the simulator main menu.

To terminate a simulation, the operator instructs the command file to abort and remove all ADEPT and FEDS tasks by entering the command

```
> @ [224,1]REMOVE
```

This command file, shown in Figure 6-8, removes all tasks that would be installed on the PDP under any configuration. FEDS tasks on the LSI are terminated by turning the power switch OFF. Figures 6-9 through 6-12 show the command files this command accesses.

```
UIC 224 2
@ADSABO
@ADSREM
UIC 224 1
@ABOFEDS
@REMFEDS
>
```

Figure 6-8. REMOVE.CMD (To Abort and Remove All Installed FEDS/ADEPT Tasks)

```
ABO/T EXEC
ABO/T DATCAP
ABO/T OUTPRO
ABO/T INPPRO
ABO/T STAPRE
ABO/T ORBIT
ABO/T DOPPRE
ABO/T DATMGR
ABO/T PREPRO
ABO/T ESTIM
ABO/T OBSMDL
>
```

Figure 6-9. ABOFEDS.CMD (Abort All FEDS Tasks)

```
ABO/T SIMCB
ABO/T ADSIM
ABO/T SIMMER
ABO/T DNLINK
ABO/T DNHIST
ABO/T SCREEN
ABO/T RECEEV
UNL SCHDULH1.DAT
```

Figure 6-10. ADSABO.CMD (Abort All Installed ADEPT Tasks)

```
REM OUTPRO
REM INPPRO
REM DATCAP
REM EXEC
REM ORBIT
REM STAPRE
REM DOPPRE
REM DATMGR
REM PREPRO
REM ESTIM
REM OBSMDL
;
SET NOPAR:GLB1
SET NOPAR:GLB2
SET NOPAR:GLB4
SET NOPAR:GLB3
SET PAR:USRGB1/BASE:3576/SIZE:1200/COM
SET PAR:USRGB2/BASE:4776/SIZE:200/COM
SET PAR:USRGB4/BASE:6176/SIZE:200/COM
;
;
>
```

Figure 6-11. REMFEDS.CMD (Remove All FEDS Tasks)

```
;   ADSREM.CMD
REM SIMCB
REM ADSIM
REM SIMMER
REM DNLINK
REM DNHIST
REM SCREEN
REM RECEEV
SET NOPAR:ADSGBL
SET PAR:USRGB5/BASE:6376/SIZE:200/COM
>
```

Figure 6-12. ADSABO.CMD (Remove All Installed ADEPT Tasks)

APPENDIX A - EXTERNAL INTERFACES

This appendix details the formats of the messages transmitted to support FEDS. Section A.1 describes the messages between FEDS and ADEPT. Section A.2 describes the messages between FEDS and the Communications Box or SIMCB. Section A.3 describes the messages between the transponder and the Communications Box.

A.1 ADEPT/FEDS INTERFACE

This section contains the uplink and downlink message formats through which ADEPT communicates with FEDS. Figure A-1 shows the standard data transmission format that is used for both uplink and downlink; Figure A-2 illustrates the transmission record format. A definition of terms used in Figure A-1 and throughout the section is provided below:

Term	Definition
Transmission	Set of one or more blocks of data that are transmitted contiguously. A transmission is always terminated by an end-of-transmission record (all -1s).
Block	Set of one or more data records that contain the same type of data.
Record	A 256-byte record containing a header (20 bytes) and one or more frames of data (see Figure D-2).
Frame	One entity of data.
Header	A 20-byte header frame that describes the contents of the record.

The message formats given here supersede those given in Appendix D of Reference 4.

A.1.1 UPLINK MESSAGES FORMATS

This section contains the uplink message formats through which data and commands are uplinked to FEDS. The format of the record header (first 20 bytes), which is common to all uplinked messages, is given on Page A-5, and the message block attributes and the frame format for each type of input data and command are presented on the following pages.

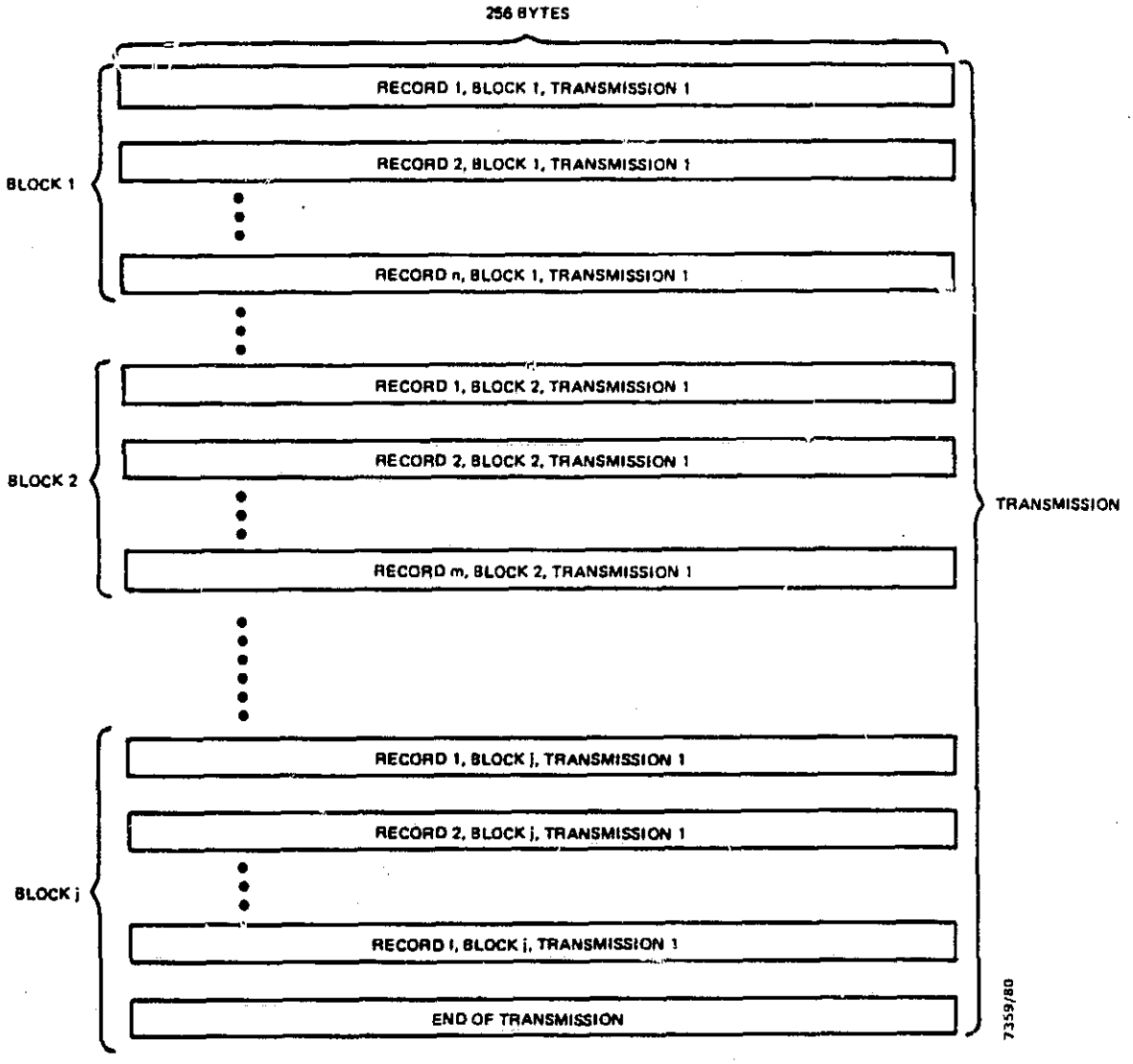


Figure A-1. Data Transmission Format

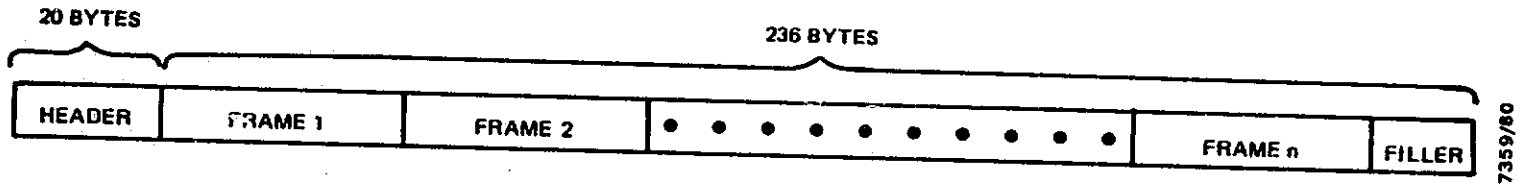


Figure A-2. Transmission Record Format

RECORD HEADER

FRAME FORMAT:

Variable	Type	Dimension	Description
IDSC	Byte	1	First synchronization code
IDEX	Byte	1	Second synchronization code
INTYPE	Byte	1	Type of input: = 1, data = 2, code = 3, command
INDATA	Byte	1	Type of data: = 1, not used in FEDS = 2, initialization table = 3, new TDRS vector = 4, estimation control parameters = 5, maneuver schedule = 6, tracking schedule = 7, miscellaneous constants = 8, station constants = 9, geopotential tables = 10, atmospheric drag = 11, timing coefficients = 12, experiment parameters
NBLOCK	Byte	1	Running number of records in block
MBLOCK	Byte	1	Total number of records in block
NTRAN	I*2	1	Running number of records in transmission
IDBLCK	I*2	1	Block ID number
NSIZE	I*2	1	Number of bytes used in record
TTRAN	R*8	1	Time of transmission (seconds from reference time)

EXPERIMENT PARAMETERS INPUT MESSAGE

1 frame = 1 experiment parameters set

1 record = header + 1 frame + fill
256 = 20 + 60 + fill
256 = 80 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
FRACC	R*8	5	Frequency associated with access method I
DLTOBS	R*8	1	Time period between observation messages from transponder
IFRACC	I*2	2	Access method associated with Ith TDRS
IDGRS	I*2	2	Ground station associated with Ith TDRS
IDT	I*2	2	TDRS ID associated with Ith TDRS

INITIALIZATION TABLE INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 initialization table (188 bytes)

1 record = 1 header + frame + fill
256 = 20 + 188 + fill
256 = 208 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
REFTM	R*8	1	Reference time
REFAPR	R*8	10	A priori state vector
REFSTD	R*8	10	A priori standard deviation
MAP	I*2	10	Solve-for/consider map

ESTIMATION CONTROL PARAMETERS INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = estimation control parameters set (152 bytes)

1 record = header + 1 frame + fill
256 = 20 + 152 + fill
256 = 172 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
DCSPAN	R*4	1	Estimation timespan (size of batch of data in seconds)
OBSSMP	R*2	1	Sample frequency for observations (seconds)
SEMULT	R*4	1	S_e multiplier for inner loop editing
TMLEAD	R*4	1	Lead time for DC precomputation (seconds)
MAXITR	I*2	1	Maximum number of iterations to be performed per slide
INLOOP	I*2	1	Maximum number of inner edit loops allowed
OBSSTD(I,J)	R*4	2,5	Observation standard deviations (only OBSSTD(2,1) is used in FEDS) I = measurement type: = 1, range = 2, Doppler J = observation type: = 1, one-way TDRSS = 2, two-way TDRSS = 3, three-way TDRSS = 4, one-way SRE = 5, two-way SRE
IROUT	I*2	1	Residuals report output control switch: = 0, no report generated = 1, report generated after last iteration on a batch of data

Variable	Type	Dimension	Description
IROUT (Cont'd)			= 2, report generated after first and last inner edit loops at each iteration on a batch of data
IDCOUT	I*2	1	DC Summary and Statistics Report output control switch: = 0, no report generated = 1, report generated after last iteration on a batch of data = 2, report generated after every iteration
RESMAX(I,J)	R*4	2,5	Maximum observed-minus-computed value for each observation type (only, RESMAX(2,1) is used in FEDS) I = measurement type J = observation type
ELVMIN	R*4	1	Maximum acceptable elevation angle for SRE data (degrees); not used in FEDS
RAYANG	R*4	1	Maximum acceptable ray path angle for TDRSS data (degrees)
RAYHGT	R*4	1	Minimum acceptable ray path height for TDRSS data types (kilometers)
EDTOL	R*4	1	Edit test tolerance
PCONV	R*4	1	Position correction convergence tolerance
VCONV	R*4	1	Velocity correction convergence tolerance
SECONV	R*4	1	S_e convergence tolerance
POSDIV	R*4	1	Maximum allowable position correction
VELDIV	R*4	1	Maximum allowable velocity correction
RATCOR	R*4	1	Position and velocity correction differences multiplier
POSLIN	R*4	1	Position linearity tolerance
VELLIN	R*4	1	Velocity linearity tolerance

NEW TDRS VECTOR(S) INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = new TDRS vector for 1 TDRS (60 bytes)

1 record = header + 1 (2) frame(s) + fill
256 = 20 + 60 (120) + fill
256 = 80 (140) + fill

1 block = 1 record (1 of 2 frames defined at transmission)

FRAME FORMAT:

Variable	Type	Dimension	Description
TDRTIM	R*8	1	TDRS reference time (YYMMDDHHMMSS.SS)
TDRSX	R*8	6	New TDRS position and velocity vectors
IDTDRS	I*2	1	TDRS ID
VECTYP	I*2	1	Type of input vector: = 0, new estimate of TDRS vector = 1, update to previous TDRS maneuver

MANEUVER SCHEDULE INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 scheduled maneuver (58 bytes)

1 record = header + 4 frames + fill
256 = 20 + 232 + fill
256 = 252 + fill

1 block = 2 records

FRAME FORMAT:

Variable	Type	Dimension	Description
TIM01	R*8	1	Time of maneuver (YYMMDDHHMMSS.SS)
XM01	R*8	6	Predicted state (position and velocity) after maneuver
MSID01	I*2	1	ID of maneuvered spacecraft (TDRS ID for TDRS, SIC and VID for user spacecraft)

TRACKING SCHEDULE INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES

1 frame = schedule for 1 tracking interval (22 bytes)

1 record = header + 8 frames + fill
256 = 20 + 8 x 22 + fill
256 = 20 + 176 + fill
256 = 196 + fill

1 block = 2 records

FRAME FORMAT:

Variable	Type	Dimension	Description
STIME	R*8	1	Start time of tracking interval (YYMMDDHHMMSS.SS)
ETIME	R*8	1	End time of tracking interval (YYMMDDHHMMSS.SS)
OBSFRQ	R*4	1	Observation frequency
IDPTDR	I*2	1	ID of TDRS to be used for one-way Doppler prediction during this interval

MISCELLANEOUS CONSTANTS INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = set of constants

1 record = header + 1 frame + fill
256 = 20 + 170 + fill = 190 bytes

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
EQTRAD	R*8	1	Equatorial radius
FLAT	R*8	1	Flattening coefficient
OMEGA	R*8	1	Rotation rate of Earth
PI	R*8	1	π
REFJUL	R*8	2	Reference time of Julian date (used with timing coefficients)
RTD	R*8	1	Radians-to-degrees conversion constant
TBIASS	R*8	1	Timing bias for user spacecraft
TFREQ(I)	R*8	5	Table used to compute pilot frequency for the following access methods (not used in FEDS) I = 1, multiple-access (MA) I = 2, S-band single-access link (SSA1) I = 3, SSA2 I = 4, K-band single-access link (KSA1) I = 5, KSA2
VLITE	R*8	1	Velocity of light
SCAREA	R*4	1	User spacecraft area
SCMASS	R*4	1	User spacecraft mass
SFLUX	R*4	1	Solar flux value
SPFREQ	R*4	1	State vector frequency in predict table (minutes) (default = 1 minute)

Variable	Type	Dimension	Description
SPINT	R*4	1	State vector frequency in predict table (minutes) (default = 30 minutes)
SOLRAD(I)	R*4	2	Solar radiation pressure for TDRS I
STEPSZ(1)	R*4	2	Integration step size: I = 1, target I = 2, TDRS
TDAREA(I)	R*4	2	Area of TDRS I
TDMASS(I)	R*4	2	Mass of TDRS I
TPAD	R*4	1	Time pad for output of predicted one-way Doppler data (minutes)
ACTFLG	Byte	1	Activity log generation switch: = 0, off = 1, on
IFRAC	Byte	1	Refraction Switch: = 0, off = 1, on
NDRAG	Byte	1	Drag switch for target: = 0, off = 1, on
NOOM(I)	Byte	2	Moon switches: I = 1, target I = 2, TDRS (= 0, off; = 1, on)
NSOLRP	Byte	1	Solar radiation pressure switch for TDRS: = 0, off = 1, on
NSUN(I)	Byte	2	Sun switches: I = 1, target I = 2, TDRS
IDEX1	Byte	1	Vehicle Id (VID) for user spacecraft
IDSC1	Byte	1	Support identification code (SIC) for use spacecraft

STATION PARAMETERS INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

Set of constants = 946 bytes

Record 1 = header + frame 1 (234 bytes) = 254 bytes + fill
Record 2 = header + frame 2 (192 bytes) = 212 bytes + fill
Record 3 = header + frame 3 (176 bytes) = 196 bytes + fill
Record 4 = header + frame 4 (200 bytes) = 220 bytes + fill
Record 5 = header + frame 5 (144 bytes) = 164 bytes + fill

1 block = 5 records

FRAME 1 FORMAT:

Variable	Type	Dimension	Description
NSTA	I*2	1	Total number of stations used
IDSTA(J)	I*2	20	Station IDs in order corresponding to constants in following arrays
STAT(I,J)	R*8	3,8	Constants for station J, where J = 1 through 8: I = 1, Earth-fixed position component-X I = 2, Earth-fixed position component-Y I = 3, Earth-fixed position component-Z

FRAME 2 FORMAT:

Variable	Type	Dimension	Description
STAT(I,J)	R*8	3,8	Constants for station J, where J = 9 through 16 (see frame 1 format, above)

FRAME 3 FORMAT:

Variable	Type	Dimension	Description
STAT(I,J)	R*8	3,4	Constants for station J, where J = 17 through 20 (see frame 1 format, above)
FREQB(J)	R*4	20	Station-dependent frequency bias (hertz) for SRE data types for station J, where J = 1 through 20; not used in FEDS

FRAME 4 FORMAT:

Variable	Type	Dimension	Description
ANTCOR(J)	R*4	20	Antenna mount correction (kilometers) for SFE data types for station J, where J = 1 through 20; not used in FEDS
MREFRC(I,J)	Byte	12,10	Monthly surface refractivity values for station J, where J = 1 through 10, and month of year I, where I = 1 to 12

FRAME 5 FORMAT:

Variable	Type	Dimension	Description
MREFRC(I,J)	Byte	12,10	Monthly surface refractivity values for station J, where J = 11 through 20 (see frame 4 format, above)
ANTALG(J)	Byte	20	Antenna alignment indicator for station J, where J = 1 through 20; not used in FEDS
TDELAY	R*4	1	User spacecraft transponder delay (kilometers)

GEOPOTENTIAL TABLES INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

Total set of constants = 1032 bytes

Record 1 = header + frame 1 (232 bytes) = 256 bytes
Record 2 = header + frame 2 (200 bytes) = 220 bytes
Record 3 = header + frame 3 (200 bytes) = 220 bytes
Record 4 = header + frame 4 (200 bytes) = 220 bytes
Record 5 = header + frame 5 (200 bytes) = 220 bytes

1 block = 5 records

FRAME 1 FORMAT:

Variable	Type	Dimension	Description
MORD(I)	Byte	2	Order of harmonic field: I = 1, target I = 2, TDRS
MDEG(I)	Byte	2	Degree of harmonic field: I = 1, target I = 2, TDRS
GM	R*8	1	Point mass
XJ	R*4	15	Zonal harmonics (J_1 through J_{15})
CS	R*4	40	First 40 C- and S-harmonic coefficients (C-harmonic coefficients in lower triangle of 15-by-16 matrix; S-harmonic coefficients in upper triangle of 15-by-16 matrix) for 15-by-15 geopotential model

FRAME 2 FORMAT:

Variable	Type	Dimension	Description
CS	R*4	50	Next 50 C- and S-harmonic coefficients (C-harmonic coefficients in lower triangle; S-harmonic coefficients in upper triangle) of 15-by-15 model

FRAME 3 FORMAT:

Variable	Type	Dimension	Description
CS	R*4	50	Next 50 C- and S-harmonic coefficients (C-harmonic coefficients in lower triangle; S-harmonic coefficients in upper triangle) of 15-by-15 model

FRAME 4 FORMAT:

Variable	Type	Dimension	Description
CS	R*4	50	Next 50 C- and S-harmonic coefficients (C-harmonic coefficients in lower triangle; S-harmonic coefficients in upper triangle) of 15-by-15 model

FRAME 5 FORMAT:

Variable	Type	Dimension	Description
CS	R*4	50	Next 50 C- and S-harmonic coefficients (C-harmonic coefficients in lower triangle; S-harmonic coefficients in upper triangle) of 15-by-15 model

ATMOSPHERIC DENSITY TABLES INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

Total set of data = 662 bytes

Record 1 = header + frame 1 (234 bytes) = 254 + fill
Record 2 = header + Frame 2 (224 bytes) = 244 + fill
Record 3 = header + Frame 3 (144 bytes) = 164 + fill

1 block = records

FRAME 1 FORMAT:

Variable	Type	Dimension	Description
NDENS	I*2	1	Number of entries in density table
NALT(J)	I*2	60	Altitude associated with density intervals (in ascending order)
DENSTY(I,J)	R*4	2,14	First 14 intervals in density table: I = 1, minimum density associated with NALT(J) I = 2, maximum density associated with NALT(J) (where J = 1 through 14)

FRAME 2 FORMAT:

Variable	Type	Dimension	Description
DENSTY(I,J)	R*4	2,28	Next 28 intervals in density table (where J = 15 through 42)

FRAME 3 FORMAT:

Variable	Type	Dimension	Description
DENSTY(I,J)	R*4	2,18	Last 18 intervals in density table (where J = 43 through 60)

TIMING COEFFICIENTS INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

Total set of data = 194 bytes

1 frame = set of timing coefficients

1 record = 1 header + 1 frame + fill
256 = 20 + 194 + fill
256 = 214 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
NDAYS	I*2	1	Number of polynomials used in TCOEFF (1 or 2)
TCOEFF(I,J)	R*4	2,2	Coefficients of polynomials approximating USNO time difference data: I = 1, modified Julian date associated with polynomial J I = 2, constant adjustment in polynomial J
NPDLHS	I*2	1	Number of polynomials used in PDELHT (1 or 2)
PDELHT(J)	R*8	2	Modified Julian date associated with PDELH polynomial J
PDELH(I,J)	R*8	10,2	Coefficients for equations of equinoxes used to correct mean GHA over a 20-day span: J = 1, first nutation polynomial J = 2, second nutation polynomial NOTE: I represents Ith coefficient of Jth polynomial

CONTROL COMMAND INPUT MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 command (20 bytes)

1 record = header + 1 frame
256 = 20 + 20 + fill
256 = 40 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
ICTYPE	I*2	1	Type of command: = 1, start = 2, stop = 3, reboot = 4, abort = 5, suspend = 6, continue = 7, mark time = 8, resume = 9, begin fast timing = 10, stop fast timing = 11, set clock = 12, status request
COMMAND(I)	Byte	20	Contents of command (depends on type of command)

A.1.2 DOWNLINK MESSAGE FORMATS

This section contains the downlink message formats through which data, reports, and messages are downlinked from FEDS. The format of the record header, which is common to all downlinked messages, is given on page A-24, and the message block attributes and frame formats for each type of output data, report, and message are presented on the following pages.

RECORD HEADER

FRAME FORMAT:

Variable	Type	Dimension	Description
IDSC	Byte	1	Spacecraft ID
IDEX	Byte	1	Experiment ID
OUTYPE	Byte	1	Type of output: = 1, spacecraft vectors = 2, Doppler observations = 3, error message = 4, activity log = 5, DC Summary and Statistics Report = 6, DC Residuals Report
	Byte	1	Blank
NBLOCK	I*2	2	Running number of records in block
NTRAN	I*2	2	Running number of records in transmission
NSIZE	I*2	2	Record size in bytes
NTOT	I*2	2	Total number of records in block
TTRAN	R*8	8	Time of transmission (seconds from reference time)

OUTPUT USER SPACECRAFT STATE VECTORS

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 state vector (58 bytes)

1 record = header + 4 frames + fill
256 = 20 + 232 + fill
256 = 252 + fill

1 block = 1 or more records

FRAME FORMAT:

Variable	Type	Dimension	Description
ID1	Byte	1	Indicator of source of initial state vector; can have values of I, U, M
ID2	Byte	1	Counter incremented when source of state vector initially used for generation is changed
TTAG	R*8	1	Time tag (YMDHMS)
PVEC	R*8	3	Position vector (x, y, z)
VVEC	R*8	3	Velocity vector ($\dot{x}$, $\dot{y}$ $\dot{z}$)

OUTPUT ONE-WAY DOPPLER OBSERVATIONS

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 observation (20 bytes)

1 record = header + 11 frames + fill
256 = 20 + 220 + fill
256 = 240 + fill

1 block = 1 or more records

FRAME FORMAT:

Variable	Type	Dimension	Description
OBTYPE	Byte	1	Observation type (= 1, TDRS one-way)
IDTDRS	Byte	1	TDRS ID
IDSTAF	I*2	1	Forward link station ID
OBTIME	R*8	1	Time tag (YMDHMS)
DOPL'	R*8	1	Doppler observation

OUTPUT ERROR MESSAGE

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 error message (50 bytes)

1 record = header + 1 frame + fill
256 = 20 + 50 + fill
256 = 70 + fill

1 block = 1 record

FRAME FORMAT:

Variable	Type	Dimension	Description
TERR	R*8	1	Time of error (YMDHMS)
NERR	I*2	1	Message number
ERRMSG	Byte	40	Message

OUTPUT FROM ACTIVITY LOG

1 frame = 1 message (50 bytes)

1 record = header + 4 frames + fill
256 = 20 + 200 + fill
256 = 220 + fill

1 block = 20 records

FRAME FORMAT:

Variable	Type	Dimension	Description
TMSG	R*8	1	Time message entered log (YMDHMS)
MSGNUM	I*2	1	Message number
MSG	Byte	40	Message

DC SUMMARY AND STATISTICS REPORT

MESSAGE BLOCK ATTRIBUTES;

Whole report = 524 bytes

Record 1 = header + frame 1 + fill = 20 + 184 + fill = 256
Record 2 = header + frame 2 + fill = 20 + 160 + fill = 256
Record 3 = header + frame 3 + fill = 20 + 196 + fill = 256

1 block = 3 records

FRAME 1 FORMAT:

Variable	Type	Dimension	Description
DCEPCH	R*8	1	Epoch
DCSTRT	R*8	1	Start time of estimation data span
DCEND	R*8	1	End time of estimation data span
SE	R*4	10	S_e at each inner loop
QSUM	R*4	10	Q summed at each inner loop
XPREV	R*8	10	Previous state vector

FRAME 2 FORMAT:

Variable	Type	Dimension	Description
XCURR	R*8	10	Current state vector
XAPR	R*8	10	A priori state vector

FRAME 3 FORMAT:

Variable	Type	Dimension	Description
RMS	R*8	10	Predicted root mean square at each inner loop
XUPD	R*8	10	State correction vector
ISTATE	I*2	10	Parameter numbers
NSTATE	I*2	1	Number of solve-for parameters
NTOTAL	I*2	1	Total number of observations available
NUSED	I*2	1	Number of observations used

Variable	Type	Dimension	Description
NITER	I*2	1	Iteration number
NBATCH	I*2	1	Slide number
ICONVG	I*2	1	Convergence/divergence indicator: = 0, no convergence/ divergence this iteration = 1, convergence (PCONV, VCONV tests) = 2, convergence (SECONV test) = 3, reduced convergence (maximum iteration reached but within tolerance of three times PCONV, VCONV, SECONV) = 4, diverged (data arc length after edit less than minimum estimation span) = 5, diverged (all new observations edited) = 6, diverged (POSDIV, VELDIV tests) = 7, diverged (RATCOR tests) = 8, diverged (maximum iterations)
NLOOP	I*2	1	Number of inner edit loop this iteration
LINTST	L*2	1	Linearity indicator: = TRUE, do not recompute partials or edit loop = FALSE, recompute partials and edit loop

DC RESIDUALS REPORT

MESSAGE BLOCK ATTRIBUTES:

1 frame = 1 line of report (48 bytes) or report descriptor information (48 bytes)

1 record = header + 4 frames + fill
256 = 20 + 192 + fill
256 = 212 + fill

1 block = up to 32 records

First record of block = header + descriptor frame + 3 data frames
All other records = header + 4 data frames

DESCRIPTOR FRAME FORMAT:

Variable	Type	Dimension	Description
REPOCH	R*8	1	Epoch (YYMMDDHHMMSS.SS)
STRES	R*8	1	Start time of batch (YYMMDDHHMMSS.SS)
ENDRES	R*8	1	End time of batch (YYMMMDDHHMMSS.SS)
RESITR	I*2	1	Iteration number
RESBAT	I*2	1	Batch number
RESINL	I*2	1	Inner loop number
SPARES	Byte	18	Spares

DATA FRAME FORMAT:

Variable	Type	Dimension	Description
IOBTYP	Byte	1	Observation type (= 1, TDRSS one-way)
IEDIT(I)	Byte	2	Edit flag (I = 1, not used; I = 2, Doppler): = 0, not edited = 1, edited by DC during edit loop = 2, edited during preprocessing = 3, edited by DC for maximum observed-minus-computed value

Variable	Type	Dimension	Description
IEDIT(I) (Cont'd)			= 4, edited by DC for minimum ray path angle (TDRSS)
ITDRSF	Byte	1	TDRSS ID (forward link)
ITDRSR	Byte	1	TDRSS ID (return link)
ISTATF	Byte	1	Forward link station ID
ISTATR	Byte	1	Return link station ID
ISPARE	Byte	1	Spare location
OBTIME	R*8	1	Time tag
COBS(1)	R*8	1	Computed range observations (not used in FEDS)
COBS(2)	R*8	1	Computed Doppler observation
RESID(1)	R*4	1	Range residual (not used in FEDS)
RESID(2)	R*4	1	Doppler residual
PRESID(1)	R*4	1	Predicted residual for range (not used in FEDS)
PRESID(2)	R*4	1	Predicted residual for Doppler

A.2 COMMUNICATIONS BOX/FEDS INTERFACE

The Communications Box and FEDS transmit and receive messages to control acquisition of a tracking signal by the transponder and to accumulate observation data. All messages between FEDS and the Communications Box are sent in an 11-byte format, shown in Figure A-3. The first seven bits of the first byte in the message constitute the function code. The first six bits define the message being sent and the seventh bit indicates the source of the message: 0 if the message is from FEDS, 1 if the message is from the Communications Box. The eighth bit of the first byte and the next five bytes are reserved for the PB5 time code. The remaining five bytes of the message constitute a data field. All unused fields in each message are zero-filled.

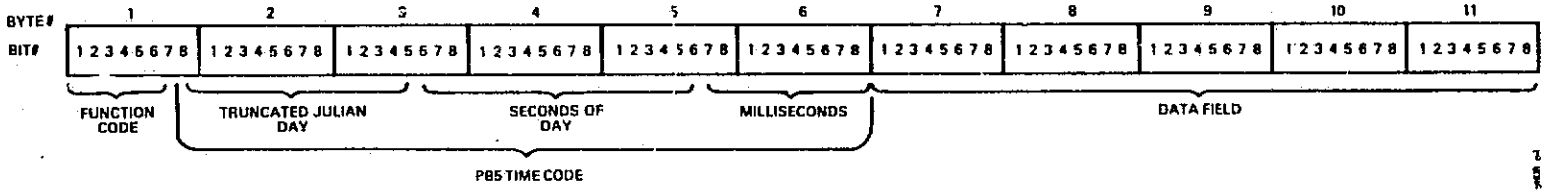


Figure A-3. Communications Box/FEDS Interface Format

A.2.1 INITIALIZATION MESSAGES

FEDS TO COMMUNICATIONS BOX

Function Code	Bytes 2 to 11	Interpretation
0 (0000000 binary)	192 (11000000 binary)	FEDS executing; verify communication with Communications Box

COMMUNICATIONS BOX TO FEDS

Function Code	Bytes 2 to 11	Interpretation
0 (0000001 binary)	Not used	Initialization message received; communication verified

A.2.2 DOPPLER OBSERVATION MESSAGES

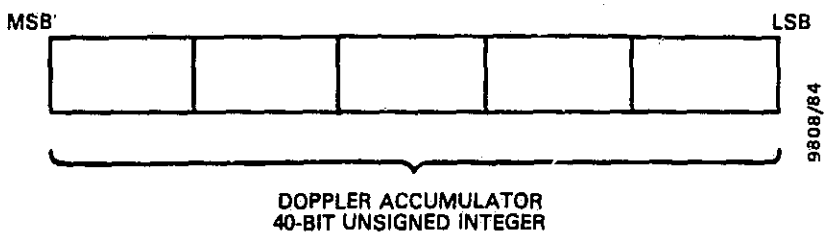
FEDS TO COMMUNICATIONS BOX

Function Code	Time Field	Data Field	Interpretation
1 (1000000 binary)	Not used	Not used	FEDS ready to receive Doppler observation

COMMUNICATIONS BOX TO FEDS

Function Code	Time Field	Data Field	Interpretation
1 (1000001 binary)	PB5 Time Code	Doppler accumulator from the transponder (format shown below)	Doppler observation at the specified time

Doppler accumulator 40-bit
Unsigned integer



A.2.3 TIME CODE MESSAGES

FEDS TO COMMUNICATIONS BOX

Function Code	Time Field	Data Field	Interpretation
2 (0100000 binary)	Not used	Not used	Request for current time from time code generator

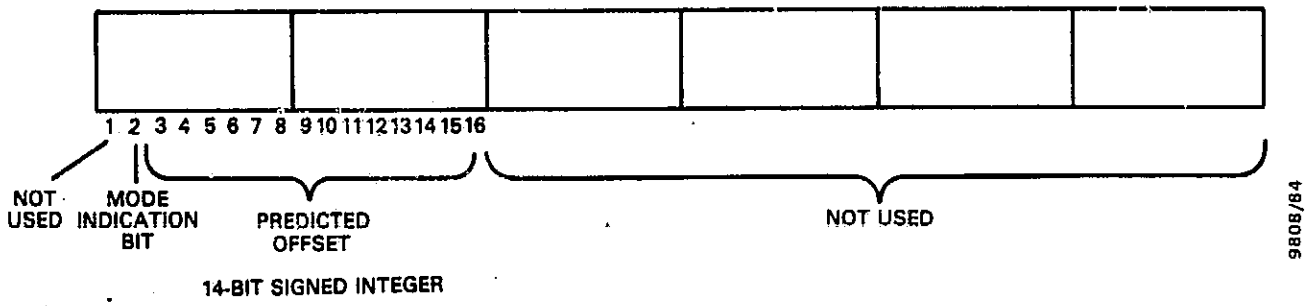
COMMUNICATIONS BOX TO FEDS

Function Code	Time Field	Data Field	Interpretation
2 (0100001 binary)	PB5 Time Code	Not used	Time field contains current time

A.2.4 PREDICTED DOPPLER MESSAGES

FEDS TO COMMUNICATIONS BOX

Function Code	Time Field	Data Field	Interpretation
3 (0010000 binary)	Not used	Frequency control word (format below)	Predicted frequency offset for use in signal acquisition by transponder



COMMUNICATIONS BOX TO FEDS

Function Code	Time Field	Data Field	Interpretation
3 (0010001 binary)	Not used	Not used	Frequency control word received from FEDS and transmitted to transponder

A.2.5 SIGNAL ACQUISITION MESSAGE

FEDS TO COMMUNICATIONS BOX

No message sent

COMMUNICATIONS BOX TO FEDS

Function Code	Time Field	Data Field	Interpretation
4 (0001001 binary)	Not used	Not used	Transponder has acquired tracking signal

A.2.6 ACCUMULATOR RESET MESSAGE

FEDS TO COMMUNICATIONS BOX

Function Code	Time Field	Data Field	Interpretation
5 (0000100 binary)	Not used	Not used	Request transponder to reset the Doppler accumulator

COMMUNICATIONS BOX TO FEDS

No Message sent

A.2.7 SIGNAL LOSS MESSAGE

FEDS TO COMMUNICATIONS BOX

No message sent

COMMUNICATIONS BOX TO FEDS

Function Code	Time Field	Data Field	Interpretation
6 (0000011 binary)	Not used	Not used	Transponder has lost the tracking signal

A.3 COMMUNICATIONS BOX/TRANSPONDER INTERFACE

Data for this section was not available when the document went to press. This appendix will be produced and distributed at a later time.

APPENDIX B - OUTPUT MESSAGE FORMATS

This Appendix gives the expanded form of messages displayed to the FEDS terminal. They come from activity log generation, execution of the executive, and communication with the Communications Box. A message will be displayed in brief form, with the message number displayed first, followed by six integers and three real numbers. If a number is not pertinent to the current message, it will be displayed as a zero. The only exceptions to this description are messages 67 and 68, which pertain to communication with the Communications Box. They are displayed with the message number first, followed by the 11-byte message displayed in octal format.

Table B-1 presents the output messages and their source.

Table B-1. FEDS Terminal Messages (1 of 6)

NO.	MESSAGE	ORIGINATOR
1	NEW TASK SCHEDULED FOR EXECUTION TASK # n , DIRECTIVE n , ACTIVITY FLAG n SECONDS FROM REFERENCE SSSS.SS	EXEC
2	TASK SCHEDULED FOR 10 CONSECUTIVE TIME SLICES Task # n , DIRECTIVE n SECONDS FROM REFERENCE SSSSS.SS	EXEC
3	INPUT QUEUE IS FULL	DATCAP
4	NEW TDRS VECTOR WAS RECEIVED	INPPRO
5	INITIALIZATION TABLE RECEIVED	INPPRO
6	MANEUVER SCHEDULE RECEIVED	INPPRO
7	ESTIMATION CONTROL PARAMETERS RECEIVED	INPPRO
8	TRACKING SCHEDULE RECEIVED	INPPRO
9	MISCELLANEOUS CONSTANTS RECEIVED	INPPRO
10	STATION CONSTANTS RECEIVE	INPPRO
11	GEOPOTENTIAL TABLES RECEIVED	INPPRO
12	ATMOSPHERIC DRAG TABLES RECEIVED	INPPRO
13	TIMING COEFFICIENTS RECEIVED	INPPRO
14	SPARE	
15	RAW OBSERVATION DATA RECEIVVED n OBSERVATION PAIRS RECEIVED n OBSERVATION PAIRS IN QUEUE n OBSERVATION PAIRS REJECTED	INPPRO INPPRO
16	OBSERVATION QUEUES ARE FULL — INPUT PROCESSOR IS WAITING	INPPRO
17	END OF TRANSMISSION RECEIVED DURING UPLINK	DATCAP
18	ORBIT FILE FOR TDRS n WAS UPDATED FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS BASED ON A NEW REFERENCE VECTOR	PREPRO
19	PREPROCESSING OF OBSERVATION BUFFER HAS BEEN COMPLETED n OBSERVATIONS ACCEPTED n OBSERVATIONS REJECTED NEW OBS. PASS IS FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS	PREPRO

Table B-1. FEDS Terminal Messages (2 of 6)

NO.	MESSAGE	ORIGINATOR
20	ALL TDRS ORBIT FILES EXTENDED TO COVER ALL OBSERVATION DATA ACTIVE TIME SPAN OF TDRS ORBIT FILES IS FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS	PREPRO
21	ORBIT FILE FOR TDRS n WAS UPDATED TOI REFLECT A MANEUVER AT YYMMDDHHMMSS.SS	PREPRO
22	ALL TDRS ORBIT FILES AND THE OBSERVATION FILE WERE PURGED	PREPRO
23	OBSERVATION FILE WAS PURGED AS A RESULT OF A USER S/C MANEUVER AT YYMMDDHHMMSS.SS. DATA COLLECTION WILL BEGIN AGAIN TDRS DATA PRIOR TO MANEUVER TIME WAS ALSO PURGED TIME SPAN OF ORBIT FILES IS FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS	PREPRO
24	PREPROCESSING OF A PASS COMPLETED n OBSERVATIONS ACCEPTED n OBSERVATIONS REJECTED NEW OBSERVATION PASS IS FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS	PREPRO
25	TRACKING PASS AT YYMMDDHHMMSS.SS COMPLETED WITH NO DATA COLLECTED	PREPRO
26	SPARE	
27	SPARE	
28	SPARE	
29	STATE VECTOR PREDICT TABLE WAS DOWNLINKED	OUTPRO
30	PREDICTED ONE-WAY DOPPLER OBSERVATIONS WERE DOWNLINKED	OUTPRO
31	ERROR MESSAGE WAS DOWNLINKED	OUTPRO
32	ACTIVITY LOG WAS DOWNLINKED	OUTPRO
33	DC SUMMARY AND STATISTICS REPORT WAS DOWNLINKED	OUTPRO
34	DC RESIDUALS REPORT WAS DOWNLINKED	OUTPRO
35	ONE WAY DOPPLER OBSERVATIONS WERE PREDICTED n OBSERVATIONS WERE GENERATED USING TDRS n FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS PASS SCHEDULED TO COMPLETE AT YYMMDDHHMMSS.SS	DOPPRE
36	ONE-WAY DOPPLER PREDICTION FAILED. REQUESTED TIME SPAN (YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS) IS NOT IN THE STATE PREDICT TABLE	DOPPRE
37	INVALID DIRECTIVE RECEIVED BY DOPPLER PREDICTOR. IDIR(6) = n	DOPPR

9808-150*1-84

Table B-1. FEDS Terminal Messages (3 of 6)

NO.	MESSAGE	ORIGINATOR
38	ONE-WAY DOPPLER PREDICTION FAILED DUE TO COMMUNICATIONS PROBLEMS WITH THE DATA MANAGER. IRET(6) = n	DOPPRE
39	ERROR OCCURRED DURING INTERPOLATION OF USER S/C STATE DURING ONE-WAY DOPPLER PREDICTION	DOPPRE
40	ERROR OCCURRED IN OBSERVATION MODEL DURING ONE-WAY DOPPLER PREDICTION. OBSERVATION WAS IGNORED	DOPPRE
41	ESTIMATION COMPLETED FOR BATCH NO. n . DC CONVERTED (CODE = n) AFTER n ITERATIONS USING n OF n AVAILABLE OBSERVATIONS. START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
42	ESTIMATION COMPLETED FOR BATCH NO. n . DC DIVERGED (CODE = n) AFTER n ITERATIONS USING n OF n AVAILABLE OBSERVATIONS START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
43	ESTIMATION COMPLETED FOR BATCH NO. n . NO CONVERGENCE/DIVERGENCE AFTER n ITERATIONS USING n OF n AVAILABLE OBSERVATIONS START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
44	ESTIMATION FAILED FOR BATCH NO. n . DC CONVERGED (CODE = n) AFTER n ITERATIONS USING n AVAILABLE OBSERVATIONS START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
45	ESTIMATION FAILED FOR BATCH NO. n . DC DIVERGED (CODE = n) AFTER n ITERATIONS USING n OF n AVAILABLE OBSERVATIONS START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
46	ESTIMATION FAILED FOR BATCH NO. n . NO CONVERGENCE/DIVERGENCE AFTER n ITERATIONS USING n OF n AVAILABLE OBSERVATIONS START TIME = YYMMDDHHMMSS.SS END TIME = YYMMDDHHMMSS.SS	ESTIM
47	SPARE	
48	ONE-WAY DOPPLER OBSERVATIONS FILE EXTENDED n OBSERVATIONS COMPUTED USING TDRS n FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS PASS SCHEDULED TO COMPLETE AT YYMMDDHHMMSS.SS	DOPPRE
49	SYNC DETECT RECEIVED FROM TRANSPONDER	DATCAP
50	TDRSS SIGNAL LOST BY TRANSPONDER	DATCAP
51	SPARE	
52	SPARE	
53	SPARE	

Table B-1. FEDS Terminal Messages (4 of 6)

NO.	MESSAGE	ORIGINATOR
54	SPARE	
55	SPARE	
56	SPARE	
57	SPARE	
58	SPARE	
59	SPARE	
60	SPARE	
61	SPARE	
62	SPARE	
63	SPARE	
64	SPARE	
65	SPARE	
66	SPARE	
67	MESSAGE RECEIVED FROM COMMUNICATIONS BOX MESSAGE TEXT (OCTAL) n n n n n n n n n n n	DATCAP
68	MESSAGE TRANSMITTED TO COMMUNICATIONS BOX MESSAGE TYPE (OCTAL) n n n n n n n n n n n	OUTPRO
69	ERROR TYPE n OCCURRED DURING GENERATION OF THE STATE VECTOR PREDICT TABLE. IT WILL BE ATTEMPTED AGAIN AT THE NEXT SCHEDULED TIME. THERE MAY BE GAPS IN THE TABLE	STAPRE
70	ERROR TYPE n OCCURRED DURING GENERATION OF THE STATE VECTOR PREDICT TABLE DUE TO A BAD STATE SOLUTION STANDARD TABLE GENERATION WILL CONTINUE WHEN SCHEDULED	STPRE
71	ERROR TYPE n OCCURRED DURING GENERATION OF THE STATE VECTOR PREDICT TABLE DUE TO A BAD INITIALIZATION TABLE STANDARD TABLE GENERATION WILL CONTINUE WHEN SCHEDULED	STAPRE
72	ERROR TYPE n OCCURRED DURING MANEUVER RECOVERY IN THE STATE PREDICTOR. THE USER S/C MANEUVER AT YYMMDDHHMMSS.SS WILL BE IGNORED. STANDARD TABLE GENERATION WILL CONTINUE WHEN SCHEDULED	STAPRE
73	THE STATE VECTOR PREDICT TABLE WAS EXTENDED FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS	STAPRE

9806-150-1-84

Table B-1. FEDS Terminal Messages (5 of 6)

NO.	MESSAGE	ORIGINATOR
74	THE STATE VECTOR PREDICT TABLE WAS GENERATED FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS BASED ON A NEW STATE SOLUTION	STAPRE
75	THE STATE VECTOR PREDICT TABLE WAS GENERATED FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS BASED ON A NEW INITIALIZATION TABLE	STAPRE
76	THE STATE VECTOR PREDICT TABLE WAS GENERATED FROM YYMMDDHHMMSS.SS TO YYMMDDHHMMSS.SS BASED ON A USER S/C MANEUVER AT YYMMDDHHMMSS.SS	STAPRE
77	INVALID DIRECTIVE RECEIVED BY OUTPUT PROCESSOR IDIR(7) = n	OUTPRO
78	PREDICTOR-CORRECTOR IN ORBIT PROPAGATOR DID NOT CONVERGE WHILE PROPAGATING TDRS. EXECUTION CONTINUED TO END TIME	PREPRO
79	NUMERICAL ERROR IN ORBIT PROPAGATOR WHILE INTEGRATING TDRS	PREPRO
80	TDRS ORBIT FILE GENERATION FAILED DUE TO SEND/RECEIVE ERRORS	PREPRO
81	INVALID DIRECTIVE RECEIVED BY DATA PREPROCESSOR. IDIR(3) = n	PREPRO
82	MESSAGES LOST DURING UPLINK n MESSAGES LOST BEGINNING WITH RECORD n OF TRANSMISSION	INPRO
83	TRANSMISSION ERROR DURING UPLINK OF DATA TYPE n ERROR CODE = n BLOCK ID NUMBER = n RECORD n OF TRANSMISSION AT TIME t RETRANSMIT ENTIRE DATA BLOCK	INPPRO
84	INVALID DIRECTIVE RECEIVED BY INPUT PROCESSOR. IDIR(2) = n	INPPRO
85	UPLINKED CONSTANTS WERE IGNORED. DATA TYPE = n AODS MUST BE SUSPENDED BEFORE UPLINKING CONSTANTS	INPPRO
86	END OF TRANSMISSION RECEIVED IN INPUT PROCESSOR n MESSAGES RECEIVED n MESSAGES REJECTED	INPPRO
87	MISCELLANEOUS CONSTANTS CANNOT BE CHANGED AFTER PROCESSING BEGINS — INPUT MISCELLANEOUS CONSTANTS WERE IGNORED	INPPRO
88	DATA CAPTURE RECEIVED INVALID DIRECTIVE. IDIR(1) = n	DATCAP
89	START COMMAND RECEIVED REFERENCE TIME IS YYMMDDHHMMSS.SS	DATCAP
90	STOP COMMAND RECEIVED	DATCAP

9808-150a]-84

Table B-1. FEDS Terminal Messages (6 of 6)

NO.	MESSAGE	ORIGINATOR
91	REBOOT COMMAND RECEIVED	DATCAP
92	ABORT COMMAND RECEIVED	DATCAP
93	SUSPEND COMMAND RECEIVED	DATCAP
94	CONTINUE COMMAND RECEIVED	DATCAP
95	MARK TIME COMMAND RECEIVVED	DATCAP
96	RESUME COMMAND RECEIVED	DATCAP
97	BEGIN FAST TIMING COMMAND RECEIVED — FAST TIMING IS ON	DATCAP
98	STOP FAST TIMING COMMAND RECEIVED — FAST TIMING IS OFF	DATCAP
99	SET CLOCK COMMAND RECEIVED. t SECONDS COMPRESSED OUT	DATCAP
100	STATUS REQUEST COMMAND RECEIVED — ACTIVITY LOG WILL BE	DATCAP

9808-150a-1-84

APPENDIX C - DATA PACKET DESCRIPTIONS

This appendix contains descriptions of the data packets used to transfer data by means of SEND and RECEEV directives between FEDS primary and secondary tasks.

C.1 DATA PACKET 1

SIZE: 73 words (146 bytes)

SENT BY: PREPRO

RECEIVED BY: DATMGR

FORMAT:

Parameter	Type	Dimension	Description
IOBTYP	I*2	1	Observation type = 1, one-way TDRSS
OBSTIM	R*8	1	Observation time tag
Spare	Byte	8	Spare
OBS	R*8	1	Doppler observation
FREQ	R*8	1	TDRSS frequency
DOPINT	R*4	1	Doppler averaging interval
Spare	Byte	1	Spare
FORANT	Byte	1	Forward station ID (internal index)
Spare	Byte	1	Spare
FORTDR	Byte	1	Forward TDRSS ID (internal index)
EDIT(I)	Byte	2	Observation data edit flag: = 0, not edited = 1, edited by DC during edit loop = 2, edited by preprocessor = 3, edited by DC for maximum observed-minus-computed value = 4, edited by DC for ray path (EDIT(1) not used in FEDS)
Spare	Byte	1	Spare
FORACC	Byte	1	Forward access method (internal index)
JPASS	Byte	1	End-of-pass indicator
BAND	Byte	1	Band frequency: = 48, S-band = 96, Ku-band

C.1 DATA PACKET 1 (Cont'd)

Parameter	Type	Dimension	Description
NEWREC	L*1	1	New record flag (= 7, record has not been processed by estimator)
Spare	Byte	97	Spare

C.2 DATA PACKET 2

SIZE: 17 words (34 bytes)

SENT BY: PREPRO

RECEIVED BY: DATMGR

FORMAT:

Parameter	Type	Dimension	Description
ITYPE	I*2	1	Type of TDRS vector: = 1, TDRS 1 = 2, TDRS 2
INPVEC	R*8	4	Input vector (time and position vector)

C.3 DATA PACKET 3

SIZE: 5 words (10 bytes)

SENT BY: DOPPRE, OBSMDL

RECEIVED BY: DATMGR

FORMAT:

Parameter	Type	Dimension	Description
NTDR	I*2	1	Type of TDRS vector: = 1, TDRS 1 = 2, TDRS 2
TTAG	R*8	1	Requesting time for a set of 10 TDRS vectors

C.4 DATA PACKET 4

SIZE: 73 words (146 bytes)

SENT BY: ESTIM, DATMGR

RECEIVED BY: DATMGR, ESTIM

FORMAT:

Parameter	Type	Dimension	Description
IOBTYP	I*2	1	Observation type: (= 1, one-way TDRSS)
OBSTIM	R*8	1	Observation time tag
Spare	Byte	8	Spare
OBS	R*8	1	Doppler observation
FREQ	R*8	1	TDRSS frequency
DOPINT	R*4	1	Doppler averaging interval
Spare	Byte	1	Spare
FORANT	Byte	1	Forward station ID (internal index)
Spare	Byte	1	Spare
FORTDR	Byte	1	Forward TDRSS ID (internal index)
EDIT(I)	Byte	2	Observation data edit flag: = 0, not edited = 1, edited by DC during edit loop = 2, edited by preprocessor = 3, edited by DC for maximum observed-minus-computed value = 4, edited by DC for ray path (EDIT(1) not used in FEDS)
Spare	Byte	1	Spare
FORACC	Byte	1	Forward access method (internal index)
JPASS	Byte	1	End-of-pass indicator
BAND	Byte	1	Band frequency: = 48, S-band = 96, Ku-band

C.4 DATA PACKET 4 (Cont'd)

Parameter	Type	Dimension	Description
NEWREC	L*1	1	New record flag (= 7, record has not been processed by estimator)
Spare	Byte	4	Spare
OBSPAR	R*4	10	Doppler observation partial derivatives
SPARE	Byte	8	Spare
OBSRES	R*8	2	Doppler observation residual

C.5 DATA PACKET 5

SIZE: 160 words (320 bytes)

SENT BY: DATMGR

RECEIVED BY: DOPPRE, OBSMDL

FORMAT:

Parameter	Type	Dimension	Description
OUTVEC(I,J)	R*8	4,10	Requested set of 10 TDRS vectors surrounding request time: I = 1, time tag associated with the vector J I = 2, x-position component of vector J I = 3, y-position component of vector J I = 4, z-position component of vector J

C.6 DATA PACKET 6

SIZE: 40 words (80 bytes)

SENT BY: PREPRO

RECEIVED BY: ORBIT

FORMAT:

Parameter	Type	Dimension	Description
ISTART	I*2	1	Start mode for propagation: = 1, use input vector = 2, use internal table
IPART	I*2	1	Variational equation control flag (= 0, do not integrate variational equation)
TTAG	R*8	1	Starting vector time tag (A.1 seconds from reference time)
X(6)	R*8	6	Starting vector (ignored if ISTART = 2)
Spare	Byte	10	Spare
ISCID	I*2	1	Spacecraft ID: = 1, TDRS 1 = 2, TDRS 2
ENDTIM	R*8	1	Requested end time of propagation (A.1 seconds from reference time)

C.7 DATA PACKET 7

SIZE: 40 words (80 bytes)

SENT BY: STAPRE

RECEIVED BY: ORBIT

FORMAT:

Parameter	Type	Dimension	Description
ISTART	I*2	1	Start mode for propagation: = 1, use input vector = 2, use internal table
IPART	I*2	1	Variational equation control flag (= 0, do not integrate variational equation)
TTAG	R*8	1	Starting vector time tag (A.1 seconds from reference time)
X(6)	R*8	6	Starting vector (ignored if ISTART = 2)
CD	R*8	1	Coefficient of drag
IMAP7	I*2	1	CD use indicator: = 0, use default coefficient of drag > 0, use CD if ISTART = 1
ISCID	I*2	1	Spacecraft ID (= 5, user predict)
ENDTIM	R*8	1	Requested end time of propagation (A.1 seconds from reference time)

C.8 DATA PACKET 8

SIZE: 40 words (80 bytes)

SENT BY: ESTIM

RECEIVED BY: ORBIT

FORMAT:

Parameter	Type	Dimension	Description
ISTART	I*2	1	Start mode for propagation (= 1, use input vector)
IPART	I*2	1	Variational equation control flag (= 0, do not integrate variational equation)
TTAG	R*8	1	Starting vector time tag (A.1 seconds from reference time)
X(6)	R*8	6	Starting vector
CD	R*8	1	Coefficient of drag
IMAP7	I*2	1	CD use indicator: = 0, use default coefficient of drag > 0, use CD
ISCID	I*2	1	Spacecraft ID (= 4, user past)
ENDTIM	R*8	1	Requested end time of propagation (A.1 seconds from reference time)

C.9 DATA PACKET 9

SIZE: 40 words (80 bytes)

SENT BY: ESTIM

RECEIVED BY: ORBIT

FORMAT:

Parameter	Type	Dimension	Description
ISTART	I*2	1	Start mode for propagation (= 1, use input vector)
IPART	I*2	1	Variational equation control flag: = 1, integrate variational equation without drag partial derivative = 2, integrate variational equation with drag partial derivative
TTAG	R*8	1	Starting vector time tag (A.1 seconds from reference time)
X(6)	R*8	6	Starting vector
CD	R*8	1	Coefficient of drag
IMAP7	I*2	1	CD use indicator: = 0, use default coefficient of drag > 0, use CD
ISCID	I*2	1	Spacecraft ID (= 4, user past)
ENDTIM	R*8	1	Requested end time of propagation (A.1 seconds from reference time)

C.10 DATA PACKET 10

SIZE: 40 words (80 bytes)

SENT BY: OBSMDL

RECEIVED BY: ORBIT

FORMAT:

Parameter	Type	Dimension	Description
ISTART	I*2	1	Start mode for propagation (= 2, use internal table)
IPART	I*2	1	Variational equation control flag: = 1, integrate variational equation without drag partial derivative = 2, integrate variational equation with drag partial derivative
TTAG	R*8	1	Starting vector time tag (A.1 seconds from reference time)
X(6)	R*8	6	Ignored because ISTART = 2
CD	R*8	1	Ignored because ISTART = 2
IMAP7	I*2	1	Ignored because ISTART = 2
ISCID	I*2	1	Spacecraft ID (= 4, user past)
ENDTIM	R*8	1	Requested end time of propagation (A.1 seconds from reference time)

C.11 DATA PACKET 11

SIZE: 204 words (408 bytes)

SENT BY: ORBIT

RECEIVED BY: PREPRO, STAPRE, ESTIM, OBSMDL

FORMAT:

Parameter	Type	Dimension	Description
NEWORB	I*2	1	Reference vector chosen by ORBIT for propagation: = 0, used internal table = 1, used input vector
IPART0	I*2	1	State transition matrix output flag: (= 0, no state transition matrix)
ENDTM1	R*8	1	End time of propagation (time tag associated with the new vector)
XOUT	R*8	6	New vector
ISCIDO	I*2	1	Spacecraft ID: = 1, TDRS 1 = 2, TDRS 2 = 4, user past = 5, user predict
IVALID	I*2	5	Validity-of-results flag:[1] = 0, no error detected = 1, input parameter error; execution continues = 50, numerical fault; execution continues = 100, input parameter error and termination = 200, input parameter out of range and termination = 500, numerical error and termination
Spare	Byte	336	Spare

[1] Up to five errors can be entered.

C.12 DATA PACKET 12

SIZE: 204 words (408 bytes)

SENT BY: ORBIT

RECEIVED BY: ESTIM, OBSMDL

FORMAT:

Parameter	Type	Dimension	Description
NEWORB	I*2	1	Vector chosen by ORBIT for startup: = 0, used stored starting vector = 1, used input vector
IPART0	I*2	1	State transition matrix output flag: = 1, state transition matrix without drag = 2, state transition matrix with drag
ENDTM1	R*8	1	End time of propagation (time tag associated with the new vector)
XOUT	R*8	6	New vector
ISCID	I*2	1	Spacecraft ID (= 4, past user orbit)
IVALID	I*2	5	Validity-of-results flag:[1] = 0, no error detected = 1, input parameter error; execution continues = 50, numerical fault; execution continues = 100, input parameter error and termination = 200, input parameter out of range and termination = 500, numerical error and termination
STM	R*8	6,7	State transition matrix at ENDTM1

[1] Up to five errors can be entered.

APPENDIX D - FEDS UPDATE PROCEDURES AND COMMAND FILES

The standard RSX-11M compilation and task building procedures are used to update FEDS and ADEPT software. Figure D-1 gives the executive command file used to compile FEDS modules. Text files used in the executive command file and task build command files for each task are given in Figures D-2 through D-75, follow, grouped by task. For completeness, command files, text files and overlay descriptor files are given to build ADEPT. In all cases, FEDS command files ending in "23" refer to command files associated with the LSI version. The command file to build the FEDS system image for the LSI is given in Figure D-75.

```
;
;       SRCLST.CMD   COMMAND FILE TO COMPILE ANY SUBSYSTEMS
;
        .ENABLE SUBSTITUTION
.START:
;
!ENTER SUBSYSTEM
        .ASKS  SUB ENTER SUBSYSTEM
        .IF SUB EQ "ORBIT" .GOTO REST
        .IF SUB EQ "FEDS" .GOTO REST
        .IF SUB EQ "ESTIM" .GOTO REST
        .IF SUB EQ "DATCAP" .GOTO REST
        .IF SUB EQ "OUTPRO" .GOTO REST
        .IF SUB EQ "INPPRO" .GOTO REST
        .IF SUB EQ "PREPRO" .GOTO REST
        .IF SUB EQ "DATMGR" .GOTO REST
        .IF SUB EQ "DOPPRE" .GOTO REST
        .IF SUB EQ "STAPRE" .GOTO REST
        .IF SUB EQ "OBSMDL" .GOTO REST
        .IF SUB EQ "EXEC"   .GOTO REST
;REQUESTED SYSTEM NOT FOUND
        .GOTO START
.REST:
;OPEN FILE FOR ALL SUBSYSTEMS
        .OPENR 'SUB'.TXT
;  COMPILE ALL MODULES
.NEXT:
        .READ MOD
        .IFT <EOF> .GOTO DONE
        .IF <FILERR> NE 1 .GOTO START
FORTRAN/F4P 'MOD'
        .IF <EXSTAT> NE <SUCCES> .GOSUB ERR
        .GOTO NEXT
.DONE:
        .CLOSE
        .EXIT
.ERR:
;RUN DB0:[201,6]BELL
        .ASKS CR CR TO CONTINUE
        .IF CR <> "" .EXIT
        .RETURN
;
```

Figure D-1. CMPFEDS.CMD (Executive Command File to Compile FEDS)

```
ABAM
ACCEL
ACTGEN
ATMOS
BEHUN
BODFIX
CKPRIO
CLKMES
CNVCW
CNVTOB
CNVTTM
CURTIM
DATCAP
DATCAP23
DATMGR
DATMGR23
DDATE
DOPLEG
DOPLTM
DOPMDL
DOPPRE
DOPPRE23
DPINIT
DSPRES
DSPOBS
DWNSND
DWNSND70
ELVANG
ESINIT
ESLIDE
ESMNVR
ESTIM
ESTIM23
EXEC
EXEC23
FORCV
GETORB
GETTDR
GHAUPD
GHAUPN
GTHEAD
INPFRM
INPPRO
INPPRO23
INTP
INTRN
INV2
IPINIT
```

Figure D-2. FEDS.TXT (Text File to Compile All FEDS Modules (1 of 3)

```
JDATE
LATLON
LGN
LOCTDR
LODACT
LODBUF
LODDCS
LODDPL
LODERR
LODOBS
LODRES
LODSEN
LODVEC
LOKAHD
LTC
LTIMT1
LUNA
MATMUL
MATPRE
MATPST
MSTEP
OBSMDL
OBSMDL23
OBSPRE
OBSRW
OBSUPD
ONELEG
OPINIT
ORBINI
ORBIT
ORBIT23
OUTPRO
OUTPRO23
OUTQIO
OUTTIM
OUTTRN
PB52CL
PPINIT
PREDIT
PREPRO
PREPRO23
PURFIL
PUROBS
PURTDR
RANGRT
REDUCE
RUKUTT
```

Figure D-2. FEDS.TXT (Text File to Compile All FEDS Modules (2 of 3)

```
SCANIN
SETCLK
SETINX
SLCORR
SLEDIT
SLEND
SLINIT
SLITER
SLOUT
SLSUMS
SLTEST
SLUPDT
SNDCMD
SOL
SOLLUN
SORBIT
SPART
SPARTV
SSTEP
STAPRE
STAPRE23
STATM
STERR
STEST
STEXP
STGEO
STINIT
STMANS
STMISC
STPV
STSTAN
STTDRS
STTIMF
STTRKS
SUMS
SYMINV
TAGOBS
TCON
TDRINT
TDRORB
TDR1WM
TDR1WP
TGTINT
TIMCON
TREF
TYMD
WTMSG
>
```

Figure D-2. FEDS.TXT (Text File to Compile All FEDS Modules (3 of 3)

```
EXEC
ACTGEN
WTMSG
CURTIM
JDATE
OUTTIM
TCON
TIMCON
TREF
TYMD
OUTQIO
```

Figure D-3. EXEC.TXT (Text File to Compile EXEC Modules)

```
EXEC/PR:0,EXEC=EXEC,OUTQIO,TIMCON,JDATE,ACTGEN,TCON,
  CURTIM,TREF,TYMD,OUTTIM,WTMSG
  SETCLK,DDATE
  [1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB2:RW
PRI=75
ASG=TI:5:6
MAXBUF=80
TASK=EXEC
LIBR= 11SRES:RO
//
```

Figure D-4. EXEC.CMD (Command File to Build EXEC Task)

```
EXEC23/PR:0,EXEC23=EXEC23,OUTQIO,TIMCON,JDATE,ACTGEN,TCON,
  CURTIM,TREF,TYMD,OUTTIM,WTMSG,SETCLK
  [1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB2:RW
PRI=70
ASG=TT3:5:6
MAXBUF=80
TASK=EXEC
LIBR= 11SRES:RO
//
>
```

Figure D-5. EXEC23.CMD (Command File to Build EXEC23 Task)

```
DATCAP
INTRN
LODOBS
CLKMES
PB52CL
DDATE
CURTIM
JDATE
LODBUF
SCANIN
SNDCMD
TCON
TREF
OUTQIO
>
```

Figure D-6. DATCAP.TXT (Text File to Compile DATCAP Modules)

```
DATCAP=DATCAP,INTRN,LODOBS,CLKMES,PB52CL,DDATE
SCANIN,SNDCMD,LODBUF
CURTIM,TCON,TREF,JDATE
MOVEC,OUTQIO,VSEND
[1,1]11SLIB/LB
/
UNITS=6
ASG=TT24:2,TT33:1,TI:5:6
COMMON= GLB1:RW
COMMON= GLB2:RW
COMMON= GLB4:RW
MAXBUF= 256
ACTFIL= 3
PRI=    51
LIBR=   11SRES:RO
//
>
```

Figure D-7. DATCAP.CMD (Command File to Build DATCAP Task)

```
DATCAP23/PR:0,DATCAP23/-SH=DATCAP23,INTRN,LODOBS,CLKMES,PB52CL,DDATE
SCANIN,SNDCMD,LODBUF
CURTIM,TCON,TREF,JDATE
MOVEC,OUTQIO
[1,1]11SLIB/LB
/
UNITS=6
ASG=TT1:2,TT0:1,TT3:5:6
COMMON= GLB1:RW
COMMON= GLB2:RW
COMMON= GLB4:RW
TASK=   DATCAP
MAXBUF= 256
ACTFIL= 3
PRI=    85
LIBR=   11SRES:RO
//
>
```

Figure D-8. DATCAP23.CMD (Command File to Build DATCAP23 Task)

```
INPPRO
IPINIT
GTHEAD
STERR
STEXP
STINIT
STEST
STTDRS
STMANS
STTRKS
STMISC
STSTAN
STGEO
STATM
STTIMF
INPFRM
LOKAHD
CURTIM
JDATE
TCON
TIMCON
TREF
OUTQIO
>
```

Figure D-9. INPPRO.TXT (Text File to Compile INPPRO Modules)

```
;
;    INPPRO.CMD     COMMAND FILE TO TASK BUILD THE INPUT PROCESSOR
;                   (PDP 11/70 VERSION)
;
INPPRO=INPPRO,IPINIT,GTHEAD,STERR,STEXP,STINIT,STEST,STTDRS,STMANS,
      STTRKS,STMISC,STSTAN,STGEO,STATM,STTIMF,INPFRM,
      LOKAHD,
      CURTIM,JDATE,TCON,TIMCON,TREF,
      [206,1]MOVEC,
      OUTQIO
      DB0:[1,1]11SLIB/LB
/
UNITS=  6
ACTFIL= 0
ASG=    TI:5
ASG=    TI:6
MAXBUF= 80
PRI=    50
TASK=   INPPRO
COMMON= GLB4:RW
COMMON= GLB1:RW
COMMON= GLB2:RW
LIBR  = 11SRES:RO
//
>
```

Figure D-10. INPPRO.CMD (Command File to Build INPPRO Task)

```
;
;    INPPRO23.CMD    COMMAND FILE TO TASK BUILD THE INPUT PROCESSOR
;                  (LSI 11/23 VERSION)
;
INPPRO23,INPPRO23=INPPRO23,IPINIT,GTHEAD,STERR,STEXP,STINIT,STEST,STTDRS,
      STMANS,STTRKS,STMISC,STSTAN,STGEO,STATM,STTIMF,INPFRM,
      LOKAHD,OUTQIO,
      CURTIM,JDATE,TCON,TIMCON,TREF,
      [206,1]MOVEC,
      DB0:[1,1]11SLIB/LB
/
UNITS=  6
ACTFIL= 0
ASG=    TT3:5:6
MAXBUF= 80
PRI=    50
TASK=   INPPRO
COMMON= GLB4:RW
COMMON= GLB1:RW
COMMON= GLB2:RW
LIBR  = 11SRES:RO
//
>
```

Figure D-11. INPPRO23.CMD (Command File to Build INPPRO23 Task)

```
PREPRO
OBSPRE
TDRORB
PPINIT
PURFIL
RANGRT
SETINX
OUTQIO
DSPOBS
SETINX
CNVTTM
CNVTOB
DDATE
JDATE
TAGOBS
TCON
TREF
PB52CL
>
```

Figure D-12. PREPRO.TXT (Test File to Compile PREPRO Modules)

```
;
;     PREPRO.CMD        COMMAND FILE TO TASK BUILD THE DATA PREPROCESSOR
;                       (PDP 11/70 VERSION)
;
PREPRO,PREPRO/-SP=PREPRO,OBSPRE,TDRORB,PPINIT,PURFIL,RANGRT,SETINX
       CNVTTM,CNVTOB,DDATE,JDATE,TAGOBS,TCON,TREF,PB52CL
       VSEND,VRCEVE,
       OUTQIO,DSPOBS
       DB0:[1,1]11SLIB/LB
/
UNITS=  6
ASG=    TI:5
ASG=    TI:6
PRI=    50
ACTFIL= 2
MAXBUF= 80
TASK=   PREPRO
COMMON= GLB1:RW
COMMON= GLB2:RW
LIBR=   11SRES:RO
//
>
```

Figure D-13. PREPRO.CMD (Command File to Build PREPRO Task)

```
;
;     PREPRO23.CMD       COMMAND FILE TO TASK BUILD THE DATA PREPROCESSOR
;                       (LSI 11/23 VERSION)
;
PREPRO23,PREPRO23/-SP=PREPRO23,OBSPRE,TDRORB,PPINIT,PURFIL,RANGRT,SETINX
       CNVTTM,CNVTOB,DDATE,JDATE,TAGOBS,TCON,TREF,PB52CL
       VSEND,VRCEVE,
       OUTQIO
       DB0:[1,1]11SLIB/LB
/
UNITS=  6
ASG=    TT3:5:6
PRI=    50
ACTFIL= 2
MAXBUF= 80
TASK=   PREPRO
COMMON= GLB1:RW
COMMON= GLB2:RW
LIBR=   11SRES:RO
//
>
```

Figure D-14. PREPRO23.CMD (Command File to Build PREPRO23 Task)

```
DATMGR
LOCTDR
PURTDR
PUROBS
OUTQIO
>
```

Figure D-15. DATMGR.TXT (Text File to Compile DATMGR Modules)

```
;
;     DATMGR.CMD     COMMAND FILE TO TASK BUILD THE DATA MANAGER
;                    (PDP 11/70 VERSION)
;
DATMGR,DATMGR/-SP=DATMGR,LOCTDR,PURTDR,PUROBS
       VSEND,VRCEVE,
       OUTQIO,
       DB0:[1,1]11SLIB/LB
/
UNITS=  6
ACTFIL= 0
ASG=    TI:5:6
PRI=    60
MAXBUF= 80
COMMON= GLB1:RW
TASK=   DATMGR
LIBR=   11SRES:RO
//
>
```

Figure D-16. DATMGR.CMD (Command File to Build DATMGR Task)

```
;
;     DATGMR23.CMD     COMMAND FILE TO TASK BUILD THE DATA MANAGER
;                     (LSI 11/23 VERSION)
;
DATMGR23,DATMGR23/-SP=DATMGR23,LOCTDR,PURTDR,PUROBS
      VSEND,VRCEVE,
      OUTQIO
      DB0:[1,1]11SLIB/LB
/
UNITS=  6
ACTFIL= 0
ASG=    TT3:5:6
PRI=    60
MAXBUF= 80
COMMON= GLB1:RW
TASK=   DATMGR
LIBR=   11SRES:RO
//
>
```

Figure D-17. DATMGR23.CMD (Command File to Build DATMGR23 Task)

```
ESTIM
ESLIDE
ESMNVR
SLINIT
SLITER
SLEND
SLSUMS
SLEDIT
SLTEST
SLUPDT
SLOUT
SLCORR
SYMINV
MATMUL
OBSRW
ESINIT
MATPRE
MATPST
OBSUPD
PREDIT
>
```

Figure D-18. ESTIM.TXT (Text File to Compile ESTIM Modules)

```
;       ESTIM.CMD   TASK BUILDING COMMAND FILE OF ESTIMATOR TO PROCESS
;                   ONLY TDRSS DATA - SRE CAPABILITY REMOVED.
;                   (PDP 11/70 VERSION)
;
ESTIM,ESTIM=ESTIM,ESLIDE,ESMNVR,SLINIT,SLITER,SLEND,SLSUMS,SLEDIT,
                SLTEST,SLUPDT,SLOUT,SLCORR,SYMINV,
                OBSRW,ESINIT,MATPRE,MATPST,MATMUL,
                OBSUPD,PREDIT,DSPOBS
;
;
; AODS EXECUTIVE ROUTINES AND BLOCK DATAS
;
                TCON,TYMD,JDATE
                OUTQIO
                VRCEVE,VSEND
;
; SYSTEM STUFF
;
                [1,1]11SLIB/LB
/
ASG=    TI:5
ASG=    TI:6
ASG=    TI:1:2
MAXBUF= 80
COMMON= GLB1:RW
LIBR=   11SRES:RO
COMMON= GLB3:RW
UNITS=  6
ACTFIL= 2
//
>
```

Figure D-19. ESTIM.CMD (Command File to Compile ESTIM Task)

```
;       ESTIM23.CMD   TASK BUILDING COMMAND FILE OF ESTIMATOR TO PROCESS
;                     ONLY TDRSS DATA - SRE CAPABILITY REMOVED.
;                     (LSI 11/23 VERSION)
;
ESTIM23,ESTIM23=ESTIM23,ESLIDE,ESMNVR,SLINIT,SLITER,SLEND,SLSUMS,SLEDIT,
                SLTEST,SLUPDT,SLOUT,SLCORR,SYMINV,
                OBSRW,ESINIT,MATPRE,MATPST,MATMUL,
                OBSUPD,PREDIT
;
;
; AODS EXECUTIVE ROUTINES AND BLOCK DATAS
;
                TCON,TYMD,JDATE
                OUTQIO
                VRCEVE,VSEND
;
; SYSTEM STUFF
;
                [1,1]11SLIB/LB
/
ASG=    TT3:5:6
MAXBUF= 80
COMMON= GLB1:RW
COMMON= GLB3:RW
TASK=   ESTIM
LIBR=   11SRES:RO
UNITS=  6
ACTFIL= 2
//
>
```

Figure D-20. ESTIM23.CMD (Command File to Build ESTIM23 Task)

```
OBSMDL
BEHUN
BODFIX
ELVANG
GETORB
GETTDR
LTIMT1
ONELEG
SORBIT
STPV
TDR1WM
TDR1WP
>
```

Figure D-21. OBSMDL.TXT (Test File to Compile OBSMDL Modules)

```
;      OBSMDL.CMD     TASK BUILDING COMMAND FILE OF MODEL TASK FOR
;                     AODS ESTIMATOR TO PROCESS TDRSS DATA ONLY
;                     (PDP 11/70 VERSION)
;
OBSMDL,OBSMDL/-SP/SH=OBSMDL,BEHUN,BODFIX,ELVANG,GETORB,
                  GETTDR,GHAUPN,LATLON,LGN,LTC
                  LTIMT1,MATMUL,ONELEG,
                  SORBIT,STPV,TDR1WM,TDR1WP
;
; AODS EXECUTIVE ROUTINES AND BLOCK DATAS
;
                  JDATE,OUTQIO,TREF,TYMD,VRCEVE,VSEND
;
; SYSTEM STUFF
;
                [1,1]11SLIB/LB
/
ASG=     TI:5
MAXBUF=  80
COMMON=  GLB1:RW
LIBR=    11SRES:RO
COMMON=  GLB3:RW
UNITS=   5
ACTFIL=  2
//
>
```

Figure D-22. OBSMDL.CMD (Command File to Build OBSMDL Task)

```
;      OBSMDL.CMD     TASK BUILDING COMMAND FILE OF MODEL TASK FOR
;                     AODS ESTIMATOR TO PROCESS TDRSS DATA ONLY
;                     (LSI 11/23 VERSION)
;
OBSMDL23,OBSMDL23/-SP/SH=OBSMDL23,BEHUN,BODFIX,ELVANG,GETORB,
                  GETTDR,GHAUPN,LATLON,LGN,LTC
                  LTIMT1,MATMUL,ONELEG,
                  SORBIT,STPV,TDR1WM,TDR1WP
;
; AODS EXECUTIVE ROUTINES AND BLOCK DATAS
;
                  JDATE,OUTQIO,TREF,TYMD,VRCEVE,VSEND
;
; SYSTEM STUFF
;
                [1,1]11SLIB/LB
/
ASG=     TT3:5
MAXBUF=  80
COMMON=  GLB1:RW
COMMON=  GLB3:RW
TASK=    OBSMDL
LIBR=    11SRES:RO
UNITS=   5
ACTFIL=  2
//
>
```

Figure D-23. OBSMDL23.CMD (Command File to Build OBSMDL23 Task)

```
DOPPRE
DOPLEG
DOPLTM
DOPMDL
DPINIT
TDRINT
TGTINT
BODFIX
GHAUPN
LTC
LATLON
ELVANG
LGN
MATMUL
BEHUN
STPV
TREF
JDATE
>
```

Figure D-24. DOPPRE.TXT (Text File to Compile DOPPRE Modules)

```
;     DOPPRE.CMD     COMMAND FILE TO TASK BUILD ONE-WAY DOPPLER
;                    PREDICTOR
;                    (PDP 11/70 VERSION)
;
DOPPRE,DOPPRE=DOPPRE,DOPLEG,DOPLTM,DOPMDL,DPINIT,TDRINT,TGTINT,
      BODFIX,GHAUPN,LTC,LATLON,ELVANG,LGN,MATMUL,BEHUN,STPV,
      TREF,JDATE,VSEND,VRCEVE,
      OUTQIO
      [1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB3:RW
MAXBUF= 80
UNITS=  6
ASG=    TI:6,TI:5,TI:1
PRI=    50
TASK=   DOPPRE
LIBR=   11SRES:RO
//
>
```

Figure D-25. DOPPRE.CMD (Command File to Build DOPPRE Task)

```
;     DOPPRE23.CMD     COMMAND FILE TO TASK BUILD ONE-WAY DOPPLER
;                      PREDICTOR
;                      (LSI 11/23 VERSION)
;
DOPPRE23,DOPPRE23=DOPPRE23,DOPLEG,DOPLTM,DOPMDL,DPINIT,TDRINT,TGTINT,
       BODFIX,GHAUPN,LTC,LATLON,ELVANG,LGN,MATMUL,BEHUN,STPV,
       TREF,JDATE,VSEND,VRCEVE,
       OUTQIO
       [1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB3:RW
MAXBUF= 80
UNITS=  6
ASG=    TT3:6,TT3:5
PRI=    50
TASK=   DOPPRE
LIBR=   11SRES:RO
//
>
```

Figure D-26. DOPPRE23.CMD (Command File to Build DOPPRE23 Task)

```
OUTPRO
OUTTRN
OPINIT
CKPRIO
LODACT
LODERR
LODDCS
LODDPL
LODRES
LODSEN
LODVEC
DWNSND70
JDATE
OUTTIM
TCON
TREF
TYMD
MOVEC
GETTIM
TYMDA
DSPRES
>
```

Figure D-27. OUTPRO.TXT (Text File to Compile OUTPRO Modules)

```
OUTPRO,OUTPRO=OUTPRO,OUTTRN,OPINIT
              CKPRIO,LODACT,LODERR,LODDCS,LODDPL,
              LODRES,LODSEN,LODVEC,DWNSND70,JDATE,
              OUTTIM,TCON,TREF,TYMD,MOVEC,
              DSPRES,
              OUTQIO,VSEND,VRCEVE,CNVCW
              DB0:[1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB3:RW
PRI=50
ACTFIL=0
MAXBUF=80
ASG=TT24:3,TI:5:6
TASK=OUTPRO
LIBR= 11SRES:RO
//
>
```

Figure D-28. OUTPRO.CMD (Command File to Build OUTPRO Task)

```
OUTPRO23,OUTPRO23=OUTPRO23,OUTTRN,OPINIT
                CKPRIO,LODACT,LODERR,LODDCS,LODDPL,
                LODRES,LODSEN,LODVEC,DWNSND,JDATE,
                OUTTIM,TCON,TREF,TYMD,MOVEC,
                OUTQIO,VSEND,VRCEVE,CNVCW,
                DB0:[1,1]11SLIB/LB
/
COMMON=GLB1:RW
COMMON=GLB3:RW
PRI=50
ACTFIL=0
MAXBUF=80
ASG=TT1:3,TT3:5:6,TT2:1
TASK=OUTPRO
LIBR= 11SRES:RO
//
```

Figure D-29. OUTPRO23.CMD (Command File to Build OUTPRO23 Task)

```
STAPRE
OUTQIO
>
```

Figure D-30. STAPRE.TXT (Text File to Compile STAPRE Modules)

```
;
;     STAPRE.CMD     COMMAND FILE TO TASK BUILD THE STATE PREDICTOR
;                    (PDP 11/70 VERSION)
;
       STAPRE=STAPRE
       VSEND,VRCEVE
       OUTQIO
       DB0:[1,1]11SLIB/LB
/
UNITS=  6
ASG=    TI:5:6
ACTFIL= 0
MAXBUF= 80
PRI=    50
TASK=   STAPRE
COMMON= GLB1:RW
COMMON= GLB3:RW
LIBR=   11SRES:RO
//
>
```

Figure D-31. STAPRE.CMD (Command File to Build STAPRE Task)

```
;
;     STAPRE23.CMD    COMMAND FILE TO TASK BUILD THE STATE PREDICTOR
;                   (LSI 11/23 VERSION)
;
       STAPRE23,STAPRE23=STAPRE23
       VSEND,VRCEVE
       OUTQIO
       DB0:[1,1]11SLIB/LB
/
UNITS=  6
ASG=    TT3:5:6
ACTFIL= 0
MAXBUF= 80
PRI=    50
TASK=   STAPRE
COMMON= GLB1:RW
COMMON= GLB3:RW
LIBR=   11SRES:RO
//
>
```

Figure D-32. STAPRE23.CMD (Command File to Build STAPRE23 Task)

```
ABAM
ACCEL
ATMOS
FORCV
GHAUPD
INTP
INV2
LUNA
MSTEP
ORBINI
ORBIT
PARTLS
REDUCE
RUKUTT
SOL
SOLLUN
SPART
SPARTV
SSTEP
STATES
SUMS
>
```

Figure D-33. ORBIT.TXT (Text File to Compile ORBIT Modules)

```
>
 ;
;  ORBIT.CMD       COMMAND FILE TO TASK BUILD THE ORBIT PROPAGATOR
;                  (PDP 11/70 VERSION)
;
ORBIT= ORBIT,ACCEL,SOLLUN,SPART,ATMOS,SPARTV
       RUKUTT,SSTEP,SUMS,ORBINI,MSTEP,INTEG,GHAUPD
       LUNA,SOL,REDUCE,ABAM,INTCON
       INV2,FORCV,PARTL,HARM,INTP,STATES
       VRCEVE,VSEND
       OUTQIO
       DB0:[1,1]11SLIB/LB
/
ACTFIL= 0
MAXBUF= 80
UNITS=  6
ASG=    TI:5
TASK=   ORBIT
PRI=    60
COMMON= GLB1:RW
LIBR=   11SRES:RO
//
```

Figure D-34(a). ORBIT23.CMD (Command File to Build ORBIT23 Task)

```
;
;  ORBIT23.CMD       COMMAND FILE TO TASK BUILD THE ORBIT PROPAGATOR
;                   (LSI 11/23 VERSION)
;
ORBIT23,ORBIT23=ORBIT23,ACCEL,SOLLUN,SPART,ATMOS,SPARTV
       RUKUTT,SSTEP,SUMS,ORBINI,MSTEP,INTEG,GHAUPD
       LUNA,SOL,REDUCE,ABAM,INTCON
       INV2,FORCV,PARTL,HARM,INTP,STATES
       VRCEVE,VSEND
       DB0:[1,1]11SLIB/LB
/
ACTFIL= 0
MAXBUF= 80
UNITS=  6
ASG=    TT3:5
TASK=   ORBIT
PRI=    60
COMMON= GLB1:RW
LIBR=   11SRES:RO
//
>
```

Figure D-34(b). ORBIT.CMD (Command File to Build ORBIT Task)

```
;
;     GLB1.FOR    COMMAND FILE TO COMPILE THE GLOBAL COMMON
;                 BLOCK DATA FILES FOR GLOBAL COMMON /GLB1/
;                 (FOR PDP 11/70 AND LSI 11/23)
;
FOR ACTLOG=ACTLOG
FOR ACTVAR=ACTVAR
FOR CONTRL=CONTRL
FOR ERRMSG=ERRMSG
FOR ESTPRM=ESTPRM
FOR INITAB=INITAB
FOR OPTAB=OPTAB
FOR PHYCON=PHYCON
FOR SYSEVN=SYSEVN
FOR TSKCOM=TSKCOM
)
```

Figure D-35. GLB1.FOR (Command File to Compile Global COMMON /GLB1/)


```
;
;  GLB1.CMD        COMMAND FILE TO TASK BUILD GLOBAL COMMON 1.
;                    (FOR PDP 11/70 AND LSI 11/23)
;
[1,1]GLB1/PI/-HD,GLB1/SH/-SP,GLB1/PI= [224,4]PHYCON,INITAB,ERRMSG,ACTLOG,
                                      [224,4]ESTPRM,CONTRL,ACTVAR,SYSEVN,EXPARM,
                                            [224,4]TSKCOM,OPTAB
/
STACK=  0
PAR=    GLB1:160000:20000
UNITS=  0
//
)
```

Figure D-36. GLB1.CMD (Command File to Build Global COMMON /GLB1/)

```
;
;     GLB2.FOR    COMMAND FILE TO COMPILE THE GLOBAL COMMON
;                 BLOCK DATA FILES FOR GLOBAL COMMON /GLB2/
;                 (FOR PDP 11/70 AND LSI 11/23)
;
FOR NEWTDR=NEWTDR
FOR MSCHED=MSCHED
FOR OBSQ=OBSQ
FOR TSCHED=TSCHED
>
```

Figure D-37. GLB2.FOR (Command File to Compile Global COMMON /GLB2/)


```
;
;  GLB2.CMD        COMMAND FILE TO TASK BUILD GLOBAL COMMON 2.
;                    (FOR PDP 11/70 AND LSI 11/23)
;
[1,1]GLB2/PI/-HD,GLB2/SH/-SP,GLB2/PI= [224,4]NEWTDR,OBSQ,TSCHED,MSCHED
/
STACK=  0
PAR=    GLB2:140000:20000
UNITS=  0
//
>
```

Figure D-38. GLB2.CMD (Command File to Build Global COMMON /GLB2/)

```
;
;     GLB3.FOR     COMMAND FILE TO COMPILE THE GLOBAL COMMON
;                  BLOCK DATA FILES FOR THE GLOBAL COMMON
;                  BLOCK /GLB3/
;
FOR BEHCOM=BEHCOM
FOR DCSTAT=DCSTAT
FOR DEBUG=DEBUG
FOR ESTLOC=ESTLOC
FOR MBATCH=MBATCH
FOR MODCOM=MODCOM
FOR OBSREC=OBSREC
FOR OUTDPL=OUTDPL
FOR OUTVEC=OUTVEC
FOR RELCOM=RELCOM
FOR RESRPT=RESRPT
>
```

Figure D-39. GLB3.FOR (Command File to Compile Global COMMON /GLB3/)

```
;
;  GLB3.CMD        COMMAND FILE TO TASK BUILD GLOBAL COMMON /GLOBL3/.
;                    (FOR PDP 11/70 AND LSI 11/23)
;
[1,1]GLB3/PI/- HD,GLB3/SH/- SP,GLB3/PI= [224,4]DCSTAT,RESRPT,OUTVEC,
                                        [224,4]OUTDPL,
                                        [224,4]ESTLOC,MODCOM,RELCOM,
                                        [224,4]BEHCOM,MBATCH,OBSREC
/
STACK= 0
PAR=   GLB3:20000:40000
UNITS= 0
//
>
```

Figure D-40. GLB3.CMD (Command File to Build Global COMMON /GLB3/)

```
;
;    GLB4.FOR     COMMAND FILE TO COMPILE THE GLOBAL COMMON
;                 BLOCK DATA FILES FOR THE GLOBAL COMMON
;                 BLOCK /GLB4/
;                 (PDP 11/70 AND LSI 11/23)
;
FOR INPBUF=INPBUF
>
```

Figure D-41. GLB4.FOR (Command File to Compile Global COMMON /GLB4/)


```
;
;  GLB4.CMD        COMMAND FILE TO TASK BUILD GLOBAL COMMON /GLOBL4/.
;                    (FOR PDP 11/70 AND LSI 11/23)
;
[1,1]GLB4/PI/-HD,GLB4/SH/-SP,GLB4/P1= [224,4]INPBUF
/
STACK= 0
PAR=   GLB4:100000:30000
UNITS= 0
//
>
```

Figure D-42. GLB4.CMD (Command File to Build Global COMMON /GLB4/)

```
TKB @EXEC
TKB @DATCAP
TKB @INPPRO
TKB @PREPRO
TKB @DATMGR
TKB @ESTIM
TKB @DOPPRE
TKB @OUTPRO
TKB @STAPRE
TKB @ORBIT
TKB @OBSMDL
>
```

Figure D-43. TKB.CMD (Command File to Build All FEDS Tasks, PDP-11/70 Version)

```
TKB @EXEC23
TKB @DATCAP23
TKB @INPPRO23
TKB @PREPRO23
TKB @DATMGR23
TKB @ESTIM23
TKB @DOPPRE23
TKB @OUTPRO23
TKB @STAPRE23
TKB @ORBIT23
TKB @OBSMDL23
>
```

Figure D-44. TKB23.CMD (Command File to Build All FEDS Tasks, LSI-11/23 Version)

```
;
;       COMPILE.CMD   COMMAND FILE TO COMPILE ANY SUBSYSTEMS
;
        .ENABLE SUBSTITUTION
.START:
;
!ENTER SUBSYSTEM
        .ASKS  SUB ENTER SUBSYSTEM
        .IF SUB EQ "ADPREP" .GOTO REST
        .IF SUB EQ "EDITSS" .GOTO REST
        .IF SUB EQ "ADSIM"  .GOTO REST
        .IF SUB EQ "RECEEV" .GOTO REST
        .IF SUB EQ "DNLINK" .GOTO REST
        .IF SUB EQ "SCREEN" .GOTO REST
        .IF SUB EQ "ADOUT"  .GOTO REST
        .IF SUB EQ "SIMMER" .GOTO REST
        .IF SUB EQ "SIMCB"  .GOTO REST
        .IF SUB EQ "DBMINI" .GOTO REST
;REQUESTED SYSTEM NOT FOUND
        .GOTO START
.REST:
;OPEN FILE FOR ALL SUBSYSTEMS
        .OPENR 'SUB'.TXT
;  COMPILE ALL MODULES
.NEXT:
        .READ MOD
        .IFT <EOF> .GOTO DONE
        .IF <FILERR> NE 1 .GOTO START
FOR 'MOD'
        .IF <EXSTAT> NE <SUCCES> .GOSUB ERR
        .GOTO NEXT
.DONE:
        .CLOSE
        .EXIT
.ERR:
RUN DB0:[201,6]BELL
        .ASKS CR CR TO CONTINUE
        .IF CR <> "" .EXIT
        .RETURN
>
```

Figure D-45. CMDADEPT.CMD (Executive Command File to Compile ADEPT)

```
ADDSUB
ADPREP
ATMPRT
BLDSIM
BUF
BUILD
CENTER
CMDCHK
CMDCOM
DATCHK
DBM
DBMEDI
DBMPRT
DCODE
DIRCOM
DIRECT
ERROR
FIELD
FILES
FLAG
GEOPRT
GETINP
GETOB
GETPRM
GETTRK
HEADER
IDCODE
LENM
LINFIL
LSTREC
MERGE
OBSCHK
OBSSCH
PAGOUT
PARA
PARAM
PARPRT
PDMP
PDUMPF
PHYCON
SFILES
SPCPRT
SRTTRK
STRPRM
TIMCHK
TRKSCH
TSORT
VERIFY
YMDHMS
>
```

Figure D-46. ADPREP.TXT (Text File to Compile ADPREP Modules)

```
EDITSS
EDIT
WRT
OPENS
GETLIN
MOVE
REMVB
DECNUM
COMPL
ADD
ADDP
BOT
DEL
DELP
EXT
KILL
LSTT
LSTP
NEX
NEXP
OVE
RET
SAV
TOP
PRI
TYP
UNS
LOC
INS
TLOC
INSBLK
INSCMD
INSCOM
INSLIN
INSOBS
CHG
SCHG
>
```

Figure D-47. EDITSS.TXT (Text File to Compile Editor Used in ADPREP Task)

```
;
;       ADPREP.TKB      COMMAND FILE TO TASK BUILD THE DATA PREPROCESSOR
;                       SUBTASK. FULL SYSTEM.
;
[224,2]ADPREP,[224,2]ADPREP/SH/-SP=[224,2]ADPREP/MP
UNITS=10
ASG=SY:1:2:3:4:7:8:9:10,TI:5:6
MAXBUF=1036
ACTFIL=7
//
)
```

Figure D-48. ADPREP.CMD (Command File to Build ADPREP Task)

```
;
;       ADPREP.ODL      COMMAND FILE THAT CONTROLS THE OVERLAY OF THE
;                       DATA PREPARATION SYSTEM.
;                       MODIFIED TO USE THE ROUTINES MODIFIED
;                       FOR FEDS
;
        .ROOT [224,2]ADPREP-ROOT1-*(SEG1-(EXT1),SEG2,SEG3-(EXT3))
EXT1:   .FCTR LEGA,LEGB-(PART1-(A,B,C),PART2),LEGC-(ANKLA,ANKLB)
EXT3:   .FCTR ARMA,ARMB
ROOT1:  .FCTR [224,2]MENU-[224,2]PAGFIL-[224,2]GETI2-ROOT2
ROOT2:  .FCTR [224,2]LENM-[224,2]CENTER-ROOT3
ROOT3:  .FCTR [224,2]ERROR-[224,2]PDUMPF-[224,2]FILES-ROOT4
ROOT4:  .FCTR [224,2]PAGOUT-[224,2]MOVEC-[224,2]PDMP-ROOT5
ROOT5:  .FCTR [224,2]LUNCOM-[224,2]CMDCOM-[224,2]YMDHMS-ROOT6
ROOT6:  .FCTR [224,2]DIRECT
SEG1:   .FCTR [224,2]FILLD-[224,2]GETPRM-[224,2]ADDSUB-SEG12
SEG12:  .FCTR [224,2]PARENS-[224,2]IBYTE-SEG13
SEG13:  .FCTR [224,2]GETINP-[224,2]LINFIL-[224,2]STRPRM
LEGA:   .FCTR [224,2]PARA-[224,2]IOFF-[224,2]SFILES-LEGA1
LEGA1:  .FCTR [224,2]IDCODE
LEGB:   .FCTR [224,2]DBM
PART1:  .FCTR [224,2]DBMEDI-[224,2]SRTTRK
PART2:  .FCTR [224,2]DBMPRT-[224,2]SPCPRT-[224,2]ATMPRT-PART2A
PART2A: .FCTR [224,2]GEOPRT-[224,2]PARPRT
A:      .FCTR [224,2]TRKSCH-[224,2]GETTRK
B:      .FCTR [224,2]OBSSCH-[224,2]GETOB
C:      .FCTR [224,2]PARAM-[224,2]DCODE
LEGC:   .FCTR [224,2]VERIFY-[224,2]TIMCHK-[224,2]MOVEB
ANKLA:  .FCTR [224,2]DATCHK-[224,2]PARPRT-ANKLA1
ANKLA1: .FCTR [224,2]ATMPRT-[224,2]GEOPRT
ANKLB:  .FCTR [224,2]CMDCHK-[224,2]OBSCHK
SEG2:   .FCTR [224,2]BUILD-[224,2]BLDSIM-[224,2]MERGE-SEG2A
SEG2A:  .FCTR [224,2]TSORT-[224,2]ADDSUB
SEG3:   .FCTR [224,2]EDITSS-[224,2]EDIT-[224,2]WRT-SEG3A
SEG3A:  .FCTR [224,2]OPENS-[224,2]GETLIN-[224,2]MOVE-SEG3B
SEG3B:  .FCTR [224,2]REMVB-[224,2]DECNUM-[224,2]COMPL
ARMA:   .FCTR [224,2]ADD-[224,2]ADDP-[224,2]BOT-ARMA1
ARMA1:  .FCTR [224,2]DEL-[224,2]DELP-[224,2]EXT-ARMA2
ARMA2:  .FCTR [224,2]KILL-[224,2]LSTT-[224,2]LSTP-ARMA3
ARMA3:  .FCTR [224,2]NEX-[224,2]NEXP-[224,2]OVE-ARMA4
ARMA4:  .FCTR [224,2]RET-[224,2]SAV-[224,2]TOP-ARMA5
ARMA5:  .FCTR [224,2]PRI-[224,2]TYP-[224,2]UNS-ARMA6
ARMA6:  .FCTR [224,2]LOC
ARMB:   .FCTR [224,2]INS-[224,2]TLOC-[224,2]TIMCHK-ARMBB
ARMBB:  .FCTR [224,2]INSBLK-[224,2]INSCMD-[224,2]INSCOM-ARMBC
ARMBC:  .FCTR [224,2]INSLIN-[224,2]INSOBS-ARMB1
ARMB1:  .FCTR [224,2]CHG-[224,2]SCHG
        .END
>
```

Figure D-49. ADPREP.ODL (Overlay Descriptor for ADPREP Task)

```
ADSIM
LENM
TIMCHK
FILES
YMDHMS
SIMINI
IDCODE
PAGOUT
MOVEC
ERROR
PDMP
PDUMPF
ESTMOD
DCODE
PARAM
GETPRM
IBYTE
STRPRM
FIELD
GETINP
LINFIL
PARENS
CENTER
DIRECT
PARA
IOFF
SIMOPT
OUTMOD
MENU
PAGFIL
GETI2
CENTER
>
```

Figure D-50. ADSIM.TXT (Text File to Compile ADSIM Modules)

```
;
;       ADSIM.CMD       COMMAND FILE TO TASK BUILD THE SIMULATION
;                       CONTROL COMPONENT.
;
ADSIM,ADSIM/SH/-SP=ADSIM/MP
LIBR=FCSRES:RO
COMMON=ADSGBL:RW:7
ACTFIL=6
MAXBUF=1036
UNITS=20
ASG=SY:1:2:3:4:7:8:9:20,TI:5:6
//
>
```

Figure D-51. ADSIM.CMD (Command File to Build ADSIM Task)

```
;
;       ADSIM.ODL       COMMAND FILE TO OVERLAY THE SIMULATION
;                       CONTROL COMPONENT.
;
          .ROOT [224,2]ADSIM-R1-*(SEGA-(P1,P2),SEGB-(PART1,PART2),SEGC)
R1:       .FCTR [224,2]LENM-[224,2]TIMCHK-[224,2]FILES-R2
R2:       .FCTR [224,2]YMDHMS-[224,2]SIMINI-[224,2]IDCODE-R3
R3:       .FCTR [224,2]PAGOUT-[224,2]MOVEC-[224,2]ERROR-R4
R4:       .FCTR [224,2]PDMP-[224,2]PDUMPF
SEGA:     .FCTR [224,2]ESTMOD
P1:       .FCTR [224,2]DCODE-[224,2]PARAM-SEGA1
SEGA1:    .FCTR [224,2]GETPRM-[224,2]IBYTE-[224,2]STRPRM-SEGA2
SEGA2:    .FCTR [224,2]FIELD-[224,2]GETINP-[224,2]LINFIL-SEGA3
SEGA3:    .FCTR [224,2]PARENS-[224,2]CENTER
P2:       .FCTR [224,2]DIRECT
SEGB:     .FCTR [224,2]PARA-[224,2]IOFF-[224,2]GETPRM-SEGB1
SEGB1:    .FCTR [224,2]IBYTE-[224,2]STRPRM-[224,2]FIELD-SEGB2
SEGB2:    .FCTR [224,2]GETINP-[224,2]LINFIL-[224,2]PARENS
PART1:    .FCTR [224,2]SIMOPT
PART2:    .FCTR [224,2]OUTMOD
SEGC:     .FCTR [224,2]MENU-[224,2]PAGFIL-[224,2]GETI2-SEGC1
SEGC1:    .FCTR [224,2]CENTER
          .END
>
```

Figure D-52. ADSIM.ODL (Overlay Descriptor for ADSIM Task)

```
SIMMER
UPLINK
SIMTIM
URAND
RTRAN
UPSEND
CRRUPT
SETRE
MSGGEN
UPCMD
UPBLOK
UPDATA
DBURST
DNOISE
GAUSS
CVTDB0
>
```

Figure D-53. SIMMER.TXT (Text File to Compile SIMMER Modules)

```
;
;       SIMMER.CMD      COMMAND FILE TO TASK BUILD THE SIMULATION
;                       CONTROL COMPONENT
;
SIMMER,SIMMER/SH/-SP=SIMMER/MP
COMMON=ADSGBL:RW:7
ACTFIL=6
PRI=75
MAXBUF=1036
UNITS=15
ASG=SY:1:2:3:4:8:14,TI:5:6,TT32:15
//
>
```

Figure D-54. SIMMER.CMD (Command File to Build SIMMER Task)

```
;
;       SIMMER.ODL      COMMAND FILE TO OVERLAY THE SIMULATION
;                       CONTROL COMPONENT.
;
            .ROOT [224,2]SIMMER-R1-*(S1-(S1A,S1B,S1C,S1D),S2)
R1:         .FCTR [224,2]UPLINK-[224,2]SIMTIM-[224,2]URAND-R2
R2:         .FCTR [224,2]RTRAN-[224,2]FILES-[224,2]SMTREF-R3
R3:         .FCTR [224,2]OUTQIO
S1:         .FCTR [224,2]UPSEND-[224,2]CRRUPT-[224,2]SETRE-S11
S11:        .FCTR [224,2]MSGGEN-[224,1]VSEND-S12
S12:        .FCTR [224,2]SIMREF-[224,2]TCON
S1A:        .FCTR [224,2]UPCMD-[224,2]CMDCOM
S1B:        .FCTR [224,2]UPBLOK-[224,2]MOVEB
S1C:        .FCTR [224,2]UPDATA
S1D:        .FCTR [224,2]TIMCHK
S2:         .FCTR [224,2]DBURST-[224,2]DNOISE-[224,2]GAUSS-S21
S21:        .FCTR [224,2]CVTDBO-[224,2]YMDHMS
            .END
>
```

Figure D-55. SIMMER.ODL (Overlay Descriptor for SIMMER Task)

```
RECEEV
SIMREF
SMTREF
TCON
OUTQIO
PRIOR
>
```

Figure D-56. RECEEV.TXT (Text File to Compile RECEEV Modules)

```
;
;       RECEEV.CMD      COMMAND FILE TO TASK BUILD THE DOWNLINK MESSAGE
;                       CAPTURE COMPONENT.
;
RECEEV/-CP,RECEEV/-SP/SH=RECEEV,SIMREF,SMTREF
                        TCON,OUTQIO,PRIOR
/
COMMON=ADSGBL:RW
UNITS=16
ASG=TI:5:11:16,TT24:12,SY:13
MAXBUF=256
ACTFIL=4
PRI=85
TASK=RECEEV
//
>
```

Figure D-57. RECEEV.CMD (Command File to Build RECEEV Task to Support FEDS on the LSI-11/23)

```
RECEEV70,RECEEV70=RECEEV70,PRIOR,SIMREF,SMTREF,TCON,
                  OUTQIO,
                  [224,1]VSEND,VRCEVE
/
COMMON=ADSGBL:RW
PRI=85
UNITS=16
ASG=TI:5;11:16,SY:13
ACTFIL=4
MAXBUF=256
TASK=RECEEV
//
>
```

Figure D-58. RECEEV70.CMD (Command File to Build RECEEV Task to Support FEDS on the PDP-11/70)

```
DNLINK
STRPTS
MSGGEN
SIMTIM
OUTQIO
DNINIT
DNMCHK
DNEXIT
GETMSG
STDATA
STMSG
TCON
SMTREF
RESPRT
DCSRPT
>
```

Figure D-59. DNLINK.TXT (Text File to Compile DNLINK Modules)

```
;
;       DNLINK.CMD      COMMAND TO TASK BUILD THE DOWNLINK MESSAGE
;                       PROCESSOR COMPONENT.
;
DNLINK,DNLINK/SH/-SP=[224,2]DNLINK,[224,2]STRPTS,
   MSGGEN,[224,2]SIMTIM,[224,1]OUTQIO,
   [224,2]DNINIT,[224,2]DNMCHK,DNEXIT,
   GETMSG,STDATA,STMSG,
   [224,2]TCON,[224,2]SMTREF,
   [224,1]VSEND,
   [224,2]RESPRT,DCSRPT
/
ACTFIL=8
COMMON=ADSGBL:RW:7
UNITS=19
MAXBUF=256
ASG=SY:13:14:16:17:18:19,TI:5:6
//
>
```

Figure D-60. DNLINK.CMD (Command File to Build DNLINK Task)

```
;
;       DNHIST.CMD      COMMAND FILE TO TASK BUILD THE SIMULATION
;                       HISTORY GENERATION COMPONENT
;
DNHIST,DNHIST/-SP=DNHIST,[224,1]VRCEVE
/
ACTFIL=3
COMMON=ADSGBL:RW:7
PRI=80
UNITS=14
ASG=SY:14,TT2:5:6
//
>
```

Figure D-61. DNHIST.CMD (Command File to Build DNHIST Task)

```
SCREEN
MOVEC
ERROR
PDUMPF
PAGOUT
CENTER
LEN
PARENS
IBYTE
GETPRM
LINFIL
FIELD
STRPRM
TPARA
TETINP
TSTMOD
TDIRCT
TDCODE
FILES
TPARAM
TIMOPT
INCMD
TMENU
LENM
PAGFIL
TGETI2
>
```

Figure D-62. SCREEN.TXT (Text File to Compile SCREEN Modules)

```
SCREEN/-CP,SCREEN/-SP/SH=SCREEN/MP
UNITS=7
MAXBUF=256
FMTBUF=270
ACTFIL=5
ASG=TT3:2,SY:3:4:5:6:7
COMMON=ADSGBL:RW
//
>
```

Figure D-63. SCREEN.CMD (Command File to Build SCREEN Task)

```
;
;       SCREEN.ODL      COMMAND FILE TO OVERLAY THE USER I/O PROCESSOR
;                       COMPONENT.
;
                .ROOT SCREEN-R1-*(S1-(S1A,S1B),S2)
R1:             .FCTR MOVEC-ERROR-PDMP-R11
R11:            .FCTR PDUMPF-PAGOUT-CENTER
S1:             .FCTR LEN-PARENS-S11
S11:            .FCTR IBYTE-GETPRM-IOFF-S12
S12:            .FCTR LINFIL-FIELD-STRPRM-S13
S13:            .FCTR TPARA-TETINP
S1A:            .FCTR TSTMOD-TDIRCT-TDCODE-S1A1
S1A1:           .FCTR FILES-TPARAM
S1B:            .FCTR TIMOPT
S2:             .FCTR INCMD-TMENU-LENM-S21
S21:            .FCTR PAGFIL-TGETI2
                .END
>
```

Figure D-64. SCREEN.ODL (Overlay Descriptor for SCREEN Task)

```
ADOUT
OUTLUN
SMHIST
DPRPT
TIMCHK
DCRPT
SVRPT
GETDCS
GETRES
ARESRP
ADCSRP
GETTIM
EPHCMP
ADDSUB
YMDHMS
CINPUT
DIF
PARA
ERROR
PDUMPF
PDMP
FIELD
GETINP
IOFF
LINFIL
PARENS
STRPRM
MOVEC
PACOUT
LEN
GETPRM
IBYTE
CMPOPT
CMPINT
EPHED
CHEAD
EPHDAT
CONVRT
RELS8
REALS4
JCOMPAR
CNTINI
CTREQ
DIFFER
DIFOUT
CREAD1
```

Figure D-65. ADOUT.TXT (Text File to Compile ADOUT Modules (1 of 2)

```
GETDAT
CSUMMR
DIFORB
DIFRM
CACCUM
DIFWRT
MENU
PAGFIL
GETI2
ERROR
LENM
PAGOUT
CENTER
>
```

Figure D-65. ADOUT.TXT (Text File to Compile ADOUT Modules (2 of 2)

```
;
;       ADOUT.CMD       COMMAND FILE FOR TASK BUILDING THE REPORT
;                       GENERATION AND ANALYSIS SUBSYSTEM
;
ADOUT,ADOUT/SH/-SP=ADOUT/MP
UNITS=13
MAXBUF=2440
FMTBUF=270
ACTFIL=7
ASG=SY:1:2:3:6:9:12:13,TI:5,MM0:10,MM1:11
//
>
```

Figure D-66. ADOUT.CMD (Command File to Build ADOUT Task)

```
;
;       ADOUT.ODL       COMMAND FILE FOR THE OVERLAY OF THE REPORT
;                       GENERATION AND ANALYSIS SUBSYSTEM.
;
          .ROOT ADOUT-R1-*(S1,S2-(C1-C1A),MENU0)
C1A:      .FCTR (C2,C3,C4)
R1:       .FCTR OUTLUN
S1:       .FCTR SMHIST-DPRPT-TIMCHK-S12
S12:      .FCTR DCRPT-SVRPT-GETDCS-S13
S13:      .FCTR GETRES-ARESRP-ADCSRP-S14
S14:      .FCTR GETTIM
S2:       .FCTR EPHCMP-ADDSUB-S21
S21:      .FCTR YMDHMS
C1:       .FCTR CINPUT-DIF-HEAD1-C11
C11:      .FCTR HEAD2-PAR0
PAR0:     .FCTR PARA-ERROR-PDUMPF-PAR1
PAR1:     .FCTR PDMP-FIELD-GETINP-PAR2
PAR2:     .FCTR IOFF-LINFIL-PARENS-PAR3
PAR3:     .FCTR STRPRM-MOVEC-PAGOUT-PAR4
PAR4:     .FCTR LEN-GETPRM-IBYTE
C2:       .FCTR CMPOPT
C3:       .FCTR CMPINT-EPHHED-CHEAD-C31
C31:      .FCTR EPHDAT-CONVRT-REALS8-C32
C32:      .FCTR REALS4
C4:       .FCTR COMPAR-CNTINI-CTREQ-C41
C41:      .FCTR DIFFER-DIFOUT-CREAD1-C42
C42:      .FCTR INTRP-CREAD2-GETDAT-C43
C43:      .FCTR CSUMMR-DIFORB-DIFRPT-C44
C44:      .FCTR CACCUM-DIFWRT
MENU0:    .FCTR MENU-PAGFIL-GETI2-MENU1
MENU1:    .FCTR ERROR-PDUMPF-PDMP-MENU2
MENU2:    .FCTR LENM-PAGOUT-CENTER-MENU3
MENU3:    .FCTR MOVEC
          .END
>
```

Figure D-67. ADOUT.ODL (Overlay Descriptor for ADOUT Task)

```
CLKTIM
FNDOBS
HMSCNV
JDATE
LCKSET
RCVMSG
REFCNV
SIMCB
SIMREF
SIMTIM
SMTREF
SNDMSG
TCON
TIMDIF
TYMD
YMDCNV
>
```

Figure D-68. SIMCB.TXT (Text File to Compile SIMCB Modules)

```
SIMCB,SIMCB=SIMCB,CLKTIM,FILES,FNDOBS,HMSCNV,JDATE,
LCKSET,MOVEB,RCVMSG,SIMTIM,SMTREF,SNDMSG,TCON,
YMDCNV,SIMREF,TYMD,OUTQIO,REFCNV,TIMDIF,RECLN,REVCW
/
ACTFIL=4
UNITS=21
ASG=TT21:21,TI:5:6
MAXBUF=356
COMMON=ADSGBL:RW
//
>
```

Figure D-69. SIMCB.CMD (Command File to Build SIMCB Task)

```
DBMINI
INIT
MENU
PAGFIL
GETI2
CENTER
PAGOUT
LENM
MOVEC
ERROR
PDMP
PDUMPF
FILES
LUNCOM
PARA
IOFF
IDCODE
PHYCON
INITAB
ESTPRM
EXPARM
PARENS
LINFIL
GETINP
FIELD
STRPRM
GETPRM
IBYTE
>
```

Figure D-70. DBMINI.TXT (Text File to Compile DBMINI Modules)

```
;
;       DBMINI.CMD      COMMAND FILE TO TASK BUILD THE DATA BASE
;                       INITIALIZATION UTILITY.
;
[224,2]DBMINI,[224,2]DBMINI/-SP/SH=[224,2]DBMINI,[224,2]INIT,[224,2]MENU,
                                [224,2]PAGFIL,[224,2]GETI2,[224,2]CENTER,
                                [224,2]PAGOUT,[224,2]LENM,
                                [224,2]MOVEC,[224,2]ERROR,[224,2]PDMP,
                                [224,2]PDUMPF,[224,2]FILES,[224,2]LUNCOM,
                                [224,2]PARA,[224,2]IOFF,[224,2]IDCODE,
                                [224,2]PHYCON,[224,1]INITAB,[224,1]ESTPRM,
                                [224,1]EXPARM,
                                [224,2]PARENS,[224,2]LINFIL,[224,2]GETINP,
                                [224,2]FIELD,[224,2]STRPRM,[224,2]GETPRM,
                                [224,2]IBYTE
/
ACTFIL=4
MAXBUF=1036
UNITS=10
ASG=SY:1:2,TI:5:6
//
>
```

Figure D-71. DBMINI.CMD (Command File to Build DBMINI Task)

```
;
;       MESSCR.CMD      COMMAND FILE TO TASK BUILD THE FORMAT MESSAGE
;                       CREATION UTILITY.
;
MESSCR,MESSCR/-SP/SH=MESSCR
/
UNITS=4
MAXBUF=256
ACTFIL=3
ASG=DB0:2,DB0:3,TI:4
//
>
```

Figure D-72. MESSCR.CMD (Command File to Build MESSCR Task)

```
;
;       ADSGBL.COM      COMMAND FILE TO PLACE A NEW /ADSGBL/ GLOBAL COMMON
;                       AREA INTO THE SYSTEM
;
SET /UIC=[224,2]
PIP ADSGBL.OBJ;*,ADSGBL.LST;*/DE
FOR ADSGBL,ADSGBL=ADSGBL
SET /UIC=[1,1]
PIP [1,1]ADSGBL.TSK;*,ADSGBL.STB;*,ADSGBL.MAP;*/DE
TKB @[224,2]ADSGBL
SET /UIC=[224,2]
)
```

Figure D-73. ADSGBL.COM (Command File to Control Building of Global COMMON /ADSGBL/)


```
;
;       ADSGBL.CMD      COMMAND FILE TO BUILD THE GLOBAL COMMON AREA
;                       /ADSGBL/
;
[1,1]ADSGBL/PI/-HD,ADSGBL/SH/-SP,ADSGBL/PI=[224,2]ADSGBL
/
STACK=0
PAR=ADSGBL:160000:20000
UNITS=0
//
)
```

Figure D-74. ADSGBL.CMD (Command File to Build Global COMMON /ADSGBL/)

```
RSX11S
SET /POOL=340
SET /MAIN=11SRES:340:145:COM
INS [1,1]11SRES
SET /MAIN=GLB1:*:136:COM
INS [1,1]GLB1/PAR=GLB1
SET /MAIN=GLB2:*:50:COM
INS [1,1]GLB2/PAR=GLB2
SET /MAIN=GLB3:*:344:COM
INS [1,1]GLB3/PAR=GLB3
SET /MAIN=GLB4:*:234:COM
INS [1,1]GLB4/PAR=GLB4
SET /MAIN=TTPAR:*:122:TASK
LOA TT:
SET /MAIN=EXEC:*:470:TASK
INS [224,1]EXEC23/PAR=EXEC
FIX EXEC
SET /MAIN=DATCAP:*:231:TASK
INS [224,1]DATCAP23/PAR=DATCAP
FIX DATCAP
SET /MAIN=INPPRO:*:310:TASK
INS [224,1]INPPRO23/PAR=INPPRO
FIX INPPRO
SET /MAIN=PREPRO:*:321:TASK
INS [224,1]PREPRO23/PAR=PREPRO
FIX PREPRO
SET /MAIN=DATMGR:*:724:TASK
INS [224,1]DATMGR23/PAR=DATMGR
FIX DATMGR
SET /MAIN=ESTIM:*:567:TASK
INS [224,1]ESTIM23/PAR=ESTIM
FIX ESTIM
SET /MAIN=OBSMDL:*:442:TASK
INS [224,1]OBSMDL23/PAR=OBSMDL
FIX OBSMDL
SET /MAIN=DOPPRE:*:347:TASK
```

Figure D-75. FEDS23.CMD (Command File to Build System Image for LSI-11/23) (1 of 2)

```
INS [224,1]DOPPRE23/PAR=DOPPRE
FIX DOPPRE
SET /MAIN=OUTPRO:*:317:TASK
INS [224,1]OUTPRO23/PAR=OUTPRO
FIX OUTPRO
SET /MAIN=STAPRE:*:154:TASK
INS [224,1]STAPRE23/PAR=STAPRE
FIX STAPRE
SET /MAIN=ORBIT:*:1262:TASK
INS [224,1]ORBIT23/PAR=ORBIT
FIX ORBIT
RUN EXEC
PAR
;
;       ASYNCHRONOUS COMMUNICATION LINES
;
SET /SLAVE=TT0:
SET /FDX=TT0:
SET /TYPEAHEAD=TT0:
;
SET /FDX=TT1:
SET /SLAVE=TT1:
SET /TYPEAHEAD=TT1:

;
SET /FDX=TT2:
SET /SLAVE=TT2:
SET /TYPEAHEAD=TT2:
;
;       OPERATOR'S CONSOLE
;
SET /FDX=TT3:
SET /TYPEAHEAD=TT3:
SET /SLAVE=TT3:
SET /CRT=TT3:
>
```

Figure D-75. FEDS23.CMD (Command File to Build System Image for LSI-11/23) (2 of 2)

APPENDIX E - SUMMARY OF FEDS REQUIREMENTS

This appendix contains the updated FEDS requirements presented in Reference 1. The FEDS requirements are presented according to level of detail, as follows:

- Section E.1 specifies the system requirements, which are the tasks the system must perform (on the highest level) to satisfy the needs and objectives of the end user.
- Section E.2 specifies the system performance requirements and limitations. These consist of the schedules on which specific requirements must be satisfied and any limitations that will affect the performance of the system.
- Section E.3 specifies the functional requirements, which are the functions the system must perform to satisfy the system requirements. These are the most detailed requirements given.

E.1 SYSTEM REQUIREMENTS

FEDS will be an onboard orbit determination system requiring periodic ground support. The objective of FEDS is to provide the outside world with orbit information (i.e., position and velocity) on a near-real-time basis that could be used for experimental data annotation.

For the ground demonstration, FEDS will be located on the ground with a transponder at GSFC. The external world including the ground support system will be simulated by ADEPT in GSFC's STL. Among other input, ADEPT will provide FEDS with an initial spacecraft state. During the experiment, the White Sands tracking station will perform Doppler compensation based on the corresponding ephemeris tape. The resulting Doppler signals will be transmitted through a TDRS

to the transponder connected to FEDS. Based on the initial state, FEDS will predict the Doppler frequency shift to enable the transponder to receive these signals. The Doppler measurements will then be used by FEDS to achieve a new best estimate of the state. The new state vector will be used on the next pass to predict the Doppler frequency shift. Figure E-1, the FEDS context diagram, shows the relationship of the ground demonstration version of FEDS to its external environment.

This section specifies the system requirements, i.e., the tasks that the prototype FEDS must perform to satisfy the needs and objectives of the ground demonstration. These requirements include the top-level FEDS requirements, presented in Section E.1.1, and the input and output requirements, presented in Sections E.1.2 and E.1.3, respectively.

E.1.1 TOP-LEVEL REQUIREMENTS

The top-level requirements of FEDS are as follows:

- FEDS will provide position and velocity on a near-real-time basis for experimental data annotation and direct downlink.
- FEDS will predict one-way Doppler observations on a scheduled basis for direct downlink to ADEPT and for transponder acquisition.
- FEDS will generate and output a state vector predict table containing vectors at a specified frequency over a specified time interval.
- FEDS will maintain and output an activity log on a regular basis and when specifically requested through a control command.
- FEDS will perform any preprocessing required to process the input one-way Doppler observations.

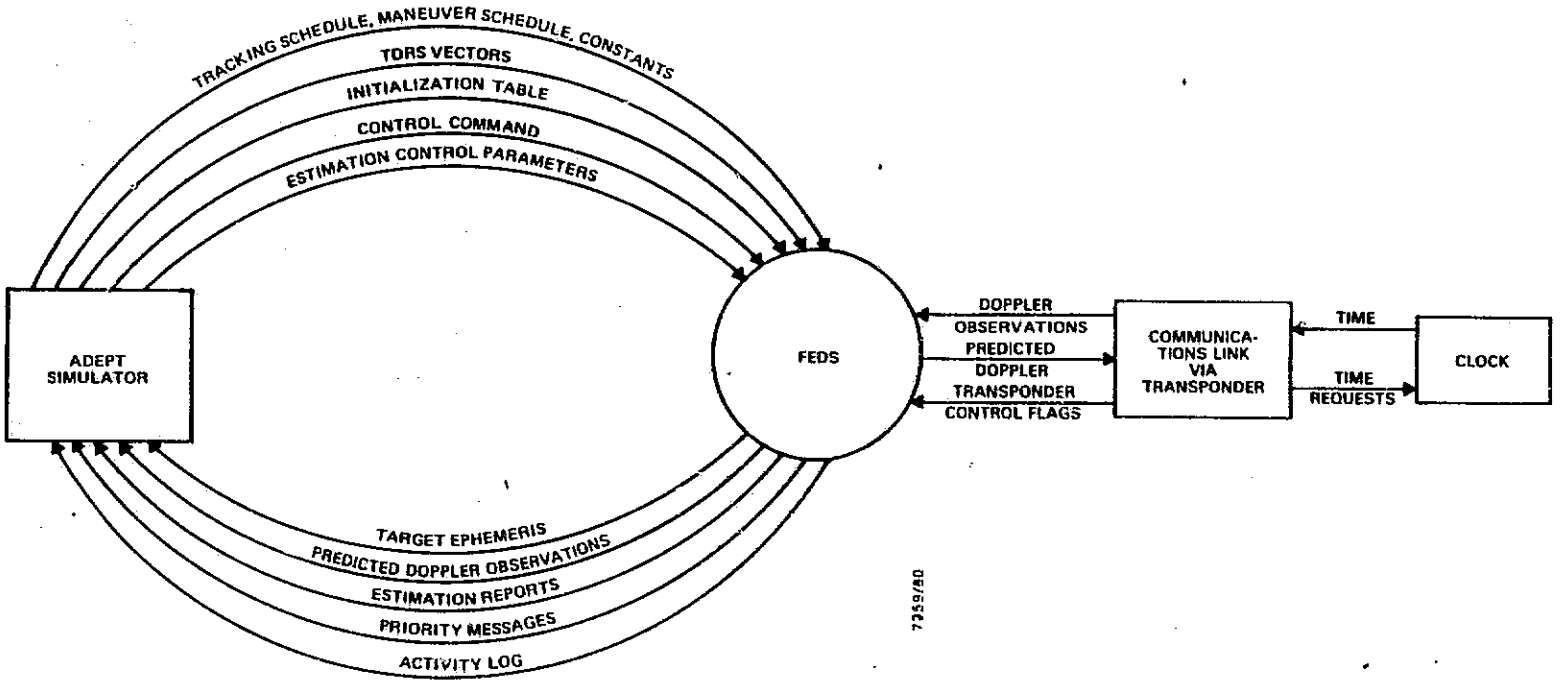


Figure E-1. FEDS Context Diagram

- FEDS will be capable of recovering from both user spacecraft and Tracking Data and Relay Satellite (TDRS) maneuvers.
- FEDS will perform orbit determination using a batch least-squares method of estimation, differentially correcting the orbit of the target (user spacecraft). FEDS will estimate the following state parameters:
  - Six parameters of the orbital state (target) (position and velocity)
  - Atmospheric drag coefficient, C_D
  - Coefficients of the frequency model for one-way TDRS System (TDRSS) data
- FEDS will process one-way TDRSS Doppler observation data.

E.1.2 INPUT REQUIREMENTS

The FEDS input requirements are as follows:

- FEDS will accept input messages containing data and control commands.
- FEDS will accept from ADEPT the following input data:
  - New TDRS vectors. These data include one state vector (position and velocity) for each active TDRS, up to two TDRSs. New TDRS vectors will be uplinked at least once per day.
  - Maneuver schedule. This schedule specifies the predicted states and times of user spacecraft and/or TDRS maneuvers. It covers up to eight maneuvers and will be uplinked as necessary. The entire maneuver schedule will be uplinked at the same time.

- Tracking schedule. This schedule is the tracking schedule for the prediction of one-way Doppler frequency shift and the annotation of observations with tracking configuration. It covers 16 tracking intervals and will be uplinked as necessary. The entire tracking schedule will be uplinked at the same time.

- Initialization table. This table specifies the initial conditions for the estimator, including the a priori state vector, which will be propagated for output until a solution is reached. This table will be uplinked at the start of FEDS execution and then later at the user's direction.

- Constants. These constants, which will be used throughout the FEDS processes, may have to be changed during long-term operations. They are categorized as follows: integration, conversion, and physical constants; station positions (minimum of 3 stations) and observation modeling constants; geopotential model constants; atmospheric drag model constants; and timing coefficients.

- Estimation control parameters. This set of parameters (e.g., maximum iterations, observation weights, convergence criteria) provides control in estimating the spacecraft state. It will be uplinked at the first estimation process and then later at the user's discretion.

- FEDS will accept Doppler observations from the communications link with the transponder consisting of a 40-bit serial word which is time tagged.

- FEDS will recognize the following control commands from ADEPT:
  - REBOOT: Reboot FEDS.
  - ABORT: Abort FEDS processing; output activity log.
  - STOP: Terminate FEDS processing in a normal manner; do not accept more data.
  - START: Start FEDS processing; accept all data. (This is a reply to commands STOP and ABORT.)
  - SUSPEND: Suspend computational processes; continue accepting data.
  - CONTINUE: Resume computations. (This is a reply to command SUSPEND.)
  - MARK TIME: Suspend all processing to allow shutdown of ground support system.
  - RESUME PROCESSING: Resume all processing: (This is a reply to command MARK TIME.)
  - BEGIN FAST TIMING: Begin fast-timing mode (i.e., compress out all idle time)
  - STOP FAST TIMING: Terminate fast-timing mode; (i.e., resume processing in real time).
  - STATUS REQUEST: Output activity log.
  - SET CLOCK: Set system clock to new time.
- FEDS will accept the following control flags from the communications link with the transponder:
  - Stop Doppler compensation indicating that the receiver carrier is locked onto the TDRS signal.

- Doppler data available flag indicating the Doppler measurement has been taken and is available for FEDS processing.

E.1.3 OUTPUT REQUIREMENTS

The FEDS output requirements are as follows:

- FEDS will periodically output an activity log containing a history of all activities it has performed.
- FEDS will output priority messages to request special ground support such as error handling, fast-timing, and so forth.
- FEDS will output tables of predicted state vectors for direct downlink to ADEPT.
- FEDS will output predicted one-way Doppler frequency shift on a scheduled basis to the transponder via the communication link for receiver acquisition.
- FEDS will output predicted one-way Doppler frequency shift on a scheduled basis for direct downlink to ADEPT.
- FEDS will output the following reports from the estimator:
  - Differential correction (DC) residuals report. This report contains information about each individual observation (e.g., tracking configuration, observation residual, editing).
  - DC summary and statistics report. This report contains DC summary information (e.g., state update, new state, standard deviations of state parameters) and DC statistics (e.g.,

current root-mean-square (rms), previous rms, batch editing statistics).

E.2 SYSTEM PERFORMANCE REQUIREMENTS AND LIMITATIONS

This section specifies those requirements that deal with system performance and the limitations associated with it. Section E.2.1 presents the system performance requirements that define the schedules on which specific requirements must be satisfied. Section E.2.2 presents the hardware and software requirements and the limitations that will affect FEDS performance.

E.2.1 SYSTEM PERFORMANCE REQUIREMENTS

The system performance requirements for FEDS are as follows:

- FEDS will capture all incoming messages upon demand.
- FEDS will service each control command immediately after reception.
- FEDS will maintain an activity log and output (downlink) it on a scheduled basis or when requested by a control command.
- FEDS will output a table of predicted user spacecraft state vectors over a specified time interval at a specified frequency. For example, if the time interval is 1/2 hour and the frequency is 1 minute, the state vector predict tables will be generated as follows:
  - Each time a new solution is reached or a new a priori state vector (initialization table) is received, a table containing state vectors at 1-minute intervals starting at the current time (t_n) and ending 1 hour later ($t_n + 1$) will be generated and output.

- Then, 1/2 hour later (t_n + 1/2), the next table will be generated and output. This table will contain state vectors at 1-minute intervals over the next 1/2 hour. The start time of this table will be the end time of the previous table (t_n + 1) and the end time will be 1/2 hour after that (t_n + 1-1/2).
- The second step will be repeated until a new solution is reached or a new a priori state vector is received, which causes the process to begin again with the first step.

- FEDS will output one-way Doppler frequency shift no later than 1 minute before the start time of the current tracking interval. The actual amount of lead time will be specified by ground control.
- FEDS will complete data preprocessing and estimation on each batch of data by the time the next pass of Doppler data is received. Since observations data will be received every revolution under normal circumstances, this processing time will be limited to the length of one revolution of the user spacecraft (nominally, 100 minutes).
- FEDS will be capable of performing batch estimation over a user-specified minimum data span that will never be larger than 24 hours. In addition, FEDS must be capable of handling a maximum of 125 observations in each batch of data.
- FEDS will be capable of generating two types of reports during each DC slide:
  - The DC residuals report, if generated, will be generated either after the last inner edit

loop of each iteration or after the last iteration on each batch of data.

- The DC summary and statistics report, if generated, will be generated either after each DC iteration or after the last iteration of each DC slide.

E.2.2 HARDWARE AND SOFTWARE REQUIREMENTS AND LIMITATIONS

The FEDS hardware and software requirements and the limitations associated with them are as follows:

- The development computer will be the Systems Technology Laboratory (STL) PDP-11/70 under the RSX-11M operating system.
- The target computer will be a PDP-11/23 under the RSX-11S operating system. It will have 256K bytes of random access memory (RAM). The only peripheral available will be a ground terminal to monitor FEDS status during testing.
- All necessary system software (i.e., the device handlers) in both the development and target computers will be available.
- Since there will be no data storage peripherals in the target system, all data must be managed in RAM. In addition, overlaying of tasks is impossible.
- A communications link with the transponder will provide time-tagged Doppler measurements and control Doppler compensation, indicate when a measurement is available, and control the Doppler accumulator.

E.3 FUNCTIONAL REQUIREMENTS DEFINITION

This section specifies the FEDS functional requirements, i.e., the functions that the system must perform to satisfy the system requirements and the performance requirements.

E.3.1 FUNCTIONAL REQUIREMENTS

The FEDS functional requirements specified in this section are presented according to functional areas, as follows:

- System control (Section E.3.1.1)
- Input processing (Section E.3.1.2)
- Data preprocessing (Section E.3.1.3)
- Data management (Section E.3.1.4)
- Estimation (Section E.3.1.5)
- One-way Doppler prediction (Section E.3.1.6)
- Output processing (Section E.3.1.7)

These functional requirements are the most detailed requirements presented. No attempt is made to define computational models or algorithms here, except where the requirements are specifically affected.

The functional requirements specified in Sections E.3.1.1 through E.3.1.7 are numbered for convenience. In the numbering system used, R indicates requirements.

E.3.1.1 System Control Functional Requirements

The functional requirements for system control are as follows:

R1.1 FEDS will maintain an activity log containing the following: system events, information messages, error messages, directives, and control commands.

R1.2 FEDS will service each control command immediately upon reception.

R1.3 FEDS will schedule maneuver recovery according to clock time and the maneuver schedule.

R1.4 FEDS maneuver recovery will consist of the following:

R1.4.1 TDRS maneuver. The predicted state after the maneuver will be given to the data preprocessor to be used for future generation of the TDRS orbit file.

R1.4.2 User spacecraft maneuver. The TDRS orbit files and the observations file will be purged. The startup procedure will be performed; estimation will be resumed only when a complete estimation span of data has been received.

R1.5 FEDS will schedule one-way Doppler prediction a user-specified number of minutes before the start time of each tracking interval in the tracking schedule.

R1.6 FEDS will schedule the output of data and messages.

R1.6.1 FEDS will schedule the output of severe errors from which the system cannot recover.

R1.6.2 FEDS will schedule the output of priority messages.

R1.6.3 FEDS will schedule the output of the activity log at a specified interval.

R1.6.4 FEDS will schedule the output of the activity log when specifically requested through a control command.

R1.6.5 FEDS will schedule the output of the predicted Doppler frequency shift at least 1 minute before the time tag of the first observation.

R1.7 FEDS will schedule the generation and output of the state vector predict table at the end of the specified interval after the last time of output.

R1.8 FEDS will schedule the generation and output of the state vector predict table immediately after a new solution is obtained.

R1.9 FEDS will schedule input processing when the input queue is full or when the input queue contains data and the system is otherwise idle.

R1.10 FEDS will schedule data preprocessing when a complete pass of data has been processed through input and estimation on the previous batch has been completed.

R1.11 FEDS will schedule data preprocessing when a TDRS maneuver occurs or when a new TDRS vector has been received.

R1.12 FEDS will schedule estimation when a new pass of data has been added to the observations data set.

R1.13 FEDS will notify ground control when it has an excessive amount of idle time for fast timing.

E.3.1.2 Input Processing Functional Requirements

The functional requirements for input processing are as follows:

R2.1 FEDS will capture all incoming messages upon demand.

R2.2 FEDS will accept, as input, messages containing data and control commands.

R2.3 FEDS will process the following types of input data: Doppler measurements, transponder control flags, new TDRS vectors, maneuver schedule, tracking schedule, initialization table, estimation control parameters, and constants (i.e., miscellaneous

constants, station constants, geopotential tables, atmospheric density tables, and timing coefficients).

R2.4 FEDS will accept the following control commands: REBOOT, ABORT, STOP, START, SUSPEND, CONTINUE, STATUS REQUEST, SET CLOCK, MARK TIME, RESUME PROCESSING, BEGIN FAST TIMING, and STOP FAST TIMING.

R2.5 Deleted.

E.3.1.3 Data Preprocessing Functional Requirements

The functional requirements for data preprocessing are as follows:

R3.1 FEDS will accept only those Doppler observation measurements that are in ascending time order and have a reasonable value.

R3.2 FEDS will convert the Doppler observation measurements and time tag to the correct engineering units.

R3.3 No smoothing of the raw observation data will be performed.

R3.4 FEDS will pregenerate TDRS orbit files from the uplinked TDRS vectors (one file for each TDRS). These files will cover the same timespan as the observations file; they will be used iteratively by the batch estimator.

R3.5 FEDS will update the TDRS orbit files when a new TDRS vector is received.

R3.6 After a TDRS maneuver, FEDS will use the predicted state vector as the base vector for generating the TDRS orbit files in the future.

R3.7 After receiving an update to a TDRS maneuver, FEDS will update the appropriate TDRS orbit file from

the maneuver time to the current processing time by propagating the input TDRS vector.

E.3.1.4 Data Management Functional Requirements

The functional requirements for data management are as follows:

R4.1 FEDS will manage all data files in memory, since no data storage peripherals will be provided.

R4.2 FEDS will have the capability to locate, read, and write observation records in the observations file.

R4.3 FEDS will have the capability to locate, read, write, and update the records of the TDRS orbit files.

R4.4 FEDS will have the capability to purge all data files.

E.3.1.5 Estimation Functional Requirements

The functional requirements for estimation are as follows:

R5.1 FEDS will perform differential correction on the most recent fixed-length minimum data span (specified through control parameters) of observation data. The observations data used will be whole passes except when data wraparound occurs.

R5.2 The method of estimation will be batch least-squares.

R5.3 Due to the real-time processing of FEDS, the estimation timespan will be slid forward to encompass each new pass of observations data. This will be referred to as a "sliding batch estimator."

R5.4 During initialization of the estimation process (defined as operations included in estimation using

a particular batch of data), the following will be performed:

R5.4.1 FEDS will initialize the estimation parameters from the initialization table and/or the estimation control parameters if either was received since the beginning of the previous estimation process.

R5.4.2 FEDS will set up the new estimation span.

R5.5 Initialization of the estimation parameters will be performed after estimation has been suspended through a control command.

R5.6 FEDS will model one-way TDRSS Doppler observations.

R5.6.1 Deleted.

R5.6.2 Unless directed otherwise, the measurement partials will be computed only on the first iteration. The linearity test described in R5.8.3 will determine whether or not recomputation is necessary.

R5.7 FEDS will perform an edit loop during the first (or, on demand, subsequent) iteration of each DC slide based on the predicted residuals and estimation statistics (specified through control parameters).

R5.7.1 The computed measurements and associated partials will remain unchanged during this process.

R5.7.2 The edit loop will terminate upon either the maximum number of loops this iteration (maximum = 10) or no observations were edited during the predicted residual versus sigma test (input parameter).

R5.8 FEDS will test for DC slide convergence, divergence, and linearity violation at the end of each iteration.

R5.8.1 FEDS will declare a new state solution at the point of convergence. Convergence is defined in Reference 5, "FEDS Estimation Logic," memorandum Sections II.A.11(a) and II.A.12(a).

R5.8.2 FEDS will remain in the propagate mode if divergence occurs. Divergence is defined in Reference 5, Sections II.A.11(b) and II.A.12(a).

R5.8.3 FEDS will perform another iteration if neither convergence nor divergence has occurred. The linearity test defined in Reference 5, Section II.A.12(b) will be performed to determine whether recomputation of partial derivatives and another edit loop will be done on the next iteration.

R5.8.4 FEDS will declare the current iteration as the last iteration of this DC slide if either convergence or divergence occurs.

R5.9 FEDS will be capable of generating a DC summary and statistics report. This report, if generated, will be generated and output either (1) after every iteration or (2) after the last iteration on each batch.

R5.10 FEDS will be capable of generating a DC residuals report. If generated, this report will be output either after the first and last edit loops of each

iteration or after the last iteration of each DC slide.

R5.11 If time allows, FEDS will precompute values needed for the next DC slide prior to the actual receipt of the next data pass. This will be done for all slides except the initial slide.

R5.11.1 The new epoch will be predetermined as the current epoch plus a fixed lead time (input parameter).

R5.11.2 Measurement residuals and partial derivatives will be computed over all observations in the previous slide.

E.3.1.6 <u>One-Way Doppler Prediction Functional Requirements</u>

The functional requirements for one-way Doppler prediction are as follows:

R6.1 FEDS will predict (simulate) one-way Doppler frequency shift over the timespans indicated by the uplinked tracking schedule.

R6.2 FEDS will use the TDRS, whose ID will be specified with each tracking interval, to predict the one-way Doppler frequency shift.

R6.3 No observation feasibility checking will be performed, since the tracking schedule will contain valid intervals for the specified TDRS.

R6.4 The target (user spacecraft) state vector used in one-way Doppler prediction will be based on the most recent state solution. When a user spacecraft maneuver has occurred or a new initialization table has been received, the most recent solution will be overridden by the new a priori estimate.

R6.5 The TDRS state vector used in one-way Doppler prediction will be based on the TDRS vector used to generate the TDRS orbit file.

E.3.1.7 Output Processing Functional Requirements

The functional requirements for output processing are as follows:

R7.1 FEDS will generate and output the state vector predict table. This table will be based on the most recent state solution. When a user spacecraft maneuver has occurred or a new initialization table has been received, the most recent solution will be overridden by the new a priori state vector.

R7.2 FEDS will output priority messages directly to the ground control (ADEPT).

R7.3 FEDS will output the activity log to ADEPT.

R7.4 FEDS will output the predicted Doppler shift to the communications link with the transponder at a specified interval.

R7.5 FEDS will output to ADEPT the DC residuals reports as they are generated by the estimator.

R7.6 FEDS will output to ADEPT the DC summary and statistics reports as they are generated by the estimator.

R7.7 FEDS will output a table of the predicted Doppler data for each tracking interval to ADEPT.

REFERENCES

1. Computer Sciences Corporation, Requirements Analysis for Automated Orbit Determination System (AODS) System Modification to Support the Ground Demonstration (memorandum), M. Regardie and D. Shank, November 1983

2. --, CSC/TM-84/6083, Flight Experiment Demonstration System (FEDS) Mathematical Specification, D. Shank, July 1984

3. --, CSC/SD-82/6068, Automated Orbit Determination System (AODS) Environment Simulator for Prototype Testing (ADEPT) User's Guide, S. Waligora, J. Fry, Jr., and Y. Ong, June 1982

4. --, CSC/SD-82-6054, Automated Orbit Determination System (AODS) Environment Simulator for Prototype Testing (ADEPT) System Description, S. Waligora, J. Fry, Jr., Y. Ong, B. Prusiewicz, and G. Klitch, June 1982

5. National Aeronautic and Space Administration, Goddard Space Flight Center, Recommended Estimation Logic for AODS (memorandum), J. Teles, January 1981 (also published as Appendix E of Systems Technology Laboratory (STL) document STL-80-003)