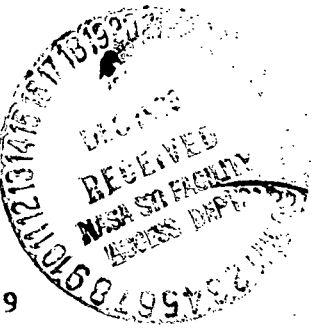
77-FM-18
Vol. III, Rev. 1

JSC-12564

Flight Design System-1
System Design Document



Processor Library
Book 3

Mission Planning and Analysis Division

October 1979

National Aeronautics and
Space Administration

Lyndon B. Johnson Space Center
Houston, Texas

77-FM-18
Vol. III, Rev. 1

JSC-12564

SHUTTLE PROGRAM

FLIGHT DESIGN SYSTEM-1
SYSTEM DESIGN

PROCESSOR LIBRARY
BOOK 3

By Software Development Branch

Approved: _____
Elric N. McHenry, Chief
Software Development Branch

Approved: _____
Ronald L. Berry, Chief
Mission Planning and Analysis Division

Mission Planning and Analysis Division

National Aeronautics and Space Administration

Lyndon B. Johnson Space Center

Houston, Texas

October 1979

PREFACE

The Flight Design System-1 (FDS-1) is a pilot project to evaluate current concepts and to determine the hardware/software capability that will be required for the operational era to support Shuttle flight planning. This software system is being implemented on a Hewlett-Packard 21MX computer with a Daconics documentation system and will provide terminal-based interactive flight planning capability.

The System Design Document (SDD) for FDS-1 is the specification for and description of this hardware/software facility. The SDD is logically organized into 10 published volumes. This organization is presented in the accompanying table. The material in the early volumes is primarily presented from the user's point of view, whereas the latter material is software-developer oriented. The SDD will be published by volumes over a period of time, and various volumes will be updated and republished during the development of FDS-1.

FDS-1 SYSTEM DESIGN DOCUMENT

Volume I	Introduction, Overview, and User Interface
*Volume II	Utility Processor Library
*Volume III	Application Processor Library
Volume IV	System Architecture and Executive
Volume V	Documentation Support
Volume VI	Standards
Volume VII	Utility Support Software
Volume VIII	Build and Delivery Procedures; Software Development, Debug, and System Build Aids
Volume IX	Executive Logic Flow - Program Design Language
Volume X	Document Change Request Procedure and Submittal Form

*Combined as one volume with title: Volume III FDS-1 Processor Library

ACKNOWLEDGMENT

This document was written and prepared by a team consisting of Johnson Space Center civil service and contractor personnel. The following organizations made significant contributions to this document:

International Business Machines, Inc.
Federal Systems Division
Mission Analysis and Engineering

Lockheed Electronics Company, Inc.
Systems and Services Division

National Aeronautics and Space Administration/Johnson Space Center
Data Systems and Analysis Directorate
Mission Planning and Analysis Division
Flight Planning Branch
Software Development Branch

The Mission Planning and Analysis Division (MPAD) directed the effort and, in addition, developed software and submitted completed draft writeups of some sections. MPAD wishes to acknowledge the excellent support received from all organizations involved in preparing this document.

CONTENTS

Sections Page

Section Page

PHASE TABLE PRINT PROCESSOR (PTP)

TABLES

Table Page

Table — Page

LAUNCH WINDOW PROCESSOR (LWP)

Table | Page

FIGURES

Figure Page

Book 3

1.0 INTRODUCTION

This volume presents the complete program documentation for the processors in the Flight Design System-1 (FDS-1). Two types of processors exist in the processor library; utility processors and application processors. Utility processors provide a general capability that is not particularly associated with flight design, such as Data Box Display (DBDSP), Data Element Definition (DEFIN), or Conditional Execution Processors (IF, ELSE, ENDIF). Application processors provide capability that is directly related to accomplishing flight design, such as Ascent (ASENT), General Purpose Maneuver Processor (GPMP), Groundtrack (GTRAK), or Mission Plan Table Processor (MPTP).

Both types of processors are presented in this volume in alphabetical order; thus, volume II of the SDD documentation is contained in this volume and will not exist separately. The only documentation that exists in addition to this volume is the software listings and their comment cards.

Because of the magnitude of this volume, it is published in three books.

2.0 PROCESSOR LIBRARY

This volume contains the complete software documentation for all FDS-1 processors. The processors are paged alphabetically according to the processor names as shown below.

Book 1

Ascent Processor	ASENT
Data Assignment Processor	ASSGN
Attitude Tables Maintenance Processor	ATM
Basetime Initialization Processor	BASTM
Consumable Analysis for Shuttle Kits Processor	CASKU
Coasting State Vector Predictor (including AEG) Processor	COAST
Data Box Display Processor	DBDSP
Data Box Extractor Processor	DBEXT
Data Box Interpolator Processor	DBINT
Data Element Definition Processor	DEFIN
Sequence Iteration Processors (DO, ENDDO)	DO
Document Processor	DOC
Deorbit Target Module Processor	DTM

Book 2

Early Repeating Groundtrack Orbits Processor	ERGO
* Finite Burn Processor	FINBN
* Fixed Magnitude Two-Burn Processor	FM2BN
Flight Plan Display Processor	FPD
General Purpose Maneuver Processor	GPMP
Groundtrack Processor	GTRAK
Conditional Execution Processors (IF, ELSE, ENDIF)	IF

*To be supplied.

Invariant Element Ephemeris Processor	INVAR
CASKU and QUIKU Output Data Display Processor	LKOUT
Landing Opportunities Processor	LOPT
Load State Vector Processor	LSV
Launch Window Processor	LWP
* Maneuver Iterator Processor	MANIT
Matrix and Attitude Support Table Processor	MAST
Master Data Temporary Print Processor	MDTP
Mission Plan Table Processor	MPTP
* Node Definer Processor	NODE
Orbital Maneuver Processor (Rendezvous)	OMP
Book 3	
Input/Output Units Specification Processor	PHYDM
* Placement Longitude Processor	PLLON
Print State Vector Processor	PSV
Phase Table Print Processor	PTP
Quick Investigation of Consumables Kits Processor	QUIKU
Parametric Scan Processors (SCAN, ENDSC)	SCAN
Sunrise/Sunset Time Predictor Processor	SRSS
* Shuttle/SUS Burn Relative Motion Processor	SSBRM
Sun Synchronous Orbits Processor	SSYN
Station Contacts Processor	STACN
Summary Table Print Processor	STP
State Vector Units Conversion Processor	SVUCP

*To be supplied.

* State Vector Coordinate Transformation Processor	TFSV
Activity Timeline Processor	TMLNU

*To be supplied.

INPUT/OUTPUT UNITS SPECIFICATION PROCESSOR (PHYDM)

1.0 PURPOSE

The PHYDM utility processor provides the FDS user with the optional capability of specifying a desired set of input/output (I/O) units to be used during a session. The default set of I/O units is as defined in section 7.0 in JSC IN 78-FM-60, volume I.

Angles ---------- Degrees

Distance ---------- Feet

Time ---------- Seconds

Velocity ---------- Feet/second

Mass ---------- Pounds mass

Force ---------- Pounds force

Length ---------- Feet

If the PHYDM processor is not executed, then this default set of I/O units will be in effect for the session. However, by using the PHYDM utility processor, the FDS user can change any or all of the I/O units for his FDS session. The unit options available to the user are defined in table 7.1-I in JSC IN 78-FM-60, volume I.

2.0 FUNCTIONAL DESCRIPTION

All user communication with the PHYDM processor is through its interface table parameters GLOCON, ANGUN, DSTUN, TIMUN, VELUN, MASUN, FORUN, LENUN, PRINT, and SESCON. GLOCON is the input parameter through which the processor receives all of the global constants data that it requires. This input is defaulted to the master data base element !!GLCN and under normal circumstances, the user need not be concerned with it. (For documentation of the contents of !!GLCN, see table 7.2-III in JSC IN 78-FM-60, vol. I.)

Through the input parameters ANGUN, DSTUN, TIMUN, VELUN, MASUN, FORUN, and LENUN, the user specifies (by mnemonic names) the I/O units desired for angles, distance, time, velocity, mass, force, and length respectively. The PHYDM processor uses this input to set the appropriate conversion factors and mnemonics in the output array SESCON. (Specifically PHYDM updates locations 2 and 13 through 40 of the SESCON array.) Note that units options for the dimension of length are provided in addition to those for the dimension of distance. The reason for this duplicity is to permit the FDS user to work in a convenient set of units when dealing with Shuttle and payload physical dimensions (such as length, area, volume, c.g. location, etc.) while at the same time, using an equally convenient set of units for distances (such as vehicle position, altitudes, etc).

The parameter PRINT is an input through which the user directs the PHYDM processor to print or not print an output display. The format for the optional output display is shown under the paragraph titled "Processor displays and display parameter definition tables."

The parameter SESCON is an array that is both input and output for the PHYDM processor. In the interface table, it is normally defaulted to the specially named AWA data element !SESCN. The name !SESCN is special because it is referenced as an input source in the default interface tables for most FDS functional processors, and under normal circumstances, the user need not be concerned with this data linkage and should not try to change it. (For a definition of the contents of the !SESCN array see table 7.2-II(a) in JSC IN 78-FM-60, vol. I.)

3.0 ASSUMPTIONS AND LIMITATIONS

a. The PHYDM processor uses the specially named AWA data element !SESCN as both an input and output array; therefore, this array must exist in the AWA prior to executing PHYDM. The array !SESCN is an output generated by the BASTM processor.

b. Where Hollerith data (i.e., character strings) are used as input, the only syntax checking that is done by the interface table editor is to ensure that the character strings are of the correct length. Therefore, if the user misspells one or more of the mnemonic inputs (or specifies a mnemonic for a nonexistent option), these errors will not be discovered until processor execution time. An error, thus found, will result in a message being issued by the processor, execution of the processor (and sequence table, if any) is terminated, and control is returned to the Executive.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The definition of the default interface table for the PHYDM processor is provided in table 4-I.

b. Interface table data array definitions - A definition of the input/output data arrays appearing in the PHYDM processor interface table is provided in table 4-II.

c. Interface table data file definitions - None.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definition tables - Only one display is generated by the PHYDM processor, the optional display that is controlled by the input parameter PRINT. The format of this display is shown in table 4-III, and a definition of the display parameters is provided in table 4-IV.

f. Processor message table - If the user misspells one or more of the units specification mnemonics or otherwise specifies an invalid mnemonic for any

dimension, the PHYDM processor will print an error message(s) on the user's terminal. The error message(s) will identify the mnemonics that are in error. The format and content of the message is given in table 4-V.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VI.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR PHYDM

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
GLOCON	AWA	Free	I	180	180	!!GLCN	Global constants array, master data base element
ANGUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for angles # = "DEG", degrees = "RAD", radius
DSTUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for distance # = "FT", feet = "M", meters = "NM", nautical miles = "ER", Earth radii = "KM", kilometers
TIMUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for time # = "SEC", seconds = "MIN", minutes = "HR", hours = "DAYS", days
VELUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for velocity # = "FT/S", feet/second = "M/S", meters/second = "NM/H", nautical miles/hour = "ER/H", Earth radii/hour = "KM/H", kilometers/hour

NOTES	CLASS	TYPE			USE	
		Free	2CH	72CH	I = Input	# - Default external unit
	AWA	Intg	6CH	Mix	O = Output	
	Disk	Real	18CH	Symb	I/O = Input/Output	
		Dubl	36CH			

TABLE 4-I.- Continued

PROCESSOR PHYDM

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
MASUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for mass # = "LBM", pounds = "KG", kilograms = "SLG", slugs
FORUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for force # = "LBF", pounds = "NWT", newtons = "PDL", poundal
LENUN	AWA	6CH	I	3	1		Mnemonic specifying the I/O units for length # = "FTL", feet = "ML", meters = "INL", inches
PRINT	AWA	2CH	I	1	1		Display print flag # = "YE", print I/O units display = "NO", don't print I/O units display
SESCON	AWA	Free	I/O	90	90	!SESCN	Session constants array

NOTES	CLASS	TYPE			USE	
		Free	2CH	72CH	I = Input	# = Default external unit
	AWA	Intg	6CH	Mix	O = Output	
	Disk	Real	18CH	Symb	I/O = Input/Output	
		Dubl	36CH			

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR PHYDM

Array name	Index location	Default value	Definition
GLOCON	(1) . . . (180)	!!GLCN	Global constants array; master data base element; see table 7.2-III in JSC IN 78-FM-60, volume I for definition of contents
SESCON	(1) . . . (90)	!SESCN	Input/output session constants array; see table 7.2-II(a) in JSC IN 78-FM-60, volume I for definition of contents. PHYDM updates location 2 and 13 through 40 of this array

TABLE 4-III.- PROCESSOR DISPLAY TABLE

PROCESSOR PHYDM

```
PHYDM HAS ESTABLISHED THE FOLLOWING
MEASUREMENT UNITS FOR FDS INPUT/OUTPUT DATA:

DIMENSION        UNITS       CONVERSION FACTOR
---------        -----       -----------------

ANGELS            XXXX          X.XXXXXXXE±XX
DISTANCE          XXXX          X.XXXXXXXE±XX
TIME              XXXX          X.XXXXXXXE±XX
VELOCITY          XXXX          X.XXXXXXXE±XX
MASS              XXXX          X.XXXXXXXE±XX
FORCE             XXXX          X.XXXXXXXE±XX
LENGTH            XXXX          X.XXXXXXXE±XX
```

TABLE 4-IV.- DISPLAY PARAMETER DEFINITION TABLE

PROCESSOR PHYDM

PHYDM PROCESSOR I/O UNITS DISPLAY	
Display parameter label	Parameter definition
UNITS	Up to four-character mnemonic that identifies the user-selected unit for the associated dimension (e.g., DEG for degree); these are the same mnemonics that PHYDM stores in the appropriate locations (13 through 26) of the I/O array SESCON.
CONVERSION FACTOR	Seven-digit conversion factor for the associated dimension; it is the numeric constant which, when multiplied times an input parameter (of the associated dimension), converts that parameter from the user-defined external units to the standard internal units for that dimension. These are the same conversion factors that PHYDM stores in the appropriate locations (27 through 40) of the I/O array SESCON.

TABLE 4-V.- PROCESSOR MESSAGE TABLE

PROCESSOR PHYDM

MSG no.	Message ID block	Message text block and explanation
1	*PHYDM*	INVALID UNITS MNEMONIC(S): XXXXX = "XXXX". The line is issued for every invalid units mnemonic. Meaning: The user either misspelled or otherwise input incorrect units mnemonics. The dimension and erroneous units mnemonics are displayed as: XXXXX = "XXXX". Severity: The processor will terminate without making any change to the !SESCN array. Control will be returned to the FDS Executive, and sequence table processing (if any) will be terminated. Action required by user: Use the Interface Table Editor to correct the erroneous mnemonic(s); then resume execution.

TABLE 4-VI.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR PHYDM

Processor name	Processor abstract prompt (maximum 256 characters)
PHYDM	PHYDM is a utility processor that provides the capability to specify a set of input/output units for use during a user's simulation.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
GLOCON	Global constants array, master data base element normally defaulted to !!GLCN
ANGUN	Mnemonic specifying the input/output units for angles: "DEG" - Degrees "RAD" - Radians
DSTUN	Mnemonic specifying the input/output units for distance: "FT" - Feet "M" - Meters "NM" - Nautical miles "ER" - Earth radii "KM" - Kilometers
TIMUN	Mnemonic specifying the input/output units for time: "SEC" - Seconds "MIN" - Minutes "HR" - Hours "DAY" - Days
VELUN	Mnemonic specifying the input/output units for velocity: "FT/S" - Feet/second "M/S" - Meters/second "NM/H" - Nautical miles/hours "ER/H" - Earth radii/hour "KM/H" - Kilometers/hour

TABLE 4-VI.- Concluded

PROCESSOR PHYDM

Processor name	Processor abstract prompt (maximum 256 characters)
PHYDM (continued)	

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
MASUN	Mnemonic specifying the input/output units for mass: "LBM" - Pounds "KG" - Kilograms "SLG" - Slugs
FORUN	Mnemonic specifying the input/output units for force: "LBF" - Pounds "NWT" - Newtons "PDL" - Poundal
LENUN	Mnemonic specifying the input/output units for length: "FTL" - Feet "ML" - Meters "INL" - Inches
PRINT	Display print flag: "YE" - Print input/output units display "NO" - Don't print input/output units display
SESCON	Session constants array; normally defaulted to !SESCN.

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM PHYDM

5.1.1 Purpose

The PHYDM utility processor provides the FDS user with the optional capability of specifying a desired set of input/output units to be used during his session. The default set of input/output units is as defined in section 1.2 (ref. 1).

Angles ---------- Degrees

Distance ---------- Feet

Time ---------- Seconds

Velocity ---------- Feet/second

Mass ---------- Pounds mass

Force ---------- Pounds force

Length ---------- Feet

If the PHYDM processor is not executed, this default set of input/output units will be in effect for the session. However, by using the PHYDM utility processor, the FDS user can change any or all of the input/output units for his FDS session. The unit options available to the user are defined in table 1.2-I (ref. 1).

5.1.2 Functional Description

Initially, the processor obtains default parameters from the master data base element !!GLCN via a call to XPGET routine, which stores them in the interface GLOCON(180) internal array, and prompts for the interface parameters ANGUN, DSTUN, TIMUN, VELUN, MASUN, FORUN, LENUN, and PRINT option. The user enters the input/output mnemonics unit desired for angles, distance, time, velocity, mass, force, and length. (See table 1.2-I of reference 1.) The user also enters the print option.

When all prompts have been answered, the processor tests each input for valid input mnemonics and, if valid, computes the session word and stores it, as well as the conversion factors mnemonics and value in the input/output array SESCON (positions 2 and 13-40).

If any of the prompt interface parameters are found to be invalid, an error message is issued by the processor, execution of the processor (and sequence table, if any) is terminated, and control is returned to the Executive with the error flag set. If there were no errors, the array SESCON is transmitted

to the user AWA via a call to XPPUT routine to be used during the user session. Execution of the processor is terminated, and normal control is returned to the Executive.

5.1.3 Assumptions and Limitations

a. The PHYDM processor uses the specially named AWA data element !SESCN as both an input and output array; therefore, this array must exist in the AWA prior to executing PHYDM. The array !SESCN is an output generated by the BASTM processor.

b. Where Hollerith data (i.e., character strings) are used as input, the only syntax checking that is done by the interface table editor is to ensure that the character strings are of the correct length. Therefore, if the user misspells one or more of the mnemonic inputs (or specifies a mnemonic for a nonexistent option) these errors will not be discovered until processor execution time. An error thus found will result in a message being issued by the processor, termination of processor execution (and sequence table, if any), and return of control to the Executive.

5.1.4 Method

The PHYDM utility processor checks user input parameter mnemonic names for ANGUN, DSTUN, TIMUN, VELUN, MASUN, FORUN, and LENUN, and sets appropriate conversion factors and the input mnemonics in the input/output array SESCON. For every valid input mnemonic the processor updates a session word and, if all input mnemonics are valid, stores it in SESCON(2) location.

Using figure 1.2-I (ref. 1), the session word is computed and stored in SESCON(2) as follows:

$$SESCON(2) = SESCON(2) + MUPi * 2^{**}IPPi$$

$$i = 1,8$$

J = MUP - Mnemonics units positions

IPP - Interface parameter positions that are constants for

ANGUN = 0	MASUN = 9
DSTUN = 1	FORUN = 11
TIMUN = 4	LENUN = 13
VELUN = 6	

5.1.5 Routine Input/Output Variables

The PHYDM input/output variables are presented in table 5.1-I.

5.1.6 Functional Logic Flow

The functional logic flow for PHYDM is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

None.

5.1.8 Special Comments

None.

5.1.9 Reference

1. Flight Design System-1, System Design Document, Standards. Volume VI, Revision 1, JSC IN 77-FM-18, January 1978.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine PHYDM

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ANGUN		6CH	I		IT	ANGUN	Mnemonic specifying the input/output units for angles.
DSTUN		6CH	I		IT	DSTUN	Mnemonic specifying the input/output units for distance.
TIMUN		6CH	I		IT	TIMUN	Mnemonic specifying the input/output units for time.
VELUN		6CH	I		IT	VELUN	Mnemonic specifying the input/output units for velocity.
MASUN		6CH	I		IT	MASUN	Mnemonic specifying the input/output units for mass.
FORUN		6CH	I		IT	FORUN	Mnemonic specifying the input/output units for force.
LENUN		6CH	I		IT	LENUN	Mnemonic specifying the input/output units for length.
PRINT		2CH	I		IT	PRINT	Display print flag.
SESCON		Free	I/O		IT	SESCON	Session constants array.
GLOCON		Free	I		IT	GLOCON	Global constants array master data base element.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

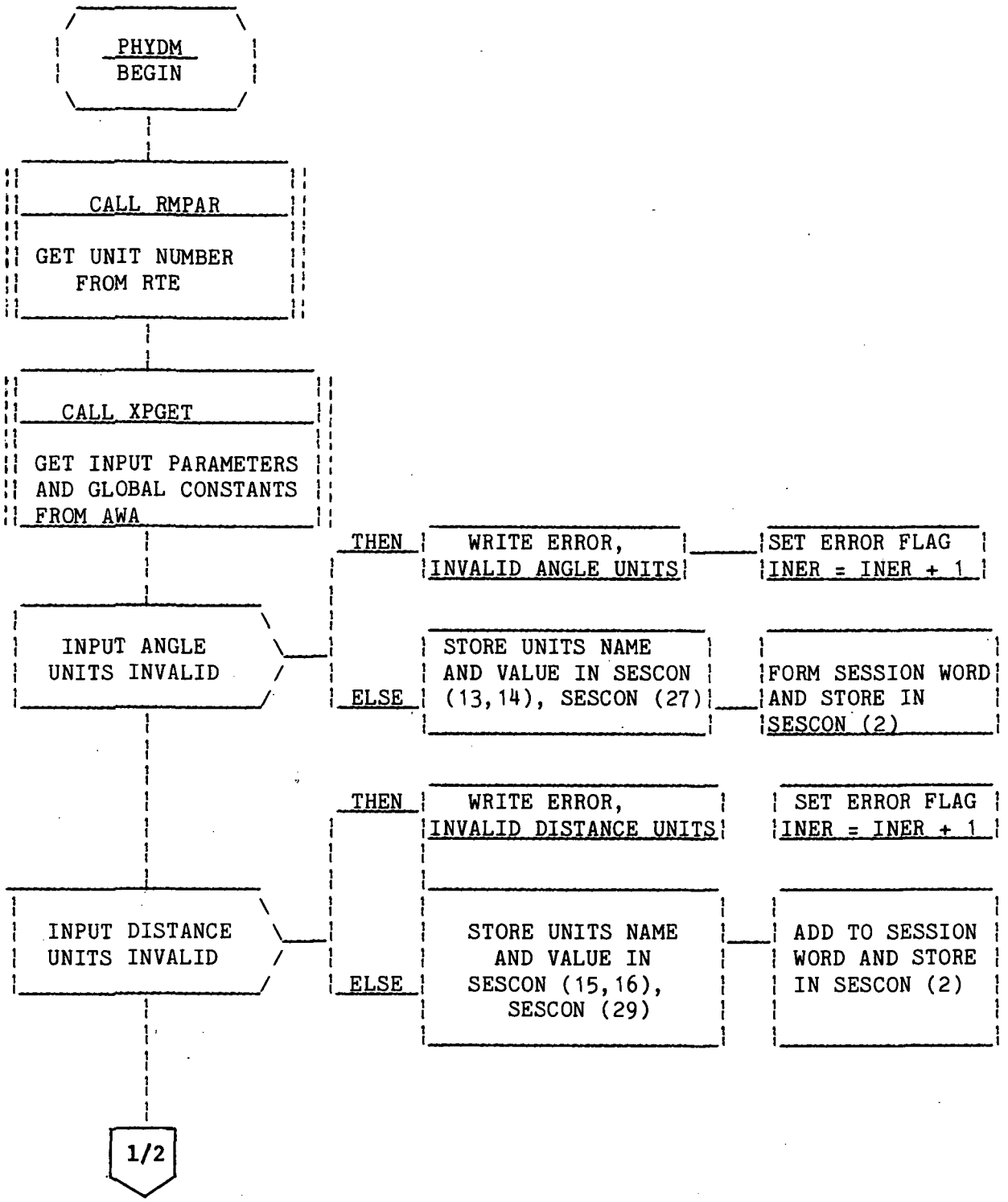


Figure 5.1-1.- PHYDM functional logic flow.

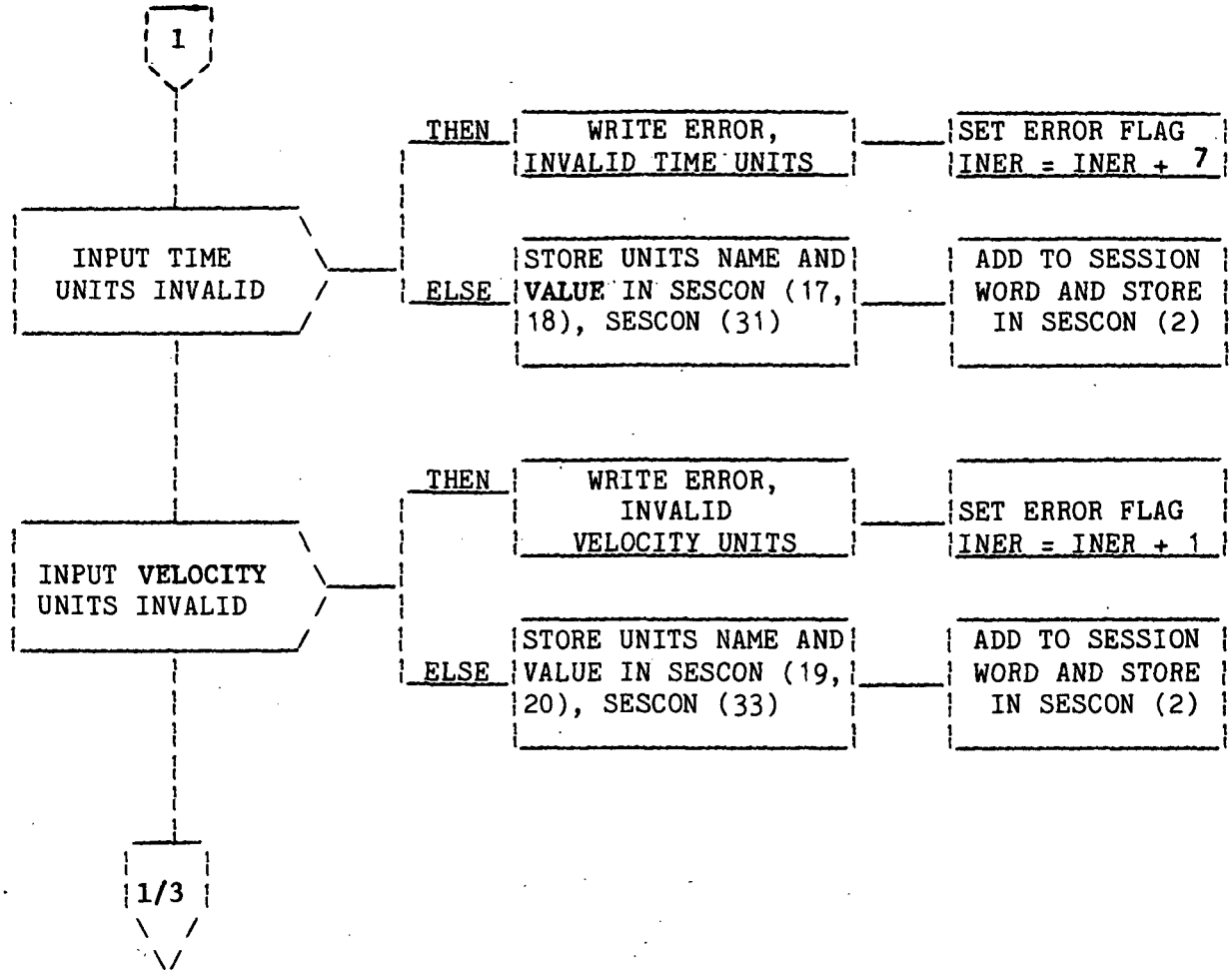


Figure 5.1-1.- Continued.

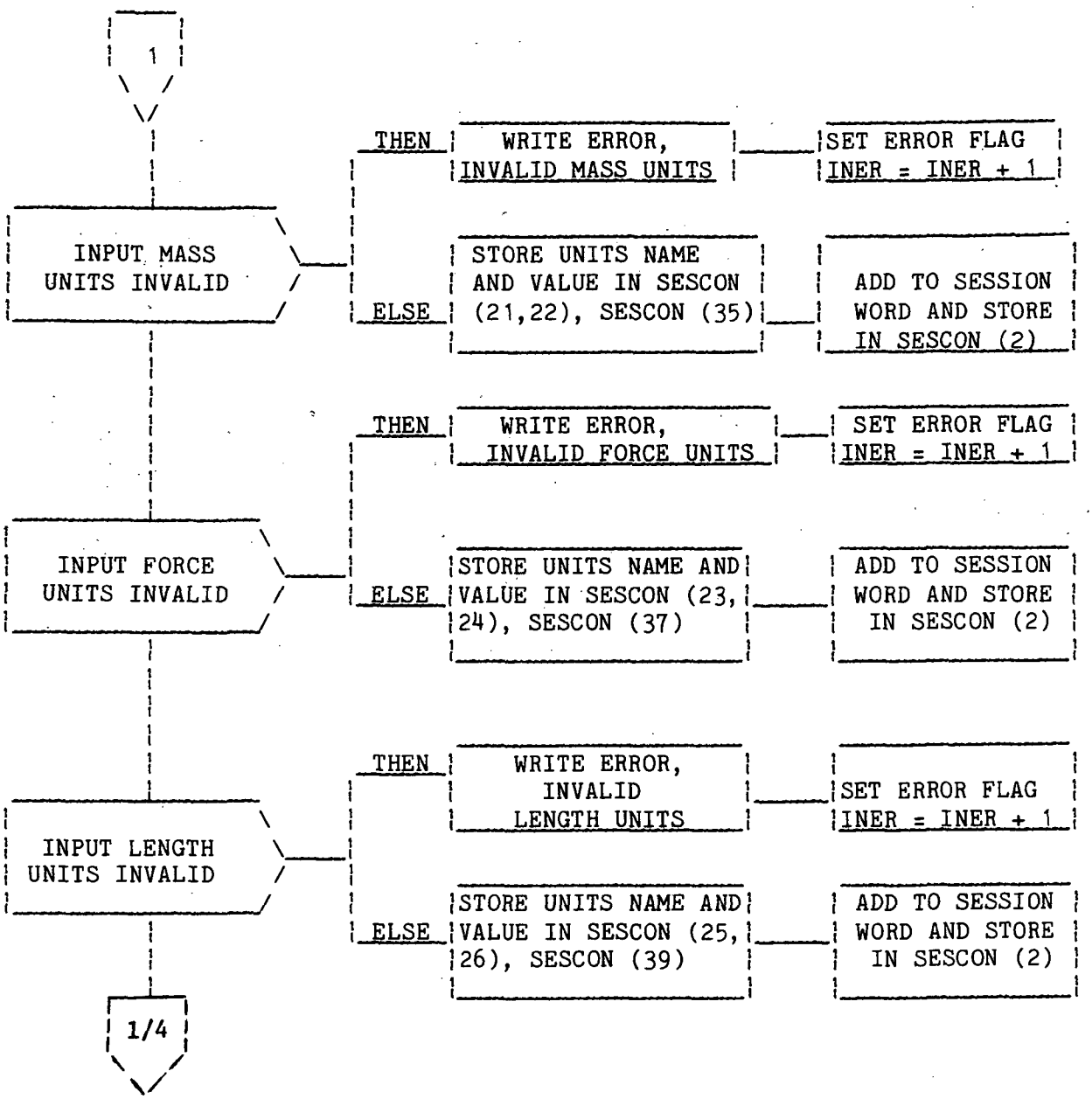


Figure 5.1-1.- Continued.

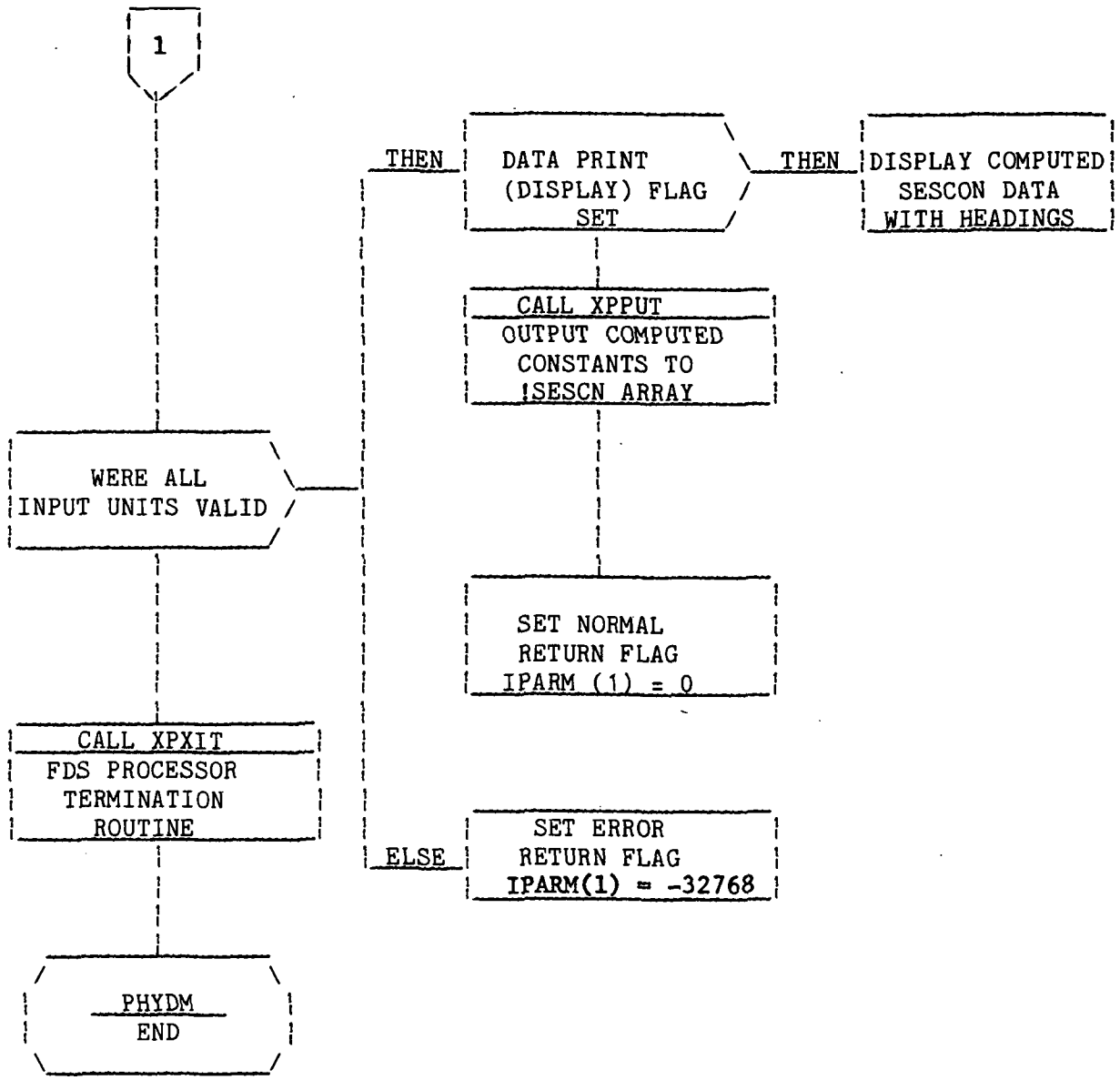


Figure 5.1-1.- Concluded.

PLACEMENT LONGITUDE PROCESSOR (PLLON)

TO BE SUPPLIED

PRINT STATE VECTOR PROCESSOR (PSV)

1.0 PURPOSE

The PSV utility processor provides the FDS user with the capability to print the contents of a position/velocity state vector data element.

2.0 FUNCTIONAL DESCRIPTION

All input to the PSV processor is through its interface table. The interface table for the PSV processor contains two parameters, SV and VNAM. SV is the parameter through which the processor receives all of the position/velocity state vector data and the information defining the reference axis and the element set. The 15 elements contain a position/velocity state in the format defined in JSC IN 78-FM-60, volume I, figure 7.3-2. The fourteenth element of the state vector contains coded information that the processor decodes to obtain the reference axis code, the element set type, and the element set code. Using the decoded values, the processor displays the position/velocity data with appropriate annotations. VNAM is an input parameter through which the user supplies a Hollerith label of up to 18 characters; this label will be displayed as the name of the state vector. In the default interface table VNAM is defaulted to blanks.

3.0 ASSUMPTIONS AND LIMITATIONS

a. No checking is performed on the state vector values.

b. The display is limited to the FDS-supported state vector coordinate sets defined in JSC IN 78-FM-60, volume I, table 7.3-VI.

c. The reference axis code, the element set type, and the element set code must represent FDS-supported coordinate sets. Validity checks are made on these values.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The definition of the processor interface table for the PSV processor is provided in table 4-I.

b. Interface table data array definitions - A definition of the input data array appearing in the PSV interface table is provided in table 4-II.

c. Interface table data file definitions - None.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definition tables - When the processor executes correctly, it will generate an output display on the user's terminal. The general format and content of the display are provided

in table 4-III, and definitions of all possible display parameters are provided in table 4-IV.

f. Processor message table - If an out-of-range value is input for one or more of the code parameters-reference axes, element set types, or element sets, the PSV processor will print an error message(s) on the user's terminal. The error message(s) will identify the parameters and value(s) that are out of range. The format and contact of the message is provided in table 4-V.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VI.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR PSV

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
SV	AWA	Real	I	30	15		Position/velocity state vector
VNAM	AWA	18CH	I	9	1	18 blanks	Vector name

NOTES	CLASS	TYPE			USE
	AWA	Free	2CH	72CH	I = Input
	Disk	Intg	6CH	Mix	O = Output
		Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR PSV

Array name	Index location	Default value	Definition
SV	(1) . . (15)		Position/velocity state; refer to JSC IN 78-FM-60, volume I, figure 7.3-2 for definition of contents. PSV uses only position/velocity state.

TABLE 4-III.- PROCESSOR DISPLAY TABLE

PROCESSOR PSV

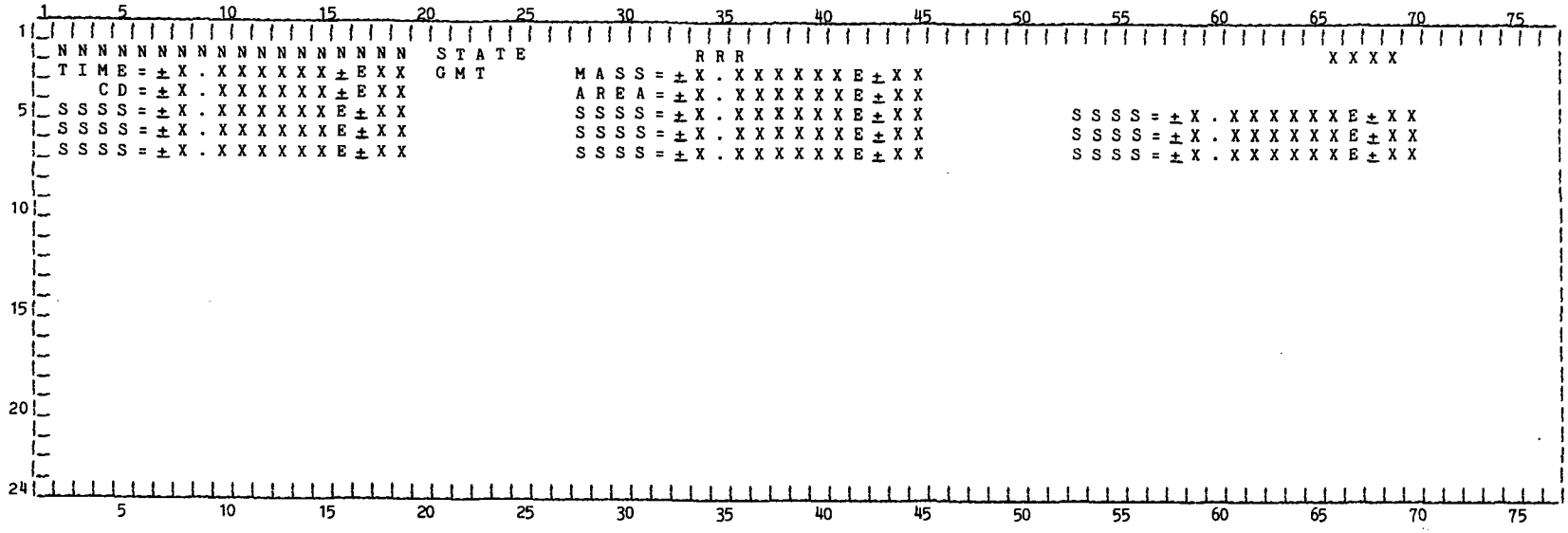


TABLE 4-IV.- DISPLAY PARAMETER DEFINITION TABLE

PROCESSOR PSV

POSITION/VELOCITY STATE VECTOR DISPLAY	
Display parameter label	Parameter definition
N-----N	Vector name (18 characters) defined by VNAM input
RRR	Reference axis (3 characters); see table 7.3-V in JSC IN 78-FM-60, volume I
EEEEE	Element set (6 characters); see table 7.3-VI in JSC IN 78-FM-60, volume I for 6 character codes and associated element type codes
TIME	Time of the state vector relative to a user-established base date
MASS	Vehicle gross weight
CD	Drag coefficient
AREA	Area for drag computation
SSSS	Six or nine elements that specify the position and velocity of the vehicle's center of mass; refer to table 7.3-VI in JSC IN 78-FM-60, volume I for definition of the mnemonic associated with each state vector component. The invariant and KS element types have nine elements; all others have six elements.

TABLE 4-V.- PROCESSOR MESSAGE TABLE

PROCESSOR PSV

MSG no.	Message ID block	Message text block and explanation
1	*SPSV*	UNSUPPORTED REFAX FLAG (JKNN), J = XX Meaning: The portion of the coded value that contains J, the reference axis flag, is not a value that FDS supports. Severity: The position/velocity state vector data will be displayed with *** replacing the reference axis and element set codes and with four blank characters replacing the remaining annotations. The PSV processor will issue an error return condition to the FDS Executive. Action required by user: Determine if the data contained in the interface table array are one of the FDS supported types, and correct the coded word to correspond to the type of data.
2	*SPSV*	UNSUPPORTED ELSET FLAG (JKNN), NN = XX. Meaning: The portion of the coded value that contains NN, the element set flag, is not a value that FDS supports. Severity: The position/velocity state vector data will be displayed with *** replacing the reference axis and element set codes and with four blank characters replacing the remaining annotations. The PSV processor will issue an error return condition to the FDS Executive. Action required by user: Determine if the data contained in the interface table array are one of the FDS supported types, and correct the coded word to correspond to the type data.
3	*SPSV*	UNSUPPORTED COMBINATION, JKNN = ±XXXX. Meaning: The reference axis flag; element set flag combination is not FDS supported. Severity: The position/velocity state vector data will be displayed with *** replacing the reference axis and element set codes and with four blank characters replacing the remaining annotations. The PSV processor will issue an error return condition to the FDS Executive. Action required by user: Determine if the data contained in the interface table array are one of the FDS supported types, and correct the coded word to correspond to the type of data.

TABLE 4-VI.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR PSV

Processor name	Processor abstract prompt (maximum 256 characters)
PSV	The PSV utility processor displays the contents of a position/velocity state vector data element.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
SV	Fifteen-element position/velocity state vector. Refer to standard state vector definition in volume VI, FDS-1 System Design Document, Standards.
VNAME	Hollerith string of up to 18 characters, which will be dislayed as the state vector name.

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM PSV

5.1.1 Purpose

The routine PSV serves as the main program of the PSV processor. It calls the utility subroutine SPSV, which performs the processor function of displaying the state vector.

5.1.2 Functional Description

The routine PSV calls the RTE operating system utility routine RMPAR to get the logical unit of the user terminal and the user qualifier code. The FDS utility routine XPGET is called to get the data contained in the interface table from the active work area (AWA). The data parameters contained in the interface table are SV and VNAM. The definitions of these parameters are given in table 5.1-I of section 5.1.5, Routine Input/Output Variables. The PSV utility processor subroutine SPSV is called to interpret a coded word to obtain the format of the data contained in the state vector, and to display the state vector with appropriate alphanumeric captions. The FDS utility routine XPXIT is called to terminate execution of the PSV processor.

5.1.3 Assumptions and Limitations

Only the FDS-supported coordinate sets, which are defined in table 1.2-VI of reference 1, are displayed with appropriate captions.

5.1.4 Method

The PSV processor subroutine SPSV performs the primary function of this processor. Section 5.2.4 provides a discussion of the method used for this function and the checks that are performed on the coded word containing the data format. The following sections describe the method used in the initialization/input logic and the general method of the subroutine SPSV.

a. Initialization/input logic - The method, conventions, and logic involved in the initialization and input to the PSV utility processor are described in this section. The logical unit number of the user terminal must be known internally to the utility processor program in order to provide displays on the user terminal and to obtain input data from the processor interface table through the AWA. The Hewlett-Packard Real-Time Executive (RTE) provides this capability to applications programs, including a FORTRAN callable routine RMPAR. The form of the call is

```
CALL RMPAR(IPARM)
```

where the statement DIMENSION IMPARM(5) must be included in the specification statements. On return from RMPAR, the logical unit number of the user terminal is contained in IPARM(1) and must be saved in a global variable for use by the processor routines as needed. The first two executable statements in the processor main program must be

```
CALL RMPAR(IPARM)

LU = IPARM(1).
```

The input data from the interface table is obtained from the AWA by calling the FDS utility subroutine XPGET. The form of the call is

```
CALL XPGET (LU,INTBUF,INTLNG,MRBUFF,N,INUMS,IN(1),------IN(N))
```

A full description of the purpose, method, and use of XPGET is provided in section 2.2, Parameter Retrieval Routine (XPGET) (ref. 2). Briefly, a variable name is provided in the calling sequence for each parameter that is to be input. On return from XPGET, each variable has been loaded with the data contained in the AWA.

b. Display of the state vector - The input parameter SV is a 15-element array containing the position/velocity state vector. The 14th element of SV is a vector code word which, when decoded, identifies the reference axis and element set of the data. The subroutine SPSV decodes the vector code word, performs validity checks on the decoded values, sets the validity check flag, displays the state vector information, and returns control to the main program.

c. Termination/output logic - On returning from the subroutine SPSV, the main program checks the validity check flag. If an invalid condition is returned by the validity check flag, the abnormal termination flag is set. If a valid condition is returned by the validity check flag, the normal termination is set. Execution of the processor is terminated by calling the FDS utility routine XPXIT. The sequence for normal termination is

```
IPARM(1) = 0

CALL XPXIT (LU,IPARM)
```

The sequence for abnormal termination is

```
IPARM(1) = -32768

CALL XPXIT (LU,IPARM)
```

5.1.5 Routine Input/Output Variables

The input/output variables for the PSV main program are presented in table 5.1-I.

5.1.6 Functional Logic Flow

Figure 5.1-1 presents the functional logic flow for the PSV main program.

5.1.7 Diagnostics and Debug

None.

5.1.8 Special Comments

None.

5.1.9 References

1. Flight Design System-1, System Design Document, Standards. Vol. VI, Rev. 1, JSC IN 77-FM-18, January 1978.

2. Flight Design System-1, System Design Document, Utility Support Software. Vol. VII, Rev. 1, JSC IN 77-FM-18, February 1978.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine PSV

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
SV	--	Real	I	--	IT	SV	Position/velocity state. Refer to figure 1.2-2 of reference 1 for definition of contents.
VNAM	--	18CH	I	--	IT	VNAM	Vector name.

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

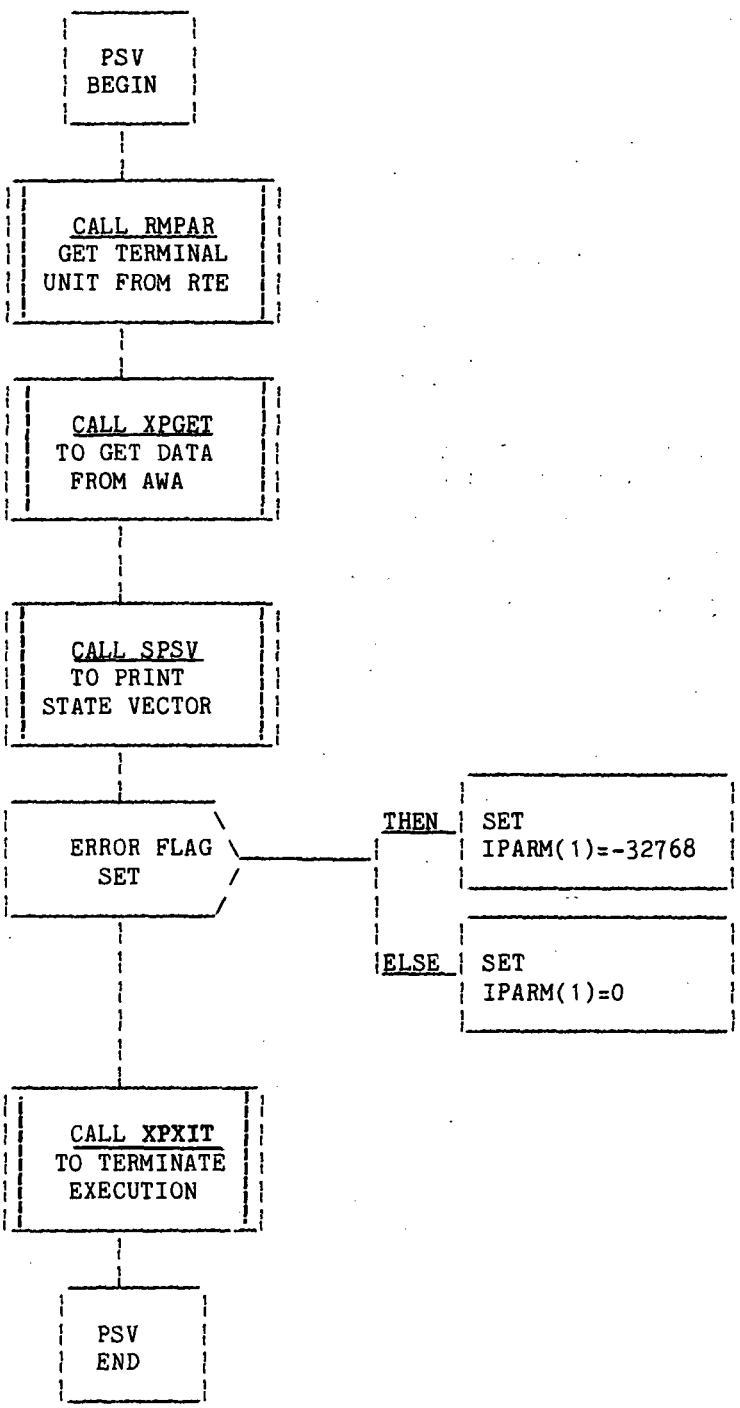


Figure 5.1-1.- PSV functional logic flow.

5.2 ROUTINE NAME - SPSV

5.2.1 Purpose

The print state vector (SPSV) subroutine prints the position/velocity state.

5.2.2 Functional Description

The 14th element of the input array SV is a vector code word that defines the element set and the reference axis. If the vector code word, SV(14), is less than zero, the error flag is set and error message 3 is displayed. Refer to table 4-V, PSV Processor Messages, for content and explanation of the error messages. The sign of SV(14) is set to plus, and the value is converted to an integer. The reference axis and the element set are extracted from the integer valued code word. The reference axis code and the element set code are checked for valid values. If the reference axis code is greater than 4, the error flag is set and error message 1 is displayed. If the element set code is greater than 12, and not equal to 19, 20, or 30, the error flag is set and message 2 is displayed. If the element set code equals 19, a flag is set to display six elements. If the element set code equals 20 or 30, a flag is set to display 9 elements. If the element set code is less than or equal to 12, a flag is set to display 6 elements.

The reference axis/element set combination is checked next. When the reference axis code equals zero, and the element set code is less than or equal to 7 or equal to 20, the error flag is set and error message 3 is displayed. When the reference axis code equals 1, and the element set code is greater than 2, the error flag is set and error message 3 is displayed. When the reference axis code equals 4, and the element set code is greater than 7, the error flag is set and error message 3 is displayed.

When the error checking is completed, the error flag is tested. If the error flag is set, another flag is set to display nine elements, the characters ***s and blanks are moved into the display arrays for the alphanumeric headings. If the error flag is not set, the appropriate reference axis headings and element set headings are moved to the display arrays for the alphanumeric headings. The vector name, reference axis mnemonic, element set mnemonic, time, mass, drag coefficient, and area of drag are displayed. The element set values and identifying mnemonic are displayed. Control is returned to the calling routine.

5.2.3 Assumptions and Limitations

Only the FDS-supported coordinate sets, which are defined in table 1.2-VI of reference 1, are displayed with appropriate captions.

5.2.4 Method

The subroutine SPSV integerizes the vector code word contained in SV(14). The integer vector code word has the form JKNN where J is the reference axis code, K is the element type code, and NN is the element set code. Validity checks are made on these codes between the ranges

$$0 \leq J \leq 4$$

$$0 \leq K \leq 2$$

$$1 \leq NN \leq 12,\ 19 \leq NN \leq 20,\ NN = 30$$

When a code fails the validity check, an error flag is set, an error message is displayed, and control is returned to the main program. Execution of the processor is terminated in the main routine, PSV, by calling the FDS utility subroutine XPXIT with the termination code set to the abnormal termination value of -32768. When the coded values pass the validity checks, the state vector values are displayed with alphanumeric annotations corresponding to the reference axis, the element type, and the element set. Control is returned to the main program and execution of the processor is terminated by calling the FDS utility subroutine XPXIT.

5.2.5 Routine Input/Output Variables

The input/output variables for subroutine SPSV are presented in table 5.2-I. The calling sequence for the subroutine is

```
CALL SPSV (LU,SV,LABEL,ERRFLG)
```

5.2.6 Functional Logic Flow

Figure 5.1-1 presents the functional logic flow for the SPSV subroutine.

5.2.7 Diagnostics and Debug

None.

5.2.8 Special Comments

None.

5.2.9 References

1. Flight Design System-1, System Design Document, Standards. Vol. VI, Rev. 1, JSC IN 77-FM-18, January 1978.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine SPSV

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LABEL	--	18CH	I	--	A		Vector name
LU	--	Intg	I	--	A		Logical unit
SV	--	Real	I	--	A		Position/velocity state
ERRFLG	--	Intg	O	--	A		Error flag = 0; no error = 1; error
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

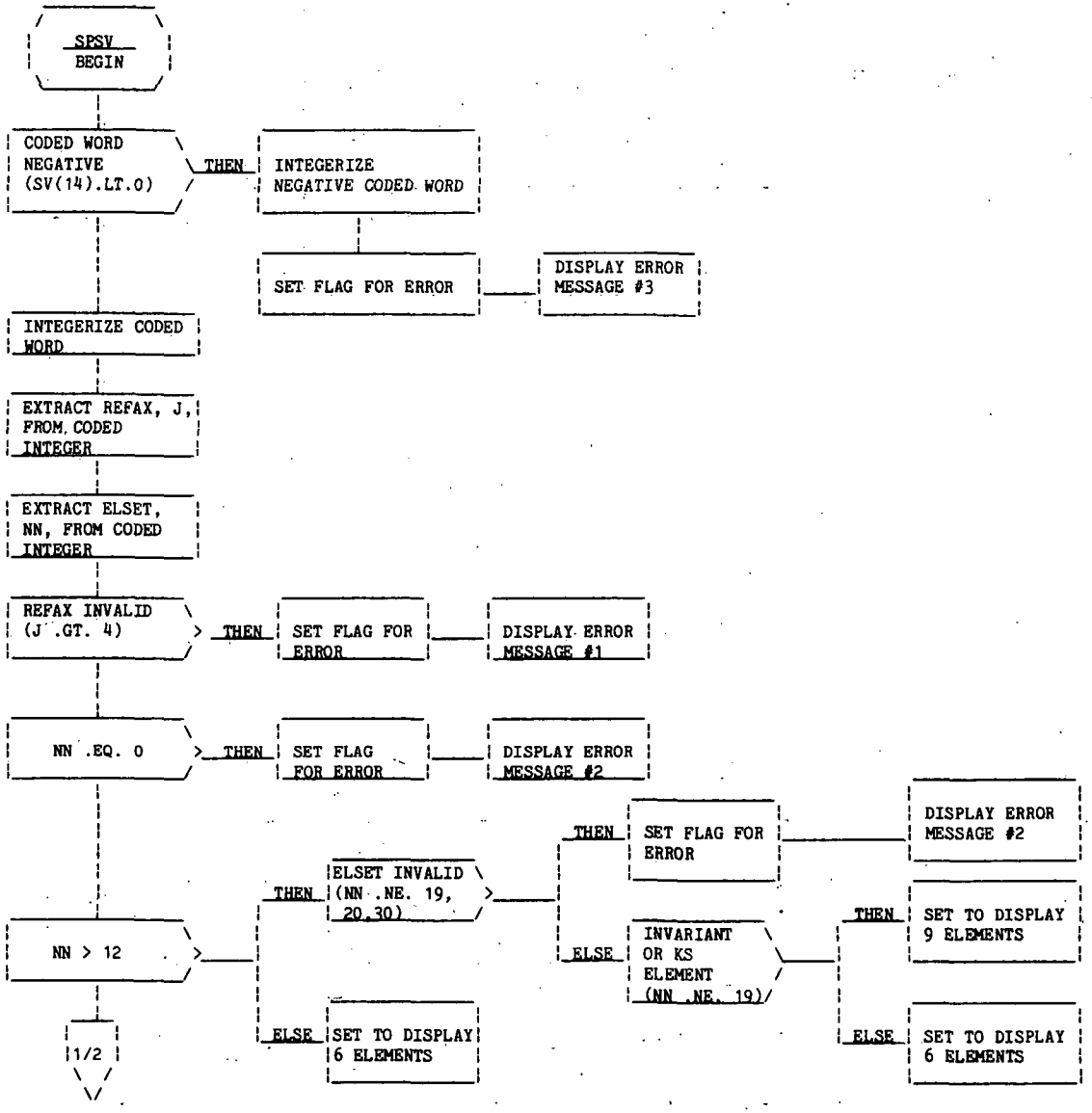


Figure 5.2-1.- SPSV functional logic flow.

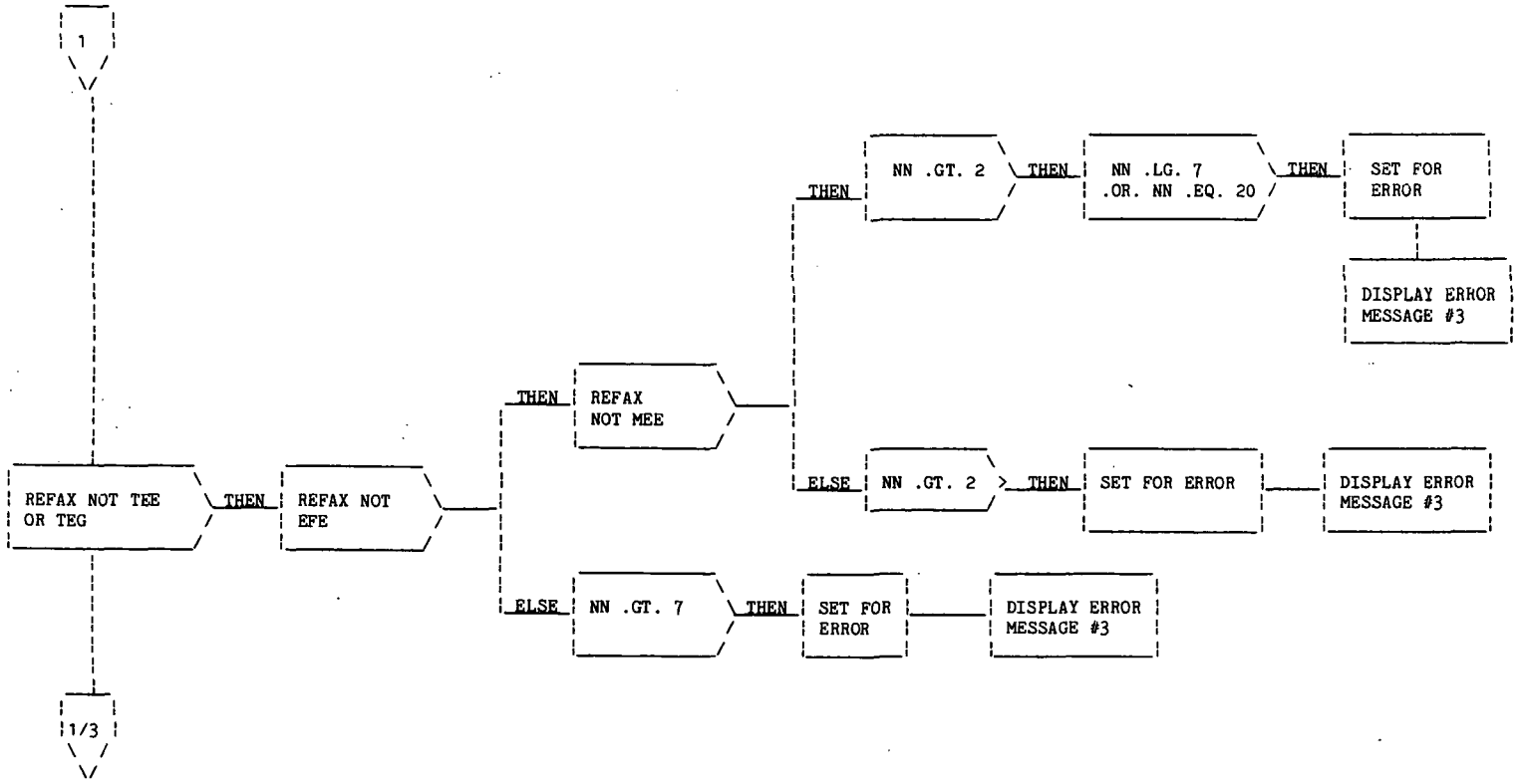


Figure 5.2-1.- Continued.

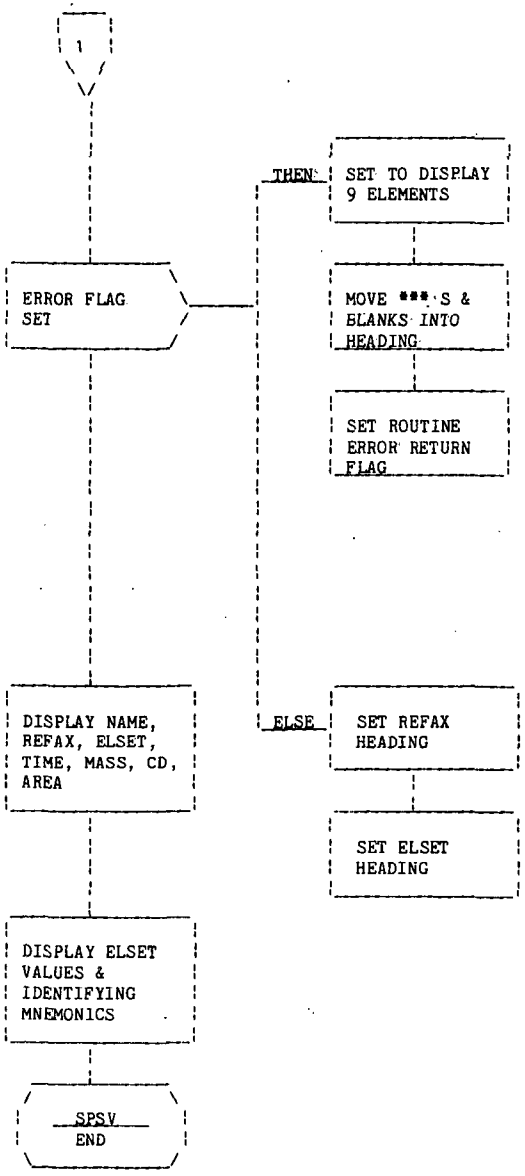


Figure 5.2-1.- Concluded.

PHASE TABLE PRINT PROCESSOR (PTP)

1.0 PURPOSE

A phase table is a sequenced array of state vectors and related propagation vectors. The PTP processor provides the FDS-1 user with the capability to display the information contained in phase tables. The phase tables reside in the user's active work area (AWA) as a disk resident data element (DRDE) or as a data element (DE).

2.0 FUNCTIONAL DESCRIPTION

There are three types of phase tables; namely, position/velocity, attitude, and mass properties. However, only position/velocity phase tables are currently processed by PTP. Position/velocity phase tables are classed as either DE's or DRDE's where the user specifies which class via the interface table. The phase table must be formatted as described in section 7.3.4 in JSC IN 78-FM-60, volume I. It is dimensioned (30,N) where N is the number of states stored. In addition to a state vector, related propagation data are also stored in the propagation vector. Phase tables also contain one or more data documentation (DD) entries. These DD entries contain base date and units information for the preceding vectors in the phase table (see fig. 7.3-16, in JSC IN 78-FM-60, vol. I). If the phase table is classed as a DE then it is limited in size to a (30,20) or 1200 words.

The user selects which phase table vectors are to be displayed by using one of two input modes. In one input mode the user may supply input data interactively as prompted by the processor. In the other input mode the user supplies input data entirely through the interface table. In either mode PTP displays the vector(s) and the DD information entry. In the first mode the user need only supply a valid input quantity for the class, phase table name, and type parameters in the interface table with the prompt parameter set for interactive prompting. PTP then will continuously prompt the user for a vector number and then display the corresponding state vector, propagation vector, and appropriate DD entry. To terminate this prompt and display sequence and normally exit the processor, the user must respond to the prompt with a blank(s) and a carriage return. In the noninteractive mode the user supplies all the interface table parameters: the class of the phase table (either DE or DRDE), the phase table name, the type of phase table, beginning vector number for the display, and the number of vectors to display with the prompt parameter set for noninteractive prompting. PTP then will display the requested vectors and pertinent DD information.

3.0 ASSUMPTIONS AND LIMITATIONS

Only position/velocity phase tables are currently supported.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The definition of the PTP interface table parameters is provided in table 4-I.

CLASS is the parameter through which the user specifies either a DE or DRDE classed phase table. If CLASS indicates that the phase table is a DE, then DENM is the name of the DE or displacement into the DE. If CLASS indicates that the phase table is a DRDE, then DRDENM is the name of the DRDE phase table to be displayed. In the default interface table the CLASS parameter defaults to "DE" and DRDENM defaults to a file named DUMY.

TYPE is the parameter through which the user specifes position/velocity, attitude, and mass properties phase tables. In the default interface table, TYPE is defaulted to position/velocity (PV) phase tables. INDEX is the parameter in which the user specifies the state vector number that begins the display. If CLASS is DE then INDEX is relative to the beginning of the DE phase table.

NUMBER is the number of state vectors to be displayed including the INDEX state vector. If the input value of NUMBER extends beyond the last valid vector in the phase table (i.e., beyond the last DD entry), then a message will be displayed indicating the logical size of the phase table and only vectors to and including the last valid vector will be displayed.

PROMPT is the parameter through which the user defines the mode of inputting data to processor PTP. If PROMPT is "OFF" then a set of state vectors and related propagation vectors is selected via the interface table parameters INDEX and NUMBER. If PROMPT is "ON" then INDEX and NUMBER are ignored and the processor prompts the user for display information via VECTOR NO.: (see processor solicited (prompted) inputs). PROCON is the processor constants array. PROCON(1) sets the cartridge reference parameter, ICR, for the DRDE phase table referred to in the DRDENM parameter. PROCON(2) defines the logical unit number for an alternate display device, should the user wish to display the phase table on some device other than the user's terminal. If the user sets the PROMPT parameter "ON" such that the user is in the interactive mode, then PROCON(2) will be ignored (i.e., selection of an alternate display device is meaningless in this mode).

b. Interface table data array definitions - The data array definitions for the interface table parameters DENM and PROCON are provided in table 4-II.

c. Interface table data file definitions - The definition of the interface table data file parameter DRDENM is provided in table 4-III.

d. Processor solicited (prompted) inputs - The processor solicited prompt is provided in table 4-IV. The prompt occurs only if the PROMPT parameter in the interface table is set "ON". In response to VECTOR NO.: the user must supply either the column number of the state vector to be displayed or a blank(s). If a column number is supplied then the state vector, related propagation data and appropriate DD entry will be displayed. The user is then reprompted with VECTOR NO.: To terminate the prompt mode and normally

exit the processor, the user must respond with a blank(s) and carriage return. If the vector number supplied by the user is less than or equal to zero or greater than the last DD entry in the phase table, an appropriate message will be displayed, and the user will be reprompted with the VECTOR NO.: prompt.

e. Processor displays and display parameter definition table - The PTP processor generates a display for position/velocity phase tables. The display contains a DD entry, a state vector, and a propagation vector. If the phase table vector number being displayed contains a DD entry, then the propagation portion of the display will be suppressed. There are four formats for this display associated with the two propagation modes and three guidance options. For the powered flight propagation mode there is one format for each of the three guidance/steering options. The fourth format is the coasting flight propagation mode. The format for a powered flight propagation mode with the impulsive maneuver guidance/steering option is shown in table 4-V(a). A definition of the displayed variables is provided in table 4-V(b). The format for a powered flight propagation mode with the inertially fixed thrust (PEG7) guidance/steering option is shown in table 4-V(c). A definition of the displayed variables is provided in table 4-V(d). The format for a powered flight propagation mode with the closed-loop guidance (PEG4) guidance/steering option is shown in table 4-V(e). A definition of the displayed variables is provided in table 4-V(f). The format for a coasting flight propagation mode is shown in table 4-V(g), and a definition of the displayed variables is provided in table 4-V(h).

f. Processor message table - The message table for the PTP processor is provided in table 4-VI.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VII.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR PTP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
CLASS	AWA	6CH	I	3	1	"DE"	DE or DRDE class indicator = "DE"; phase table resides in a DE = "DRDE"; phase table resides in a DRDE file
DENM	AWA	Real	I	1200	(30,20)		Phase table name for DE's
DRDENM	Disk	Real	I	—	—	DUMY	Phase table name for DRDE's
TYPE	AWA	2CH	I	1	1	"PV"	Type of phase table = "PV"; for position/velocity phase tables = "AT"; for attitude phase tables = "MP"; for mass properties phase tables
INDEX	AWA	Intg	I	1	1		Beginning vector number in the display
NUMBER	AWA	Intg	I	1	1		Number of vector(s) to display
PROMPT	AWA	6CH	I	3	1	"OFF"	Prompt option flag = "OFF"; PTP does not prompt user for VECTOR NO.: = "ON"; PTP prompts user for VECTOR NO.:
PROCON	AWA	Free	I	2	2		Processor constants array

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR PTP

Array name	Index location	Default value	Definition
DENM	(1,1) . . . (30,1) (1,2) . . . (30,2) . . . (1,20) . . . (30,20)		A position/velocity phase table residing in a DE within the user's AWA and formatted as described in section 7.3.4 in JSC IN 78-FM-60, volume I. DENM may contain multiple DD entries.
PROCON	1	20	Cartridge reference number for DRDE files.
	2	0	Logical unit number for display output; 0 = user's terminal.

TABLE 4-III.- INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR PTP

DRDE DATA FILE DRDENM

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file
	4-6	Interface table variable creating file
	7-9	Processor last changing file
	10-12	Interface table variable last changing file
2-N	1-60	Position/velocity phase table column (see section 7.3.4.b, in JSC IN 78-FM-60, volume I). Any column may contain a DD entry.

TABLE 4-IV.- PROCESSOR SOLICITED (PROMPTED) INPUTS

PROCESSOR PTP

Prompt	Meaning	Valid responses
VECTOR NO.:	Column number in the phase table of the vector to be displayed	= X where X = 1-N and N = number of columns in the phase table
)	Terminates the processor	= "Ø"

TABLE 4-V.- PROCESSOR DISPLAYS AND DISPLAY PARAMETER DEFINITION TABLES

(a) Impulsive maneuver guidance/steering option

PROCESSOR PTP

```
                              PHASE TABLE αααααα

BASE TIME  = YEAR XXXX , DAY XXX   , HOUR ±X.XXXXXXE±XX
UNITS      = AAAA, DDDD, TTTT, VVVV, MMMM, FFFF, LLLL

PHASE TABLE αααααα VECTOR αααααα                                  EEEEEE
                   STATE          RRR
TIME=±X.XXXXXXE±XX GMT     MASS=±X.XXXXXXE±XX
  CD=±X.XXXXXXE±XX         AREA=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX         SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX         SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX         SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX
PHASE      = POWERED FLIGHT   BURN TIME    =±X.XXXXXXE±XX    DRAG   =αααααααα
PROCESSOR  = αααααα           PROPAGATION  =αααααααααααα
GUIDANCE          = IMPULSIVE MANEUVER AT STATE TIME
PROPULSION        = ααααααα
THRUST  REF.AXIS  = ααα
       LVLH                        INERTIAL
DVX  =±X.XXXXXXE±XX           VX  =±X.XXXXXXE±XX       THRUST  =±X.XXXXXXE±XX
DVY  =±X.XXXXXXE±XX           VY  =±X.XXXXXXE±XX       ISP     =±X.XXXXXXE±XX
DVZ  =±X.XXXXXXE±XX           VZ  =±X.XXXXXXE±XX
```

TABLE 4-V.- Continued

(b) Display parameter definition for the impulsive maneuver guidance/steering option

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition
YEAR	Base time, year
DAY	Base time, day
HOUR	Base time, hour past midnight
AAAA	Mnemomic for angle units
DDDD	Mnemonic for distance units
TTTT	Mnemonic for time units
VVVV	Mnemonic for velocity units
MMMM	Mnemonic for mass units
FFFF	Mnemonic for force units
LLLL	Mnemonic for length units
	Data documentation entry
aaaaaa	Name of phase table being displayed
aaaaaa	Vector number being displayed
RRR	Reference axis (3 characters); see table 7.3-V in JSC IN 78-FM-60, volume I
EEEEEE	Element set (6 characters); see table 7.3-VI in JSC IN 78-FM-60, volume I
TIME	Time of the state vector relative to a user-established base date
MASS	Vehicle gross mass
CD	Drag coefficient
AREA	Area for drag computation
SSSS	Six or nine elements that specify the position and velocity of the vehicle's center of mass; refer to table 7.3-VI in JSC IN 78-FM-60, volume I for definition of the mnemonic associated with each state vector component. The invariant and KS element types have nine elements; all others have six elements.
PHASE	Propagation code defined as POWERED FLIGHT for table V(a) of this processor.
BURN TIME	Burn time to next state
DRAG	Drag option = constant computed
PROCESSOR	Processor for simulator code; see table 7.3-VII in JSC IN 78-FM-60, volume I
PROPAGATION	Type of propagation; see table 7.3-VII in JSC IN 78-FM-60, volume I
GUIDANCE	Type of guidance = Impulsive maneuver at state time
PROPULSION	Type of propulsion; see table 7.3-IX in JSC IN 78-FM-60, volume I
THRUST REF. AXIS	Coordinate system flag for thrust alinement; see table 7.3-V in JSC IN 78-FM-60, volume I

TABLE 4-V.- Continued

(b) Concluded

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition		
LVLH (DVX)	ΔV_x	LVLH	For guidance = Impulsive maneuver at state time
LVLH (DVY)	ΔV_y	LVLH	
LVLH (DVZ)	ΔV_z	LVLH	
INERTIAL (VX)	V_x	INERTIAL	
INERTIAL (VY)	V_y	INERTIAL	
INERTIAL (VZ)	V_z	INERTIAL	
THRUST	Thrust		
ISP	Isp		

TABLE 4-V.- Continued

(c) Inertially fixed thrust (PEG7) guidance/steering option

PROCESSOR PTP

```
                               PHASE TABLE αααααα

 BASE  TIME  = YEAR XXXX , DAY XXX   , HOUR ±X.XXXXXXE±XX
 UNITS       = AAAA, DDDD, TTTT, VVVV, MMMM, FFFF, LLLL

 PHASE TABLE αααααα VECTOR αααα                                        EEEEEE
                    STATE           RRR
 TIME=±X.XXXXXXE±XX GMT      MASS=±X.XXXXXXE±XX
   CD=±X.XXXXXXE±XX          AREA=±X.XXXXXXE±XX
 SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX              SSSS=±X.XXXXXXXE±XX
 SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX              SSSS=±X.XXXXXXXE±XX
 SSSS=±X.XXXXXXE±XX          SSSS=±X.XXXXXXE±XX              SSSS=±X.XXXXXXXE±XX
 PHASE       = POWERED FLIGHT    BURN TIME      =±X.XXXXXXE±XX
 PROCESSOR   = αααααα            PROPAGATION    =αααααααααααα
 GUIDANCE          = INERTIALLY FIXED THRUST DIRECTION - PEG7
 PROPULSION        = ααααααα
 THRUST REF. AXIS  = ααα
           LVLH                  INERTIAL
 DVX  =±X.XXXXXXE±XX         VX  =±X.XXXXXXE±XX            TIG  =±X.XXXXXXE±XX
 DVY  =±X.XXXXXXE±XX         VY  =±X.XXXXXXE±XX
 DVZ  =±X.XXXXXXE±XX         VZ  =±X.XXXXXXE±XX
```

TABLE 4-V.- Continued

(d) Display parameter definition table for the inertially fixed thrust (PEG7) guidance/steering option

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition
YEAR	Base time, year
DAY	Base time, day
HOUR	Base time, hour past midnight
AAAA	Mnemonic for angle units
DDDD	Mnemonic for distance units
TTTT	Mnemonic for time units
VVVV	Mnemonic for velocity units
MMMM	Mnemonic for mass units
FFFF	Mnemonic for force units
LLLL	Mnemonic for length units
	Data documentation entry
aaaaaa	Name of phase table being displayed
aaaaaa	Vector number being displayed
RRR	Reference axis (3 characters); see table 7.3-V in JSC IN 78-FM-60, volume I
EEEEEE	Element set (6 characters); see table 7.3-VI in JSC IN 78-FM-60, volume I
TIME	Time of the state vector relative to a user-established base date
MASS	Vehicle gross mass
CD	Drag coefficient
AREA	Area for drag computation
SSSS	Six or nine elements that specify the position and velocity of the vehicle's center of mass; refer to table 7.3-VI in JSC IN 78-FM-60, volume I for definition of the mnemonic associated with each state vector component. The variant and KS element types have nine elements; all others have six elements.
PHASE	Propagation code defined as POWERED FLIGHT for table V(c) of this processor.
BURN TIME	Burn time to next state
DRAG	Drag option = constant computed
PROCESSOR	Processor for simulator code; see table 7.3-VII in JSC IN 78-FM-60, volume I
PROPAGATION	Type of propagation; see table 7.3-VII in JSC IN 78-FM-60, volume I
GUIDANCE	Type of guidance = Inertially fixed thrust direction - PEG7
PROPULSION	Type of propulsion; see table 7.3-IX, in JSC IN 78-FM-60, volume I
THRUST ALIGNMENT	Coordinate system flag for thrust alinement; see table 7.3-V in JSC IN 78-FM-60, volume I

TABLE 4-V.- Continued

(d) Concluded

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition		
LVLH (DVX)	ΔV_x	LVLH	For guidance = Inertially fixed thrust direction - PEG7
LVLH (DVY)	ΔV_y	LVLH	
LVLH (DVZ)	ΔV_z	LVLH	
INERTIAL (VX)	V_x	Inertial at state time	
INERTIAL (VY)	V_y	Inertial at state time	
INERTIAL (VZ)	V_z	Inertial at state time	
TIG	Tig		

TABLE 4-V.- Continued

(e) Closed-loop guidance (PEG4) guidance/steering option

PROCESSOR PTP

```
                              PHASE TABLE αααααα

BASE TIME = YEAR XXXX , DAY XXX , HOUR ±X.XXXXXXE±XX
UNITS     = AAAA, DDDD, TTTT, VVVV, MMMM,FFFF, LLLL

PHASE TABLE αααααα VECTOR αααααα
                   STATE            RRR                                 EEEEEE
TIME=±X.XXXXXXE±XX GMT        MASS=±X.XXXXXXE±XX
  CD=±X.XXXXXXE±XX            AREA=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX            SSSS=±X.XXXXXXE±XX             SSSS=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX            SSSS=±X.XXXXXXE±XX             SSSS=±X.XXXXXXE±XX
SSSS=±X.XXXXXXE±XX            SSSS=±X.XXXXXXE±XX             SSSS=±X.XXXXXXE±XX
PHASE     = POWERED FLIGHT        BURN TIME   = ±X.XXXXXXE±XX     DRAG = αααααα
PROCESSOR = ααααα                 PROPAGATION = ααααααααααααα
GUIDANCE        = CLOSED LOOP GUIDANCE - PEG4
PROPULSION      = ααααααα
THRUST REF. AXIS= ααα
                                   INERTIAL
HEI  =±X.XXXXXXE±XX           VV =±X.XXXXXXE±XX              C1 =±X.XXXXXXE±XX
QEI  =±X.XXXXXXE±XX           VY =±X.XXXXXXE±XX              C2 =±X.XXXXXXE±XX
FUEL =±X.XXXXXXE±XX           VZ =±X.XXXXXXE±XX
```

TABLE 4-V.- Continued

(f) Display parameter definition table for the closed-loop (PEG4) guidance/steering option

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition
YEAR	Base time, year
DAY	Base time, day
HOUR	Base time, hour past midnight
AAAA	Mnemonic for angle units
DDDD	Mnemonic for distance units
TTTT	Mnemonic for time units
VVVV	Mnemonic for velocity units
MMMM	Mnemonic for mass units
FFFF	Mnemonic for force units
LLLL	Mnemonic for length units
	Data documentation entry
aaaaaa	Name of the phase table being displayed
aaaaaa	Vector number being displayed
RRR	Reference axis (3 characters); see table 7.3-V in JSC IN 78-FM-60, volume I
EEEEEE	Element set (6 characters); see table 7.3-VI in JSC IN 78-FM-60, volume I
TIME	Time of the state vector relative to a user-established base date
MASS	Vehicle gross mass
CD	Drag coefficient
AREA	Area for drag computation
SSSS	Six or nine elements that specify the position and velocity of the vehicle's center of mass; refer to table 7.3-VI in JSC IN 78-FM-60, volume I for definition of the mnemonic associated with each state vector component. The variant and KS element types have nine elements; all others have six elements.
PHASE	Propagation code defined as POWERED FLIGHT for table V(c) of this processor.
BURN TIME	Burn time to next state
DRAG	Drag option = constant computed
PROCESSOR	Processor for simulator code; see table 7.3-VII in JSC IN 78-FM-60, volume I
PROPAGATION	Type of propagation; see table 7.3-VII in JSC IN 78-FM-60, volume I
GUIDANCE	Type of guidance = Closed-loop guidance - PEG4
THRUST ALIGNMENT	Coordinate system flag for thrust alignment; see table 7.3-V in JSC IN 78-FM-60, volume I

TABLE 4-V.- Continued

(f) Concluded

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition		
HEI	Hei		
QEI	Qei		
FUEL	Ifuel		
INERTIAL (VX)	V_x	Inertial at TIG	For guidance = closed loop - PEG4
INERTIAL (VY)	V_y		
INERTIAL (VZ)	V_z		
C1	C_1		
C2	C_2		

TABLE 4-V.- Continued

(g) Coasting flight propagation mode

PROCESSOR PTP

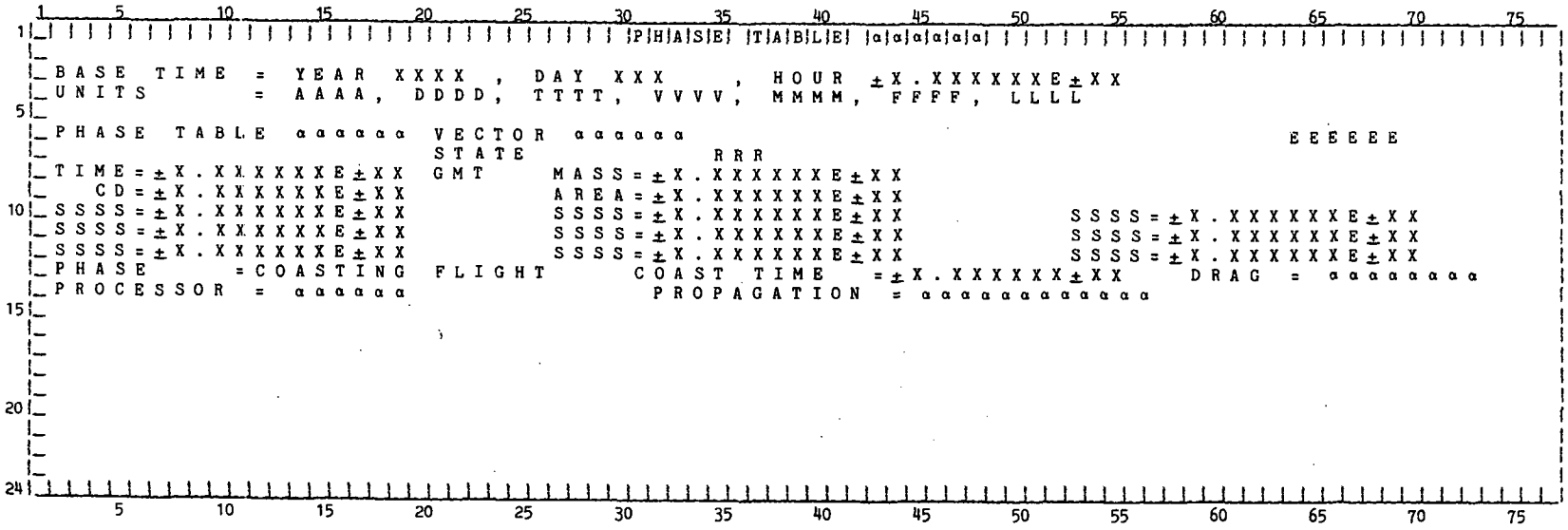


TABLE 4-V.- Concluded

(h) Display parameter definition table for the coasting flight propagation mode

PROCESSOR PTP

PHASE TABLE

Display parameter label	Parameter definition
YEAR	Base time, year
DAY	Base time, day
HOUR	Base time, hour
AAAA	Mnemonic for angle units
DDDD	Mnemonic for distance units
TTTT	Mnemonic for time units
VVVV	Mnemonic for velocity units
MMMM	Mnemonic for mass units
FFFF	Mnemonic for force units
LLLL	Mnemonic for length units
	Data documentation entry
PTPTPT	Name of phase table being displayed
VVVVVV	Vector number being displayed
RRR	Reference axis (3 characters); see table 7.3-V in JSC IN 78-FM-60, volume I
EEEEEE	Element set (6 characters); see table 7.3-VI in JSC IN 78-FM-60, volume I
TIME	Time of the state vector relative to a user-established base date
MASS	Vehicle gross mass
CD	Drag coefficient
AREA	Area for drag computation
SSSS	Six or nine elements that specify the position and velocity of the vehicle's center of mass; refer to table 7.3-VI in JSC IN 78-FM-60, volume I for definitions of the mnemonic associated with each state vector component. The variant and KS element types have nine elements; all others have six elements
PHASE	Propogation code defined as COASTING FLIGHT for table V(g) of this processor.
COAST TIME	Coast time to next state
DRAG	Drag option = constant computed
PROCESSOR	Processor for simulator code; see table 7.3-VII in JSC IN 78-FM-60, volume I
PROPAGATION	Type of propagation; see table 7.3-VII in JSC IN 78-FM-60, volume I

TABLE 4-VI.- PROCESSOR MESSAGE TABLE

PROCESSOR PTP

MSG no.	Message ID block	Message text block and explanation
1	*PTP*	ABORT, CLASS MUST EQUAL DE OR DRDE Meaning: CLASS must equal "DE" or "DRDE". Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid response to CLASS.
2	*PTP*	ABORT, PROMPT MUST BE ON OR OFF Meaning: PROMPT must equal "ON" or "DRDE". Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid response to PROMPT.
3	*PTP*	WARNING, INTERACTIVE MODE - PROCON(2) IGNORED Meaning: PTP ignored user's selection of an alternate display device. Severity: Does not interfere with normal processor execution. Action required by user: None
4	*PTP*	ABORT, TYPE PARAMETER NOT VALID Meaning: TYPE must equal "PV", because only position/velocity phase tables are currently processed. Severity: PTP aborted, no output generated Action required by user: Use the Interface Table Editor to enter a valid response to TYPE.
5	*PTP*	ABORT, (MP) TYPED PHASE TABLE NOT IMPLEMENTED ABORT, (AT) TYPED PHASE TABLE NOT IMPLEMENTED Meaning: Mass properties and attitude phase tables are not currently processed by PTP. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid response to TYPE.
6	*PTP*	ABORT, DRDE FILE IS NOT OF TYPE 2 Meaning: User selected a DRDE file that was not a type 2 file in response to DRDENM parameter. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid phase table DRDE file name to DRDENM parameter.

TABLE 4-VI.- Continued

PROCESSOR PTP

MSG no.	Message ID block	Message text block and explanation
7	*PTP*	ABORT, DRDE FILE RECORD LENGTH NOT EQUAL TO 60 Meaning: User selected a DRDE file whose record length was not equal to 60. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid phase table DRDE file name to DRDENM parameter.
8	*PTP*	ABORT, FILE OPEN ERROR = IIII, FILENAME = aaaaaa Meaning: The DRDE file phase table is not in the proper configuration. Severity: PTP aborted, no output generated. Action required by user: Check the filename response to DRDENM.
9	*PTP*	ABORT, FILE READ ERROR = IIII, RECORD = IIII, FILENAME = aaaaaa Meaning: The DRDE file phase table is not in the proper configuration or there was a hardware failure. Severity: PTP aborted - may have generated output. Action required by user: Check the filename response to DRDENM or investigate hardware problems.
10	*PTP*	WARNING, NO OF DD'S IN PHASE TABLE aaaaaa EXCEED AVAILABLE SPACE Meaning: Internal space limits the number of DD entries in a DRDE phase table to 55 entries. If more than 55 entries are found they are ignored. As a result the phase table column number of the 55th DD entry is the last logical column in the phase table. Severity: Does not interfere with normal processor execution. Action required by user: None
11	*PTP*	ABORT, PHASE TABLE aaaaaa DOES NOT CONTAIN ANY DD INFORMATION Meaning: PTP aborted because the phase table had no valid DD information in any column. Severity: PTP aborted, no output generated. Action required by user: Check the phase table and verify that it is a phase table because it must contain DD information.

TABLE 4-VI.- Continued

PROCESSOR PTP

MSG no.	Message ID block	Message text block and explanation
12	*PTP*	ABORT, USER'S RESPONSE TO INDEX OUT OF RANGE Meaning: PTP aborted because user's response to INDEX in the interface table was outside the limits of the table. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter a valid number to NUMBER and INDEX, then execute PTP.
13	*PTP*	WARNING, VECTOR RESPONSE OUT OF RANGE Meaning: User's response to "VECTOR NO.:" prompt was not valid. Severity: Does not interfere with normal processor execution. Action required by user: Respond to "VECTOR NO.:" with a valid response. See table 4-IV.
14	*PTP*	ABORT CODE = I FROM XPRDS CALL Meaning: Erroneous response to the "VECTOR NO.:" prompt Severity: Error message displayed and prompt is reissued. Action required by user: Respond as shown in table 4-IV to next prompt.
15	*PTP*	DEBUG ABORT, THERE IS NO NEXT DD Meaning: Mistake in logic within PTP. Severity: PTP aborted due to irrecoverable error. Action required by user: Consult MPAD personnel, R. S. Davis.
16	*PTP*	ABORT, DENM RESPONSE REQUIRES A NAME NOT LITERAL DATA Meaning: PTP aborted because the user entered literal data not alphanumeric name to DENM prompt. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter an alphanumeric name to DEDM prompt, then execute PTP.

TABLE 4-VI.- Continued

PROCESSOR PTP

MSG no.	Message ID block	Message text block and explanation
17	*PTP*	ABORT, PHASE TABLE aaaaaa MUST BE TYPED REAL IN AWA Meaning: PTP aborted because user entered a phase table name and the phase table was not typed Real in the AWA. The name may not be the name of a phase table. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter the name of a correctly typed phase table.
18	*PTP*	ABORT, PHASE TABLE aaaaaa MATRIX COLUMN LENGTH NOT EQUAL TO 30 Meaning: User entered a phase table name and the phase table does not have an I dimension equal to 30. The name may not be a phase table name. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter the name of a valid phase table.
19	*PTP*	ABORT, aaaaaa SUBSCRIPTS NOT IN FORMAT (1,N) WHERE N=1, # COLUMNS Meaning: User entered a subscripted phase table name where subscripts do not fall on a phase table column boundary in response to DENM parameter. Severity: PTP aborted, no output generated. Action required by user: Use the Interface Table Editor to enter the name and valid subscripts to the DENM parameter.
20	*PTP*	ERROR, PHASE TABLE aaaaaa VECTOR (II,JJ) INVALID ±.XXXXXXE±XX Meaning: An error was detected in Vector (II,JJ) of the phase table aaaaaaa. The value of the invalid vector = ±X·XXXXXXE±XX·***'s will represent this value in displays. Severity: If this vector is used in subsequent calculations, they too may be invalid, otherwise this error does not interfere with normal processor execution. Action required by user: Check the invalid vector and determine if it is appropriate to have that vector in the phase table.
21	*PTP*	PHASE TABLE aaaaaa CREATED BY aaaaaa/aaaaaa Meaning: If the phase table is classed as DRDE then the above message indicates what processor and interface table variable created the phase table. Severity: Does not interfere with normal processor execution. Action required by user: None.

TABLE 4-VI.- Concluded

PROCESSOR PTP

MSG no.	Message ID block	Message text block and explanation
22	*PTP*	PHASE TABLE aaaaaa LAST CHANGED BY aaaaaa/aaaaaa Meaning: If the phase table is classed as DRDE then the above message indicates what processor and interface table variable last changed the phase table. Severity: Does not interfere with normal processor execution. Action required by user: None.
23	*PTP*	PHASE TABLE aaaaaa IS A (30,NN) ARRAY Meaning: There are NN columns in phase table aaaaaa. Severity: Does not interfere with normal processor execution. Action required by user: None.

TABLE 4-VII.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR PTP

Processor name	Processor abstract prompt (maximum 256 characters)
PTP	The PTP processor provides the FDS-1 user with the capability to display a phase table residing in a DE or DRDE file in the user's AWA.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
CLASS	Class indicating where the phase table resides. = "DE" ; phase table resides in a DE = "DRDE"; phase table resides in a DRDE file
DENM	Name for phase tables that are DE's
DRDENM	Name for phase tables that are DRDE's
TYPE	Type of phase table. = "PV"; position/velocity phase table = "AT"; attitude phase tables = "MP"; mass properties phase table
INDEX	Column number of the first state vector to be displayed
NUMBER	Number of vectors to be displayed
PROMPT	Flag defining input mode. = "OFF"; all input is through interface table. = "ON" ; input is via interface table and VECTOR NO.: prompt.
PROCON	Processor constants array

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM PTP

5.1.1 Purpose

The routine PTP serves as the main program of the PTP processor. It obtains most of the interface table parameters and then calls the appropriate routines to decode and display the phase table.

5.1.2 Functional Description

The routine PTP calls the utility routine RMPAR to get the logical unit number of the user's terminal. The FDS utility routine XPGET is called to get the CLASS, TYPE, INDEX, NUMBER, PROMPT, and PROCON parameters. The definitions of these parameters are given in table 4-I. The TYPE parameter is then tested for a valid phase table. Only the position/velocity phase table is valid for FDS-1. If the type is either a mass property or attitude phase table, then the processor displays a message stating that these phase tables will be valid for FDS-2, and the processor terminates execution. Otherwise, an invalid TYPE parameter message is displayed and the processor execution terminates.

The PROMPT parameter contains a value of "ON" or "OFF", and indicates whether the user selected the interactive prompting mode or not. If "ON", PROCON(2) or the alternate print device's logical unit number is ignored and the default device becomes the logical unit number of the user's terminal. If the noninteractive prompting mode was selected, the alternate logical unit number is set to PROCON(2), if present; otherwise, it defaults to the logical unit number of the user's terminal. The CLASS parameter is compared with DE or DRDE to determine whether the phase table resides as an AWA data element or as a disk resident data element. If neither, an error message is displayed indicating that the CLASS parameter is invalid, and execution of the processor is terminated with the error abort code set to -32768. If the phase table is a data element, routine DE is called. If the phase table is a disk resident data element, routine DRDE is called. The FDS utility subroutine XPXIT is called to terminate the processor.

5.1.3 Assumptions and Limitations

a. There will be no more than 55 DD entries in a DRDE classed phase table.

b. Only position/velocity phase tables are displayed for FDS-1.

5.1.4 Method

See Functional Description, section 5.1.2. (For further information, see reference 1.)

5.1.5 Routine Input/Output Variables

The input/output variables for the PTP routine are presented in table 5.1-I.

5.1.6 Functional Logic Flow

The functional logic flow for the PTP routine is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

None.

5.1.8 Special Comments

None.

5.1.9 References

1. Flight Design System-1, System Design Document, Standards. Vol. VI, Rev. 1, JSC IN 77-FM-18, January 1978.

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR SVUCP

Array name	Index location	Default value	Definition
GLOCON	(1) . . . (180)	!!GLCN	Global constants array data base element. See JSC IN 78-FM-60, volume I, table 7.2-III for definition of contents.
SESCON	(1) . . . (90)	!SESCN	Input/output session constants array. See JSC IN 78-FM-60, volume I, table 7.2-II(a) for definition of contents.
SVIN	(1) . . . (15)		The standard position/velocity state vector. See section 7.3 in JSC IN 78-FM-60, volume I.
SVOUT	(1) . . . (15)		The standard position/velocity state vector. See section 7.3 in JSC IN 78-FM-60, volume I.

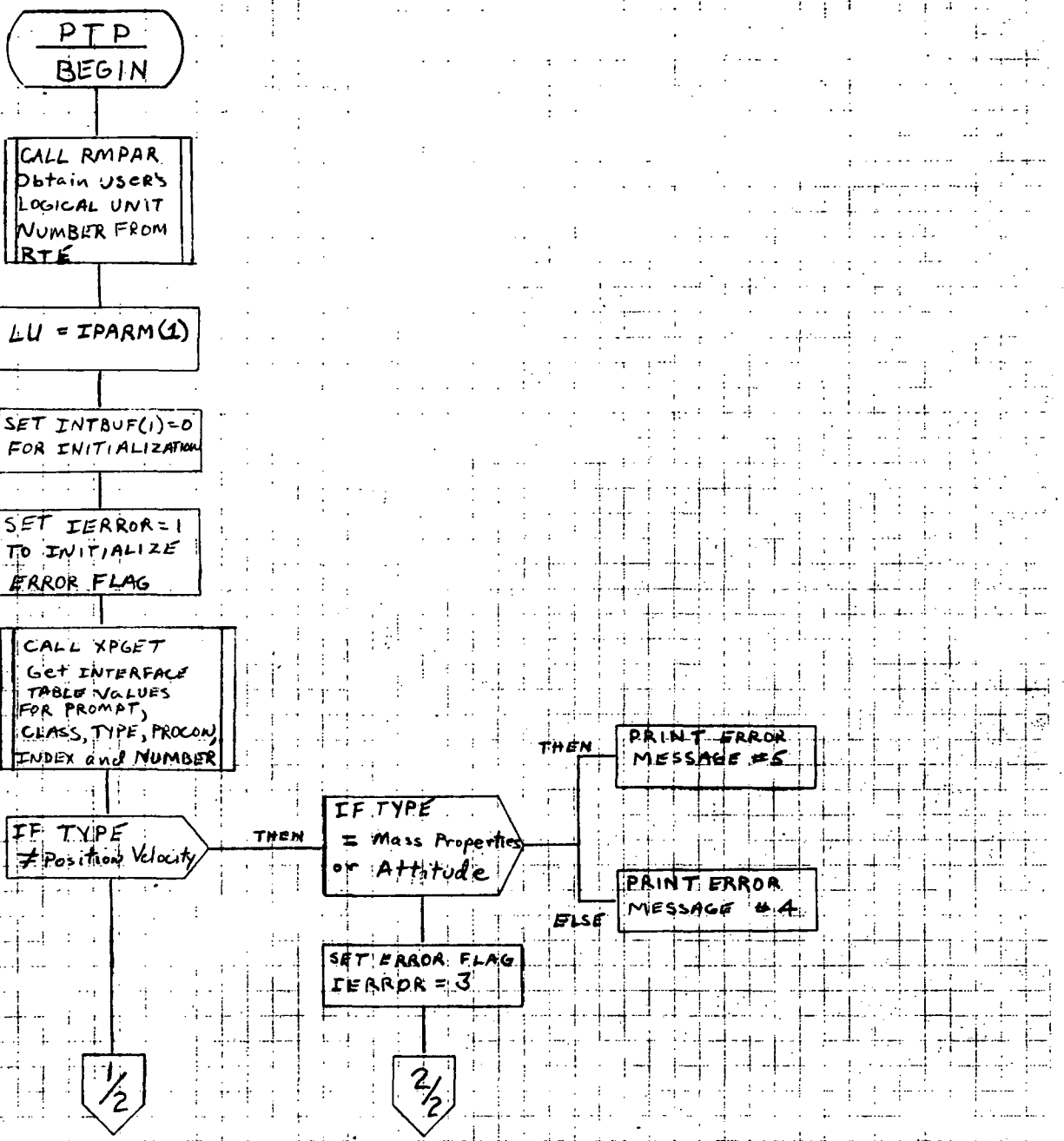


Figure 5.1-1.- PTP functional logic flow.

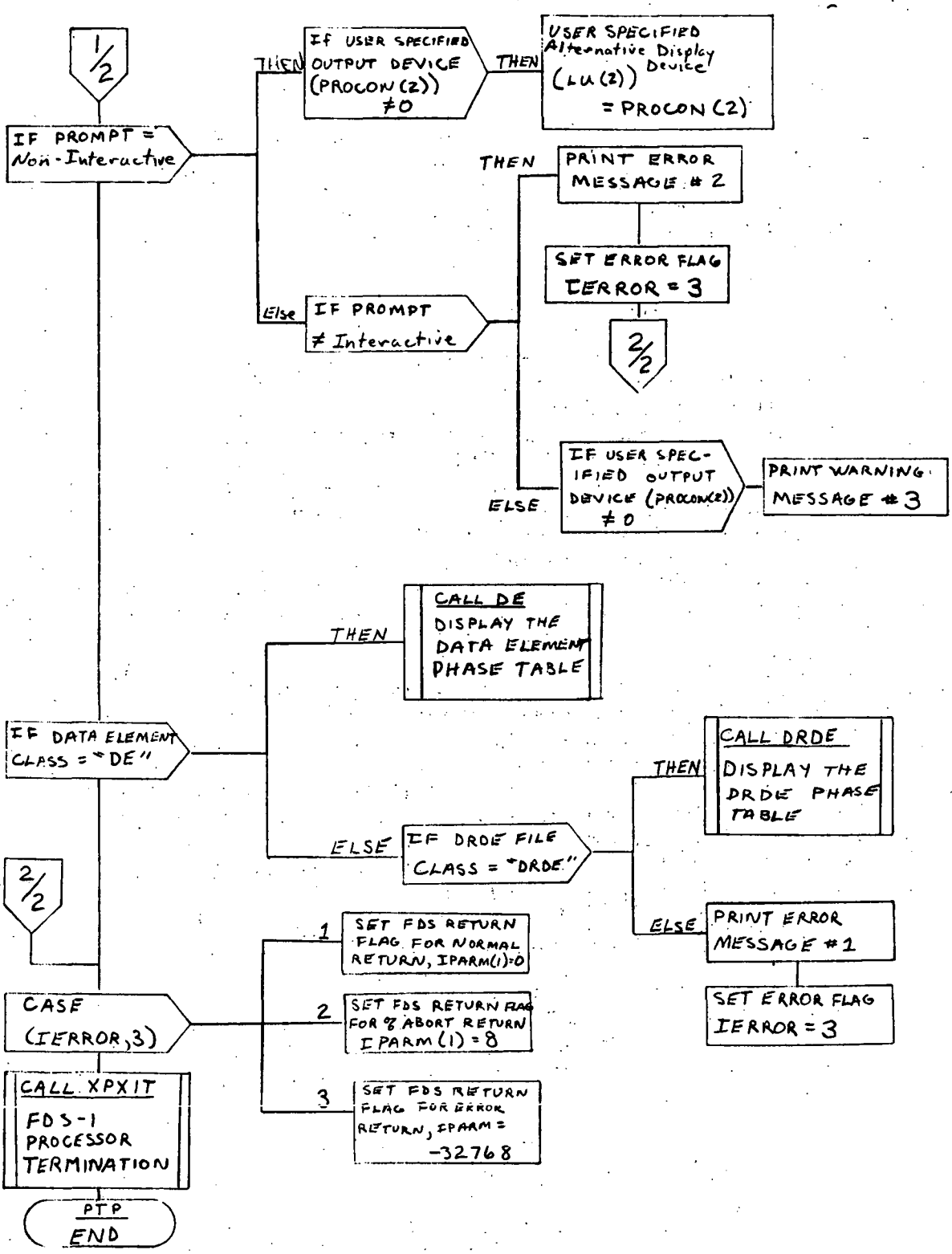


Figure 5.1-1.- Concluded.

5.2 ROUTINE NAME - DRDE

5.2.1 Purpose

The disk resident data element (DRDE) subroutine reads the user-selected DRDE file. Using the parameter PROMPT, the routine uses either interface preset variable data or prompts the user for the variable data. The DRDE routine then calls an appropriate routine to display data definition (DD) information and state and propagation vectors from the DRDE file.

5.2.2 Functional Description

The routine DRDE calls the utility routine XPATR to obtain the attributes of the DRDE file. If the file type is not a type two, execution of the processor is terminated with the error code set to -32768. If the record size is not equal to 60, execution of the processor is terminated with the error code set to -32768. The File Management Package (FMP) routine OPEN is called to open the DRDE file for processing. The header record is read from the DRDE file and displayed on the user's terminal. A search of the DRDE file is made to find and store all occurrences of data definition information. If none are found, execution of the processor is terminated with the error code set to -32768. The last logical column in the DRDE phase table is set to the column number of the last DD entry in the table, and the logical size of the phase table is then displayed to the user. A test is made to determine whether the user is in the interactive mode or the noninteractive mode. If the user selected the noninteractive mode, the interface table parameters INDEX and NUMBER determine what vectors are to be displayed. If INDEX is negative, or greater than the last logical column in the DD array, execution of the processor is terminated with the error code set to -32768. The last user-selected column in the phase table is set to the sum of INDEX + column number - 1. If the last user-selected column exceeds the last logical column, the last user-selected column is set to the last logical column. The start index for displaying the phase table is set at INDEX. The ending index is set at the first DD column number that is greater than the starting index column number. The appropriate DD entry is displayed on the terminal screen via a subroutine call to DDOUT. A record is read from the DRDE file and displayed on the terminal screen via a call to the VCOUT routine.

If the user selected the interactive mode, a call is made to the FDS utility routine XPRDS to display the prompt "VECTOR NO." and waits for the user's response. If data are between one and the last logical column in the DRDE phase table, a search of the DD array is made and the ending index is set equal to the first DD index greater than data.

The appropriate DD entry is displayed on the terminal screen via a call to the DDOUT routine. A record is read from the DRDE file corresponding to data, and is displayed via a call to the display routine VCOUT. If data were not between one and the last logical column, a warning message is displayed on the terminal screen. Another call is made to the XPRDS routine for "VECTOR NO." data.

If either a % or a blank(s) and carriage return is entered, the processor's execution is terminated with the error code set to 8 if %, or 0 if blank(s) and carriage return.

5.2.3 Assumptions and Limitations

See Assumptions and Limitations in section 5.1.3.

5.2.4 Method

See Functional Description in section 5.2.2.

5.2.5 Routine Input/Output Variables

The input/output variables for the DRDE routine are presented in table 5.2-I.

5.2.6 Functional Logic Flow

The functional logic flow for the DRDE routine is presented in figure 5.2-1.

5.2.7 Diagnostics and Debug

None.

5.2.8 Special Comments

None.

5.2.9 References

None.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine DRDE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	—	Intg	I/O	—	A	--	Logical unit no. of user's terminal
MODE	—	Intg	I	—	A	PROMPT	Flag for PROMPT option
INTBUF	—	Intg	I	--	A	—	Interface table header buffer
MRBUFF	--	Intg	I	--	A	--	Manager request buffer
ICR	—	Intg	I	--	A	PROCON(1)	Cartridge reference parameter
TYPE	--	Intg	I	—	A	TYPE	Type of phase table
DD	—	Real	I	—	A	—	Real DD array
IDD	--	Intg	I	—	A	--	Integer equivalent for DD
IERROR	—	Intg	I/O	—	A	—	Subroutine error flag
STRING	—	12CH	I	—	A	--	Character data for VECTOR NO.: prompt
INDEX	—	Intg	I	—	A	INDEX	Start vector number in display
NUMBER	--	Intg	I	—	A	NUMBER	Number of vectors to be displayed
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Continued

Routine DRDE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
NAME	—	6CH	I	--	IT	DRDENM	Name of DRDE file
PTYPE	—	Intg	I	—	IT	--	Data type code for DRDENM
SIZE	—	Intg	I	--	IT	--	Number of blocks in DRDE file
IDIM	--	Intg	I	—	IT	--	Maximum record size
DSPTYP	--	Intg	I	--	IT	—	RTE file manager file type
IDCB	—	Intg	I/O	—	—	--	Data control block array
IERR	—	Intg	I	--	—	--	FMP error code
IOPTN	--	Intg	O	—	—	--	Open options for FMP call, OPEN
ISECU	—	Intg	O	--	—	--	Security code for file to be opened
IBUF	--	Intg	I	—	—	--	User's buffer for FMP calls
IL	—	Intg	O	--	—	--	No. of words to be read per READF call
LEN	—	Intg	I	—	—	--	Actual no. of words read
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Concluded

Routine DRDE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
RECNO	—	Intg	O	--	—	--	Actual record number
DATA	--	Intg	I	—	—	--	User's response to VECTOR NO.: prompt

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

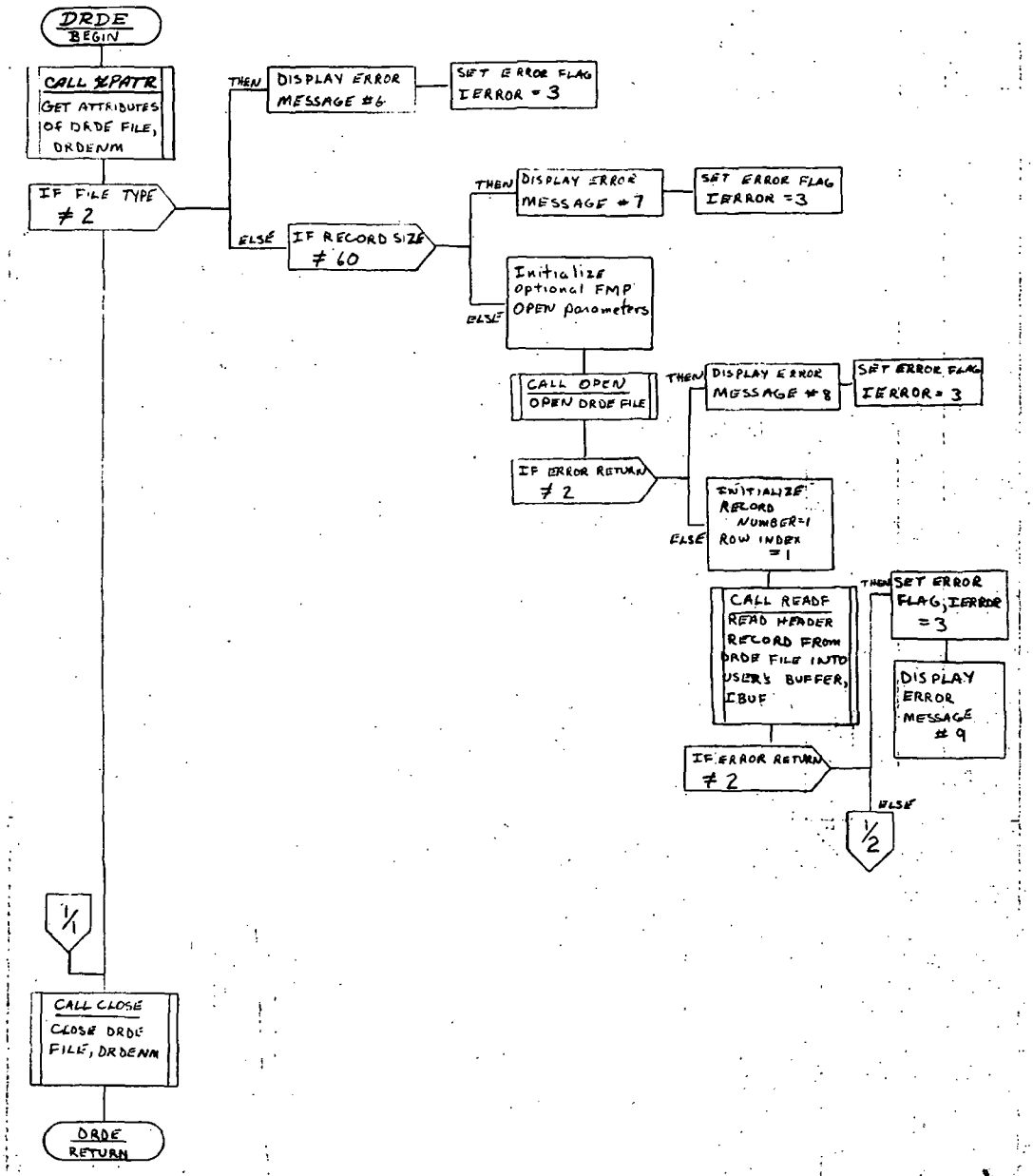


Figure 5.2-1.- DRDE functional logic flow.

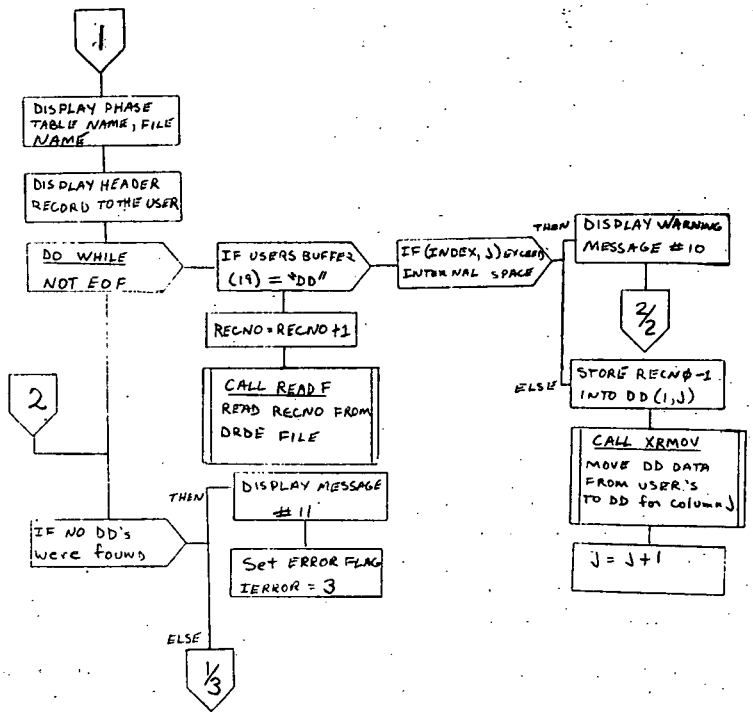


Figure 5.2-1.- Continued.

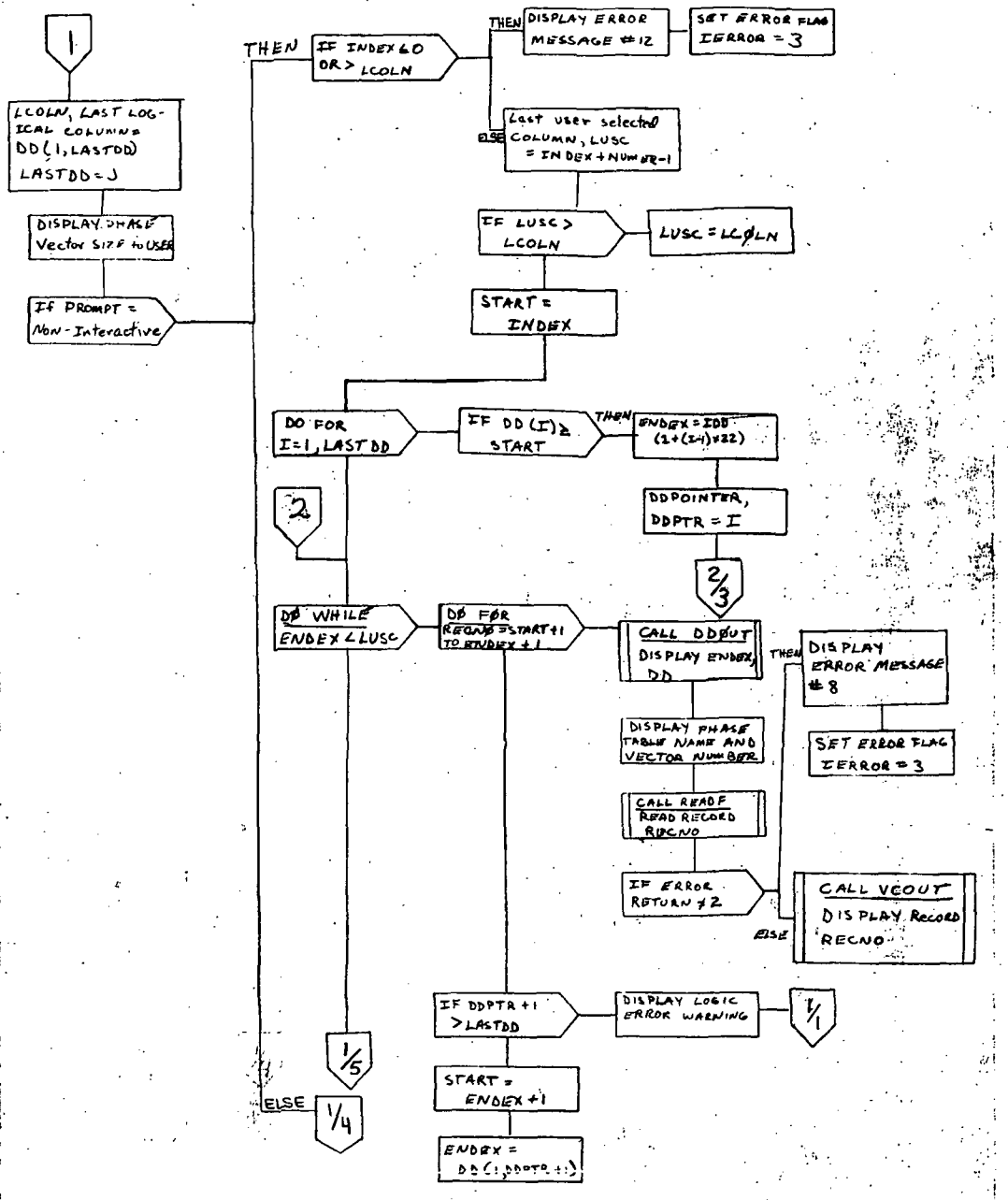


Figure 5.2-1.- Continued.

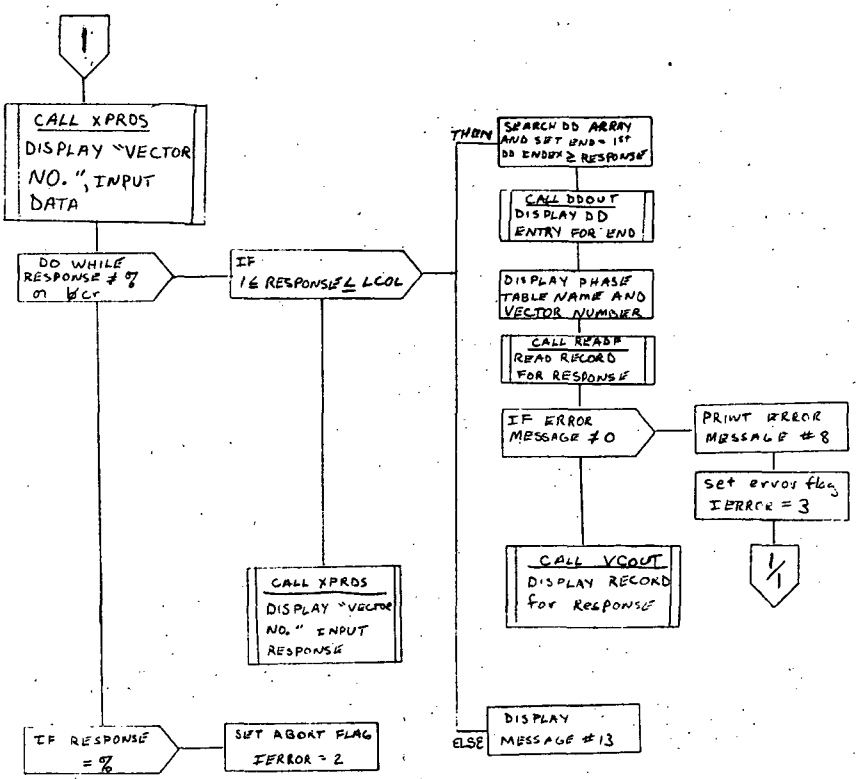


Figure 5.2-1.- Continued.

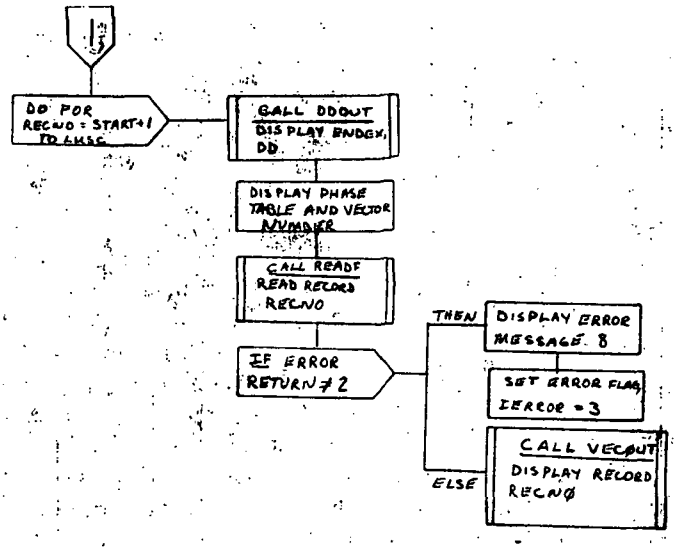


Figure 5.2-1.- Concluded.

5.3 ROUTINE NAME - DE

5.3.1 Purpose

The data element (DE) subroutine reads the user-selected data element from the AWA. Using display parameters from the interface table, or prompting the user for display information, DE calls the appropriate subroutines to display data definition information and state and propagation vectors from the data element.

5.3.2 Functional Description

The routine DE calls the FDS utility routine XPATR to obtain the attributes of the data element phase table, DENM. If the user-supplied literal data to the DENM parameter, execution of the processor is terminated with the error code set to -32768. If the user's phase table is not typed real, execution of the processor is terminated with the error code set to -32768. If the phase table is not a (30,N) array, execution of the processor is terminated with the error code set to -32768. If the user-supplied subscripts are not on a phase table column boundary, execution of the processor is terminated with the error code set to -32768. The size of the phase table is recomputed to equal the size returned from the XPATR minus the displacement resulting from user-supplied subscripts, if any. The entire DE phase table is read from the AWA via an XPGTI.

A search of the DE phase table is made to find and store all occurrences of data definition information. If none are found, execution of the processor is terminated with the error code set to -32768. The last logical column in the phase table is set to the column number of the last DD entry in the phase table, or the last value stored in the DD array, and the logical size of the phase table is then displayed to the user. A test is made to determine whether the user is in the interactive mode or the noninteractive mode. If the user selected the noninteractive mode, the interface table parameters INDEX and NUMBER determine what vectors are to be displayed. If INDEX is negative, or greater than the last logical column in the DD array, execution of the processor is terminated with the error code set to -32768. The last user-selected column in the phase table is set to the sum of INDEX + column number - 1. If, however, this value exceeds the last logical column, the last user-selected column is set equal to the last logical column. The start index for displaying the phase table is set to INDEX. The ending index is set to the first DD column number that is greater than the starting index column number. The appropriate DD entry is displayed on the terminal screen via a subroutine call to DDOUT. Vector I in the phase table is displayed via a subroutine call to VCOUT. This ends the noninteractive mode.

If the user selected the interactive mode, a call is made to the FDS utility routine XPRDS to display the prompt "VECTOR NO." and wait for the user's response. If data are between one and the last logical column in the data element phase table, a search of the DD array is made and the ending index is set equal to the first DD index greater than data. The appropriate DD entry is displayed on the terminal screen via a call to the DDOUT routine. The DE phase table entry

corresponding to data is displayed on the terminal screen via a call to the display routine VCOUT. If data were not between one and the last logical column, a warning message is displayed on the terminal screen. Another call is made to the XPRDS routine for "VECTOR NO.:" entry. If either a % or a blank(s) and carriage return is entered, the processor's execution is terminated with the error code set to 8 if %, or 0 if blank(s) and carriage return.

5.3.3 Assumptions and Limitations

See Assumptions and Limitations in section 5.1.3.

5.3.4 Method

See Functional Description in section 5.3.2.

5.3.5 Routine Input/Output Variables

The input/output variables for the DE routine are presented in table 5.3-I.

5.3.6 Functional Logic Flow

The functional logic flow for the DE routine is presented in figure 5.3-1.

5.3.7 Diagnostics and Debug

None.

5.3.8 Special Comments

None.

5.3.9 References

None.

TABLE 5.3-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine DE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	--	*Intg*	*I/O*	--	A	--	*Logical unit no. of user's terminal*
MODE	--	Intg	I	--	A	PROMPT	Flag for PROMPT option
INTBUF	--	Intg	I	--	A	--	Interface table header buffer
MRBUFF	--	Intg	I	--	A	--	Manager request buffer
TYPE	--	Intg	I	--	A	TYPE	Type of phase table
DENM	--	Real	I	--	A	--	Real array for DE storage
IDENM	--	Intg	I	--	A	--	Integer equivalent for DENM
IERROR	--	Intg	I/O	--	A	--	Subroutine error flag
STRING	--	12CH	I	--	A	--	Char. data for VECTOR NO.: prompt
INDEX	--	Intg	I	--	A	INDEX	Start vector no. in display
NUMBER	--	Intg	I	--	A	NUMBER	No. of vectors to be displayed
NAME	--	6CH	I	--	IT	DENM	Name of data element phase table

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.3-I.- Concluded

Routine DE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
PTYPE	--	Intg	I	--	IT	--	Data type code for DE
SIZE	--	Intg	I	--	IT	--	Size of DE phase table
IDIM	--	Intg	I	--	IT	--	I dimension of DE phase table
DSPTYP	--	Intg	I	--	IT	--	The phase table's displacement into the DE
DATA	--	Intg	I	--	--	--	User's response to VECTOR NO.: prompt
RETC	--	Intg	I	--	--	--	Return code from XPRDS call
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

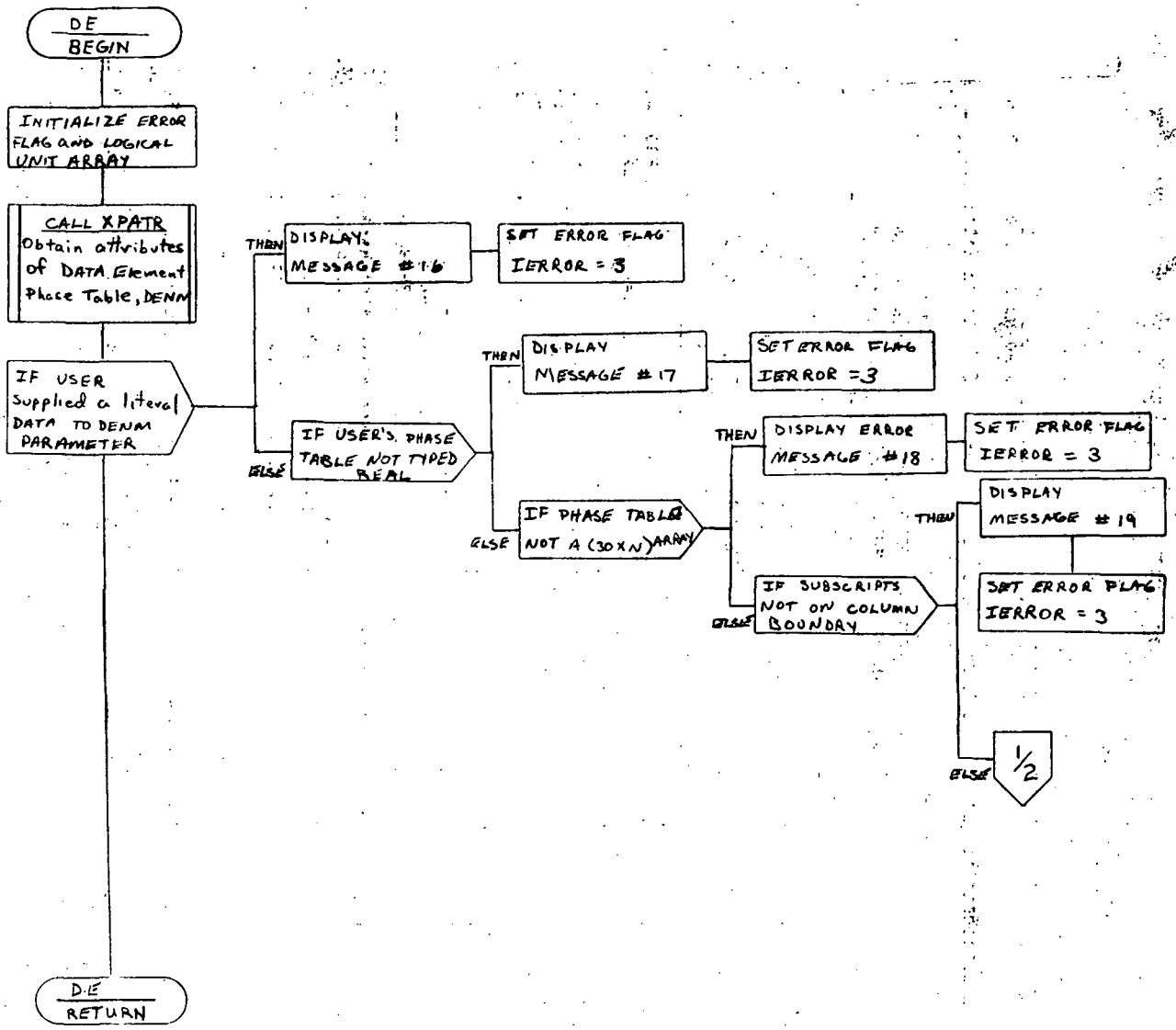


Figure 5.3-1.- DE functional logic flow.

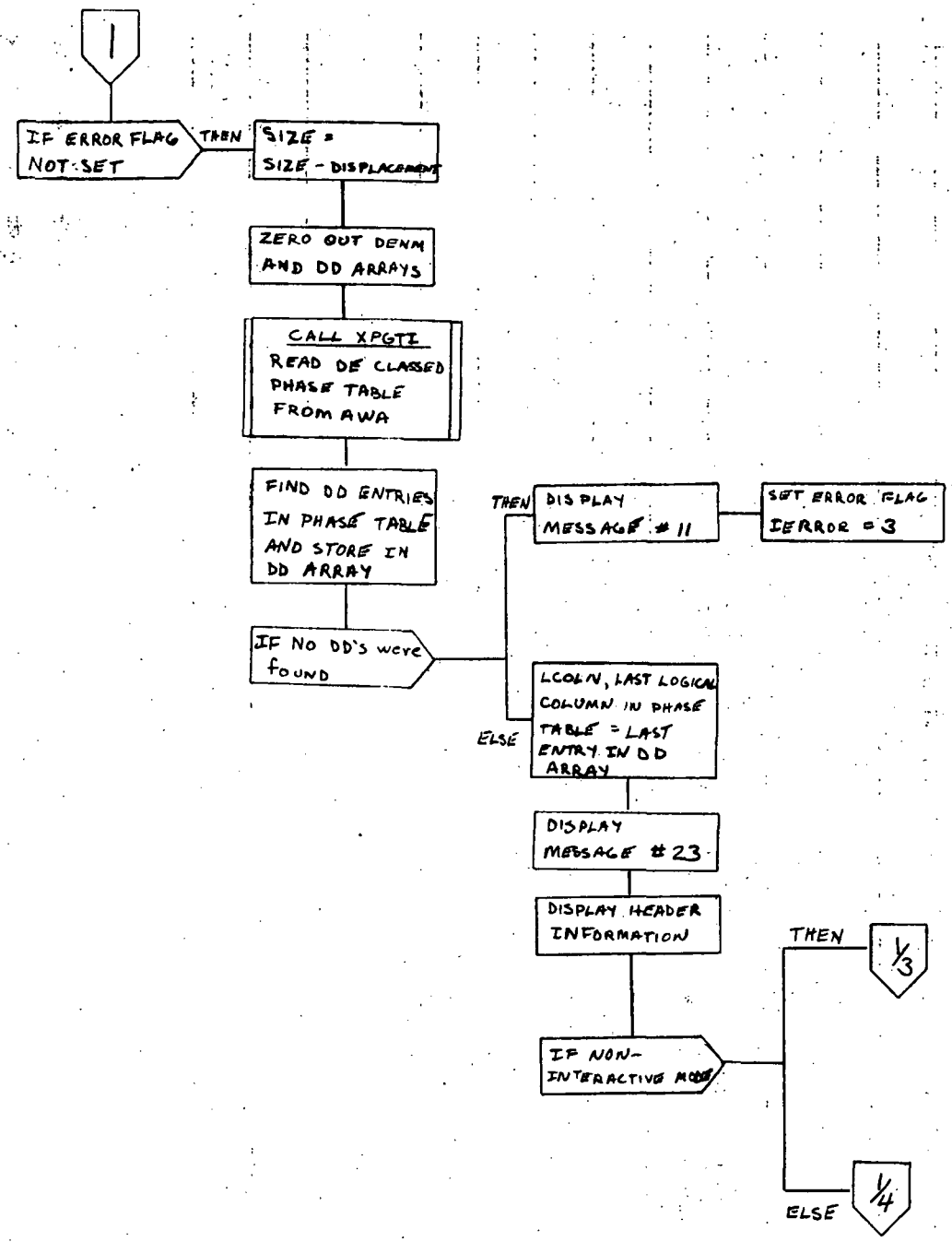


Figure 5.3-1.- Continued.

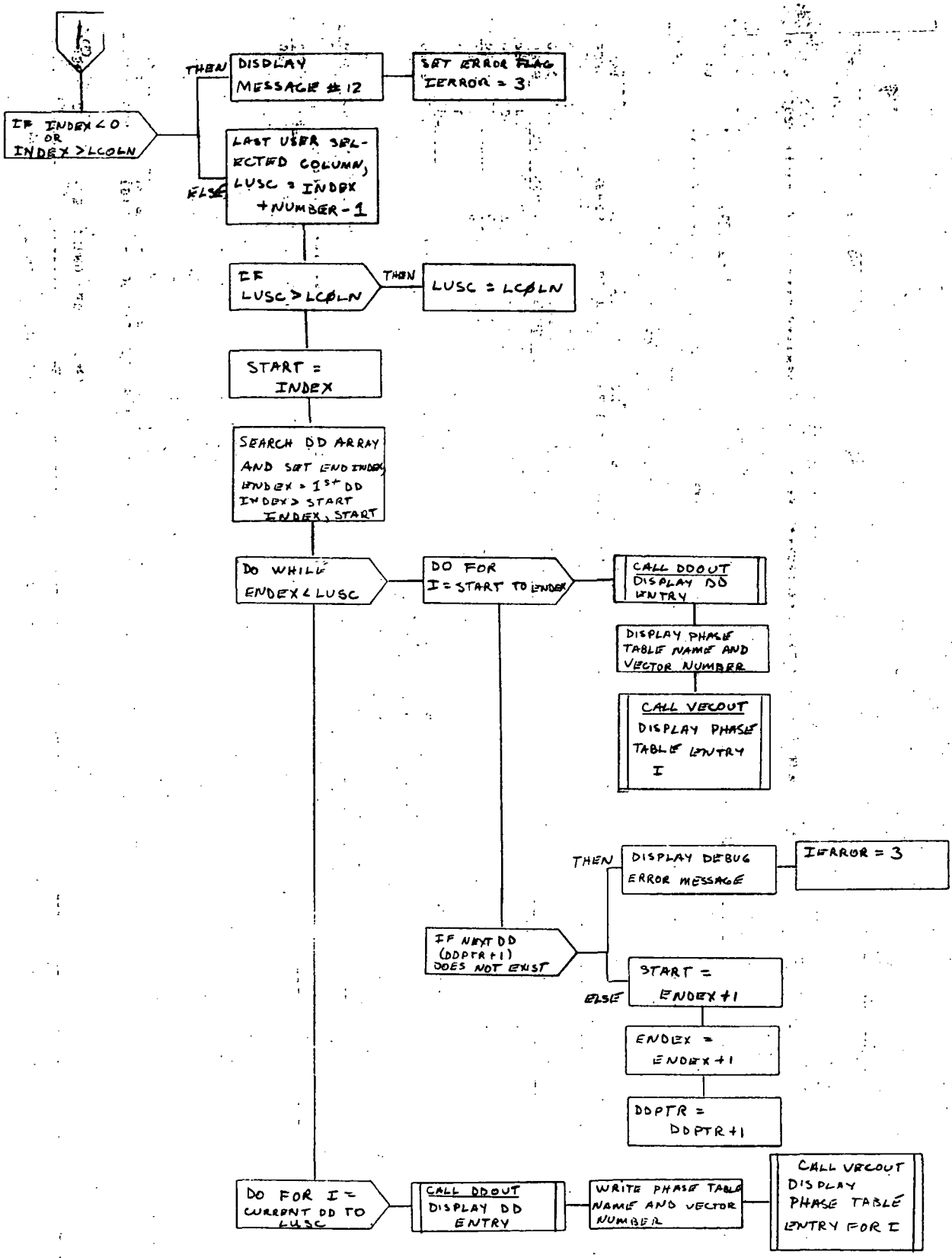


Figure 5.3-1.- Continued.

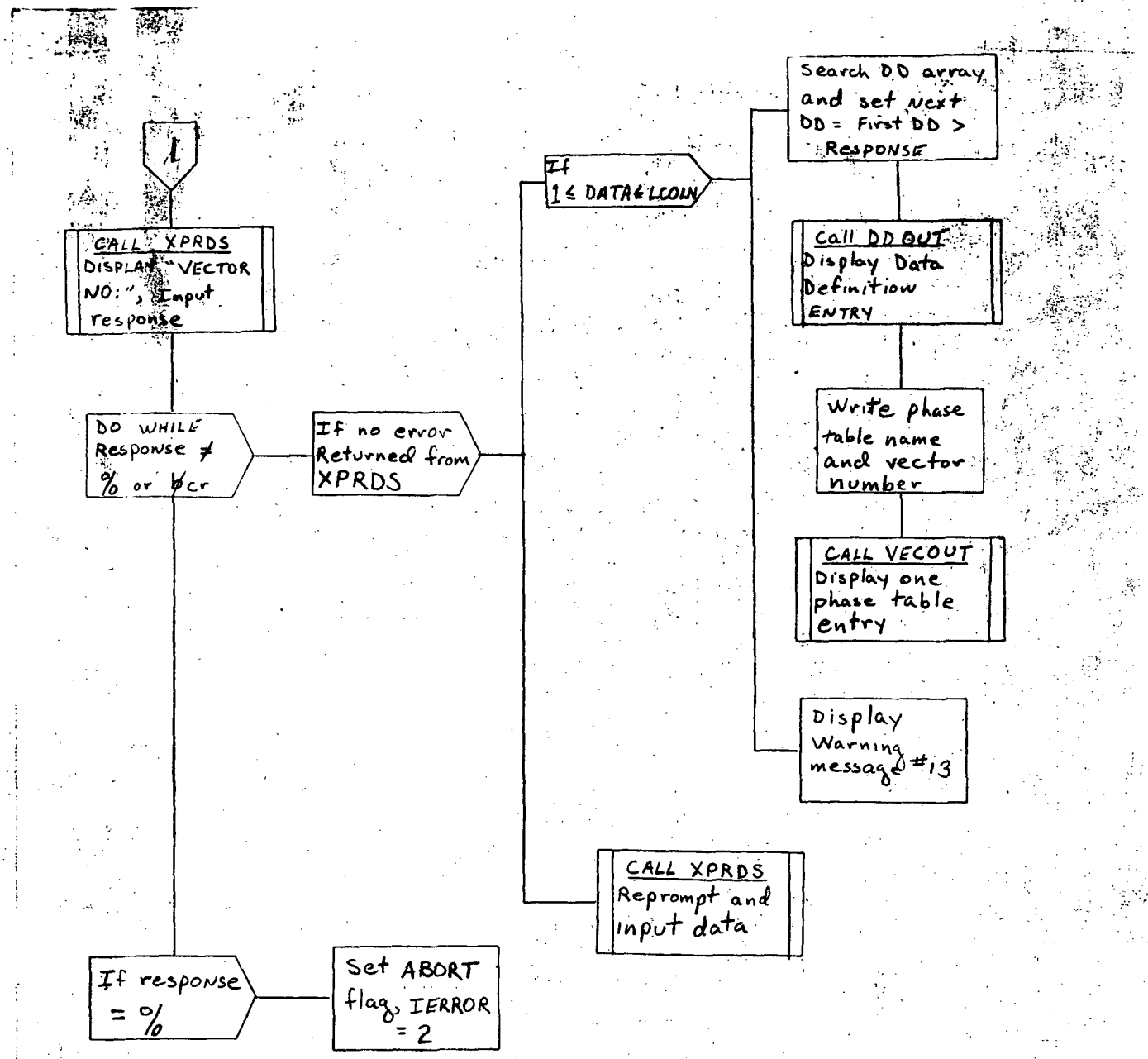


Figure 5.3-1.- Concluded.

5.4 ROUTINE NAME - DDOUT

5.4.1 Purpose

The data definition output (DDOUT) subroutine displays a data definition entry from any phase table in the following format:

BASE TIME = YEAR YYYY, DAY DDD, HOUR ±X.XXXXXXE±XX

UNITS = AAAA, DDDD, TTTT, VVVV, MMMM, FFFF, LLLL

5.4.2 Functional Description

The routine DDOUT calls the FDS utility routine XRMOV to move data internally from the DARRAY to the BASTM and UNIT arrays so that the data can be equivalenced to internal variables. The base time information is displayed from the BASTM array and the units information is displayed from the UNITS array.

5.4.3 Assumptions and Limitations

None.

5.4.4 Method

See Functional Description in section 5.4.2.

5.4.5 Routine Input/Output Variables

The input/output variables for the DDOUT routine are presented in table 5.4-I.

5.4.6 Functional Logic Flow

The functional logic flow for the DDOUT routine is presented in figure 5.4-1.

5.4.7 Diagnostics and Debug

None.

5.4.8 Special Comments

None.

5.4.9 References

None.

TABLE 5.4-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine DDOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	—	Intg	I	--	A	--	Logical unit no. of user's terminal
DARRAY	—	Real	I	—	A	--	DD array
MODE	—	Intg	I	--	A	PROMPT	Flag for PROMPT option
BASTM	—	Real	I/O	—	—	--	Array containing base time
UNITS	—	Intg	I/O	--	—	--	Array containing units
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

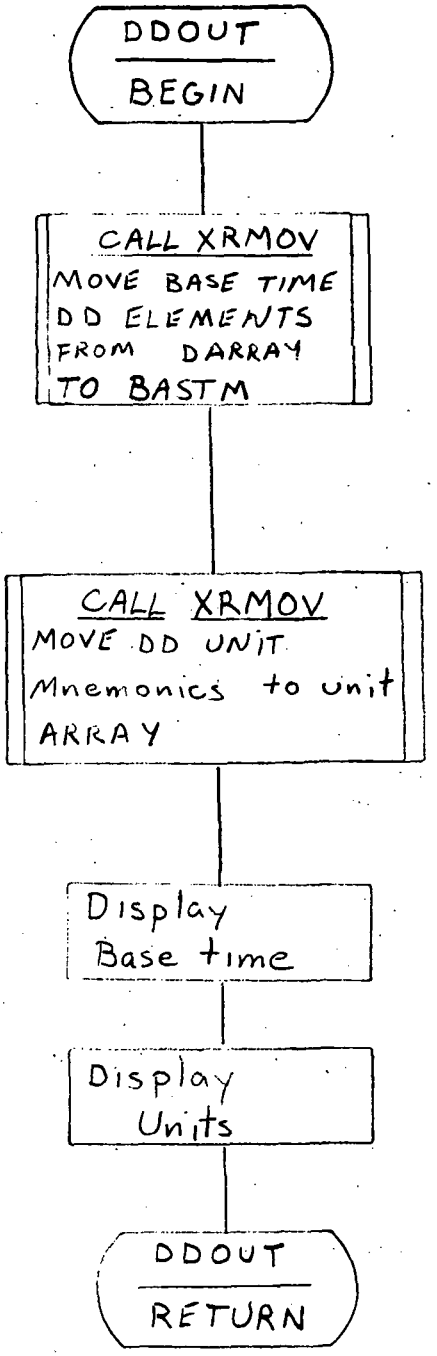


Figure 5.4-1.- DDOUT functional logic flow.

5.5 ROUTINE NAME - VCOUT

5.5.1 Purpose

The vector output (VCOUT) subroutine displays a state vector and a related propagation vector from a phase table in a manner presented in table 4-V(a) through (h), beginning with

PHASE TABLE PTPTPT VECTOR VVVVV

5.5.2 Functional Description

The routine VCOUT checks phase table type for position/velocity. If the type is not position/velocity but an attitude or mass properties, no display is generated and execution of the processor is terminated. For position/velocity phase tables the state vector is checked for zero. If zero, a message is displayed on the terminal screen and control is returned to the calling program. The FDS functional utility routine SPSV is called to display the state vector part of the phase table entry that contains both the state and propagation vectors. If an error is returned from SPSV, the error flag is incremented by four. If a DD entry is contained in the lower half of the phase table entry instead of propagation information, control is returned to the calling program. The coasting or powered flight simulator code number is converted from a real to an integer number and the processor and propagation codes are extracted. The phase code and the drag option are integerized. The processor code is checked for validity. If it is not an integer value from 1 to 22, the error flag is incremented by 8, a message is displayed to the user, and the internal variable IXX is set to zero. IXX is the index into the data array containing the processor codes. The propagation code is checked for validity. If it is not an integer value from 1 to 12, the error flag is incremented by 16, a message is displayed to the user, and the internal variable IYY is set to zero. IYY is the index into the data array containing the propagation codes. The drag option is checked for validity. If it is not 1 or 2, the error flag is incremented by 32, a message is displayed to the user, and IDRAG is set to zero. IDRAG is the index into the data array containing the drag options. The phase code is checked for validity. If it is not equal to 1 or 2, the error flag is incremented by 64 and control is returned to the calling program. If the phase code is equal to one, then the data indicates that the display will be for a coasting flight and the appropriate data are displayed to the user via the terminal screen. If the phase code is equal to two, the display is a powered flight. The guidance/steering option code is integerized, as well as the propulsion system code. If the guidance/steering option is not 1, 2, or 3, it is invalid; the error flag is incremented by 128, and IGUID is set to zero. IGUID is the index of the data array containing the guidance/steering options. If the propulsion system code is not between 1 and 5, it is invalid; the error flag is incremented by 256, a message is displayed to the user, and the internal variable IENG is set to zero. IENG is the index into the data array containing the propulsion system codes. Next, the coordinate system flag is checked for thrust alinement. If it is negative, the error flag is incremented by 512, an error

message is displayed to the user, and the internal variable JP1 is set to zero. JP1 is J + 1 where J is the index into the data array that contains the reference axis code for thrust alinement. The coordinate system flag for thrust alinement is JK where J is the reference axis designator and K is the type of element designator. If K is not equal to zero, the error flag is incremented by 512, an error message is displayed on the terminal screen, and JP1 is set to zero. Otherwise, a check is made to determine if J is between 0 and 4. If it is, JP1 is set to J + 1. If not, an error flag is incremented by 512, an error message is displayed to the user, and JP1 is set to zero.

This is a powered flight; therefore, the appropriate powered flight data are displayed to the user via the terminal screen.

5.5.3 Assumptions and Limitations

None.

5.5.4 Method

See Functional Description in section 5.5.2.

5.5.5 Routine Input/Output Variables

The input/output variables for the VCOUT routine are presented in table 5.5-I.

5.5.6 Functional Logic Flow

The functional logic flow for the VCOUT routine is presented in figure 5.5-1.

5.5.7 Diagnostics and Debug

None.

5.5.8 Special Comments

None.

5.5.9 References

See references in section 5.1.9.

TABLE 5.5-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine VCOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	—	Intg	I	--	A	--	Logical unit number of user's terminal
VECTOR	--	Real	I/O	--	A	--	Phase table entry to be displayed
TYPE	—	Intg	I	—	A	TYPE	Type of phase table
ERROR	--	Intg	I/O	—	A	--	Subroutine error flag
NAME	—	6CH	I/O	—	A	DENM/DRDENM	Name of phase table
INO	--	Intg	I/O	—	A	--	Vector number
MODE	--	Intg	I	--	A	PROMPT	Flag for PROMPT option
LUM	--	Intg	O	—	—	--	Logical unit no. of print device
DUM	—	18CH	O	--	—	--	Dummy char. data sent to SPSV
IERR	--	Intg	I	—	—	--	Error flag returned from SPSV
DRAG	—	24CH	O	--	—	--	Character data for drag options
PROCES	--	150CH	O	—	—	--	Character data for processor codes

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.5-I.- Concluded

Routine VCOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
GUID	—	128CH	O	--	—	--	Character data for guidance options
ENGIN	—	60CH	O	—	—	--	Character data for propulsion system codes
REFAX	—	24CH	O	—	—	--	Character data for reference axis
INTEG	—	156CH	O	--	—	--	Character data for propagation codes

NOTES:

TYPE

Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

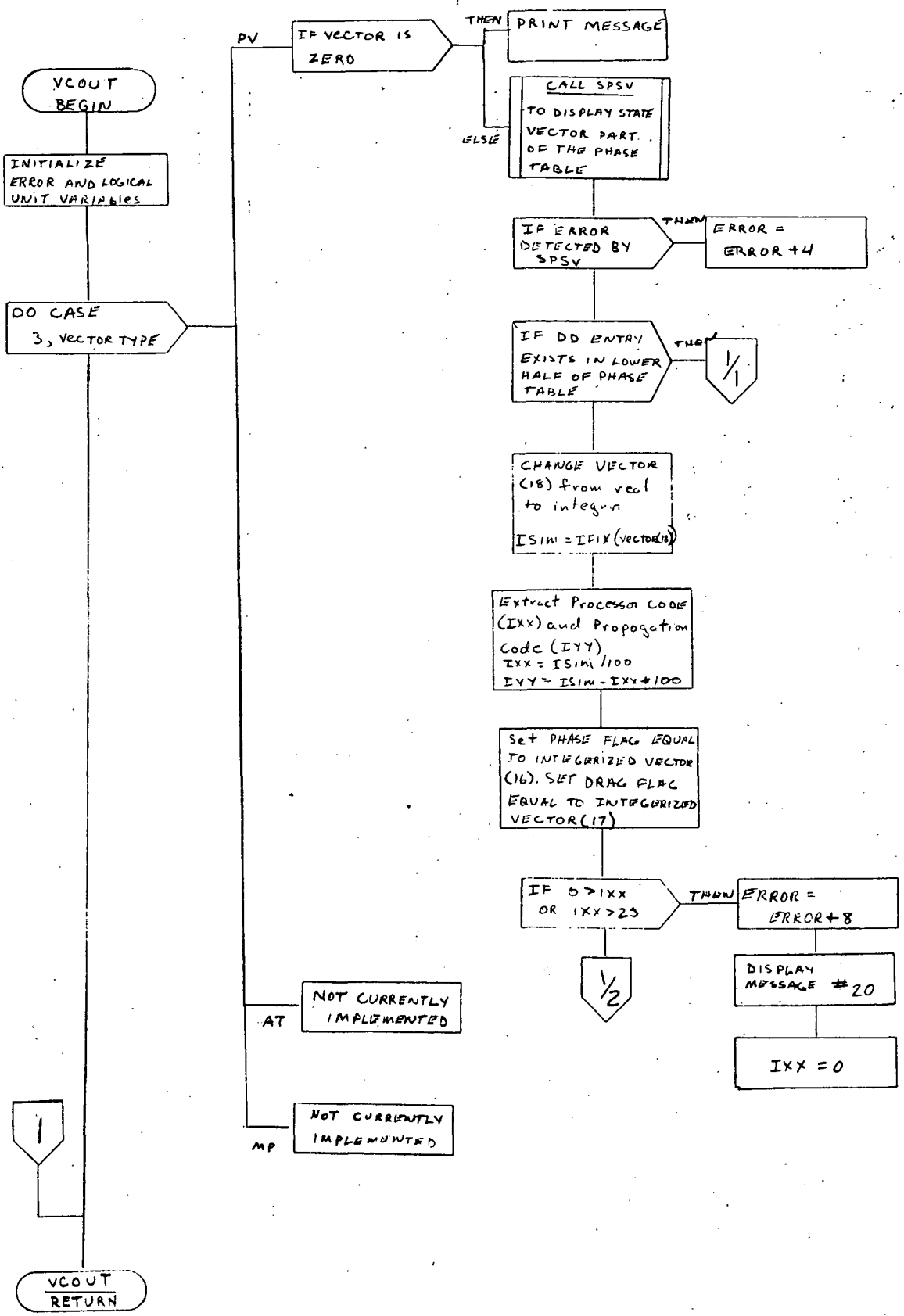


Figure 5.5-1.- VCOUT functional logic flow.

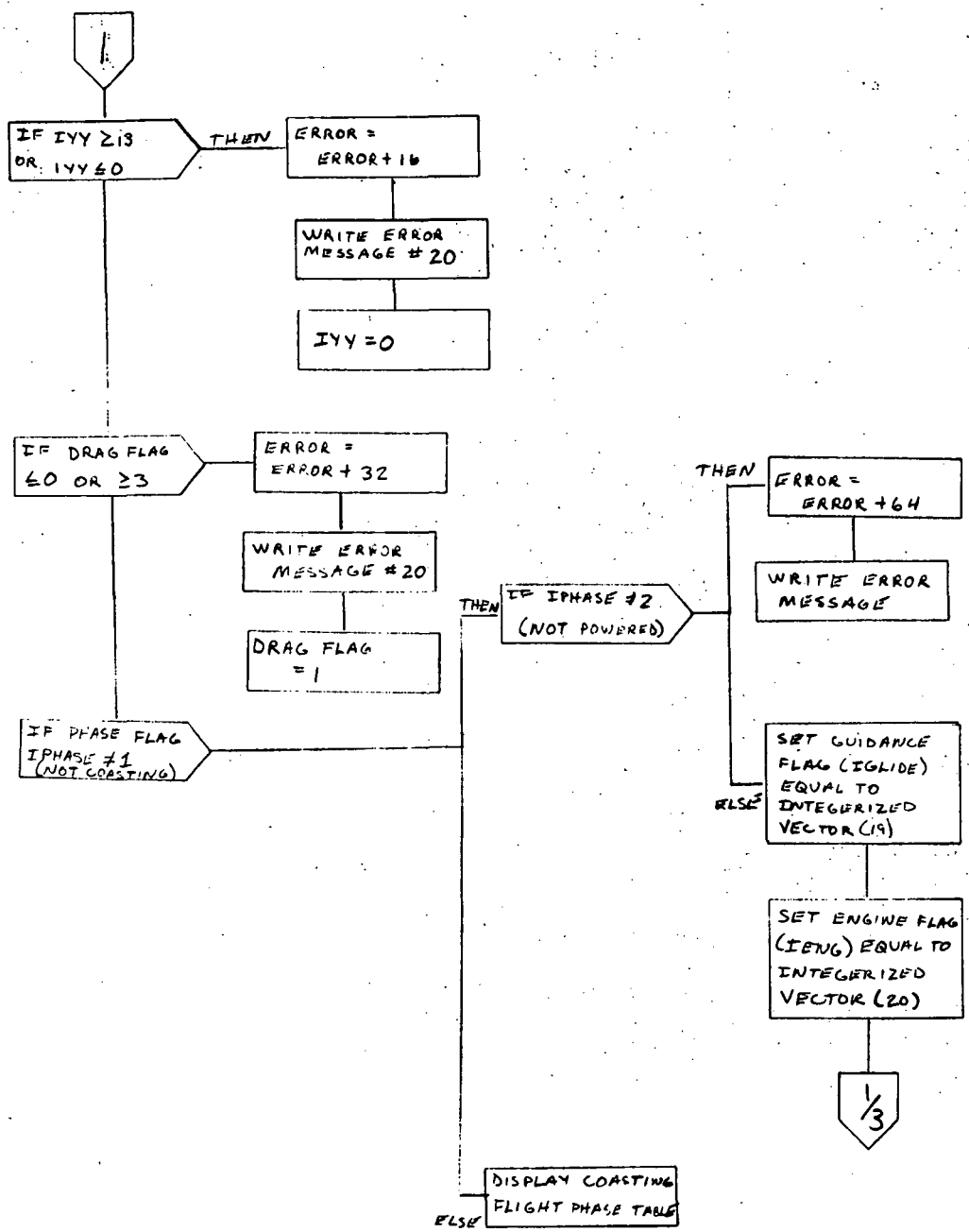


Figure 5.5-1.- Continued.

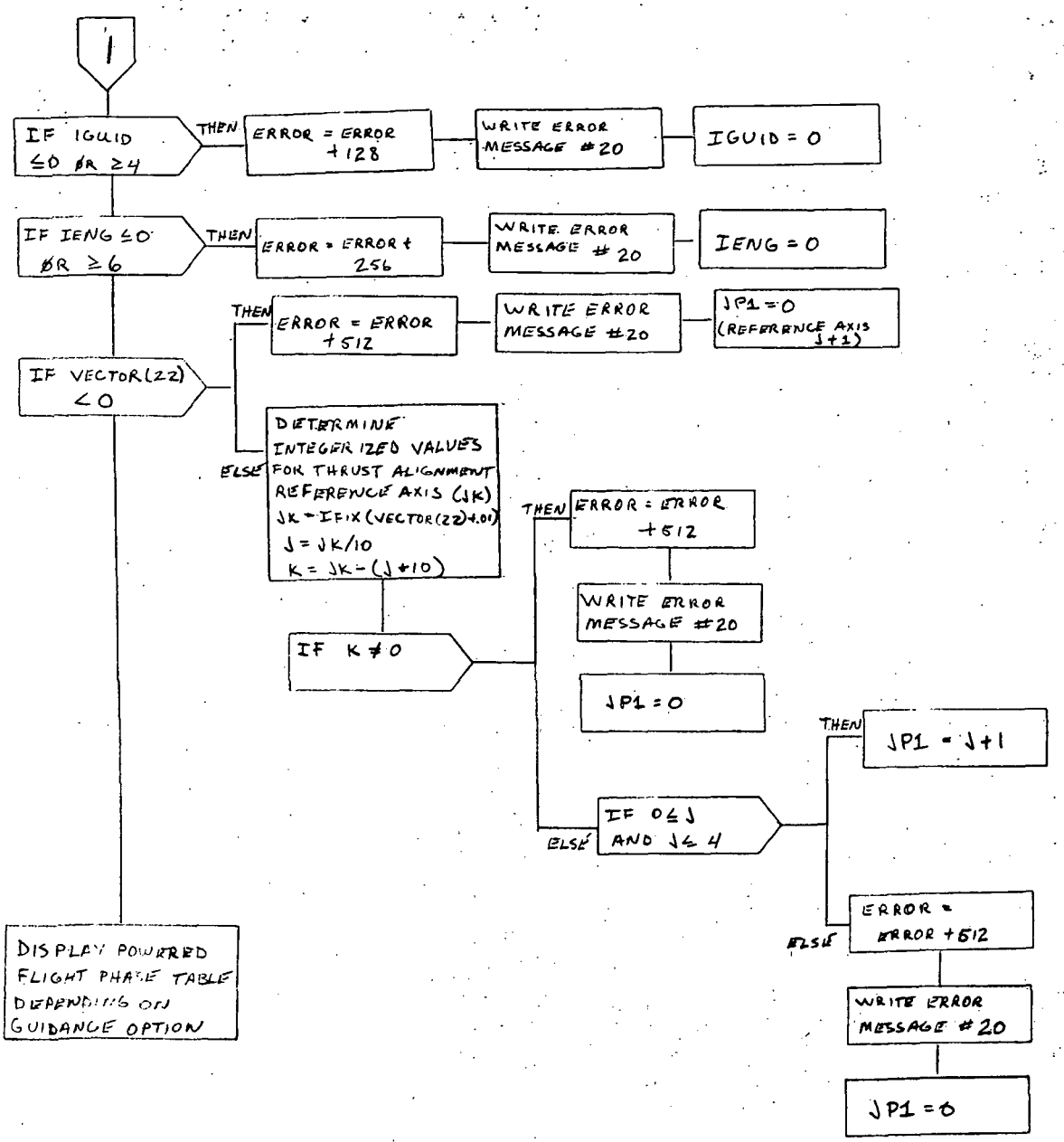


Figure 5.5-1.- Concluded.

QUICK INVESTIGATION OF CONSUMABLES KITS PROCESSOR (QUIKU)

1.0 PURPOSE

The QUIKU applications processor provides a fast and efficient study for several of the nonpropulsive consumables and systems constraints. Specifically, the nonpropulsive consumables considered are oxygen, hydrogen, nitrogen, ammonia, waste water, and potable water. Constraint data are provided for the Environmental Control System (ECS) and the Electrical Power System (EPS).

2.0 FUNCTIONAL DESCRIPTION

After receiving the user-provided input from the interface table, the QUIKU processor will convert the data from the disk files into the proper form for internal use. The processor then analyzes a mission activities time line for the nonpropulsive consumables (oxygen, hydrogen, nitrogen, ammonia, waste water, and potable water) using the activity load block data from the Activity Block Data processor (ABLKS) and the systems characteristics from the Systems Characteristics Update processor (SYDUS). The processor then builds a load profile for each point on the mission activities time line. This load profile is then evaluated for electrical power system (EPS) requirements using algorithms developed in the Consumables Analysis for Shuttle Kits (CASKU) processor analyses. The heat load profile is adjusted to the power requirements determined and evaluated for the cooling requirements using algorithms developed from detailed analysis of the Orbiter's cooling system. Concurrent with this, the processor considers and evaluates the cabin atmospheric requirements. The processor writes an output record each time a system constraint is discovered and at each point on the mission activities time line.

3.0 ASSUMPTIONS AND LIMITATIONS

a. It is assumed that the user desires a fast analysis of the time line that was developed with the TMLNU processor and does not need or desire detailed printouts of the EPS or ECS solutions at each point on the time line.

b. The user is assumed to be familiar with the files used to support the execution of the processor.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The definition of the processor interface table parameters are provided in table 4-I.

   Specific constants required by the processor are maintained through the default values stored in the processor constants array, PROCON. Specifically, these constants are the cartridge reference numbers of the system data files, the DRDE data files, and the number of blocks required for the output DRDE file.

The user specifies the name of the time-line file by entering the four-character name in the TIMFIL parameter. The output file name is specified by entering the four-character name in the DATAFL parameter.

File names for the ORBITER Activity Data file, PAYLOAD Activity Data file, and the SYSTEMS CHARACTERISTIC Data file are entered through the ORBACT, PAYACT, and SYSFIL parameters, respectively. If a payload activity data file is not used the user must enter blanks in the name field.

Through the parameter CONTNU the user specifies the desire to continue processing should the internal arrays containing output data become full.

b. Processor interface table data array definitions - The definitions of the data arrays for the QUIKU processor are provided in table 4-II.

c. Processor interface table data file definitions - The definition of the input and output DRDE files used by the QUIKU processor are provided in table 4-III.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definition table - None.

f. Processor message table - The messages displayed by the QUIKU processor in the event of errors are provided in table 4-IV.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-V.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR QUIKU

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
TIMFIL	Disk	Intg	I	-	-		Name of the users DRDE time-line file
DATAFL	Disk	Intg	O	-	-		User selected name of the output DRDE file
ORBACT	AWA	6CH	I	3	1		Name of the ORBITER Activity Data file
PAYACT	AWA	6CH	I	3	1		Name of the PAYLOAD Activity Data file
SYSFIL	AWA	6CH	I	3	1		Name of the SYSTEMS CHARACTERISTICS Data file
PROCON	AWA	Intg	I	3	3	16,20,400	Cartridge reference number for system and DRDE files, output file block size
CONTNU	AWA	2CH	I	1	1		Users desire to continue processing if arrays become full "YE" or "NO"

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR QUIKU

Array name	Index location	Default value	Definition
PROCON	1	16	Catridge reference number for data files
	2	20	Cartridge reference number for DRDE files
	3	400	Blocks to allocate for the output DRDE file

TABLE 4-III.- INTERFACE TABLE DATA FILE DEFINITIONS

(a) Time-line file

PROCESSOR QUIKU

DRDE DATA FILE /XXXXC

Record number	Integer word allocations	Content and definition
1	1-3	"TMLNU" - Name of the FDS-1 processor that created the file
	4-6	TIMFIL - Interface table variable name through which the file was created
	7-9	"TMLNU" - Name of the FDS-1 processor that last updated the file
	10-12	TIMFIL - Interface table variable name through which the file was changed
2-n	1	MACT - Activity number
	2-3	STIME - Activity start time
	4-5	ETIME - Activity stop time

TABLE 4-III.- Continued

(b) Output data file

PROCESSOR QUIKU

DRDE DATA FILE /XXXXC

Record number	Integer word allocations	Content and definition
1	1-3	"QUIKU" - Name of the FDS-1 processor that created the file
	4-6	DATAFL - Interface table variable name through which the file was created
	7-9	"QUIKU" - Name of the FDS-1 processor that last updated the file
	10-12	DATAFL - Interface table variable name through which the file was changed
2	1-5	JTIME - Year, Julian calendar day, hour, minute, and second that the file was created
	6-8	TIMFIL - Name of the users time-line file
	9-11	ORBACT - Name of the ORBITER Activity Data file
	12-14	PAYACT - Name of the PAYLOAD Activity Data file
	15-17	SYSFIL - Name of the SYSTEMS CHARACTERISTICS Data file
	18-20	PROGNM - Name of the processor
3	1-2	RTIME - The time that a system constraint limit was surpassed
	3-4	KIND - Name of the system that caused the constraint
	5	KITS - Number of hydrogen kits required
	6-7	AQTY - Value that caused the system constraint
	8-9	QTY - The system limit that was surpassed
4	1-2	RTIME - The time that a system constraint limit was surpassed
	3-4	KIND - Name of the system that caused the constraint
	5	I - A value of zero

TABLE 4-III.- Continued

(b) Continued

PROCESSOR QUIKU

DRDE DATA FILE /XXXXC

Record number	Integer word allocations	Content and definition
	6-7	VAL - Value that caused the system constraint
	8-9	VALMT - The system limit that was surpassed
5-n	1-2	OUT(1) - Time from user's time line for the event set equal to TIME
	3-4	OUT(2) - Oxygen quantity at TIME
	5-6	OUT(3) - Hydrogen quantity at TIME
	7-8	OUT(4) - Nitrogen quantity at TIME
	9-10	OUT(5) - Ammonia quantity at TIME
	11-12	OUT(6) - Waste water quantity at TIME
	13-14	OUT(7) - Potable water quantity at TIME
	15-20	OUT(8-10) - Power values at TIME
	21-22	OUT(11) - Not used
	23-28	OUT(12-14) - Source voltages at TIME
	29-34	OUT(15-17) - Not used
	35-40	OUT(18-20) - Source currents at TIME
	41-46	OUT(21-23) - Not used
	47-48	OUT(24) - Heat load at TIME
	49-50	OUT(25) - Radiator heat load at TIME
	51-52	OUT(26) - Not used
	53-54	OUT(27) - Ascent cooler heat load at TIME

TABLE 4-III.- Continued

(b) Concluded

PROCESSOR QUIKU

DRDE DATA FILE /XXXXC

Record number	Integer word allocations	Content and definition
	55-56	OUT(28) - Ammonia boiler heat load at TIME
	57-76	OUT(29-38) - Not used
	77-78	OUT(39) - Statistical peak power at TIME
	79-80	OUT(40) - Statistical minimum power at TIME
	81-82	OUT(41) - System thermal capacity at TIME
	83-84	OUT(42) - System thermal margin at TIME

TABLE 4-IV.- PROCESSOR MESSAGE TABLE

PROCESSOR *QUIKU*

MSG no.	Message ID block	Message text block and explanation
1	*QUIKU*	ERROR XXXX CANNOT OPEN aaaaaa Meaning: The file named could not be opened and the error number was as shown Severity: Terminal Action required by user: Determine cause of the error and reinitiate processor
2	*QUIKU*	OUTPUT FILE PROB. IN PLDAT XXXX Meaning: An error has occurred while attempting to write to the output file; error number is XXXX Severity: Terminal Action required by user: Determine the cause or the error and reinitiate processor
3	*QUIKU*	ACT. FILE PROB. XXXX Meaning: An error has occurred while attempting to read the activity file; error number is XXXX Severity: Terminal Action required by user: Determine the cause of the error and reinitiate processor
4	*QUIKU*	TIMLN FILE PROB. XXXX Meaning: An error has occurred while attempting to read the TIMLN file; error number is XXXX Severity: Terminal Action required by user: Determine the cause of the error and reinitiate the processor
5	*QUIKU*	OUTPUT FILE PROB. IN TKLMT XXXX Meaning: An error has occurred while attempting to write to the output file; error number is XXXX Severity: Terminal Action required by user: Determine the cause of the error and reinitiate the processor
6	*QUIKU*	OUTPUT FILE PROB. IN EPLMT XXXX Meaning: An error has occurred while attempting to write to the output file; error number is XXXX Severity: Terminal Action required by user: Determine the cause of the error and reinitiate processor

TABLE 4-IV.- Concluded

PROCESSOR QUIKU

MSG no.	Message ID block	Message text block and explanation
7	*QUIKU*	COULDN'T OPEN XXXX Meaning: Could not open file whose name was just entered in response to a processor prompt XXXX = ERROR Severity: Warning/Terminal Action required by user: Verify spelling of file name. If in error, correct when reprompted. If correct, terminate processor and determine cause
8	*QUIKU*	COULDN'T FIND ACTIVITY XXXX Meaning: The activity number shown was not found in the data file Severity: Warning Action required by user: Verify activity number and modify TIMELINE file
9	*QUIKU*	TIMELINE TO LARGE FOR ARRAY Meaning: User has exceeded array dimensions with TIMELINE Severity: Terminal Action required by user: Reaccomplish time line

TABLE 4-V.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR QUIKU

Processor name	Processor abstract prompt (maximum 256 characters)
QUIKU	QUIKU provides a fast, efficient analysis for oxygen, hydrogen, nitrogen, ammonia, waste water, and potable water. System constraints are also provided.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
TIMFIL	Name of the TIMELINE file to be used
DATAFL	Name of the output data file
ORBACT	Name of the ORBITER ACTIVITY data file
PAYACT	Name of the PAYLOAD ACTIVITY data file
SYSFIL	Name of the SYSTEMS CHARACTERISTICS data file
PROCON	Processor constants for cartridge references and output data file block size
CONTNU	Does the user desire to continue processing the data even though the internal arrays are full? YE or NO.

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM QUIKU

5.1.1 Purpose

The routine QUIKU serves as the main program of the QUIKU processor. It provides the executive control for the execution of the processor by loading the segments when needed.

5.1.2 Functional Description

The main routine QUIKU obtains the RTE system parameters through a call to RMPAR. The logical unit number variable, LU, is then set to IPARM(1). The routine then brings into memory the preparation segment, QPRPU, through a call to the RTE service routine LDSEG.

Upon completion of the preparation segment, control returns and a test on the termination flag IPARM(1) is made. If no error condition exists, the processor then brings into memory the computational segment, QSEGU, through a call to the RTE service routine, LDSEG.

Upon completion of the computation segment, control returns and a test is made on the termination flag IPARM(1). If no error exists, the processor calls the FDS termination routine XPXIT and exits normally.

If an error condition exists, as indicated by IPARM(1) being equal to -32768, the processor calls the FDS termination routine XPXIT and exits abnormally.

5.1.3 Assumptions and Limitations

None.

5.1.4 Method

None.

5.1.5 Routine Input/Output Variables

The input/output variables for the main program, QUIKU, are presented in table 5.1-I.

5.1.6 Functional Logic Flow

A functional logic flow is not provided. The functional level PDL for QUIKU is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

None.

5.1.8 Special Comments

This F milestone will only contain documentation on routines that have been extensively modified.

5.1.9 References

None.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine QUIKU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	—	Intg	I/O	--	C	--	Logical unit number of user's terminal.
IPARM	—	Intg	I/O	—	C	--	Five-element system and FDS-1 parameter list.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

```
1*
1 BEGIN QUIKU
2     CALL RMPAR FOR SYSTEM PARAMETERS
2     SET LU NUMBER TO SYSTEM PARAMETER ONE
2     CALL LDSEG FOR PREPERATION SEGMENT
2     IF(IPARM ONE = ZERO)
2     THEN
3        BRANCH TO -LBL100-
2     ELSE
3        EXIT TO -EX8000-
2     ENDIF
2*
2 -LBL100-
2     CALL LDSEG FOR COMPUTATIONAL SEGMENT
2     IF(IPARM ONE = -32768)
2     THEN
3        EXIT TO -EX8000-
2     ELSE
3        EXIT TO -EX9000-
2     ENDIF
2*
2 -EX8000-
3     EXIT TO -EX9010-
2*
2 -EX9000-
3     SET IPARM ONE TO ZERO
2*
2 -EX9010-
3     CALL CLOSE FOR OUTPUT FILE
3     CALL CLOSE FOR ORBITER ACTIVITY DATA FILE
3     CALL CLOSE FOR PAYLOAD ACTIVITY DATA FILE
2     CALL XPXIT
1 END QUIKU
1*
```

Figure 5.1-1.- QUIKU functional level PDL.

5.2 ROUTINE NAME - QPRPU

5.2.1 Purpose

The quick preparation segment (QPRPU) retrieves the appropriate data from the disk files named in the interface table after having initialized the processor parameters. This routine processes the users' TIMELINE file into an array containing time and activity block data location. Other internal arrays are also filled with data from the disk files.

5.2.2 Functional Description

Initially, the variable that contains the number of entries in the time line array, IARAY, is set to zero. An FDS-1 input buffer, INTBUF(1), is also set to zero. A call is made to the FDS-1 utility routine XPGET to obtain interface table variables. Processor subroutine ACTFL is called to obtain ORBITER ACTIVITY file data. An error check is made on the return from ACTFL, and an error condition will cause an abnormal termination.

A test is made on the Hollerith array PAYACT to see if the PAYLOAD ACTIVITY file is used. If it is, a call is made to ACTFL to obtain PAYLOAD ACTIVITY file data.

A Hollerith array TFILNM is initialized to the TIMELINE file name from the common array TIMFIL. A call is made to the file manager service routine OPEN to open the TIMELINE file. If an error condition exists after the open call, all files are closed and an abnormal termination follows.

A loop follows in which a record is read from the TIMELINE file via the file manager service routine READF. Error conditions are checked for and, if detected, an error message is printed and an abnormal termination follows. A test is then made to determine whether this terminates the record sets. If not, it is determined whether the activity is for the Orbiter or a payload. An attempt is then made to locate the activity in the data base array. If it is not there, an explanatory message is displayed and control returns to the top of the loop. Otherwise, array pointers are incremented and tested for a valid activity. If the array pointer is greater than the number of activity blocks to be scanned, control returns to the top of the loop. If the activity block number read from the input time line does not equal the activity block number from the orbital or payload data base of that index, control returns to the top of the loop. Otherwise, the start time and the stop time into the array are set and the index incremented until the index pointer is greater than the number of activity blocks to be scanned, or until the activity block numbers match. Execution then returns to the top of the loop.

There are three exits from this record reading loop: an error condition in the file read, too many records in the TIMELINE file for the internal array, or an end of file (EOF) condition.

The error condition results in an error message being displayed and an abnormal termination.

If either the TIMELINE file was too big for the internal array, or an activity was not found, an appropriate message is printed. If the user specified "YE" to the Hollerith interface table variable CONTNU, the processor will continue on the data it has; otherwise, an abnormal termination will follow.

If the processor detects an EOF flag while reading the TIMELINE file (the first word of the input record set to 9999), the reading loop is exited normally. The last record values of the internal array are set to EOF values and the file is closed by the file manager service routine CLOSE.

The internal time line is now chronologically ordered by a processor subroutine ORDER. A time variable TIME is set to the beginning of the timeline.

The SYSTEMS data file is opened via file manager service routine OPEN. If an error condition is returned, an error message is printed and an abnormal termination follows. The file is then positioned via the file manager service routine POSNT. Again, the error condition is checked, and upon detection results in a message being printed and an abnormal termination.

The SYSTEMS data file is then read into the internal array RMISC via the file manager service routine READF. Any possible error condition is checked upon return of READF, and the error detection result is an error message being printed and an abnormal termination.

The values in RMISC are used to determine base quantities, kit quantities, and various other parameters that are significant in the analysis. Specifically, these are atmospheric leak rate of cabin (lb/hr/psi), metabolic rate of the Shuttle crew (SQMET), radiator cooling capacity (btu/hr), available potable water quantity, minimum allowable quantity of potable water, maximum allowable quantity of potable water, rate at which the water flows through the overboard vent lines, a flag that specifies whether the flash evaporators are to be used for the water dump, pressure of the oxygen in the cabin, pressure of nitrogen in the cabin, baseline usable tank capacities for each consumable, and the usable tank capacity per kit for each consumable. The SYSTEMS data file is then closed via a call to the file manager service routine CLOSE.

The oxygen atmospheric usage is determined as a factor of the leakage rate of the cabin, the metabolic rate of the crew, and the oxygen pressure in the cabin. The nitrogen atmospheric usage is then determined as a factor of the pressure of nitrogen in the cabin and the atmospheric leak rate of the cabin.

The clock time is determined for reference purposes through a call to the RTE executive.

An output data file is created by first establishing the file type, blocks, and maximum record size. The FDS-1 utility routine XPATR is called to retrieve parameter attributes of the file being created. These parameters are then stored in the AWA by the FDS-1 utility processor XPPUT. The file is then created using the file manager service routine CREAT, using the user-supplied name of the

output DRDE file. A test is made to see if an error occurred while creating the file and, if so, an error message is printed and an abnormal termination follows.

If the file is created properly, it is opened by using the file manager service routine OPEN. A test is made for an error condition, and if one exists a message is printed and abnormal termination follows.

Otherwise, data file names are moved to header records. These records are then written to the output file via the file manager service routine WRITF. An error condition is then tested and if an error occurred while writing the file, an error message is printed and an abnormal termination follows. Otherwise, normal termination follows. This procedure consists of setting IPARM(1) to zero, setting the return linkage through the RTE service routine RETRN, and calling the calling routine, QUIKU.

5.2.3 Assumptions and Limitations

Because of internal size limits on arrays TIM and JACT, a maximum of 900 activities can be processed.

5.2.4 Method

None.

5.2.5 Routine Input/Output Variables

The input/output variables for routine QPRPU are presented in table 5.2-I.

5.2.6 Functional Logic Flow

A functional logic flow is not provided. The functional level PDL for QPRPU is presented in figure 5.2-1.

5.2.7 Diagnostics and Debug

Error messages are displayed for the following problems:

a. Inability to open a file

b. File problem (problem in a read, positioning or write)

c. Inability to find an activity for a given number

d. The time line is larger than the internal arrays

5.2.8 Special Comments

None.

5.2.9 References

None.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine QPRPU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
BSLQTY	--	Real	O	--	C	--	Baseline usable tank capacities for each consumable
CONTNU	--	2CH	I	--	IT	--	User's desire to continue processing if arrays become full
FLNAME	--	6CH	O	--	C	--	Name of the output data file
FLSDMP	--	Logical	O	--	C	--	Specifies if flash evaporators are to be used for the water dump
H2O	--	Real	O	--	C	--	Available potable water quantity
H2OHI	--	Real	O	--	C		Maximum available quantity of potable water
H2OLO	--	Real	O	--	C	--	Minimum available quantity of potable water
IFILE	--	Intg	O	--	C	--	Data control buffer for the output file
INTBUF	--	Intg	O	--	C	--	FDS input buffer placed by the preparation segment
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Continued

Routine QPRPU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IOACT	—	Intg	O	--	C	--	Data control buffer for the ORBITER ACTIVITY file.
IPACT	—	Intg	O	--	C	—	Data control buffer for the PAYLOAD ACTIVITY file.
IPARM	—	Intg	I	--	C	--	System and FDS-1 parameters.
ISIZE	--	Intg	O	—	IT	--	Contains the file type, blocks, and maximum record size in XPPUT call.
JACT	--	Intg	O	—	C	--	Array containing activity numbers.
LTR	—	2CH	I	—	C	—	ASCII character describing the activity type.
LU	—	Intg	I	--	C		Logical unit number of user.
N2RT	--	Real	O	—	C		Nitrogen flow rate into cabin.
PROCON	—	Real	I	—	C		Processor constants, cartridge reference numbers for system and DRDE files, and the number of blocks in the output file.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Concluded

Routine QPRPU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
QTYKIT	—	Real	O	--	C	--	Usable tank capacity per kit for each consumable.
RADCAP	—	Real	O	--	C	--	Radiator cooling capacity (btu/hr).
SQMET	—	Real	O	—	C	--	Metabolic rate of Shuttle crew.
TIM	—	Real	O	--	C	--	Array containing both start and stop times.
TIME	—	Real	O	--	C	--	Point on time line presently being evaluated.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

```
1*
1 BEGIN QPRPU
2     SET COUNTERS AND INTERFACE TABLE BUFFER
2     CALL XPGET FOR INTERFACE TABLE PARAMETERS
2     PERFORM ACTFL FOR ORBITER ACTIVITY DATA FILE
2     IF(NO ERROR RETURN)
2     THEN
3        BRANCH TO -LBL200-
2     ELSE
3        EXIT TO -EX8000-
2     ENDIF
2*
2 -LBL200-
2     IF(PAYLOAD DATA FILE NAME NOT = BLANKS)
2     THEN
3        PERFORM ACTFL FOR THE PAYLOAD DATA FILE
3        IF(NO ERROR RETURN)
3        THEN
4           BRANCH TO -LBL300-
3        ELSE
4           EXIT TO -EX8000-
3        ENDIF
2     ELSE
3        BRANCH TO -LBL300-
2     ENDIF
2*
2 -LBL300-
2     SET TIMELINE FILE NAME
2     CALL OPEN FOR TIMELINE FILE
2     IF(NO ERROR)
2     THEN
3        READ HEADER RECORD
3        IF(NO ERROR)
3        THEN
4           DOWHILE(MORE RECORDS REMAIN)-
5              CALL READF FOR A TIMELINE RECORD
5              IF(NO ERROR)
5              THEN
6                 SET FLAGS FOR ORBITER ACTIVITY
6                 IF(ACTIVITY = PAYLOAD ACTIVITY)
6                 THEN
7                    RESET FLAGS FOR PAYLOAD ACTIVITY
7                    BRANCH TO -LBL380-
6                 ELSE
7                    BRANCH TO -LBL380-
6                 ENDIF
6*
```

Figure 5.2-1.- QPRPU functional level PDL.

```
6 -LBL380-
7                DOFOR I = 1,COUNTER,1
8                   IF(ACTIVITY NUMBER NOT= ARRAY NUMBER)
8                   THEN
9                      CONTINUE DOFOR
8                   ELSE
9                      BRANCH TO -LBL400-
8                   ENDIF
7                ENDDO
6                WRITE ACTIVITY NOT FOUND MESSAGE
6                SET FLAG
6                CONTINUE DOWHILE
6 -LBL400-
6                SET COUNTERS FOR NEXT DO LOOP
6 -LBL410-
6                INCREMENT COUNTERS
7                DOUNTIL(COUNTER > LIMIT OR ACTIVITY NUMBER
7                        NOT= ACTIVITY ARRAY NUMBER)
8                   IF(ENDTIME < START TIME + DELTA START TIME)
8                   THEN
9                      CONTINUE DOUNTIL
8                   ELSE
9                      INCREMENT COUNTERS
9                      SET START TIME INTO TIME ARRAY
9                      SET ACTIVITY NUMBER INTO ACTIVITY NUMBER
9                         ARRAY
9                      IF(LETTER ARRAY VALUE = TEST VALUE
9                      THEN
10                        CONTINUE DOUNTIL
9                      ELSE
10                        INCREMENT COUNTERS
10                        SET ACTIVITY NUMBER INTO ACTIVITY
10                           NUMBER ARRAY
10                        IF(LETTER ARRAY VALUE = TEST VALUES)
10                        THEN
11                           SET START TIME INTO TIME ARRAY
11                           IF(TIME ARRAY < END TIME)
11                           THEN
12                              CONTINUE DOUNTIL
11                           ELSE
12                              SET END TIME INTO TIME ARRAY
11                           ENDIF
10                        ELSE
11                           SET END TIME INTO TIME ARRAY
11                           CONTINUE DOUNTIL
10                        ENDIF
9                      ENDIF
8                   ENDIF
```

Figure 5.2-1.- Continued.

```
6           -LBL470-
6                 CALL CLOSE FILE
6                 IF( READ ERROR PRESENT)
6                 THEN
7                    WRITE READ ERROR MESSAGE
7                    EXIT TO -EX8000-
6                 ELSE
7                    BRANCH TO -LBL500-
6                 ENDIF
5              ENDIF
4           ENDDO
3        ELSE
4           WRITE READ ERROR MESSAGE
4           EXIT TO -EX8000-
3        ENDIF
2     ELSE
3        WRITE OPEN ERROR MESSAGE
3        EXIT TO -EX8000-
2     ENDIF
2*
2 -LBL500-
2     IF(ERROR FLAG = TEST VALUE)
2     THEN
3        IF(CONTINUE FLAG = YES)
3        THEN
4           PERFORM ORDER
3        ELSE
4           EXIT TO -EX8000-
3        ENDIF
2     ELSE
3        PERFORM ORDER
2     ENDIF
2        SET TIME TO FIRST TIME ARRAY VALUE
2*
2     CALL OPEN FOR SYSTEMS CHARACTERISTICS DATA FILE
2     IF(NO ERROR)
2     THEN
3        CALL POSNT TO POSITION FILE FOR FIRST DATA RECORD
3        IF(NO ERROR)
3        THEN
4           CALL READF FOR A RECORD
4           IF(NO ERROR)
4           THEN
5              SET INTERNAL VARIABLES
6              DOFOR I = 1,5,1
7                 SET INTERNAL VARIABLE ARRAYS
6              ENDDO
5              CALL CLOSE FILE
4           ELSE
```

Figure 5.2-1.- Continued.

```
5               WRITE READ ERROR MESSAGE
5               CALL CLOSE FILE FOR SYSTEM CHARACTERISTICS
5               EXIT TO -EX8000-
4            ENDIF
3         ELSE
4            WRITE POSITION ERROR MESSAGE
4            CALL CLOSE FILE FOR SYSTEMS CHARACTERISTICS
4            EXIT TO -EX8000-
3         ENDIF
2      ELSE
3         WRITE OPEN ERROR MESSAGE
3         EXIT TO -EX8000-
2      ENDIF
2*
2      CALCULATE VARIABLES
2      CALL EXEC FOR CLOCK TIME
2      SET VARIABLES TO TIME
2      SET FDS VARIABLES
2      CALL XPATH FOR OUTPUT FILE NAME
2      CALL XPPUT TO INPUT FILE CHARACTERISTICS
2      CALL CREAT TO CREATE THE OUTPUT FILE
2      IF(ERROR)
2      THEN
3         IF(DUPLICATE FILE NAME)
3         THEN
4            CALL OPEN FOR THE OUTPUT FILE
4            IF(NO ERROR)
4            THEN
5               BRANCH TO -LBL800-
4            ELSE
5               EXIT TO -EX8000-
4            ENDIF
3         ELSE
4            EXIT TO -EX8000-
3         ENDIF
2      ELSE
3         BRANCH TO -LBL800-
2      ENDIF
2*
2 -LBL800-
3      DOFOR I = 1,3,1
4         SET DRDE HEADER RECORD VALUES
3      ENDDO
2      CALL WRITE TO OUTPUT DRDE HEADER RECORD
2      IF(NO ERROR)
2      THEN
3         CALL WRITE TO OUTPUT INFORMATION HEADER RECORD
3         IF(ERROR)
3         THEN
```

Figure 5.2-1.- Continued.

```
4           EXIT TO -EX8000-
3        ELSE
4           EXIT TO -EX9000-
3        ENDIF
2     ELSE
3        EXIT TO -EX8000-
2     ENDIF
2*
2 -EX8000-
3     SET IPARM ONE TO TERMINAL VALUE
3     EXIT TO -EX9999-
2*
2 -EX9000-
3     SET IPARM ONE TO NORMAL RETURN VALUE
2*
2 -EX9999-
3     CALL RETRN
3     CALL MAIN PROGRAM QUIKU
2*
1 END QPRPU
1*
```

Figure 5.2-1.- Concluded.

5.3 ROUTINE NAME - QSEGU

5.3.1 Purpose

The purpose of quick segment (QSEGU) routine is to provide the executive control to sequence the evaluation model appropriately.

5.3.2 Functional Description

The QSEGU routine is basically an outer loop controlling the reading of the time line. It evaluates certain consumption rates enclosing an inner loop that outputs to a file and does some other computations. The inner loop continues while the time of the point on the time line currently being evaluated is less than the time of the next time-line entry. The outer loop continues until the last record is found.

Initially, the outer loop calls processor subroutine PROFL, which reads the time line array and builds a load profile for the EPS and ATCS model. A test is made for an error return from PROFL and if one is detected, the loop is exited and the processor is abnormally terminated.

If no error condition is present, a call is made to processor subroutine EPS to determine power requirements and calculate fuel cell parameters. A call is then made to processor subroutine ECS to determine consumable usage rates affected by the ECS.

At this point, the inner loop begins with a call to processor subroutine PLDAT, which outputs to file the consumable requirements at each time-line entry point. A test for an error condition is made at the return from PLDAT, and if an error is detected, the loop is exited and an abnormal termination follows.

Otherwise, a test is made to see if the last record has been processed (the time of the next time-line entry being greater than 99999). If not, processor subroutine TANK is called to determine the total requirements for each consumable being evaluated. A test for an error condition is made at the return from TANK, and if an error exists, the loop is exited and an abnormal termination follows.

At this point a test is made to see if current position on the time line is less than the time held in the TNEXT variable (time of the next time-line entry). If so, control is returned to the beginning of the inner loop; otherwise, control returns to the outer loop.

When the outer loop is exited, the variable representing the current time on the time line is set to 99999. A call is made to PLDAT to output data.

The processor truncates the output file by calling the file manager routine LOCF to determine the length of the file. The number of excess blocks is determined and truncated through the third parameter ITRUNC in the file manager routine CLOSE. The new file characteristics are placed in the interface table through the FDS-1 utility processor XPPUT. The ORBITER ACTIVITY file and the PAYLOAD

ACTIVITY file are closed via CLOSE, the return linkage is set via the RTE service routine RETRN, and a call is made to the calling routine QUIKU.

5.3.3 Assumptions and Limitations

None.

5.3.4 Method

None.

5.3.5 Routine Input/Output Variables

The QSEGU input/output variables are presented in table 5.3-I.

5.3.6 Functional Logic Flow

A functional logic flow is not provided. The functional level PDL for QSEGU is presented in figure 5.3-1.

5.3.7 Diagnostics and Debug

None.

5.3.8 Special Comments

None.

5.3.9 References

None.

TABLE 5.3-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine QSEGU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IPARM	—	Intg	I/O	—	C	--	System and FDS parameters.
TIME	--	Real	I/O	—	C	--	Point on time line presently being evaluated.
TNEXT	—	Real	I/O	—	C	--	Time of next time-line entry.
IFILE	—	Intg	I	--	C	--	Data control buffer for the output file.
IOACT	—	Intg	I	—	C	--	Data control buffer for the ORBITER ACTIVITY file.
IPACT	—	Intg	I	—	C	--	Data control buffer for the PAYLOAD ACTIVITY file.
LU	--	Intg	I	—	C	--	User's logical unit number.
INTBUF	--	Intg	I/O	--	C	--	FDS-1 input buffer.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

```
1*
1 BEGIN QSEGU
2 -LBL100-
3    DOWHILE(MORE INPUT RECORDS REMAIN)
4       PERFORM PROFL
4       IF(NO ERROR RETURN)
4       THEN
5          PERFORM EPS
5          PERFORM ECS
5*
5       -LBL300-
5          PERFORM PLDAT
5          IF(NO ERROR RETURN)
5          THEN
6             IF(NEXT TIME > TEST VALUE)
6             THEN
7                BRANCH TO -LBL600-
6             ELSE
7                PERFORM TANK
7                IF(NO ERROR RETURN)
7                THEN
8                   IF(TIME < NEXT TIME)
8                   THEN
9                      BRANCH TO -LBL300-
8                   ELSE
9                      CONTINUE DOWHILE
8                   ENDIF
7                ELSE
8                   EXIT TO -EX8000-
7                ENDIF
6             ENDIF
5          ELSE
6             EXIT TO -EX8000-
5          ENDIF
4       ELSE
5          EXIT TO -EX3000-
4       ENDIF
3    ENDDO
2*
```

Figure 5.3-1.- QSEGU functional level PDL.

```
2 -LBL600-
2     SET TIME TO LAST VALUE
2     PERFORM PLOAT
2     EXIT TO -EX9000-
2*
2 -EX8000-
3     SET IPARM ONE TO TERMINAL VALUE
3     CALL CLOSE FOR ORBITER ACTIVITY FILE
3     CALL CLOSE FOR PAYLOAD ACTIVITY FILE
3     CALL CLOSE FOR OUTPUT FILE
3     EXIT TO -EX9040-
2*
2 -EX9000-
3     SET IPARM ONE TO NORMAL RETURN VALUE
3     CALL LOCF FOR NUMBER OF BLOCKS USED
3     IF(NO ERROR)
3     THEN
4        COMPUTE AMOUNT TO TRUNCATE
4        CALL CLOSE TO TRUNCATE OUTPUT FILE
4        IF(ERROR)
4        THEN
5           EXIT TO -EX8000
4        ELSE
5           SET NEW FILE CHARACTERISTICS
5           CALL XPPUT TO INSERT NEW FILE CHARACTERISTICS
5           CALL CLOSE FOR ORBITER ACTIVITY FILE
5           CALL CLOSE FOR PAYLOAD ACTIVITY FILE
4        ENDIF
3     ELSE
4        EXIT TO -EX3000-
3     ENDIF
2*
2 -EX9040-
3     CALL RETRN
3     CALL MAIN PROGRAM QUIKU
2*
1 END QSEGU
1*
```

Figure 5.3-1.- Concluded.

PARAMETRIC SCAN PROCESSORS (SCAN/ENDSC)

1.0 PURPOSE

The SCAN/ENDSC utility processors, used as a pair of sequence table entries, provide the FDS user with the capability to perform parametric scans with one or more processors within a sequence table, creating a data box (a set of summary tables collected into a disk resident data element).

2.0 FUNCTIONAL DESCRIPTION

The processors SCAN/ENDSC create and control an interactive loop over a series of processors as defined by a set of sequence table entries. The scan loop begins at the sequence number of the processor immediately subsequent to the SCAN and continues to the sequence number of the corresponding ENDSC.

Within the scan loop, one or two independent variables can be designated to be varied. SCAN interface table parameter keywords provide the identification of the independent variables, their centroid values, increment values, and the number of steps (iterations) to be executed on either side of the centroid. This information plus additional scan information is saved in a manager data array named &SCNTB.

As each cycle of the iteration loop is being executed, the processors involved must generate output data that are stored in a data element identified by the interface table parameter keyword SUMTAB (summary table). The summary table consists of a name, units mnemonics, and a value for a predefined set of parameters (the predefined set depends on the processor(s) involved in the scan).

Each execution of ENDSC gets information from the current &SCNTB and outputs the current contents of the summary table defined by SCAN parameter keyword DATBOX. New values for the independent variables are computed, &SCNTB is updated, and control is given to the processor at the top of the scan loop.

Iteration of the scan loop is performed in a nested fashion (i.e., the first variable (XSCAN) will be cycled for each value of the second variable (YSCAN)). At the end of the parametric scan, all values will be left at the centroid position and control is given to the processor immediately subsequent to the ENDSC processor.

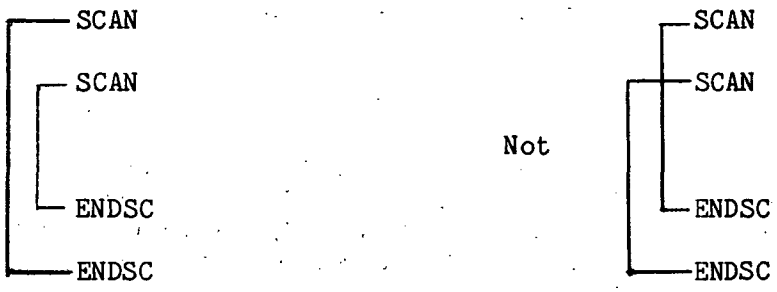
3.0 ASSUMPTIONS AND LIMITATIONS

Listed as follows are the assumptions and limitations associated with the SCAN/ENDSC processors.

a. SCAN and ENDSC must be utilized as a pair to ensure correct results for a parametric scan. Use of SCAN with no ENDSC results in no data box generation; use of only ENDSC results in an error message and execution is terminated.

b. There is no interface table for ENDSC. SCAN builds &SCNTB control block where ENDSC receives all inputs.

c. Parametric scans are processed in a nested fashion; i.e., the first ENDSC encountered will be matched with the last previous SCAN encountered; for example:



The maximum number of nested SCANs is 4, and each nested SCAN must output to a different data box (DATBOX).

NOTE: When executing SCAN in the semiautomatic mode, the user should be aware of the following:

(1) When a SCAN is followed by another SCAN in a predefined sequence table, the first SCAN is executed with the second SCAN as the reset sequence number, and the second SCAN is executed with the following processor as the reset sequence number. In this manner the execution of these two SCANs is nested.

(2) When any processor except SCAN is overridden by the user with a SCAN, the SCAN is executed and the reset sequence number is the original processor, not the original SCAN. If an ENDSC is also inserted, the net effect is that the two SCANs will not be nested, but executed sequentially with the same reset sequence number.

d. The summary table parameter keyword SUMTAB in SCAN must be satisfied by an AWA element name; not a literal. It need not be present in the AWA at SCAN execution but must exist when ENDSC is executed. The summary table can be subscripted, but the subscripts must index the beginning of an entry (name, units, values). The summary table can be any size; if it is less than or equal to 32 elements, the entire table is saved in the data box, but if it is greater than 32 elements, only the first 32 elements (after subscripts are applied) are stored in the data box.

e. The number of scan variables, parameter keyword NOVAR, can only have an integer value of 1 or 2.

f. The number of steps, parameter keywords XSTEPS and YSTEPS, can only have an integer value 0-5 inclusive.

4.0 PROCESSOR INPUT/OUTPUT

Listed below are the inputs required and outputs for the SCAN/ENDSC processors.

a. Processor interface table - The interface table for SCAN is defined in table 4-I and contains:

   (1) Summary table to be output to data box

   (2) Data box to be created for this SCAN

   (3) Variables to be scanned

   (4) Scan variable parameters (centroids, increments, number of steps)

b. Processor interface table data array definitions - The format of the summary table defined by parameter keyword SUMTAB depends on the processor(s) involved in the SCAN.

c. Processor interface table data file definitions - The format (type 2; record length, 64 words) and definitions of the output data file is provided in table 4-II.

d. Processor displays and display parameter definition table - None.

e. Processor message table - The SCAN/ENDSC processor error messages are provided in table 4-III.

f. Scan control table - The format of the SCAN control table is given in table 4-IV. It is created by SCAN, updated by ENDSC and deleted by ENDSC when the parametric scan is completed. It is 173 words per active SCAN (maximum is 4 active), and resides in the AWA as a manager data array.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-V.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR SCAN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
PROCON	AWA	Intg	I	1	1	20	Cartridge number for data box file
SUMTAB	AWA	Free	I	(variable)	8	--	Summary table
NOVAR	AWA	Intg	I	1	1	1	Number of scan variables (1 or 2)
XUNIT	AWA	6CH	I	3	1	--	Units of X-scan variable
XCENTR	AWA	Real	I	2	1	--	X centroid
XINCR	AWA	Real	I	2	1	--	X increment
XSTEPS	AWA	Intg	I	1	1	--	Number of steps for X (0-5)
YUNIT	AWA	6CH	I	3	1	"Ø"	Units of Y-scan variable
YCENTR	AWA	Real	I	2	1	0	Y centroid
YINCR	AWA	Real	I	2	1	0	Y increment
YSTEPS	AWA	Intg	I	1	1	0	Number of steps for Y (0-5)

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-I.- Concluded

PROCESSOR SCAN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I, J)	Values stored in default interface table	Definition
DATBOX	Disk	Free	0	(variable)	-	--	Data box
XSCAN	AWA	Real	0	2	1	--	X-scan variable
YSCAN	AWA	Real	0	2	1	DUMMY	Y-scan variable

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- PROCESSOR INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR SCAN/ENDSC

DRDE DATA FILE DATBOX

Record number	Integer word allocations	Content and definition
1	64	(1) Name of FDS processor creating the file (4) Interface table variable name for this file (7) Name of FDS processor updating file (3 ASCII words of blanks) (10) Interface table variable name for this update (3 ASCII words of blanks)
2	64	(1) Number of entries in SUMTAB (integer 1-32) (2) X scan variable (6 CHAR) (5) X first subscript (integer or zero if none) (6) X second subscript (integer or zero if none) (7) X units (6 CHAR) (10) X centroid (Real) (12) X increment (Real) (14) X number of steps (integer 0-5) (15) Y scan variable (6 CHAR) or zero if none (18) Y first subscript (integer or zero if none) (19) Y second subscript (integer or zero if none) (20) Y units (6 CHAR) (23) Y centroid (Real) (25) Y increment (Real) (27) Y number of steps (integer 0-5)
3	64	First 64 words of SUMTAB variable names
4	64	Last 32 words of SUMTAB variable names First 32 words of SUMTAB variable units
5	64	Last 64 words of SUMTAB variable units
6	64	Values for scan variable(s) X_1 or X_1Y_1

TABLE 4-II.- Concluded

PROCESSOR SCAN/ENDSC

DRDE DATA FILE DATA BOX

Record number	Integer word allocations	Content and definition
7	64	Values for scan variable(s) X_2 or X_2Y_1
.		
.		
.		
N+5*	64	Values for scan variable(s) X_n or X_nY_1
N+6*	64	(If there is no Y scan variable, record N+5 is the last record, ELSE) Values for scan variables X_1Y_2
.		
.		
.		
N+M+5	64	Values for scan variables X_nY_m

*NOTE: N = Number of values of X
M = Number of values of Y

TABLE 4-III.- PROCESSOR MESSAGE TABLE

PROCESSOR SCAN/ENDSC

MSG no.	Message ID block	Message text block and explanation
1	*SC01*	"SUMTAB MUST BE AN AWA ELEMENT, NOT A LITERAL." Meaning: SUMTAB parameter keyword contains literal data instead of an AWA data element name. Severity: Fatal, sequence is aborted. Action required by user: Use Interface Table Editor to make SUMTAB refer to an AWA data element and reexecute sequence table.
2	*SC02*	"SUMTAB SUBSCRIPTS DO NOT INDEX A SUMMARY TABLE ENTRY." Meaning: SUMTAB subscripts do not index a name, units, value entry within a summary table. Severity: Fatal, sequence is aborted. Action required by user: Use Interface Table Editor to correct subscripts and reexecute the sequence table.
3	*SC03*	"CAN ONLY SCAN 1 OR 2 VARIABLES." Meaning: NOVAR parameter keyword <1 or >2. Severity: Fatal, sequence is aborted. Action required by user: Use Interface Table Editor to set NOVAR to either 1 or 2 and reexecute the sequence table.
4	*SC04*	"NUMBER OF STEPS MUST BE BETWEEN 0 and 5, INCLUSIVE." Meaning: Either the XSTEPS or YSTEPS parameter keyword is a value <0 or >5. Severity: Fatal, sequence is aborted. Action required by user: Use Interface Table Editor to set XSTEPS and/or YSTEPS to a valid value and reexecute the sequence table.
5	*SC05*	"MAXIMUM NUMBER OF NESTED SCANS IS 4." Meaning: Sequence table contains more than 4 entries to execute the SCAN processor. Severity: Fatal, sequence is aborted. Action required by user: Use the Sequence Table Editor to modify the sequence table to contain 4 or less scans and reexecute the sequence table.

TABLE 4-III.- Continued

PROCESSOR SCAN/ENDSC

MSG no.	Message ID block	Message text block and explanation
6	*SC06*	"NESTED SCANS CANNOT OUTPUT TO SAME DATA BOX." Meaning: There are two nested scans whose interface tables refer to the same data box in the parameter keyboard DATBOX. Severity: Fatal, sequence is aborted. Action required by user: Either use the Interface Table Editor to change one of the DATBOX names or use the Sequence Table Editor to execute one of the scans with a different interface table; then reexecute the sequence table.
7	*SC07*	"SCAN CANNOT BE LAST OR ONLY ENTRY IN A SEQUENCE TABLE." Meaning: Either a sequence table contains a SCAN as the last entry, or SCAN was executed in the manual mode. Severity: Fatal, sequence is aborted. Action required by user: Only execute SCAN from a predefined sequence table, ensuring the presence of an ENDSC.
8	*SC08*	"NO SCAN WAS EXECUTED BEFORE THE ENDSC." Meaning: Either a sequence table contains an ENDSC with no SCAN, or ENDSC was executed in the manual mode. Severity: Fatal, sequence is aborted. Action required by user: Only execute ENDSC from a predefined sequence table, ensuring the presence of a previous SCAN.
9	*SC09*	"SUMTAB MUST BE PRESENT IN AWA AT END SCAN TIME." Meaning: ENDSC could not find the summary table referenced in SUMTAB SCAN parameter keyword in the AWA. Severity: Fatal, sequence is aborted. Action required by user: Ensure that a processor is executed prior to ENDSC that outputs a summary table whose name is the same as referenced by SCAN's SUMTAB.
10	*SC10*	"NO AWA SPACE TO STORE THIS SCAN'S CONTROL TABLE." Meaning: There is not enough AWA space for &SCNTB. Severity: Fatal, sequence is aborted. Action required by user: Clear enough room in the AWA for 173 free words for each SCAN to be executed.

TABLE 4-III.- Concluded

PROCESSOR SCAN/ENDSC

MSG no.	Message ID block	Message text block and explanation
11	#SC11#	"FMGR ERROR -NN CCCCC." Meaning: A File Manager error -NN occurred trying to do a CCCCC operation where CCCCC is WRITE, READ, etc. Severity: Fatal, sequence is aborted. Action required by user: Notify FDS system maintenance personnel.

TABLE 4-IV.- SCAN CONTROL TABLE DEFINITION

PROCESSOR SCAN/ENDSC

Array name	Index location	Default value	Definition	
&SCNTB	1		SUMTAB	6-CHAR name of summary table
	4		DISPL	Displacement into SUMTAB
	5		DATBOX	6-CHAR qualified name of data box
	8		RESET	Reset sequence number
	9		CENREC	Centroid record number
	10		XSCAN	Name of X-scan variable
	13		XDISPL	Displacement into XSCAN
	14		XCENT	Centroid value for X
	16		XINCR	Increment value for X
	18		XSTEPS	Number of steps for X
	19		XCUR	Current step number for X
	20		YSCAN	Name of Y-scan variable
	23		YDISPL	Displacement into YSCAN
	24		YCENT	Centroid value for Y
	26		YINCR	Increment value for Y
	28		YSTEPS	Number of steps for Y
	29		YCUR	Current step number for Y
	30		IDCB	DATBOX DCB (144 words)

TABLE 4-V.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR SCAN

Processor name	Processor abstract prompt (maximum 256 characters)
SCAN	The SCAN processor executes a series of processors iterating on XSCAN and YSCAN (if entered) values computed using the centroid, increment, and current step number used in creating a data box file.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
PROCON	PROCON is one integer value; the cartridge number for the data box file.
SUMTAB	SUMTAB is an 8 by 32 word (free) array; value is a summary table.
DATBOX	DATBOX is a DRDE file; contents are five header records and one summary table (one record) for each iteration of SCAN.
NOVAR	NOVAR is one integer value; number of variables to SCAN (one or two).
XSCAN	XSCAN is one real value; the AWA element used as the first SCAN variable.
XUNIT	XUNIT is a six-character name; value is the units of X variable.
XCENTR	XCENTR is one real value; the centroid value for X.
XINCR	XINCR is one real value; the increment used to obtain the next value of X to be used in SCAN.
XSTEPS	XSTEPS is one integer value; the number of steps on either side of the centroid to be executed for X.
YSCAN	YSCAN is one real value; the AWA element used as the second SCAN variable.
YUNIT	YUNIT is a six-character name; value is the units of the Y variable.

TABLE 4-V.- Continued

PROCESSOR SCAN

Processor name	Processor abstract prompt (maximum 256 characters)
Parameter keyword name	**Parameter definition prompt (maximum 256 characters)**
YCENTR	YCENTR is one real value; the centroid value for Y.
YINCR	YINCR is one real value; the increment used to obtain the next value of Y to be used in SCAN.
YSTEPS	YSTEPS is one integer value; the number of steps on either side of the centroid to be executed for Y.

TABLE 4-V.- Concluded

PROCESSOR ENDSC

Processor name	Processor abstract prompt (maximum 256 characters)
ENDSC	The ENDSC processor is used in conjunction with SCAN only. It marks the end of the sequence of processors to be scanned. It has no interface table.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)

5.0 PROCESSOR ROUTINES

The only available routine documentation is contained on the comment cards in the software listing. Additional material may be found in JSC IN 77-FM-18, vol. IV, rev. 2 dated April 1978, and in JSC IN 77-FM-18, vol. IX (to be published).

SUNRISE/SUNSET TIME PREDICTOR PROCESSOR (SRSS)

1.0 PURPOSE

The sunrise/sunset time predictor processor computes, for an Earth orbit, the times of occurrence of terminator crossings, sunrise/sunset, effective sunrise/sunset, orbital midnight, and orbital noon.

2.0 FUNCTIONAL DESCRIPTION

The trajectory information may be input either through a trajectory DRDE, which has been generated by the INVAR processor, or by inputting a single invariant element state vector.

The processor uses the input to compute, on an orbit-by-orbit basis, eight quantities for each of eight lighting conditions. The eight lighting conditions are shown in figure 2-1.

Effective sunrise/sunset is computed by considering a spherical Earth with an effective radius defined as the sum of the Earth radius plus the atmospheric altitude. The sunrise/sunset quantities are either penumbra, umbra, or point of tangency.

The eight quantities computed for each occurrence of a lighting condition are:

a. Orbit number
b. Ground elapsed time
c. Greenwich mean time
d. Declination
e. Longitude
f. Altitude of vehicle
g. Local vertical, local horizontal pitch angle to Sun
h. Local vertical, local horizontal yaw angle to Sun

The eight calculated quantities are the outputs of the processor and are displayed on the user's terminal. The user may select to store these quantities as a DRDE.

3.0 ASSUMPTIONS AND LIMITATIONS

Refraction is not considered in the calculations. The analytic Sun ephemeris routine, SUN, is used in making the calculations. Orbital propagation using invariant elements does not reflect changes to the orbit due to atmospheric drag

between the states stored in the trajectory DRDE. Therefore, the vectors in the trajectory DRDE should be stored close enough together so that these inaccuracies will be small enough for the user's requirements. Periodic perturbations to the orbit are not considered. Only elliptical orbit formulation is provided.

The BASTM processor should be run prior to running the SRSS processor in order to obtain the solar ephemeris coefficients from the master data base element !SESCN.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The input the sunrise/sunset processor is through its interface table and through the trajectory DRDE that has been generated by the FDS-1 application processor INVAR. Optionally, a single invariant element state vector may be input instead of the trajectory DRDE. The input parameters are IVEC, SVIN, VECFIL, GETS, GETF, ALTEFF, INORB, OPT, OUTFLG, GMTR, and SARRAY. In addition, 10 conversion factors and constants contained in the master data base elements !!GLCN and !SESCN are input through the interface table. For the specific constants and conversion factors input, refer to table 4-I. The sunrise/sunset quantities are either penumbra, umbra, or point of tangency based on the setting of the flag OPT.

   Through the input parameters GETS and GETF, the user specifies the start time (GETS) relative to the base time and the final time (GETF) relative to the base time. The altitude (ALTEFF) for the effective sunrise/sunset computations and the orbit counter (INORB) corresponding to the initial orbit for GETS are specified.

   Through the input parameter IVEC, the user sets a flag to indicate the source of data for the element state vector. The valid settings for the input flag and their meanings are shown in the definition block of the sunrise/sunset interface table (table 4-I). Depending on the setting of IVEC, the invariant element state vector is input through the parameter SVIN or from the DRDE specified by VECFIL. The contents of the invariant element state vector are defined in table 7.3-VI JSC IN 78-FM-60, of volume I.

   Parameter OUTFLG indicates whether or not the output quantities are to be written to the DRDE specified by the parameter SUNFIL. When DRDE output is specified, the same quantities that are displayed are written to the DRDE in the format presented in table 4-IV with the exception of time, which is written to the DRDE as Greenwich mean time (GMT) in seconds.

b. Interface table data array definitions - The definition of the input data arrays appearing in the sunrise/sunset interface table is provided in table 4-II.

c. Interface table data file definitions - The formats of the position velocity state vector DRDE and the output DRDE are provided in tables 4-III(a) and 4-III(b).

d. Processor solicited (prompted) inputs - The processor solicited prompt, its meanings, and the valid response are given in table 4-IV. As each line of the display is written, the line counter is incremented. When the number of lines displayed is equal to the maximum number of lines per page, as specified in the PROCON array, the processor solicited prompt "MAXIMUM LINE NUMBER" is displayed and execution of the processor pauses. This pause gives the user the opportunity to review the contents of the display and to make a hard copy of the display. The user clears the screen and enters a blank character and a carriage return to continue execution of the processor.

e. Processor displays and display parameter definition tables - When the processor executes correctly, it will generate an output display on the user's terminal. The format and definitions of the display are given in tables 4-V and 4-VI.

f. Processor message table - The messages that may be displayed on the user's terminal during execution of the processor are provided in table 4-VII. An explanation of the message, the severity of the problem, and the action required by the user are also provided in the table.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VIII.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR SRSS

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
IVEC	AWA	2CH	I	1	1		Vector source flag "IN" = Single input state (SVIN) "DF" = Trajectory DRDE as specified by VECFIL
SVIN	AWA	Real	I	30	15		Input invariant element state vector. Required when IVEC = "DF".
VECFIL	Disk	Real	I	--	--		Name of invariant element DRDE. Required when IVEC = "DF".
OPT	AWA	2CH	I	1	1		Option for specifying sunrise/sunset quantities "UM" = Umbra "PE" = Penumbra "TA" = Tangent
GETS	AWA	Real	I	6	3		Start time (hr., min., sec.) relative to base time.
GETF	AWA	Real	I	6	3		Final time (hr., min., sec.) relative to base time.
ALTEFF	AWA	Real	I	2	1		Altitude above Earth's surface for use in effective sunrise/sunset calculations.
INORB	AWA	Intg	I	1	1		Initial orbit counter. Corresponds to the orbit counter for GETS time.

NOTES	CLASS	TYPE			USE
	AWA	Free	2CH	72CH	I = Input
	Disk	Intg	6CH	Mix	O = Output
		Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-I.- Continued

PROCESSOR SRSS

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
OUTFLG	AWA	2CH	I	1	1	"NO"	Flag that indicates whether or not the output quantities are to be output to the DRDE specified by SUNFIL. "NO" ; = no output generated "YES"; = output generated
SUNFIL	DISK	Free	O	--	—		Name of output DRDE.
SCOEF	AWA	Real	I	44	22	!SCOEF	Sun constants for analytic ephemeris model.
GMTR	AWA	Real	I	2	1	!SESCN(11)	Reference time for GET computations.
CFA	AWA	Real	I	2	1	!SESCN(27)	Conversion factor for angles - external to internal.
CFD	AWA	Real	I	2	1	!SESCN(29)	Conversion factor for distance.
CFT	AWA	Real	I	2	1	!SESCN(31)	Conversion factor for time.
ROMEGE	AWA	Real	I	2	1	!SESCN(81)	Rotation rate of the Earth.
PI	AWA	Real	I	2	1	!!GLCN(59)	Ratio of circle circumference to diameter.
TWOPI	AWA	Real	I	2	1	!!GLCN(61)	Twice pi

NOTES	CLASS	TYPE			USE
	AWA	Free	2CH	72CH	I = Input
	Disk	Intg	6CH	Mix	O = Output
		Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-I.- Concluded

PROCESSOR SRSS

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
DTWOPI	AWA	Dubl	I	3	1	!!GLCN(58)	Double precision TWOPI
CIRRAD	AWA	Real	I	2	1	!!GLCN(21)	Circular Earth radius
AMIN	AWA	Real	I	2	1	!!GLCN(97)	Seconds per minute
AHOUR	AWA	Real	I	2	1	!!GLCN(99)	Seconds per hour
ADAY	AWA	Real	I	2	1	!!GLCN(101)	Seconds per day
PROCON	AWA	Free	I	15	15		SRSS constants

NOTES	CLASS	TYPE			USE
	AWA	Free	2CH	72CH	I = Input
	Disk	Intg	6CH	Mix	O = Output
		Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR SRSS

Array name	Index location	Default value	Definition	
SVIN	1		Vector time (standard state vector format)	Standard position/velocity state vector definition
	2		Invariant elements semimajor axis	
	3		Invariant elements eccentricity	
	4		Invariant elements inclination	
	5		Invariant elements argument of perigee	
	6		Invariant elements argument of ascending node	
	7		Invariant elements mean anomaly	
	8		Invariant elements mean motion	
	9		Invariant elements rate of change of perigee	
	10		Invariant elements rate of change of the ascending node	
	11		CD-drag coefficient	
	12		Area for drag computation	
	13		Vehicle gross mass	
	14		Vector type code (=3120.)	
	15		Vector name code	
GET	1		Start time relative to reference time (hour)	
	3		Start time relative to reference time (minutes)	
	5		Start time relative to reference time (seconds)	
GETF	1		Final time relative to reference time (hours)	
	3		Final time relative to reference time (minutes)	
	5		Final time relative to reference time (seconds)	
PROCON	1	0	Debug print flag 0 = no print 1 = debug print	
	2	0	Output destination flag 0 = terminal positive number = LU of output device	
	3	5000.	Iteration tolerance for radius iterations, ft.	
	5	2.0	Iteration tolerance for time iteration, sec.	
	7	24	Maximum number of lines per page	
	8	20	Cartridge reference number for DRDE	
	9	150	Maximum number of blocks for the DRDE	
	10	1.5754062	Angle from Sun to terminator for umbra solutions, rad	
	12	1.5707963	Angle from Sun to terminator for tangency solutions, rad	
	14	1.5661011	Angle from Sun to terminator for penumbra solutions, rad	

TABLE 4-III.- INTERFACE TABLE DATA FILE DEFINITIONS

(a) Position/velocity state vector DRDE.

PROCESSOR SRSS

DRDE DATA FILE VECFIL

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file
	4-6	Interface table variable creating file
	7-9	Processor last changing file
	10-12	Interface table variable last changing file
2-N	1-60	Position/velocity phase table values. Refer to figure 7.3-13 of JSC IN 78-FM-60, volume I

TABLE 4-III.- Continued

(b) Output DRDE

PROCESSOR SRSS

DRDE DATA FILE SUNFIL

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file ("SRSS")
	4-6	Interface table variable creating file
	7-9	Processor last changing file
	10-12	Interface table variable last changing file
2-N	1	Orbit number of terminator set
	2-3	Greenwich mean time of terminator set
	4-5	Latitude of terminator set
	6-7	Longitude of terminator set
	8-9	Altitude of terminator set
	10-11	LVLH pitch angle to Sun
	12-13	LVLH yaw angle to Sun
	14	Orbit number of effective sunset
	15-16	Greenwich mean time of effective sunset
	17-18	Latitude of effective sunset
	19-20	Longitude of effective sunset
	21-22	Altitude of effective sunset
	23-24	LVLH pitch angle to Sun
	25-26	LVLH yaw angle to Sun
	27	Orbit number of sunset
	28-29	Greenwich mean time of sunset
	30-31	Latitude of sunset
	32-33	Longitude of sunset
	34-35	Altitude of sunset
	36-37	LVLH pitch angle to Sun
	38-39	LVLH yaw angle to Sun
	40	Orbit number of orbital midnight
	41-42	Greenwich mean time of orbital midnight
	43-44	Latitude of orbital midnight
	45-46	Longitude of orbital midnight
	47-48	Altitude of orbital midnight
	49-50	LVLH pitch angle to Sun
	51-52	LVLH yaw angle to Sun
	53	Orbital number of sunrise
	54-55	Greenwich mean time of sunrise
	56-57	Latitude of sunrise
	58-59	Longitude of sunrise
	60-61	Altitude of sunrise
	62-63	LVLH pitch angle to Sun

TABLE 4-III.- Continued

(b) Continued

PROCESSOR SRSS

DRDE DATA FILE SUNFIL

Record number	Integer word allocations	Content and definition
2-N (Cont'd)	40	Orbit number of orbital midnight
	41-42	Greenwich mean time of orbital midnight
	43-44	Latitude of orbital midnight
	45-46	Longitude of orbital midnight
	47-48	Altitude of orbital midnight
	49-50	LVLH pitch angle to Sun
	51-52	LVLH yaw angle to Sun
	53	Orbit number of sunrise
	54-55	Greenwich mean time of sunrise
	56-57	Latitude of sunrise
	58-59	Longitude of sunrise
	60-61	Altitude of sunrise
	62-63	LVLH pitch angle to Sun
	64-65	LVLH yaw angle to Sun
	66	Orbit number of effective sunrise
	67-68	Greenwich mean time of effective sunrise
	69-70	Latitude of effective sunrise
	71-72	Longitude of effective sunrise
	73-74	Altitude of effective sunrise
	75-76	LVLH pitch angle to Sun
	77-78	LVLH yaw angle to Sun
	79	Orbit number of terminator rise
	80-81	Greenwich mean time of terminator rise
	82-83	Latitude of terminator rise
	84-85	Longitude of terminator rise
	86-87	Altitude of terminator rise
	88-89	LVLH pitch angle to Sun
	90-91	LVLH yaw angle to Sun
	92	Orbit number of orbital noon
	93-94	Greenwich mean time of orbital noon
	95-96	Latitude of orbital noon
	97-98	Altitude of orbital noon
	101-102	LVLH pitch angle to Sun
	103-104	LVLH yaw angle to Sun

TABLE 4-IV.- PROCESSOR SOLICITED (PROMPTED) INPUTS

PROCESSOR SRSS

Prompt	Meaning	Valid responses
"MAXIMUM LINE NUMBER"	The maximum number of output lines specified in PROCON(7) has been reached	After printing a copy of the terminal screen, clear the screen, and enter a space and a carriage return to obtain the remaining output.

TABLE 4-V.- PROCESSOR DISPLAY FORMAT

PROCESSOR SRSS

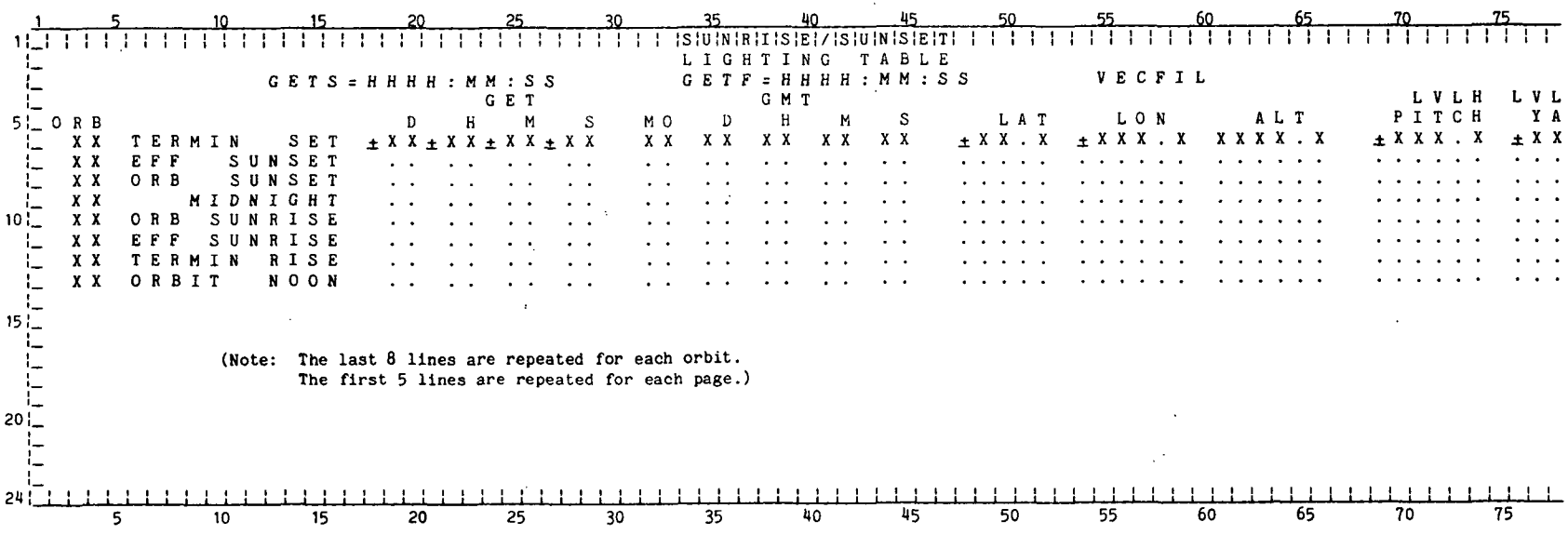


TABLE 4-VI.- DISPLAY PARAMETER DEFINITION TABLE

PROCESSOR SRSS

Display name	
Display parameter label	Parameter definition
GETS	Start time (hrs., min., sec.) relative to base time
GETF	Final time (hrs., min., sec.) relative to base time
VECFIL	Name of invariant element trajectory DRDE. Blanks are displayed when there is no DRDE.
ORB	Orbit number
GET	Ground elapsed time (day, hours, minutes, seconds)
GMT	Greenwich mean time (months, days, hours, minutes, seconds)
LAT	Latitude, degrees
LON	Longitude, degrees
ALT	Altitude, nautical miles
LVLH PITCH	Local vertical, local horizontal (LVLH) pitch, deg
LVLH YAW	Local vertical, local horizontal (LVLH) yaw, deg

TABLE 4-VII.- PROCESSOR MESSAGE TABLE

PROCESSOR SRSS

MSG no.	Message ID block	Message text block and explanation
1	*SRSS*	STATE VECTOR CODE NOT INVARIANT ELEMENT Meaning: Output other than invariant element state vector is specified. Severity: Processor terminates. Action required by user: Input invariant element state vector.
2	*SRSS*	OPEN FILE ERROR = IIII FILE NAME = AAAAAA Meaning: The state vector DRDE is not in the proper configuration. Severity: Processor terminates DRDE. Action required by user: Check the DRDE name, etc., of the state vector DRDE. Rerun the processor.
3	*SRSS*	FILE CREATE ERROR + IIII FILE NAME = AAAAAA Meaning: Error occurred when an attempt was made to create the DRDE. Severity: Processor terminates. No DRDE is created. Action required by user: Check the DRDE name, etc., of the DRDE file for redundancy. Rerun the processor.
4	*SRSS*	FILE READ ERROR = IIII FILE NAME = AAAAAA Meaning: Error rending vector DRDE. Severity: Processor terminates. Action required by user: Regenerate DRDE and rerun the processor.
5	*SRSS*	FILE WRITE ERROR = IIII FILE NAME = AAAAAA Meaning: Error writing DRDE. Severity: Processor terminates. Action required by user: Rerun the processor.
6	*SRSS*	INPUT OPTION DOES NOT SPECIFY UMBRA, TANGENCY OR PENUMBRA SOLUTION. OPT = AA Meaning: OPT is not set to "UM", "TA", or "PE" to indicate type of solution. Severity: Processor terminates. Action required by user: Correct OPT and rerun processor.

TABLE 4-VIII.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR SRSS

Processor name	Processor abstract prompt (maximum 256 characters)
SRSS	The SRSS processor computes, for an Earth orbit, the times of occurrence of terminator crossings, sunrise/sunset, effective sunrise/sunset, orbital midnight, and orbital noon.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
	Use definitions as provided in table 4-I, Processor Interface Table, in addition to the following.
PROCON	SRSS constants: (1) Debug (2) Output LU (3) Iteration tolerance for radius (5) Iteration tolerance for time (7) Max lines/page (8) DRDE cartridge reference number (9) Max number of blocks in DRDE (10) (12) Sun to terminator angle/umbra, tangent, penumbra (14)

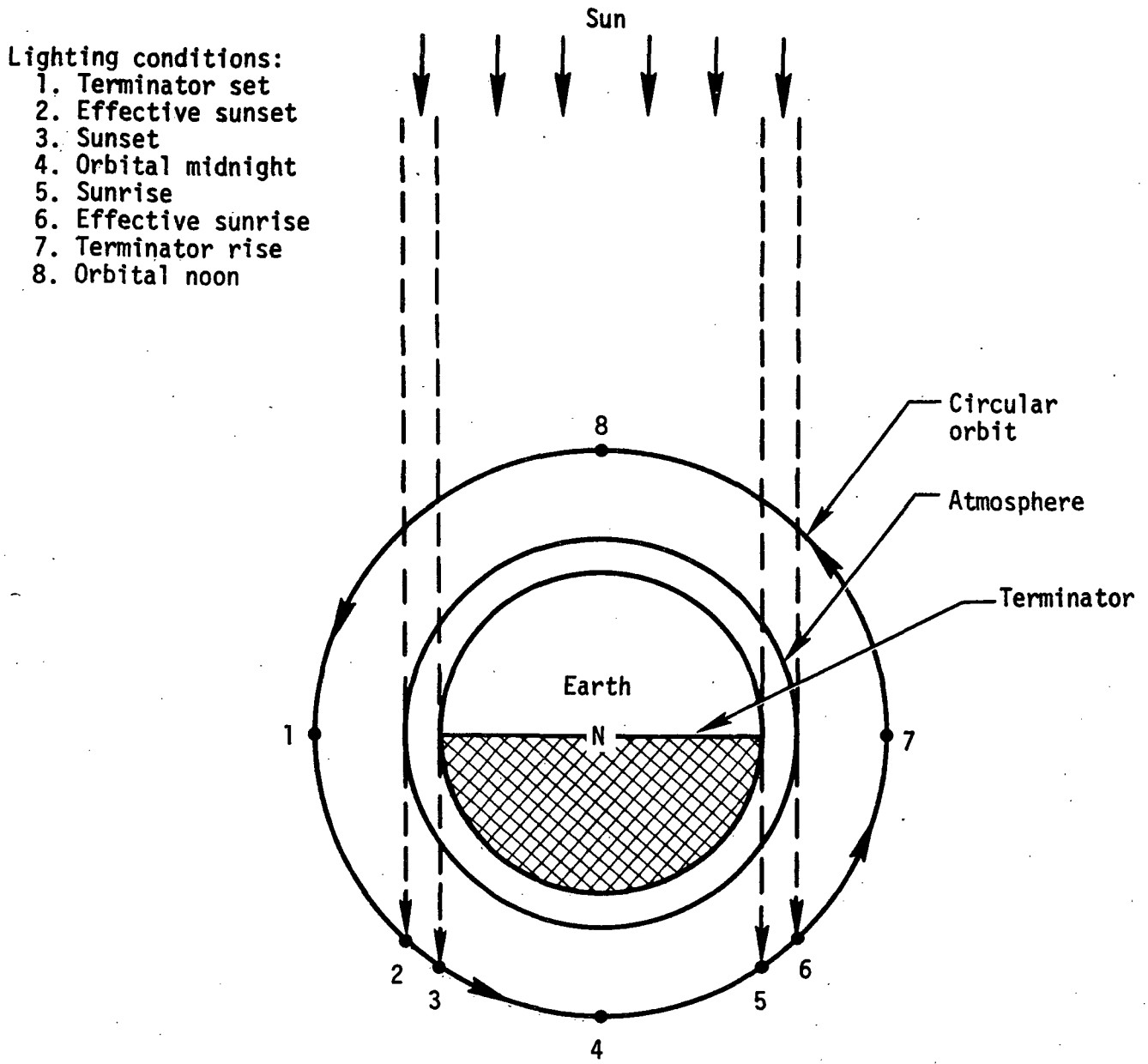


Figure 2-1.- Eight lighting conditions for the SRSS processor.

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM SRSS

5.1.1 Purpose

The routine SRSS serves as the main program of the sunrise/sunset (SRSS) processor. The routine provides the calls to appropriate subroutines for the computations made by the processor. These computations are

a. Orbit number
b. Ground elapsed time
c. Greenwich mean time
d. Latitude
e. Longitude
f. Altitude
g. Local vertical/local horizontal pitch angle to Sun
h. Local vertical/local horizontal yaw angle to Sun

The above eight quantities are computed at the eight lighting conditions

a. Terminator set
b. Effective sunset
c. Sunset
d. Orbital midnight
e. Effective sunrise
f. Sunrise
g. Terminator rise
h. Orbital noon

In addition, the routine SRSS calls the routine DSPLA, which displays the data and writes the data to an output file.

5.1.2 Functional Description

The Hewelett-Packard RTE-III routine RMPAR is called to obtain the logical unit number of the user's terminal. The FDS-1 utility routine XPGET is called to obtain the input data from the AWA. The state vector data are input directly through the AWA, or read from the DRDE file that has been written by the INVAR processor. The input data are converted to internal units where necessary. The output display device is set. The value of the angle used in the sunrise/sunset calculations is set corresponding to the desired condition of umbra, tangency, or penumbra.

The initial orbit number and the argument of latitude corresponding to the start time are determined for use in the orbit number computation. Next, a loop is executed until the current time exceeds the final time. Inside this loop, the eight quantities for the eight lighting conditions are computed. At the end of the loop, the computed quantities are displayed and written to the output DRDE

file when the output file option is turned on. Within the loop, the times of arrival at terminator set, orbital midnight, terminator rise, and orbital noon are computed by subroutine ARIV. Also within the loop, the times of arrival at effective sunrise/sunset and orbital sunrise/sunset are computed by subroutine RISE. Subroutine ARIV is described in section 5.2, and subroutine RISE is described in section 5.3.

After the arrival time is computed for each lighting condition, the latitude, longitude, and altitude are computed. The computed time of arrival is tested and the orbit number is increased by one if the next orbit has been started.

At the completion of the loop, if the abnormal termination flag is set, the termination variable is set to the error condition code of -32768. A final record is written to the output DRDE file. The first word of the final record contains -1. The FDS utility routine XPXIT is called to return control to the FDS Executive.

5.1.3 Assumptions and Limitations

Section 3.0 describes the assumptions and limitations of the processor.

5.1.4 Method

The routine SRSS is the control program for the processor. The orbit number, altitude, latitude, and longitude are computed by SRSS for each lighting condition. Subroutine LVLH, which is called by SRSS, constructs the local vertical/local horizontal transformation matrix. The local vertical/local horizontal pitch and yaw are computed by routine PYCAL, which is called by SRSS. Section 5.5 provides a description of subroutine PYCAL, and section 5.6 provides a description of subroutine LVLH.

To initialize the orbit number, subroutine ADVU is called to determine the argument of latitude at the start time. (See processor LOPT for a description of subroutine ADVU.) To ensure that the initial orbit number is before the first terminator set, the argument of latitude for the initial orbit is tested. If it is greater than 180 degrees, its value is reduced by 180 degrees.

The right ascension (δ) and declination (ϕ) are computed as

$$\delta = \tan^{-1} (-a_{32}/-a_{31}) - \omega_E t$$

$$\phi = \sin^{-1} (-a_{33})$$

where

a_{31}, a_{32}, a_{33} are the Z-axis of the local vertical/local horizontal matrix (computed by routine LVLH)

ω_E = rotation rate of the Earth

The altitude (h) is computed by

$$h = R_{orb} - R_E$$

where

R_{orb} = vehicle orbital radius

R_E = circular Earth radius

Table 5.1-I provides a list of mathematical code symbols versus internal code symbols.

5.1.5 Routine Input/Output Variables

The SRSS input/output variables are presented in table 5.1-II.

5.1.6 Functional Logic Flow

The functional logic flow for SRSS is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

Additional values may be displayed for debug purposes by setting the input interface table parameter PROCON(1) to a nonzero value. The debug values are displayed by routines RVECF (see processor LOPT for description) and ARIV (sec. 5.2).

5.1.8 Special Comments

None.

5.1.9 References

None.

TABLE 5.1-I.- MATHEMATICAL CODE SYMBOLS VERSUS INTERNAL CODE SYMBOLS

Math symbol	Internal code symbol
a	AMAT
δ	Array (3)
h	Array (4)
ω_E	ROMEGE
ϕ	Array (2)
R_E	CIRRAD
R_{orb}	ROUT
t	Array (1)

TABLE 5.1-II.- ROUTINE INPUT/OUTPUT VARIABLES

Routine SRSS

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ADAY	--	Real	I	sec	IT	!!GLCN(101)	Seconds per day
AHOUR	--	Real	I	sec	IT	!!GLCN(99)	Seconds per hour
ALTEFF	--	Real	I	feet	IT	--	Altitude above Earth's surface for use in effective sunrise/sunset calculations
AMILE	--	Real	I	feet	IT	!!GLCN(89)	Feet per nautical mile
AMIN	--	Real	I	sec	IT	!!GLCN(97)	Seconds per minute
ANGP	--	Real	I	rad	IT	PROCON(14)	Angle from Sun to terminator for penumbra solutions
ANGT	--	Real	I	rad	IT	PROCON(12)	Angle from Sun to terminator for tangency solutions
ANGU	--	Real	I	rad	IT	PROCON(10)	Angle from Sun to terminator for umbra solutions
CFA	--	Real	I	--	IT	!SESCN(27)	Conversion factor for angles (external units to internal)
CFD	--	Real	I	--	IT	!SESCN(29)	Conversion factor for distance

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.1-II.- Continued

Routine SRSS

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
CFT	--	Real	I	--	IT	!SESCN(31)	Conversion factor for time
CIRRAD	--	Real	I	feet	IT	!GLCN(21)	Circular Earth radius
GD2PI	2π	Dubl	I	--	IT	!!GLCN(68)	2π
GETF	--	Real	I	--	IT	--	Final time (hr, min, sec) relative to base time
GETS	--	Real	I	--	IT	--	Start time (hr, min, sec) relative to base time
GMTR	--	Real	I	--	IT	!SESCN(1)	Reference time for GET computations
GPI	π	Real	I	--	IT	!!GLCN(59)	π
G2PI	2π	Real	I	--	IT	!!GLCN(61)	2π
GLOCON	--	Free	I	--	IT	!!GLCN	Global constants array
ICR	--	Intg	I	--	IT	PROCON(8)	Cartridge reference number for DRDE
INORB	--	Intg	I	--	IT	--	Initial orbit count
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-II.- Continued

Routine SRSS

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IDAY	—	Intg	I	--	IT	!!SESCN(5)	Day of base date
IPRNTL	—	Intg	I	—	IT	PROCON(1)	Debug print flag
IVEC	—	2CH	I	--	IT	—	Vector source flag "IN" = single input state (SVIN) "DF" = vector DRDE as specified by VECFIL
IYEAR	—	Intg	I	--	IT	!SESCN(3)	Year of base date
MLINE	—	Intg	I	—	IT	PROCON(7)	Maximum number of lines per page
MONTH	—	Intg	I	--	IT	!SESCN(4)	Month of base date
NOBLKS	--	Intg	I	—	IT	PROCON(9)	Maximum number of blocks for the output file
OLU	--	Intg	I	—	IT	PROCON(2)	Output destination flag 0 = user's terminal >0 = logical unit of output device
OPT	—	2CH	I	—	IT	—	Option for specifying angle from Sun to terminator "UM" = umbra "TA" = tangency "PE" = penumbra
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-II.- Concluded

Routine SRSS

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
OUTFLG	--	2CH	I	--	IT	--	Flag to indicate whether DRDE is to be written "NO" = Do not write DRDE "YE" = Write DRDE
PROCON	--	Free	I	--	IT	--	SRSS constants array
RTOL	--	Real	I	feet	IT	PROCON(3)	Iteration tolerance for radius iteration
SCOEF	--	Real	I	--	IT	!SCOEF	Sun constants for analytic ephemeris model
SESCON	--	Free	I	--	IT	!SESCN	Session constants array
SVIN	--	Real	I	--	IT	--	Input invariant element state vector.
TTOL	--	Real	I	sec	IT	PROCON(5)	Iteration tolerance for time iteration.
VECFIL	--	6CH	I	--	IT	--	Name of invariant element DRDE
SUNFIL	--	6CH	O	--	IT	--	Name of output DRDE

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

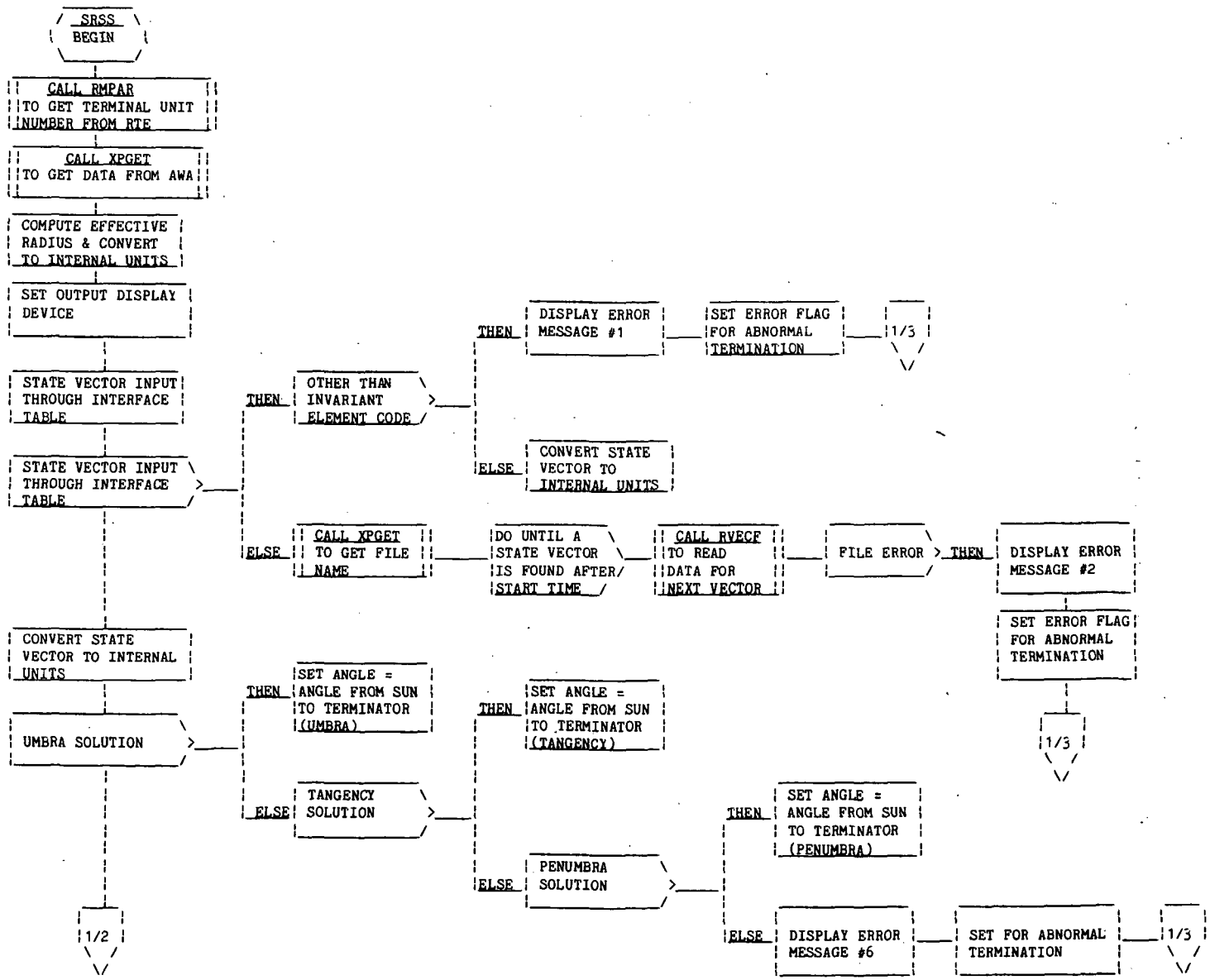


Figure 5.1-1.- SRSS functional logic flow.

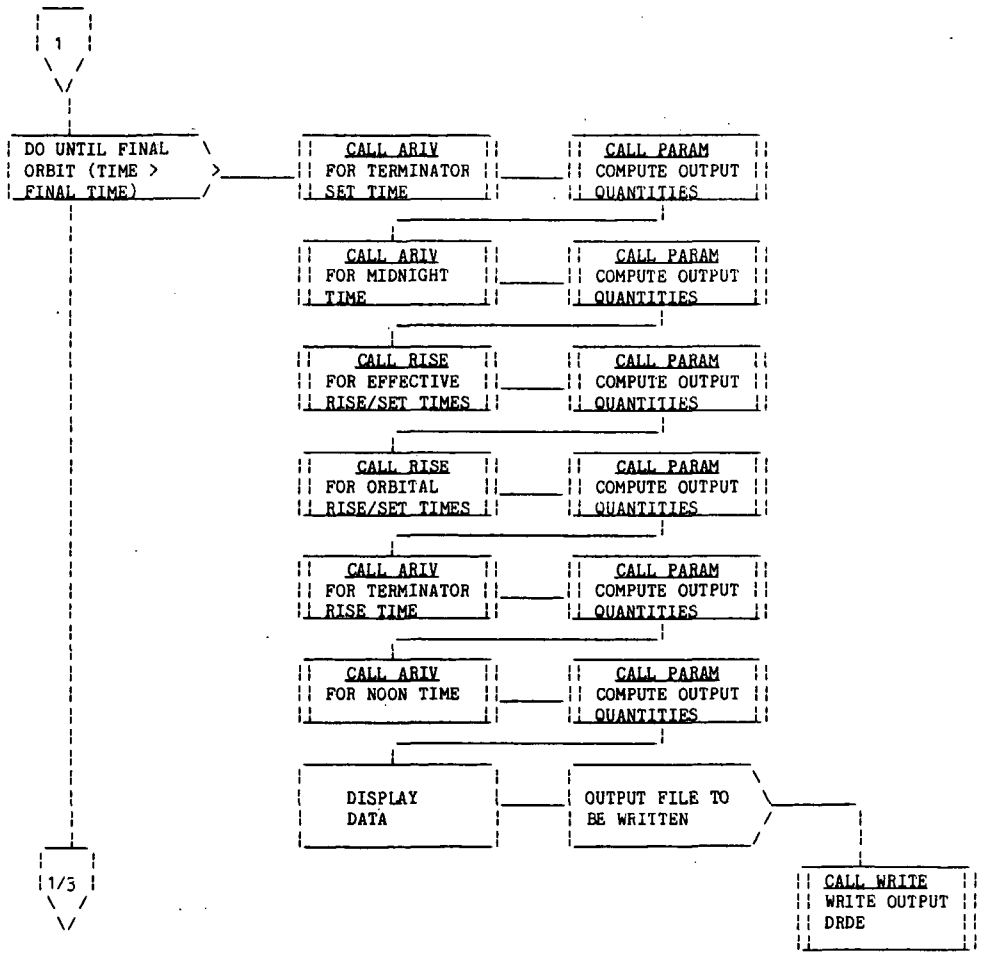


Figure 5.1-1.- Continued.

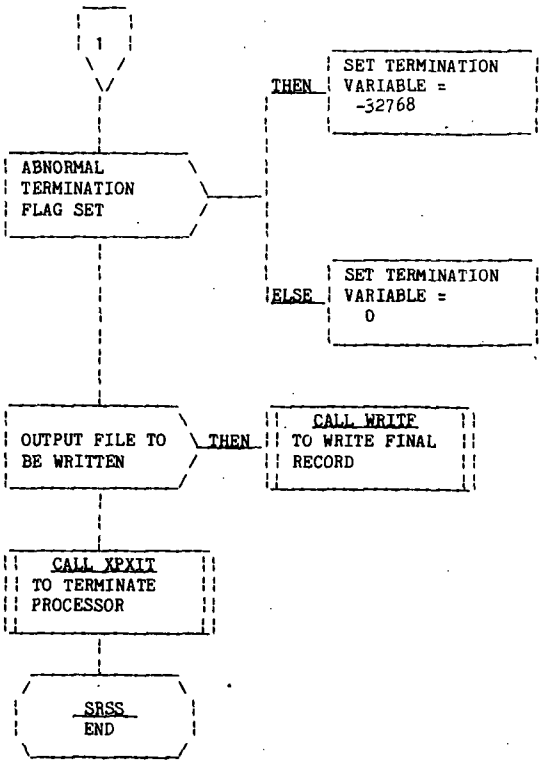


Figure 5.1-1.- Concluded.

5.2 ROUTINE NAME - ARIV

5.2.1 Purpose

The time of arrival subroutine (ARIV) computes the time of arrival of the vehicle at an input angle from the Sun or an input ground site. If the input angle is zero or pi, the arrival time is the closest point of approach (CPA) position or the midnight position. The argument of latitude, the orbital radius, and the position vector from the Earth to the Sun at the solution time are computed.

5.2.2 Functional Description

The subroutine ADVU is called to compute the argument of latitude at the current time. The argument of the ascending node is computed. The target option is tested for Earth or Sun. If the option is Earth, the vector to the ground site is computed. If the option is Sun, the FDS-1 utility routine SUN is called to compute the vector to the Sun, and the FDS-1 utility routine UNIT is called to unitize the Sun vector. The routine CPA is called to compute the closest point of approach. The argument of latitude is computed, and subroutine TAU is called to compute the time of arrival at the argument of latitude. (See processor LOPT for a description of subroutine TAU.)

5.2.3 Assumptions and Limitations

None.

5.2.4 Method

For a given time, routine ARIV uses routine CPA (sec. 5.4) to compute the argument of latitude (U_{CPA}) of the orbital closest point of approach to the Sun or an input ground site. If the requested angle is 0 or 180 degrees, the desired argument of latitude (UT) is either U_{CPA} or $U_{CPA} + 180$. If the desired angle is not 0 or 180, then the in-plane angle (γ) is computed as

$$\cos \gamma = \cos \alpha / \cos \beta$$

Where

α = the desired angle from the vehicle to the Sun or ground site.

β = the angle the Sun or ground site is out of the orbit plane (computed in routine CPA, section 5.4)

If the above equation results in a value whose absolute value is greater than 1, then no solution exists. The sign of γ is set to the same sign as α. The desired argument of latitude is computed as

$$U_T = U_{CPA} + \gamma$$

Routine TAU is used to find the time of arrival at the desired argument of latitude. This process is repeated with the new time until two successive times are computed with a change less than a tolerance.

5.2.5 Routine Input/Output Variables

The ARIV input/output variables are presented in table 5.2-I. The calling sequence is

```
CALL ARIV (TIMS,ANG,TOPT,GSIT,ISET,TTOL,TOUT,UOUT,ROUT,SCOEF,TVEC)
```

5.2.6 Functional Logic Flow

The functional logic flow for ARIV is presented in figure 5.2-1.

5.2.7 Diagnostics and Debug

When the optional debug flag (sec. 5.1) is turned on, additional values are displayed for debug purposes. The debug values displayed are the desired angle from the target, the current time, the computed time of arrival, the desired argument of latitude, the argument of latitude of the closest point of approach, the tolerance on the time iteration, and the angle used in computing the position vector from the Earth to the Sun.

5.2.8 Special Comments

None.

5.2.9 References

None.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine ARIV

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ANG	--	Real	I	rad	A		Desired angle from target
GSIT	--	Real	I	--	A		Geocentric radius, latitude, and longitude of ground site if TOPT = -1.
HDOT		Real	I		C		Rate of change of argument of node
HK		Real	I		C		Argument of ascending node at time of vector
IK		Real	I		C		Inclination angle
IPRNT	—	Intg	I		C		Debug print flag = 0, no debug print = 1, debug print
ISET	--	Intg	O	--	A		Solution indicator = 1, solution found = -1, no solution found
LUO	—	Intg	I	--	C		Alternate display device
PI	π	Real	I	--	C		Ratio of circle circumference to circle radius
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Continued

Routine ARIV

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ROMEGE		Real	I		C		Rotation rate of Earth
ROUT		Real	O	feet	A		Orbital radius at the solution time
SCOEF		Real	I	--	A		Solar coefficient array (from BASTM processor)
TIMS		Real	I	sec	A		Start time; the solution will be within $\pm \pi$ of the start time
TIMV		Real	I	sec	C		Time of vector
TOPT		Intg	I	--	A		Target option = 1, Sun = -1, ground site
TOUT		Real	O	sec	A		Solution time
TTOL		Real	I	sec	A		Tolerance for time iteration
TVEC		Real	O	--	A		Position vector from Earth to Sun
TWOPI	2π	Real	I	--	C		2*PI
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Concluded

Routine ARIV

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
UOUT		Real	O	rad	A		Argument of latitude at the solution time

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

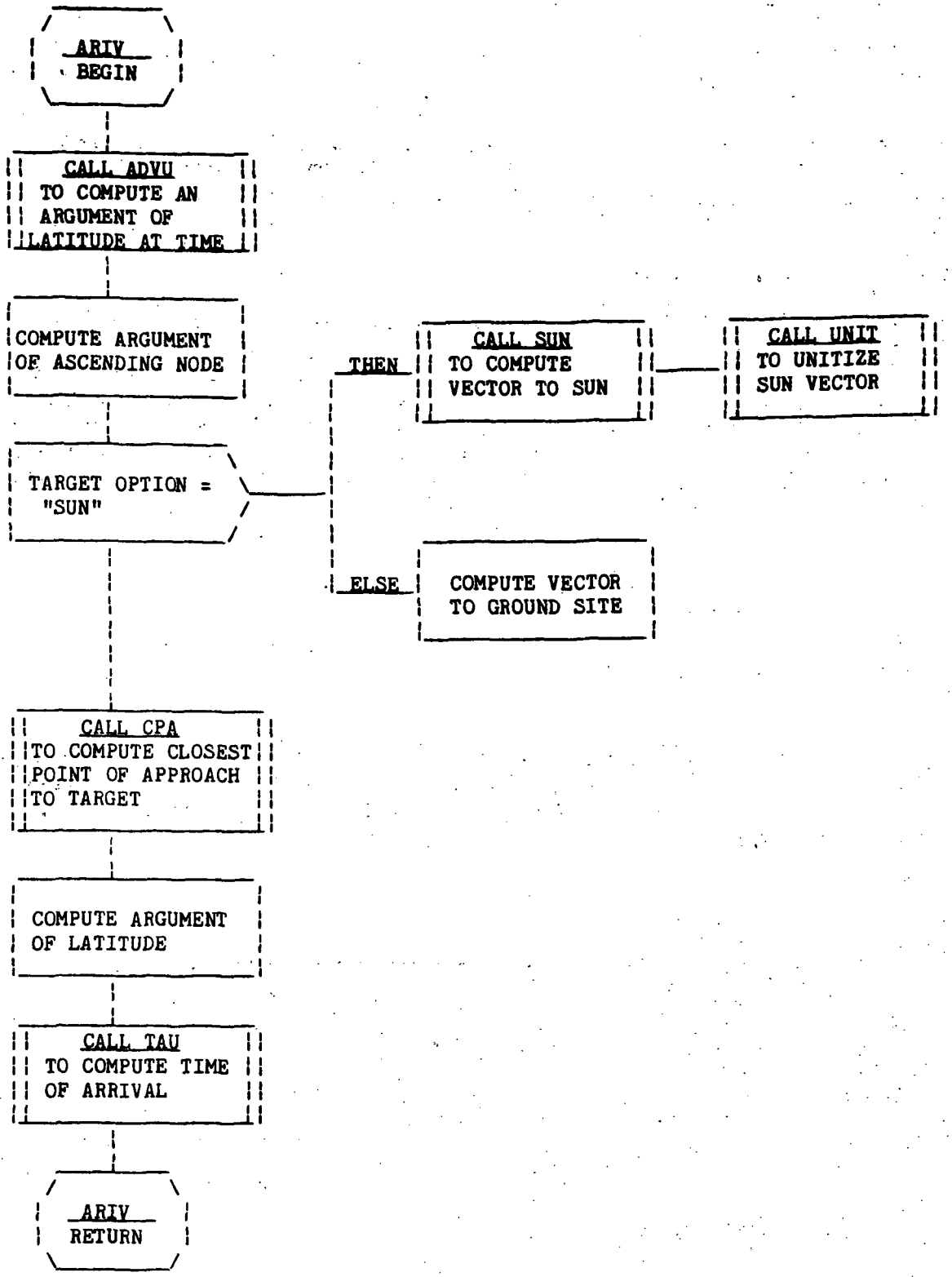


Figure 5.2-1.- ARIV functional logic flow.

5.3 ROUTINE NAME - RISE

5.3.1 Purpose

The sunrise and sunset subroutine (RISE) finds the orbital rise and set times with respect to the Sun or a geosynchronous position above the surface of the Earth.

5.3.2 Functional Description

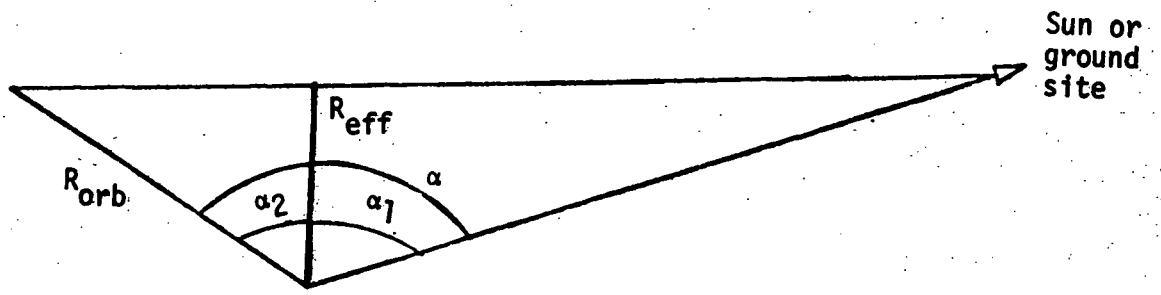
The computations are made inside a loop that is executed twice; once for sunset and once for sunrise. The argument of the ascending node is computed, and the target option is tested for an Earth site or the Sun. If the option is Earth, the vector to the ground site is computed. If the option is Sun, the FDS-1 utility routine UNIT is called to unitize the Sun vector. The routine CPA is then called to compute the closest point of approach, and the orbital radius is computed within a tolerance. The routine ARIV is called to compute the time of arrival at the target angle.

5.3.3 Assumptions and Limitations

None.

5.3.4 Method

Routine RISE computes the angle between the vehicle and the Sun or ground site at effective sunrise or sunset time and then calls routine ARIV to compute the time of arrival at this angle. The desired angle (α) is computed as shown in the following figure.



$$\alpha = \pm (\alpha_1 + \alpha_2)$$

where

α_1 = the angle between the vehicle and the terminator point

α_2 = the input angle between the terminator point and the Sun or ground site

The (+) sign is used for computing the sunset time, and the (-) sign is used for computing the sunrise time. α_2 is an input angle that is used to determine whether the solution represents umbra, penumbra, or a point of tangency. α_1 is computed by

$$\alpha_1 = \cos^{-1} (R_{eff}/R_{orb})$$

where

R_{eff} = input radius of the effective atmosphere

R_{orb} = radius of the vehicle

R_{orb} is initially approximated by the semimajor axis (a_k). R_{orb} is then recomputed by

$$R_{orb} = a_k (1 - e_k^2)/(1 + e_k \cos f)$$

where

a_k = semimajor axis

e_k = eccentricity

f = true anomaly

f is computed by

$$f = U_{CPA} + \alpha - g$$

where

U_{CPA} = argument of latitude at closest point of approach (computed by routine CPA)

g = argument of perigee

α_1 is then recomputed, and a corresponding new R_{orb} is computed. This process continues until the change in R_{orb} is less than a tolerance.

Table 5.3-I provides a list of mathematical code symbols versus internal code symbols.

5.3.5 Routine Input/Output Variables

The RISE input/output variables are presented in table 5.3-II. The calling sequence is

```
CALL RISE (TIMS,RADEFF,IOPT,GSIT,ANG,TTOL,RTOL,
           TSET,TRISE,RORB,SCOEF,TVEC)
```

5.3.6 Functional Logic Flow

The functional logic flow for RISE is presented in figure 5.3-1.

5.3.7 Diagnostics and Debug

None.

5.3.8 Special Comments

None.

5.3.9 References

None.

TABLE 5.3-I.- MATHEMATICAL CODE SYMBOLS VERSUS INTERNAL CODE SYMBOLS

Math symbol	Internal code symbol
α	TANG
α_1	ANG1
α_2	ANG
a_k	AK
e_k	EK
f	TA
g	GC
R_{eff}	RADEFF
R_{orb}	RORB
U_{CPA}	UCPA

TABLE 5.3-II.- ROUTINE INPUT/OUTPUT VARIABLES

Routine RISE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
AK		Real	I		C		Semimajor axis
ANG		Real	I	rad	A		Desired angle from target
DTWOPI	2π	Dubl	I	--	C		Double-precision 2π
EK		Real	I	rad	C		Eccentricity
GDOT		Real	I		C		Invariant element rate of change of perigee
GK		Real	I		C		Invariant element argument of perigee
GSIT	--	Real	I	--	A		Geocentric radius, latitude, and longitude of ground site if TOPT = -1
HDOT		Real	I		C		Rate of change of argument of node
HK		Real	I		C		Argument of ascending node at time of vector
IK		Real	I		C		Inclination angle
MK		Real	I		C		Mean anomaly

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.3-II.- Continued

Routine RISE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
RADEFF		Real	I		A		Effective radius
ROMEGE		Real	I		C		Rotation rate of Earth
RORB		Real	O	feet	A		Orbital radius at the solution time
RTOL		Real	I	feet	A		Tolerance on orbital radius
SCOEF		Real	I	--	A		Solar coefficient array
TIMS		Real	I	sec	A		Start time; the solution will be within $\pm \pi$ of the start time
TIMV		Real	I	sec	C		Time of vector
TRISE		Real	O	sec	A		Solution time of sunrise
TSET		Real	O	sec	A		Solution time of sunset
TTOL		Real	I	sec	A		Tolerance for time iteration
TVEC		Real	O	--	A		Position vector from Earth to Sun

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.3-II.- Concluded

Routine RISE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
TWOPI	2π	Real	I	--	C		2*π

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

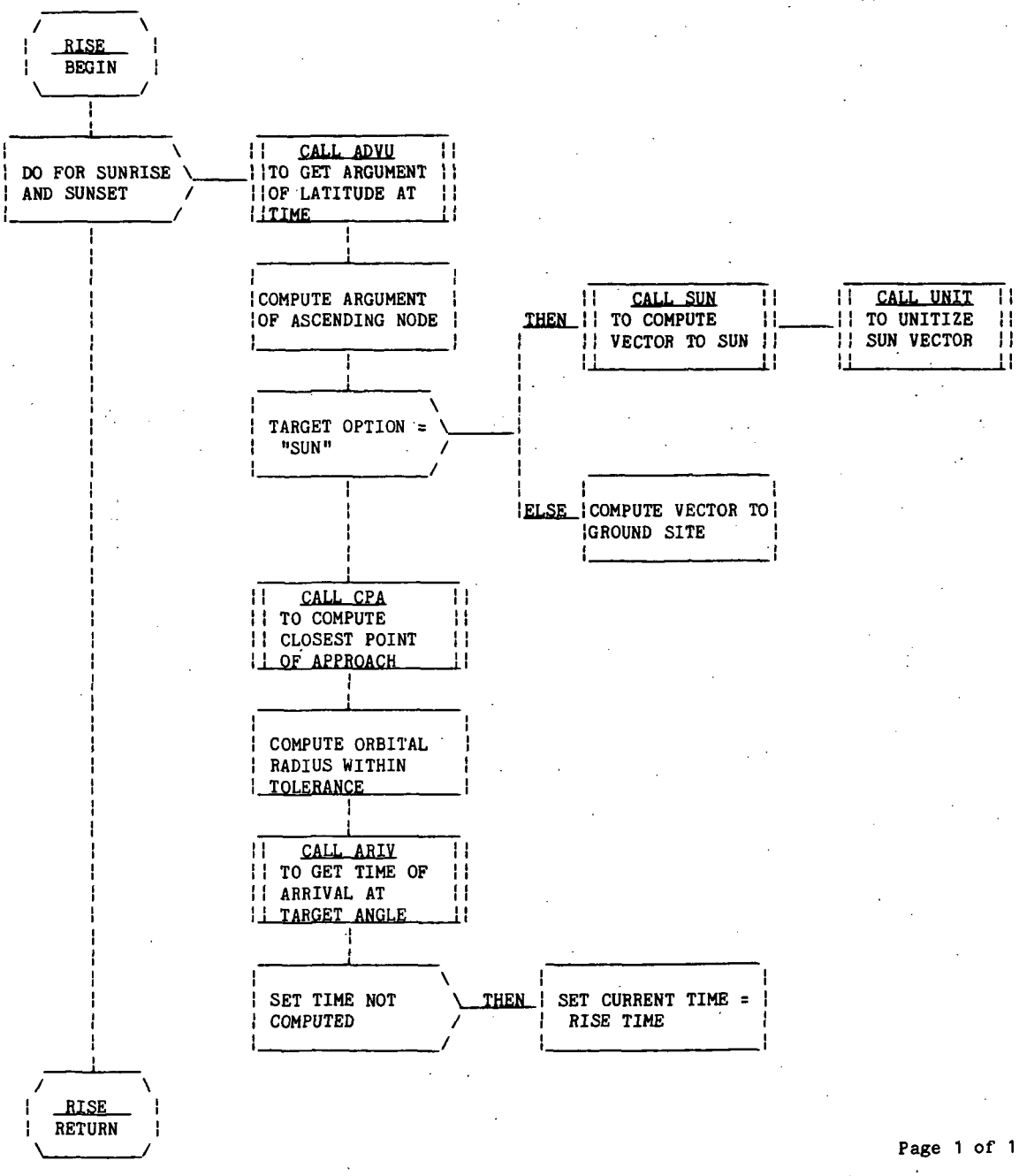


Figure 5.3-1.- RISE functional logic flow.

5.4 ROUTINE NAME - CPA

5.4.1 Purpose

The closest point of approach subroutine (CPA) computes the closest point of approach of an orbiting vehicle to a ground site.

5.4.2 Functional Description

None.

5.4.3 Assumptions and Limitations

None.

5.4.4 Method

The method used in this subroutine is taken in part from the method used for the CPA computation in the main program of processor LOPT. (Refer to the Closest Point of Approach section in the main program text.)

5.4.5 Routine Input/Output Variables

The CPA input/output variables are presented in table 5.4-I. The calling sequence is

```
CALL CPA (H,TAR,ARGL,SINB,TWOPI)
```

5.4.6 Functional Logic Flow

None.

5.4.7 Diagnostics and Debug

None.

5.4.8 Special Comments

None.

5.4.9 References

None.

TABLE 5.4-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine CPA

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ARGL		Real	O	--	A		Argument of latitude at closest point of approach
H		Real	I		A		Angular momentum vector
SINB		Real	O	--	A		(H vector) · (S vector)
TAR		Real	I	--	A		Position vector from Earth to Sun
TWOPI	2π	Real	I	--	A		$2*\pi$

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

5.5 ROUTINE NAME - PYCAL

5.5.1 Purpose

The pitch/yaw calculation subroutine (PYCAL) makes the pitch/yaw calculations for a given local vertical/local horizontal (LVLH) transformation matrix, and for a vector describing the position of the Earth to the Sun.

5.5.2 Functional Description

Routine PYCAL initially defines some intermediate variables using the LVLH transformation matrix and the Sun TEG vector. The pitch and yaw values are then directly determined through trigonometric manipulation.

5.5.3 Assumptions and Limitations

None.

5.5.4 Method

The Sun vector is a three-real-word linear array that contains the X, Y, and Z coordinates in the Time Equatorial Greenwich (TEG) coordinate system. The Sun vector is developed in the FDS-1 utility processor SUN. This Sun vector is initially combined with the LVLH transformation matrix (defined in subroutine LVLH) as follows:

$$X_1 = SUN(1)\ AMAT(1,1) + SUN(2)\ AMAT(1,2) + SUN(3)\ AMAT(1,3)$$

$$Y_1 = SUN(1)\ AMAT(2,1) + SUN(2)\ AMAT(2,2) + SUN(3)\ AMAT(2,3)$$

$$Z_1 = SUN(1)\ AMAT(3,1) + SUN(2)\ AMAT(3,2) + SUN(3)\ AMAT(3,3)$$

where AMAT is a 3x3 LVLH transformation matrix

Pitch is then defined as

$$Pitch = ATANZ\ (-Z_1,\ X_1)$$

Yaw is defined as

$$Yaw = \text{arc sine}\ (Y_1)$$

5.5.5 Routine Input/Output Variables

The PYGAL input/output variables are presented in table 5.5-I.

5.5.6 Functional Logic Flow

None.

5.5.7 Diagnostics and Debug

None.

5.5.8 Special Comments

None.

5.5.9 References

None.

TABLE 5.5-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine PYCAL

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
SUNA	--	Real	I	--	A	--	Position vector from the Earth to the Sun in the TEG coordinate system.
AMAT	--	Real	I	--	A	--	LVLH transformation matrix
PITCH	--	Real	O	--	A	--	Calculated pitch.
YAW	--	Real	O	--	A	--	Calculated yaw.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

5.6 ROUTINE NAME - LVLH

5.6.1 Purpose

The local vertical/local horizontal calculation (LVLH) subroutine constructs a local vertical/local horizontal transformation matrix from the argument of the node, argument of latitude, and the sine and cosine of inclination.

5.6.2 Functional Description

The LVLH subroutine is a straightforward computation of the local vertical/local horizontal transformation matrix. Initially, the sine and cosine of the argument of the ascending node and the argument of latitude are determined to facilitate computation of the matrix. The FORTRAN library subroutines SIN and COS are used to obtain the sine and cosine of these values.

5.6.3 Assumptions and Limitations

None.

5.6.4 Method

The local vertical/local horizontal transformation matrix is computed as follows:

$$[A] = \begin{bmatrix} -\sin\Omega\cos\omega - \cos\Omega\cos i\sin\omega & -\sin\Omega\sin\omega + \cos\Omega\cos i\cos\omega & \cos\Omega\sin i \\ -\sin i\sin\omega & \sin i\cos\omega & -\cos i \\ -\cos\Omega\cos\omega + \sin\Omega\cos i\sin\omega & -\cos\Omega\sin\omega - \sin\Omega\cos i\cos\omega & -\sin\Omega\sin i \end{bmatrix}$$

where i is the inclination angle

ω is the argument of the ascending node

Ω is the argument of latitude

Table 5.6-I provides a list of mathematical code symbols versus internal symbols.

5.6.5 Routine Input/Output Variables

The LVLH input/output variables are presented in table 5.6-II. The subroutine calling sequence is

```
CALL LVLH (HC,COSI,SINI,UC,AMAT)
```

5.6.6 Functional Logic Flow

The functional logic flow for LVLH is presented in figure 5.6-1.

5.6.7 Diagnostics and Debug

None.

5.6.8 Special Comments

None.

5.6.9 References

None.

TABLE 5.6-I.- MATHEMATICAL CODE SYMBOLS VERSUS INTERNAL CODE SYMBOLS

Math symbol	Internal code symbol
Ω	UC
ω	HC
cos Ω	COSU
cos ω	COSH
sin Ω	SINU
sin ω	SINH

TABLE 5.6-II.- ROUTINE INPUT/OUTPUT VARIABLES

Routine LVLH

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
HC	Ω	Real	I	--	A	--	Argument of ascending node
COSI	COS(I)	Real	I	--	A	--	Cosine of inclination
SINI	SIN(I)	Real	I	--	A	--	Sine of inclination
UC	ω	Real	I	--	A	--	Argument of latitude
AMAT	—	Real	O	--	A	--	3 x 3 transformation matrix
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory	

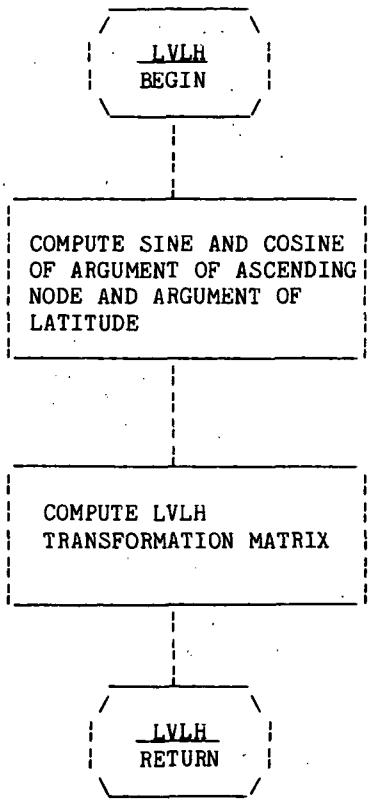


Figure 5.6-1.- LVLH functional logic flow.

5.7 ROUTINE NAME - DSPLA

5.7.1 Purpose

The display (DSPLA) subroutine displays a heading describing one of eight lighting conditions (terminator set, effective sunset, sunset, orbital midnight, effective sunrise, sunrise, terminator rise, and orbital noon) and the eight quantities computed for the lighting condition. The quantities displayed are: orbital number, ground elapsed time (GET) (in days, hours, minutes, and seconds), Greenwich mean time (GMT) (in months, days, hours, minutes, and seconds), latitude, longitude, altitude, local vertical/local horizontal pitch and yaw.

5.7.2 Functional Description

The DSPLA subroutine initially calls the FDS-1 utility subroutine DATE to determine the GMT month, day, hour, minute and second. FDS-1 utility subroutine GMTIM is then called to compute the GET in days, hours, minutes, and seconds. In both instances, the seconds are converted from real to integer by the FORTRAN IFIX library routine.

The DSPLA routine then displays the heading, orbit number, GET (day, hour, minute and second), GMT (month, day, hour, minute and second), latitude, longitude, altitude, LVLH pitch, and LVLH yaw. All displays are sent to the logical unit number specified in the parameter list.

5.7.3 Assumptions and Limitations

None.

5.7.4 Method

None.

5.7.5 Routine Input/Output Variables

The DSPLA input/output variables are presented in figure 5.7-1. The subroutine calling sequence is

```
CALL DSPLA (LUO,IORB,HEAD,AR,GMTR,BDATE,AMILE,CFA,PI,TWOPI)
```

5.7.6 Functional Logic Flow

The functional logic flow for DSPLA is presented in figure 5.7-1.

5.7.7 Diagnostics and Debug

None.

5.7.8 Special Comments

None.

5.7.9 References

None.

TABLE 5.7-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine DSPLA

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
AMILE		Real	I	ft/ n. mi.	A		Conversion factor; feet per nautical mile
AR		Mix	I	--	A		Array containing data to be displayed
BDATE		Real	I	--	A		Year, month, day of base date
CFA		Real	I	--	A		Conversion factor for angles
GMTR		Real	I	sec	A		Reference Greenwich mean time
HEAD			I	--	A		Array containing alphanumeric description of display line
IORB	--	Intg	I	--	A		Orbit number
LUO	--	Intg	I	--	A		Alternate display device
PI	π	Real	I	--	A		Ratio of circle circumference to diameter
TWOPI	2π	Real	I	--	A		$2*\pi$
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

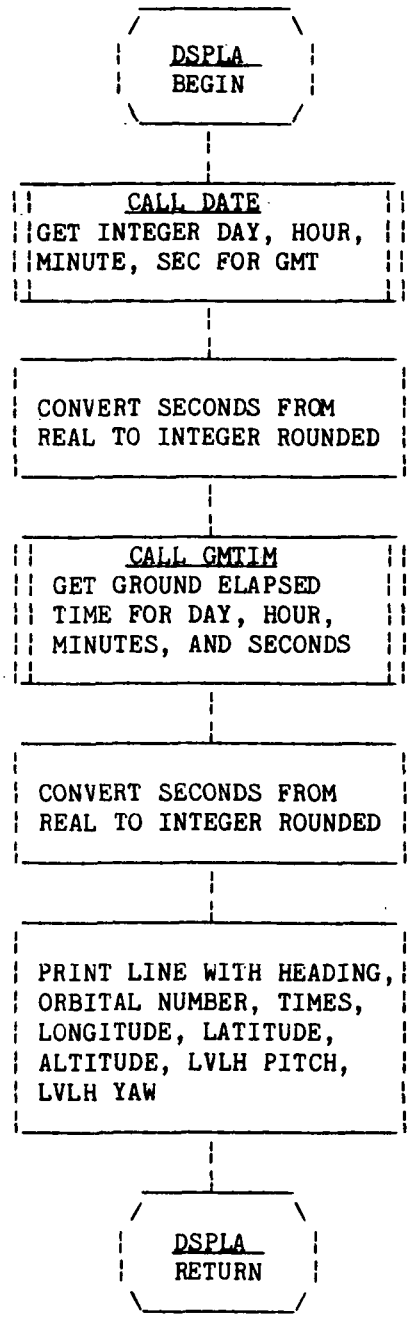


Figure 5.7-1.- DSPLA functional logic flow.

SHUTTLE/SUS BURN RELATIVE MOTION PROCESSOR (SSBRM)

TO BE SUPPLIED

SUN-SYNCHRONOUS ORBITS PROCESSOR (SSYN)

1.0 PURPOSE

The SSYN processor computes the semimajor axis and the inclination required to achieve a Sun-synchronous orbit.

2.0 FUNCTIONAL DESCRIPTION

The SSYN processor has the capability of computing two types of Sun-synchronous orbits. Given the desired apogee and perigee altitudes as input, SSYN will compute the inclination necessary to achieve a Sun-synchronous orbit, regardless of the groundtrack repeatability. If, instead, the number of orbits in which the groundtrack repeats and the day on which the groundtrack repeats are given, SSYN will compute the semimajor axis and inclination that will allow the groundtrack to repeat in exactly 24-hour multiples. This type orbit is called a Sun-synchronous repeating orbit.

3.0 ASSUMPTIONS AND LIMITATIONS

The nodal precession rate of the vehicle orbit must be equal to the approximate Earth's angular travel rate around the Sun (i.e., $2\pi/365.25$). Consequently, all Sun-synchronous orbits have inclinations that are greater than 90^{o}.

The maximum value for the semimajor axis occurs when the inclination is 180^{o}.

SSYN does not generate a vehicle state vector.

When ORBID = "CSR" is requested, a table is generated containing information about circular orbits that are Sun-synchronous and have repeating groundtracks. The parameters START, END, and STEP control the number of orbits that are generated. SSYN assumes that START $\leq$ END, and STEP = 1, 2, 3,...,n.

The orbital rates parameters, apsidal precession, and nodal period are calculated using invariant elements with the zonal harmonics incorporated. Similarly, the equation for nodal precession incorporates the zonal harmonics.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The definition of the SSYN processor interface table parameters is provided in table 4-I. GLOCON is a set of constants, universal to all the processors, that is maintained in the master data base. The SSYN processor accesses this array for the constants that it requires. The default values are stored in !!GLCN.

SSYN accesses the SESCON array to obtain the session related constants that are generated by the user upon execution of the system utility processor BATSM. The default values are stored in !SESCN and are standard for all processors.

Specific constants and tolerances required by SSYN are maintained in PROCON. These parameters are used mainly for iteration tolerances and internal print analysis.

ORBID is the identification code for the type of Sun-synchronous orbit desired. Either a Sun-synchronous orbit (not necessarily circular or groundtrack repeating) or a circular, Sun-synchronous, repeating orbit may be chosen.

DAY is the integer value for the day on which it is desired to have the groundtracks repeat. The base day has a value of zero. Thus, for a one-day repeating orbit, DAY needs a value of 1.

When the circular, Sun-synchronous, repeating groundtrack option is chosen, a table of solutions is generated. The range of solutions is determined by the input parameters START, END, and STEP. START is the starting orbit number for the range of solutions. A semimajor axis and inclination is computed that allows a vehicle to repeat its groundtrack in START orbits on DAY day.

Similarly, END is the last orbit number for which a solution is calculated.

STEP is the orbit number increment within the interval between START and END. Thus, the table of solutions is generated over the following range of orbit numbers (NUMORB):

NUMORB = START, START + STEP, START + (2) (STEP), START + (3) (STEP), ..., END

The default value for STEP is 1.

HA and HP are the height of apogee and height of perigee, respectively, measured above a reference radius (the equatorial Earth radius for a Fischer ellipsoid that is input through GLOCON). They are used as input when the Sun-synchronous orbit option is requested.

b. Processor interface table data array definitions - The definition of the input/output data arrays appearing in the SSYN interface table is provided in table 4-II.

c. Processor interface table data file definitions - None.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definitions - Two displays are generated by SSYN (one each for the options defined by ORBID). The format of the display for ORBID = "SSO" is shown in table 4-III(a), an example is shown in table 4-III(b), and a definition of the display variables is provided in table 4-III(c). All the display parameters are in the user-selected

external units with the following exceptions: TSID and TNODE are in minutes. Also, the reference radius is in internal units. Note that R is the radius magnitude and A is the semimajor axis.

The format of the display for ORBID = "CSR" is shown in table 4-III(d), an example is shown in table 4-III(e), and a definition of the display variables is provided in table 4-III(f).

f. Processor message table - In general, SSYN does not generate any messages to the user during execution. However, in the cases of certain types of user input errors, an error message will be displayed as information to the user. Depending upon the severity of the error, execution may be terminated at that point. The messages the user may expect in those cases are shown in table 4-IV.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-V.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR SSYN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
GLOCON	AWA	Free	I	180	180	!!GLCN	Global constants array, master data base element
SESCON	AWA	Free	I	90	90	!SESCN	Session constants array
PROCON	AWA	Free	I	20	20		SSYN constants and tolerance
ORBID	AWA	6CH	I	3	1		Orbit type identification code "SSO"; Sun-synchronous orbit "CSR"; Circular, Sun-synchronous, repeating orbits
DAY	AWA	Intg	I	1	1		Day (from base day of zero) on which to compute the "CSR" orbits
START	AWA	Intg	I	1	1		Starting orbit number for table (used when ORBID = "CSR")
END	AWA	Intg	I	1	1		Ending orbit number for table (used when ORBID = "CSR")
STEP	AWA	Intg	I	1	1	1	Orbit number increment for table between START and END (used when ORBID = "CSR")

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- PROCESSOR INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR SSYN

Array name	Index location	Default value	Definition
GLOCON	(1) . . . (180)	!!GLCN	Global constants array, master data base element; see table 7.2-III in JSC IN 78-FM-60, volume I for definition of contents.
SESCON	(1) . . . (90)	!SESCN	Input/output session constants array; see table 7.2-II(a) in JSC IN 78-FM-60, volume I for definition of contents.
PROCON	(1)	0	Output unit on which to write all print; the default value is a flag to use the user terminal; a positive value identifies the output.
	(2)	0	Not used
	(3)	0	Internal print flag option = 0 no internal print = 1 internal print
	(4)	0	Not used
	(5)	20	Maximum iterations limit
	(6)	0	Not used
	(7)	0	Not used
	(8)	0	Not used
	(9)	2.0E-4	Transfer angle tolerance, radians
	(11)	6.0E-4	Eccentricity tolerance
	(13)	1.0E-4	Convergence tolerance
	(15)	0	Not used
	(16)	0	Not used
	(17)	0	Not used
	(18)	0	Not used
	(19)	250.	Radius tolerance, feet

TABLE 4-III.- PROCESSOR DISPLAYS AND DISPLAY PARAMETER DEFINITION TABLES

(a) Sun-synchronous orbit

PROCESSOR SSYN

```
                             SUN-SYNCHRONOUS ORBIT

****** INITIAL INPUT TO PROGRAM ******
ORBID,DAY                            AAAAAA                     XXX
START,END,STEP                       XXX          XXX           XXX
HA,HP                        +XXXXXXXXX.X       +XXXXXXXXX.X
ALTITUDE MEASURED ABOVE SPHERE WITH RADIUS OF ±XXXXXXXXX.X

                     ******** ORBITAL PARAMETERS ********
HA,HP                                +XXXXXX.XXXX        +XXXXXX.XXXX
INCLINATION                          +XXXX.XXXXXX
R,ALTITUDE,ECCENTRICITY              +XXXXXXX.XXXX       +XXXXX.XXXXX       +.XXXXXXXXXE+XX
A,MEAN A                             +XXXXXXX.XXXX       +XXXXXXX.XXXX
MEAN MOTION                          +XXX.XXXXXXX
OMEGAD,FREQ.RECUR.NODE               +X.XXXXXXXXX        +XXXXXX.XXXX
TSID,TNODE                           +XXXX.XXXXXX        +XXXX.XXXXXX
```

TABLE 4-III.- Continued

(b) Example of the Sun-synchronous orbit

SUN-SYNCHRONOUS ORBIT

```
****** INITIAL INPUT TO PROGRAM ******
 ORBID,DAY                          SSO                0
 START,END,STEP                       0        0         1
 HA,HP                              561.9              200.0
 ALTITUDE MEASURED ABOVE SPHERE WITH RADIUS OF  20925740.0

                        ******** ORBITAL PARAMETERS ********
 HA,HP                               561.9000          200.0000
 INCLINATION                        98.200302
 R,ALTITUDE,ECCENTRICITY            3643.9341         200.00000      .48302650E-01
 A,MEAN A                           3828.8794         3824.9966
 MEAN MOTION                        3.6402826
 OMEGAD,FREQ.RECUR.NODE            .000684455         1440.0000
 TSID,TNODE                         98.893402         98.952072
```

TABLE 4-III.- Continued

(c) Display parameter definition table for the Sun-synchronous orbit display

PROCESSOR SSYN

SUN-SYNCHRONOUS ORBIT	
Display parameter label	Parameter definition
HA	Height of apogee
HP	Height of perigee
INCLINATION	Inclination of the orbit plane
R	Radius magnitude
ALTITUDE	Altitude above a reference equatorial radius (Fischer ellipsoid)
ECCENTRICITY	Eccentricity of orbit
A	Semimajor axis
MEAN A	Mean semimajor axis
MEAN MOTION	Mean motion of the orbit
OMEGAD	Nodal precession rate of orbit
FREQ. RECUR.	Time required for the longitude of the ascending node to recur
NODE	
TSID	Inertial period, minutes
TNODE	Nodal period, minutes

TABLE 4-III.- Continued

(d) Circular Sun-synchronous repeating orbits

PROCESSOR SSYN

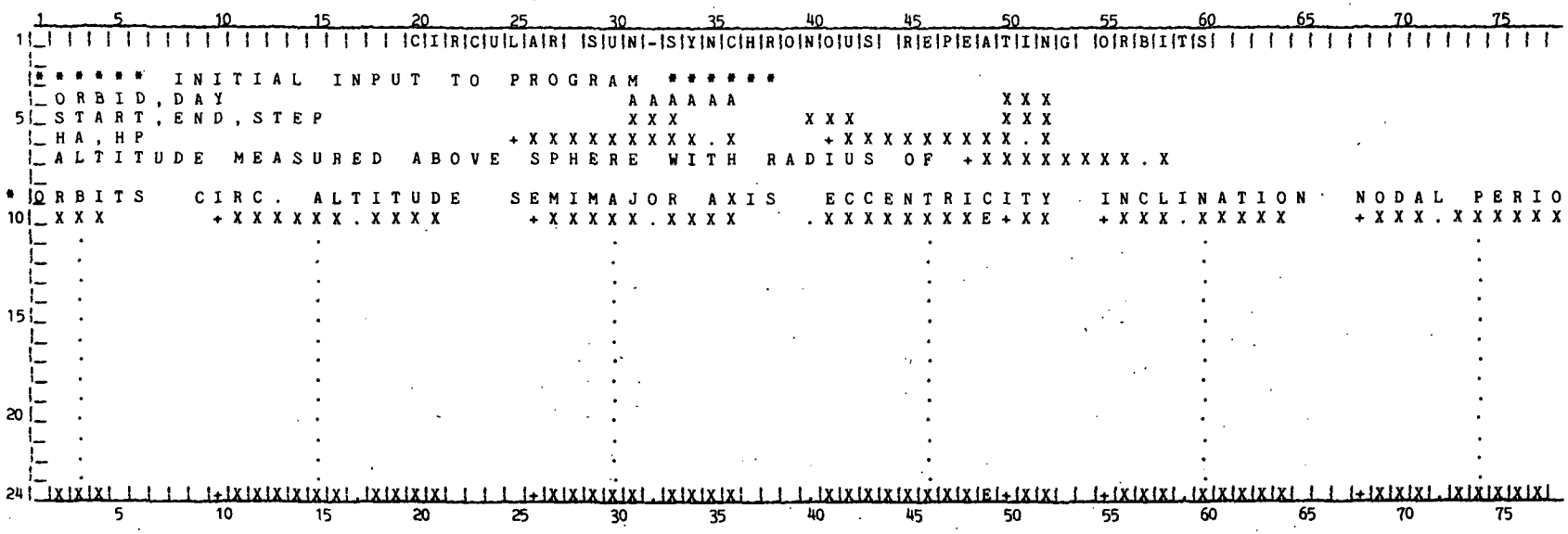


TABLE 4-III.- Continued

(e) Example of the circular Sun-synchronous repeating orbits display

CIRCULAR SUN-SYNCHRONOUS REPEATING ORBITS

****** INITIAL INPUT TO PROGRAM ******
ORBID,DAY CSR 3
START,END,STEP 19 51 1
HA,HP .0 .0
ALTITUDE MEASURED ABOVE SPHERE WITH RADIUS OF 20925740.0

ORBITS	CIRC. ALTITUDE	SEMIMAJOR AXIS	ECCENTRICITY	INCLINATION	NODAL PERIOD
19	3221.9976	6668.7979	.42986870E-03	173.56439	227.370209
20	2997.4565	6443.9395	.39553642E-03	151.96085	216.000519
21	2790.9839	6237.2275	.37026405E-03	142.03040	205.715240
22	2600.3188	6046.3809	.35190582E-03	135.04334	196.361110
23	2423.5937	5869.5137	.33831596E-03	129.66385	187.825745
24	2259.2769	5705.0889	.32913685E-03	125.32106	179.999329
25	2106.0410	5551.7705	.32353401E-03	121.72122	172.800354
26	1962.6824	5408.3506	.32067299E-03	118.67952	166.153778
27	1828.2681	5273.8896	.32007694E-03	116.07433	160.000580
28	1701.8914	5147.4795	.32138824E-03	113.81775	154.285797
29	1582.8284	5028.3926	.32413006E-03	111.84625	148.965057
30	1470.4534	4916.0020	.32854080E-03	110.11145	144.000305
31	1364.1294	4809.6689	.33366680E-03	108.57454	139.354462
32	1263.3889	4708.9248	.34010410E-03	107.20595	134.999451
33	1167.7659	4613.3018	.34737587E-03	105.98119	130.908386
34	1076.8933	4522.4326	.35500526E-03	104.88083	127.059464
35	990.3289	4435.8760	.36370754E-03	103.88751	123.428665
36	907.8018	4353.3584	.37276745E-03	102.98813	119.999893
37	829.0112	4274.5791	.38230419E-03	102.17108	116.756485
38	753.7098	4199.2920	.39243698E-03	101.42674	113.684433
39	681.6349	4127.2314	.40280819E-03	100.74652	110.769608
40	612.5645	4058.1763	.41353703E-03	100.12326	108.000046
41	546.3274	3991.9570	.42486191E-03	99.55098	105.366104
42	482.7177	3928.3647	.43606758E-03	99.02414	102.857025
43	421.6049	3867.2705	.44775009E-03	98.53835	100.465530
44	362.8022	3808.4868	.45967102E-03	98.08922	98.182144
45	306.1713	3751.8755	.47183037E-03	97.67322	95.999680
46	251.6239	3697.3486	.48434734E-03	97.28743	93.913010
47	199.0061	3644.7505	.49674511E-03	96.92883	91.914627
48	148.2401	3594.0049	.50938129E-03	96.59515	90.000244
49	99.1930	3544.9790	.52237511E-03	96.28397	88.163437
50	51.7851	3497.5913	.53524971E-03	95.99339	86.400040
51	5.9334	3451.7607	.54848194E-03	95.72173	84.705841

TABLE 4-III.- Concluded

(f) Display parameter definition table for the circular Sun-synchronous repeating orbits display

PROCESSOR SSYN

CIRCULAR SUN-SYNCHRONOUS REPEATING ORBITS	
Display parameter label	Parameter definition
ORBITS CIRC.	Number of orbits in which groundtracks will repeat
ALTITUDE	Circular altitude (above Fischer ellipsoid reference radius) of orbit
SEMIMAJOR AXIS	Semimajor axis of orbit
ECCENTRICITY	Eccentricity
INCLINATION	Inclination of orbit
NODAL PERIOD	Time to travel from one ascending node to next, minutes

TABLE 4-IV.- PROCESSOR MESSAGE TABLE

PROCESSOR SSYN

MSG no.	Message ID block	Message text block and explanation
1	*SSYN*	*SUNIN* "AAAAAA" IS INVALID INPUT FOR "BBBBBB". Meaning: The user has supplied an incorrect orbit type code. Severity: The processor will terminate, and control will be returned to the FDS Executive. Action required by user: Use the Interface Table Editor to correct the erroneous code; then resume execution.
2	*SSYN*	*SYNC* INVALID SOLUTION. A .GT. AMAX. PROCESSOR CONTINUING. Meaning: The semimajor axis exceeded the maximum allowed. An invalid solution for inclination greater than 180 deg. was calculated. Severity: If more data points have been requested, the processor will continue. Action required by user: None
3	*SSYN*	*SYNC* MAXIMUM ITERATIONS. ITER = XX. Meaning: SSYN did not converge on a solution for semimajor axis and inclination. Severity: Warning only. The last solution calculated will be displayed. The processor will continue. Action required by user: Check iteration tolerances that are input in the PROCON array.

TABLE 4-V.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR SSYN

Processor name	Processor abstract prompt (maximum 256 characters)
SSYN	SSYN computes the semimajor axis and inclination required to achieve a Sun-synchronous orbit.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
GLOCON	Global constants array, master data base element normally defualted to !!GLCN
SESCON	Session constants array, normally defaulted to !SESCN
PROCON	SSYN processor constants and tolerances
ORBID	Orbit-type ID code "SSO" = Sun-synchronous orbit "CSR" = Circular, Sun-synchronous repeating orbits
DAY	Day on which to compute the repeating orbits
START	Starting orbit number for range of solutions in display when ORBID = "CSR"
END	Stopping orbit number for range of solutions in display when ORBID = "CSR"
STEP	Orbit number increment over range of solutions in display when ORBID = "CSR"
HA	Height of apogee when ORBID = "SSO"
HP	Height of perigee when ORBID = "SSO"

5.0 PROCESSOR ROUTINES

The only available routine documentation is contained on the comment cards in the software listing.

STATION CONTACT PROCESSOR (STACN)

1.0 PURPOSE

The STACN applications processor computes the time at which an orbiting vehicle is above an input elevation angle with respect to a groundtracking station.

2.0 FUNCTIONAL DESCRIPTION

There are two analytic methods and one step function method of determining the times when an orbiting vehicle is above the desired elevation angle. The step function method simply steps along the trajectory and computes the elevation angle at each point. The times of acquisition of signal (AOS) and loss of signal (LOS) are determined by times when the computed elevation is greater than the desired elevation angle.

The trajectory information may be input either through a trajectory DRDE, which has been generated by the INVAR processor, or by inputting a single invariant element state vector. For the analytic method, the processor then generates a table that contains the times and longitudes of the ascending node, and the corresponding sets of invariant elements for all orbits that span the start and stop times. The processor then cycles through logic, which determines the station contacts on a site-by-site basis. For each site, the processor computes a range of ascending nodes, for which the elevation angle constraint will be satisfied. This range of nodes is an approximation, because the flight time from the ascending node to closest point of approach must be approximated. The computed range of nodes is therefore biased to ensure that all possible orbits that may contain solutions are considered. However for very high elliptical orbits, the bias number may need to be increased to catch all possible solutions.

If the ascending node for a given orbit is within the range of nodes, an iterative method is used to find the orbital closest point of approach (CPA). If the elevation angle at CPA is less than the elevation angle constraint, the output quantities are computed and displayed.

The input to the STACN processor, which is described in section 4.0, is used to compute the following quantities.

a. Ground elapsed time to acquisition of signal

b. Ground elapsed time to loss of signal

c. Maximum elevation angle

d. Azimuth at acquisition of signal

e. Range at acquisition of signal

f. Minimum range

Of the two analytic methods, the quasianalytic method executes in a shorter time and produces less accurate results. The iterative method produces more accurate results but requires more computation time.

The calculated quantities along with the station ID, the orbit number, and the delta-T, are the outputs of the processor and are displayed on the user's terminal. The input parameter OUTFLG indicates whether or not the output quantities are to be written to the DRDE specified by the parameter STAFIL. When disk output is specified, the same quantities that are displayed, with the exception of the delta-T value, are written to the DRDE in the chronological order of the ground elapsed time (GET) to acquisition of signal. The quantities that are displayed as GET are written to the disk output as Greenwich mean time (GMT).

3.0 ASSUMPTIONS AND LIMITATIONS

Orbital propagation using invariant elements does not reflect changes to the orbit due to atmospheric drag between the states stored in the trajectory file. Therefore, the vectors in the trajectory file should be stored close enough together so that these inaccuracies will be small enough for the user's requirements. Periodic perturbations to the orbit are not considered. Only elliptical orbit formulation is provided. Due to array storage constraints, the processor cannot process more than 150 orbits or a trajectory segment requiring more than 30 invariant element arrays.

For a near circular orbit, the times computed by the quasianalytic method are accurate to approximately 20 seconds. For the iterative technique, the times are accurate to approximately 5 seconds or less.

The two analytic methods are not reliable for orbits which approach a ground track reversal condition. The step function method should be used for orbits which have a semimajor axis greater than 10 000 nautical miles and an eccentricity greater than 0.5. The step function method may miss contacts of shorter duration than the time step taken.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The input to the STACN processor is through its interface table and through the trajectory disk file which has been generated by the FDS-1 application processor INVAR. Optionally, a single invariant element state vector may be provided. The parameters input through the interface table are IVEC, SCVEC, VECFIL, ISIT, SITNAM, GSIT, SITFIL, GETS, GETF, ELC, INORB, OPTC, OUTFLG, and GMTR. In addition, ten conversion factors and constants contained in the master data base elements !!GLCN and !SESCN are input through the interface table. For the specific constants and conversion factors input, refer to table 4-I.

   Through the input parameters GETS and GETF, the user specifies the start time (GETS) relative to the base time and the final time (GETF) relative to the base time. The elevation angle (ELC) and the initial orbit counter (INORB) corresponding to the initial orbit for GETS are specified.

Through the two input parameters, IVEC and ISIT, the user sets flags to indicate the source of data for the element state vector and the station contact sites. The valid settings for the IVEC and ISIT input flags and their meanings are shown in the definition block of the STACN processor default interface table (table 4-I). The source of data is through either associated interface table parameters or a DRDE. Depending on the setting of IVEC, the invariant element state vector is input through the parameter SCVEC or from the DRDE VECFIL. The contents of the invariant element state vector are defined in table 7.3-VI of JSC IN 78-FM-60, volume I. Similarly, depending on the setting of ISIT, the radius, the geocentric latitude, and the longitude of the station contact site are input through the parameter GSIT or from the non-DRDE disk file identified by the parameter SITFIL.

The computation method is selected by the input interface table parameter OPTC. When OPTC is set to "FA" for "fast", a quasianalytic method that executes in a short time is used to make the calculations. When OPTC is set to "SL" for "slow", a method using iterative techniques is used to obtain a more accurate solution. When OPTC is set to "HO", the step function method is used. This option should be used for "high" orbits. The TFREQ parameter determines the time step used for the "HO" option. The defaulted value of TFREQ assumes seconds for the external time units.

The definition of the interface table for the STACN processor is provided in table 4-I.

b. Interface table data array definitions - The definition of the input data arrays appearing in the STACN interface table is provided in table 4-II.

c. Interface table data file definitions - The contents of the element state vector file, the station contact file, and the output file are provided in tables 4-III, 4-IV, and 4-V.

d. Processor solicited (prompted) inputs - The processor solicited prompt, its meaning, and the valid response are given in table 4-VI. As each line of the display is written, the line counter is incremented. When the number of lines displayed is equal to the maximum number of lines per page, the processor solicited input message "MAXIMUM LINE NUMBER" is displayed and execution of the processor pauses. This pause gives the user the opportunity to review the contents of the display and to make a hard copy of the display. The user clears the screen and enters a blank character and a carriage return to continue execution of the processor.

e. Processor displays and display parameter definition table - When the processor executed correctly, it will generate an output display on the user's terminal. The format and content of the display are given in tables 4-VII and 4-VIII.

f. Processor message table - The messages that may be displayed on the user's terminal during execution of the processor are provided in table 4-IX. An explanation of the message, the action required by the user, and the severity of the problem are provided in the table.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-X.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR STACN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
IVEC	AWA	2CH	I	1	1		Vector source flag "IN" = single input state (SCVEC) "DF" = vector DRDE as specified by VECFIL
SCVEC	AWA	Real	I	30	15		Input invariant element state vector. Required when IVEC = "IN".
VECFIL	Disk	Real	I	-	-		Name of invariant element DRDE. Required when IVEC = "DF".
ISIT	AWA	2CH	I	1	1		Flag indicating source of station contact site data "IN" = single input site "DF" = station contact site disk file as specified by SITFIL.
SITNAM	AWA	6CH	I	3	3		Name of input station contact site. Required when ISIT = "IN".
GSIT	AWA	Real	I	6	3		Radius, geocentric latitude, and longitude of input station contact site. Required when ISIT = "IN".
OPTC	AWA	2CH	I	1	1		Option specifying station contact calculation. "FA" = Fast "SL" = Slow "HO" = High orbit

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-I.- Continued

PROCESSOR STACN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
SITFIL	AWA	6CH	I	-	-		Name of station contact site disk file. Required when ISIT = "IN".
GETS	AWA	Real	I	6	3		Start time (hours, minutes, seconds) relative to base time
GETF	AWA	Real	I	6	3		Final time (hours, minutes, seconds) relative to base time
ELC	AWA	Real	I	2	1		Elevation angle
INORB	AWA	Intg	I	1	1		Initial orbit counter. Corresponds to the orbit counter for GETS.
OUTFLG	AWA	2CH	I	1	1	"NO"	Flag which indicates whether or not the output quantities are to be output to the DRDE specified by STAFIL.
STAFIL	Disk	Free	O	-	-		Name of output DRDE
TFREQ	AWA	Real	I	1	1	300.	Time step fequency for the "high" orbit option. The default value assumes seconds for the time units.
GMTR	AWA	Real	I	2	1	!SESCN(11)	Reference time for GET computations

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-I.- Concluded

PROCESSOR STACN

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
CFA	AWA	Real	I	2	1	!SESCN(27)	Conversion factor for angles - external to external
CFD	AWA	Real	I	2	1	!SESCN(29)	Conversion factor for distance
CFT	AWA	Real	I	2	1	!SESCN(31)	Conversion factor for time
ROMEGE	AWA	Real	I	2	1	!SESCN(81)	Rotation rate of Earth
PI	AWA	Real	I	2	1	!!GLCN(59)	Ratio of circle circumference to diameter
TWOPI	AWA	Real	I	2	1	!!GLCN(61)	Twice pi
DTWOPI	AWA	Dubl	I	3	1	!!GLCN(68)	Double precision TWOPI
AMILE	AWA	Real	I	2	1	!!GLCN(89)	Feet per nautical mile
AMIN	AWA	Real	I	2	1	!!GLCN(97)	Seconds per minute
AHOUR	AWA	Real	I	2	1	!!GLCN(99)	Seconds per hour
PROCON	AWA	Free	I	9	9		STACN constants

NOTES

CLASS
AWA
Disk

TYPE
Free 2CH 72CH
Intg 6CH Mix
Real 18CH Symb
Dubl 36CH

USE
I = Input
O = Output
I/O = Input/Output

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR STACN

Array name	Index location	Default value	Definition	
SCVEC	1		Vector time (standard state vector format)	Standard position/velocity state vector definition
	2		Invariant elements semi-major axis	
	3		Invariant elements eccentricity	
	4		Invariant elements inclination	
	5		Invariant elements argument of perigee	
	6		Invariant elements argument of ascending node	
	7		Invariant elements mean anomaly	
	8		Invariant elements mean motion	
	9		Invariant elements rate of change of perigee	
	10		Invariant elements rate of change of the ascending node	
	11		C_D - drag coefficient	
	12		Area for drag computation	
	13		Vehicle gross mass	
	14		Vector type code (=3120.)	
	15		Vector name code	
GSIT	1		Radius of input station contact site	
	3		Geocentric latitude of input station contact site	
	5		Longitude of input station contact site	
GETS	1		Start time relative to reference time (hours)	
	3		Start time relative to reference time (minutes)	
	5		Start time relative to reference time (seconds)	
GETF	1		Final time relative to reference time (hours)	
	3		Final time relative to reference time (minutes)	
	5		Final time relative to reference time (seconds)	
PROCON	1	0	Debug print flag 0 = no print 1 = debug print	

TABLE 4-II.- Concluded

PROCESSOR STACN

Array name	Index location	Default value	Definition
PROCON	2	0	Output destination flag 0 = terminal positive number = LU of output device
	3	1.7453292×10^{-2}	Bias which is applied to nodal range values to account for uncertainty in nodal computations due to uncertainty in flight time (default = 1 degree) (Units: radians)
	5	2.	Iteration tolerance for CPA iteration
	7	24	Maximum number of lines per page
	8	20	Cartridge reference number for data files
	9	100	Maximum number of blocks for the output DRDE

TABLE 4-III.- INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR STACN

DRDE DATA FILE VECFIL

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file
	4-6	Interface table variable creating file
	7-9	Processor last changing file
	10-12	Interface table variable last changing file
2-N	1-60	Position/velocity phase table values. Refer to section 7.3.3 in JSC IN 78-FM-60, volume I for form and content.

TABLE 4-IV.- INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR STACN

NON-DRDE DATA FILE SITFIL

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file
	4-6	Interface table variable creating file
	7-9	Processor last changing file
	10-12	Interface table variable last changing file
2-N	1-2	Contact station site name
	3-4	Geocentric latitude
	5-6	Longitude
	7-8	Radius

TABLE 4-V.- INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR STACN

NON-DRDE DATA FILE STAFIL

Record number	Integer word allocations	Content and definition
1	1-3	Processor creating file - STACN
	4-6	Interface table variable creating file - STAFIL
	7-9	Processor last changing file - STACN
	10-12	Interface table variable last changing file - STAFIL
2-N	1-3	Station contact site identification
	4	Orbit number
	5-6	Greenwich mean time of acquisition of signal
	7-8	Greenwich mean time of loss of signal
	9-10	Maximum elevation angle
	11-12	Azimuth at acquisition of signal
	13-14	Range at acquisition of signal
	15-16	Minimum range

TABLE 4-VI.- PROCESSOR SOLICITED (PROMPTED) INPUTS

PROCESSOR STACN

Prompt	Meaning	Valid responses
Maximum line number	The maximum number of output lines specified in PROCON(7) has been reached.	After printing a copy of the terminal screen, clear the screen and enter a space and a carriage return to obtain the remaining output.

TABLE 4-VII.- STACN PROCESSOR DISPLAY FORMAT

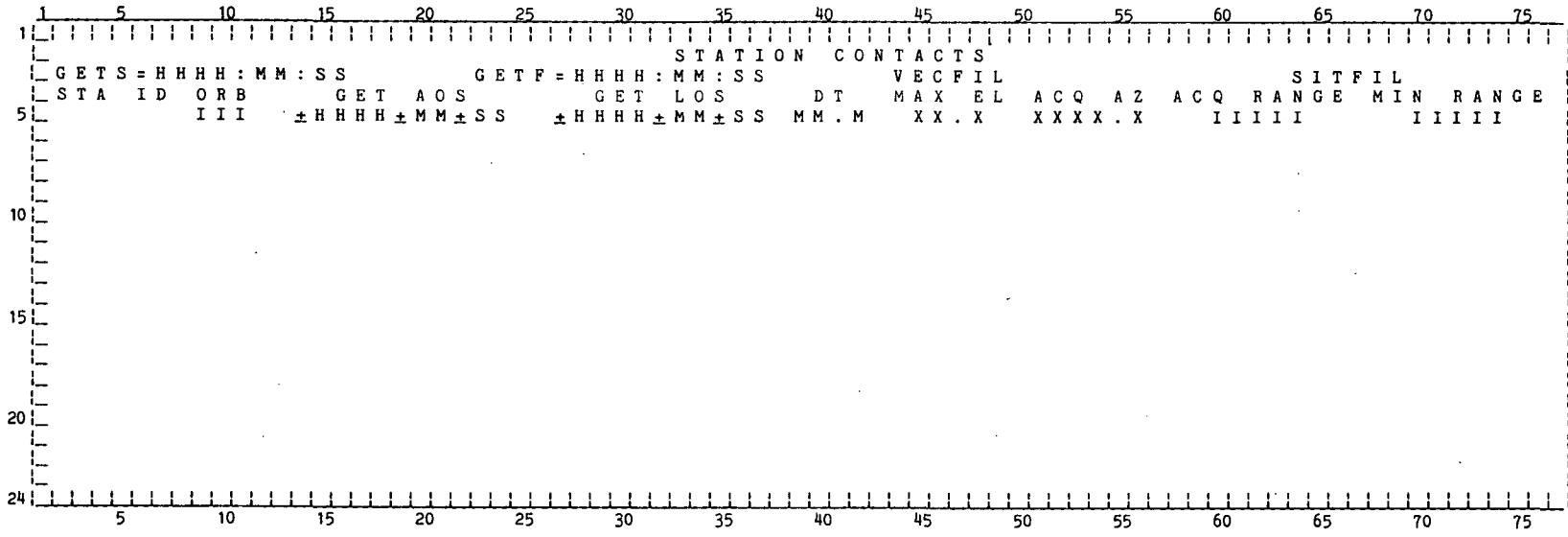


TABLE 4-VIII.- DISPLAY PARAMETER DEFINITION TABLE

PROCESSOR STACN

Display name	
Display parameter label	Parameter definition
GETS	Start time (hours, minutes, seconds) relative to base time
GETF	Final time (hours, minutes, seconds) relative to base time
STA ID	Station contact site identification
ORB	Orbit number
GET AOS	GET to acquisition of signal (hours, minutes, seconds)
GET LOS	GET to loss of signal (hours, minutes, seconds)
DT	Delta time (minutes)
MAX EL	Maximum elevation angle
ACQ AZ	Azimuth at acquisition of signal
ACQ RANGE	Range at acquisition of signal (nautical miles)
MIN RANGE	Minimum range (nautical miles)

TABLE 4-IX.- PROCESSOR MESSAGE TABLE

PROCESSOR STACN

MSG no.	Message ID block	Message text block and explanation
1	*STACN*	STATE VECTOR CODE NOT INVARIANT ELEMENT Meaning: Other than invariant element state vector is specified. Severity: Processor terminates. Action required by user: Input invariant element state vector. Rerun the processor.
2	*STACN*	OPEN FILE ERROR= IIII FILE NAME= AAAAAA Meaning: Either the state vector DRDE or the station contact site file is not in the proper configuration. Severity: Processor terminates. Action required by user: Check the file name, etc., of the trajectory or station contact site files. Rerun the processor.
3	*STACN*	FILE CREATE ERROR= IIII FILE NAME= AAAAAA Meaning: Error occurred when an attempt was made to create the output DRDE or a scratch file. Severity: Processor terminates. Action required by user: Check the name, etc., of the output DRDE for redundancy or contact SDB/FM6 to purge the scratch file. Rerun the processor.
4	*STACN*	NO STATION CONTACTS: OUTSIDE TRAJECTORY LIMITS Meaning: Station contact site is outside the limits of the trajectory and no output data will be generated for that site. Severity: None Action required by user: None
5	*STACN*	EXCEEDED MAX NO. OF ORBITS: Meaning: The trajectory spans more than the allowable number of orbits. Severity: Only time span corresponding to the maximum number of orbits will be processed. Action required by user: Rerun processor to obtain desired time span not included in the output.

TABLE 4-IX.- Concluded

PROCESSOR STACN

MSG no.	Message ID block	Message text block and explanation
6	*STACN*	EXCEEDED MAX NO. OF VECTOR ENTRIES IN ARRAYS Meaning: The time span requires more vector storage area than available. Severity: Processor only generates data for the time span over which the vector storage capability is not exceeded. Action required by user: Rerun processor with a different time span to obtain desired data not included.
7	*STACN*	FILE READ ERROR= IIII FILE NAME= AAAAAA Meaning: Error reading vector DRDE or site file. Severity: Processor terminates. Action required by user: Regenerate vector DRDE or site file and then rerun processor.
8	*STACN*	FILE WRITE ERROR= IIII FILE NAME= AAAAAA Meaning: Error writing output DRDE. Severity: Processor terminates. Action required by user: Rerun the processor.
9	*STACN*	SCRATCH FILE ERROR= IIII Meaning: Error reading, writing, or positioning scratch file. Severity: Processor terminates. Action required by user: Rerun the processor.

TABLE 4-X.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR STACN

Processor name	*Processor abstract prompt (maximum 256 characters)*
STACN	The STACN processor computes the time at which an orbiting vehicle is above an input elevation angle with respect to a ground tracking station.

Parameter keyword name	*Parameter definition prompt (maximum 256 characters)*
	Use definitions as provided in table 4-I, Processor Interface Table in addition to the following.
PROCON	STACN constants: (1) Debug (2) Output logical unit (3) Node range bias angle (rad) (5) Iteration tolerance (7) Maximum lines/page (8) Output file cartridge reference number (9) Maximum number of blocks in output file

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM STACN

5.1.1 Purpose

The routine STACN serves as the main program of the STACN processor. The routine provides most of the computational functions of the application processor STACN. The computational parameters are

a. Orbit number

b. Time of acquisition of signal (AOS)

c. Time of loss of signal (LOS)

d. Delta time (LOS time - AOS time)

e. Maximum elevation angle of the orbiting vehicle above the ground station

f. Azimuth at acquisition of signal

g. Range at acquisition of signal

h. Minimum range between the orbiting vehicle and the ground station

In addition, the routine STACN provides the linkage to the routines that sort the data by AOS times, provide the display of the data, and write the data to an output file.

5.1.2 Functional Description

The procedure of STACN parallels the landing opportunities processor (LOPT) computational procedure through the calculations for the closest point of approach (CPA). (See processor LOPT for the CPA functional description.) The acquisition and loss of signal calculations follow the CPA calculations. As stated in section 2.0, two methods of making the AOS/LOS calculations are available depending on the degree of accuracy required and the computational speed desired. One method is a quasianalytic method that executes in a short time. The other method uses iterative techniques to obtain a more accurate solution. When a station contact occurs, the data associated with that contact is written to a temporary file.

The AOS/LOS calculations are displayed and written to the output disk file in chronological order of AOS time making necessary a sort by AOS time of the data on the temporary file. Because of the HP21MX core limitations, the sort and display functions are accessed through a separate program scheduled by the routine

STACN. Calls to the RTE-III routine EXEC enable execution transfers and transmit necessary data parameters to and from the scheduled program STALK. Refer to section 5.2 for a description of STALK.

5.1.3 Assumptions and Limitations

The following assumptions and limitations apply to the STACN processor.

a. Elliptical orbit propagation is assumed.

b. Periodic perturbations to the orbit are not included in the computations.

c. For a near circular orbit, the values of time computed by the quasianalytic method are accurate to approximately 20 seconds. For the iterative technique, the values of time are accurate to approximately 5 seconds or less.

5.1.4 Method

The routine STACN performs many of the same calculations as the main routine of the FDS-1 applications processor LOPT. (Refer to processor LOPT for the method used and for descriptions of the subroutines RVECF, CNODS, TAU, ADVU, and CPA.) As stated previously, there are two methods available for the computation of the times of the acquisition of signal and loss of signal. Each method will be described separately.

a. Iteration method for AOS/LOS computation - The iterative method, or "slow" method, involves iteration on the value of time until a tolerance specified by input is met. First, the angle from the ground site to the vehicle at AOS is computed by the following equation (fig. 5.1-1).

$$\alpha = \pi/2 - e - \sin^{-1}\left[(R_{site}/R_{vehicle})\cos e\right]$$

where e is the elevation angle

R_{site} is the radius of the site

$R_{vehicle}$ is the radius of the vehicle to the center of the Earth

Next, using the value of the out-of-plane angle (β) obtained in the CPA iteration, the in-plane angle from CPA to AOS is computed as follows (fig. 5.1-2).

$$\alpha_2 = \cos^{-1}(\cos\alpha/\cos\beta)$$

The argument of latitude at AOS is computed from

$$UT = UT_{CPA} - \alpha_2$$

where UT_{CPA} is the argument of latitude at CPA obtained from CPA iteration.

The time of arrival and the radius of the vehicle at the desired argument of latitude (UT) are computed using subroutine TAU. However, since the computation of α requires an approximation of the radius of the vehicle (semimajor axis), the computation is not exact. Thus, the above computations are repeated using the new radius of the vehicle until the change in time of arrival is less than an input tolerance.

Next, the angular momentum vector ($\vec{H}$) and the ground site vector are computed at the new time by

$$\vec{H} = \begin{bmatrix} \sin(HC)\ \sin(i) \\ -\cos(HC)\ \sin(i) \\ \cos(i) \end{bmatrix}^T$$

where i is the inclination angle. The argument of the ascending node HC at the current time is computed

$$HC = h_k + \dot{h}(t - t_{vector})$$

where h_k is the argument of the ascending node at the vector time, t is the current time, t_{vector} is the vector time, and $\dot{h}$ is the rate of change of the argument of the ascending node.

The vector of the ground site $\vec{S}$ is

$$\vec{S} = \begin{bmatrix} \cos\ (L_{site})\ \cos\ (\Lambda + TU\Omega_e) \\ \cos\ (L_{site})\ \sin\ (\Lambda + TU\Omega_e) \\ \sin\ (L_{site}) \end{bmatrix}^T$$

where

L_{site} is the latitude of the site

Λ is the longitude of the site

TU is the time of arrival

Ω_e is the rotation rate of the Earth

A new out-of-plane angle is computed by

$$\sin \beta = \vec{H} \cdot \vec{S}$$

A new value of UT_{CPA} is computed using subroutine CPA. A new time of arrival is computed using the method outlined above with the new parameters β and UT. If the change in time from the previously computed time is less than a tolerance, this time is accepted as the AOS time (TA). The same computational procedure is used to obtain the LOS time (TL), except that the desired argument of latitude of LOS is computed as

$$UT = UT_{CPA} + \alpha_2$$

The range and azimuth at AOS are computed below. The unit vector of the vehicle $\vec{R}$ is computed as

$$\vec{R} = \begin{bmatrix} \cos(HC)\cos(UT) - \sin(HC)\cos(i)\sin(UT) \\ \sin(HC)\cos(UT) + \cos(HC)\cos(i)\sin(UT) \\ \sin(i)\sin(UT) \end{bmatrix}$$

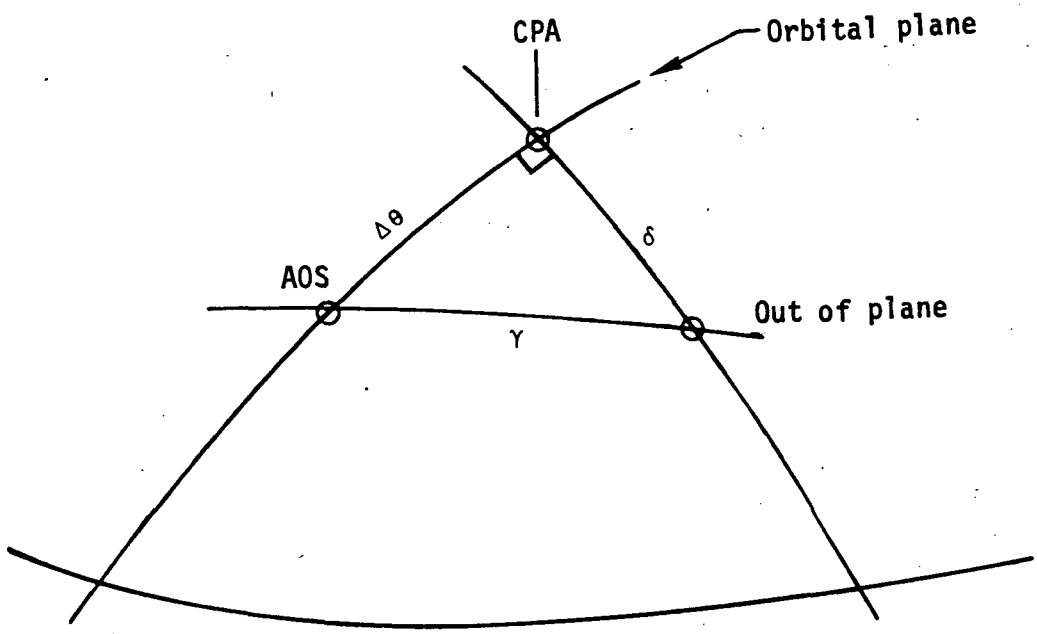
UT is the argument of latitude. Using the law of cosines, the range at AOS is computed.

$$RAOS = \left[R_{site}^2 + R_{vehicle}^2 - 2R_{site}\, R_{vehicle} \cos \beta\right]$$

The azimuth at AOS is computed in the AZAOS subroutine. (Refer to section 5.5 for a description of this computation.)

Table 5.1-I provides a list of mathematical symbols versus code symbols.

b. Quasianalytic method for AOS/LOS computations - The quasianalytic, or "fast" method, involves iteration on the orbital path of the vehicle, θ, from the CPA to obtain the AOS. When $|\Delta\theta - \Delta\theta_{previous}| <$ tolerance, the iteration is complete and the AOS time is computed.



Using spherical trigonometry, and assuming a zero degree elevation angle

$$\cos\gamma = \cos\Delta\theta \cos\delta = r_{STA}/r_v \quad (1)$$

$$r_v = P/\left[1 + e\cos(\theta_{CPA} - \Delta\theta)\right] \quad (2)$$

where r_{STA} is the radius of the site

r_v is the radius of the vehicle

P is the semilatus rectum of orbit

Combining equations (1) and (2), and solving for cos $\Delta\theta$

$$\cos\Delta\theta_2 = \frac{r_{STA}}{P\cos\delta}\left[1 + e\cos(\theta_{CPA} - \Delta\theta_1)\right]$$

$\Delta\theta_1$ is initially set to zero to start the iteration.

If $|\Delta\theta_2 - \Delta\theta_1| <$ tolerance, the iteration is complete. If $|\Delta\theta_2 - \Delta\theta_1| \geq$ tolerance, set $\Delta\theta_1 = \Delta\theta_2$ and recompute cos $\Delta\theta_2$.

The time increment for AOS is

$$\Delta t = \frac{1}{\eta}\left[\Delta\theta_2 + 2e(\sin\theta_1 - \sin\theta_2) + \frac{3}{4}e^2(\sin 2\theta_2 - \sin 2\theta_1)\right]$$

$$T_{AOS} = T_{CPA} - \Delta t$$

where

e is the eccentricity

η is the mean motion

T_{AOS} is the time at AOS

T_{CPA} is the time at CPA

The following procedure is used by the LOS time computation.

$$\theta_2 = \theta_{CPA} + \Delta\theta_2$$

$$\Delta t = \frac{1}{\eta}\left[\Delta\theta_2 + 2e(\sin\theta_2 - \sin\theta_{CPA}) + \frac{3}{4}e^2(\sin 2\theta_2 - \sin 2\theta_{CPA})\right]$$

$$T_{LOS} = T_{CPA} + \Delta t$$

The range at acquisition of signal, RAOS, and the azimuth at acquisition of signal are computed as described in section 5.1.4a.

Table 5.1-II provides a list of mathematical symbols versus program symbols.

5.1.5 Routine Input/Output Variables

The input/output variables for the routine STACN are given in table 5.1-III.

5.1.6 Functional Logic Flow

The functional logic flow for STACN is provided in figure 5.1-3.

5.1.7 Diagnostics and Debug

Additional values may be displayed for debug purposes by setting the input interface table parameter PROCON(1) to a nonzero value. The debug values displayed are the orbit number, the invariant element set number, the time of the node, and the longitude of the ascending node. These four values are displayed each time through the loop that generates the ascending nodes. Another set of debug parameters are displayed after each call to the subroutine CNODS during the range of nodes computations. The values displayed are the crossrange angle, four values of the range of nodes, and the flag indicating whether or not the site is within the trajectory limits.

5.1.8 Special Comments

None.

5.1.9 References

None.

TABLE 5.1-I.- MATH SYMBOLS VERSUS INTERNAL CODE SYMBOLS
AOS/LOS CALCULATIONS, SLOW METHOD

Math symbol	Internal code symbol	Math symbol	Internal code symbol
α	ANG	cos i	COSI
α_2	ANG2	sin i	SINI
cos β	COSB	L_{site}	S4
sin β	SINB	Λ	LON
cos A	DOT (RVEC, SVEC)	$[R]$	RVEC
e	ELC	RAOS	RAOS
cos e	COSEL	$R_{vehicle}$	RAD
γ	GAM	Ω_e	ROMEGE
$\dot{h}$	HDOT	π	PI
h_k	HK	t	TU
H	H	t_{vector}	TIMV
HC	HC	TA	TA
i	IK	TL	TL
β	ALP	R_s	SMAG

TABLE 5.1-II.- MATH SYMBOLS VERSUS CODE SYMBOLS
AOS/LOS CALCULATIONS, FAST METHOD

Math symbol	Internal code symbol	Math symbol	Internal code symbol
e	EK	η	ETA
$\cos \delta$	COSB	P	PK
θ_1	THTA	r_{STA}	SMAG
$\Delta\theta_1$	DTHA	r_V	RAD
θ_2	THTN	T_{AOS}	TA
$\Delta\theta_2$	DTHN	T_{CPA}	TU
θ_{CPA}	TAC	T_{LOS}	TL
$\sin \theta_{CPA}$	SITC	Δt	DANG*ETA
$\sin 2\theta_{CPA}$	SITTC		

TABLE 5.1-III.- ROUTINE INPUT/OUTPUT VARIABLES

Routine STACN

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ABIAS	—	Real	I	rad	IT	PROCON(3)	Bias to account for uncertainty in nodal computations due to uncertainty in flight time
AHOUR	--	Real	I	sec	IT	--	Seconds per hour
AMILE	--	Real	I	feet	IT	--	Feet per nautical mile
AMIN	--	Real	I	sec	IT	--	Seconds per minute
CPA	—	Real	I	--	IT	--	Conversion factor for angles - external units to internal
CFD	—	Real	I	—	IT	--	Conversion factor for distance
CFT	--	Real	I	—	IT	--	Conversion factor for time
DTWOPI	--	Dubl	I	--	IT	--	2π
ELC	--	Real	I	deg	IT	--	Elevation angle
GETF	—	Real	I	--	IT	—	Final time (hr, min, sec) relative to base time

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.1-III.- Continued

Routine STACN

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
GETS	--	Real	I	--	IT	--	Start time (hr, min, sec) relative to base time
GMTR	--	Real	I	sec	IT	--	Reference time for GET computations
GSIT	--	Real	I	--	IT	--	Radius, geocentric latitude, longitude of input station
ICR	--	Intg	I	--	IT	PROCON(8)	Cartridge number for output file
INORB	--	Intg	I	--	IT	--	Initial orbit count
IPRNTL	--	Intg	I	--	IT	PROCON(1)	Debug print flag
ISIT	--	2CH	I	--	IT	--	Flag indicating source of station contact site data "IN" = single input site "DF" = site disk file
IVEC	--	2CH	I	--	IT	--	Flag indicating source of vector data "IN" = single input state vector "DF" = vector disk file
MLINE	--	Intg	I	--	IT	PROCON(7)	Maximum number of lines per display
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-III.- Continued

Routine STACN

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
NOBLKS	--	Intg	I	--	IT	PROCON(9)	Maximum number of blocks for the output file
OLU	--	Intg	I	--	IT	PROCON(2)	Display destination flag 0 = terminal >0 = unit number of display device
OPTC	--	2CH	I	--	IT	--	Specifies AOS/LOS calculation option "SL" = slow method "FA" = fast method
OUTFLG	--	2CH	I	--	IT	--	Indicator for writing output disk file "NO" = do not write file "YE" = write file
PI	--	Real	I	--	IT	--	π
PROCON	--	Intg	I	--	IT	--	Processor constants
ROMEGE	--	Real	I	rad/sec	IT	--	Rotation rate of Earth
RTOL	--	Real	I	sec	IT	PROCON(5)	Iteration tolerance for CPA and AOS/LOS loops
SCVEC	--	Real	I	--	IT	--	State vector file when not on disk file
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-III.- Concluded

Routine STACN

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
SITNAM	--	6CH	I	--	IT	--	Name of station contact site file
TWOPI	--	Real	I	--	IT	--	2π
AZAS	AZAS	Real	O	rad	F	--	Azimuth at AOS
EL2	e	Real	O	rad	F	--	Maximum elevation angle
IONA1	--	2CH	O	--	F	IRECO(1)	Site name (characters 1,2)
IONA2	--	2CH	O	--	F	IRECO(2)	Site name (characters 3,4)
IONA3	--	2CH	O	--	F	IRECO(3)	Site name (characters 5,6)
IREV	--	Intg	O	--	F	IRECO(4)	Orbit number of contact
RAOS	RAOS	Real	O	feet	F	IRECO(13)	Range at AOS
RNG	R_{min}	Real	O	feet	F	IRECO(15)	Minimum range
TA	TA	Real	O	sec	F	IRECO(5)	Time at acquisition of signal
TL	TL	Real	O	sec	F	IRECO(7)	Time at loss of signal
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

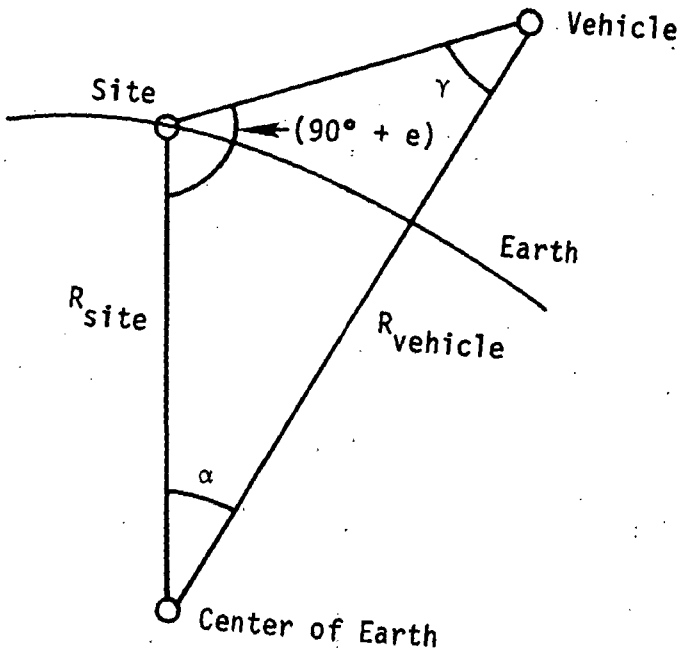
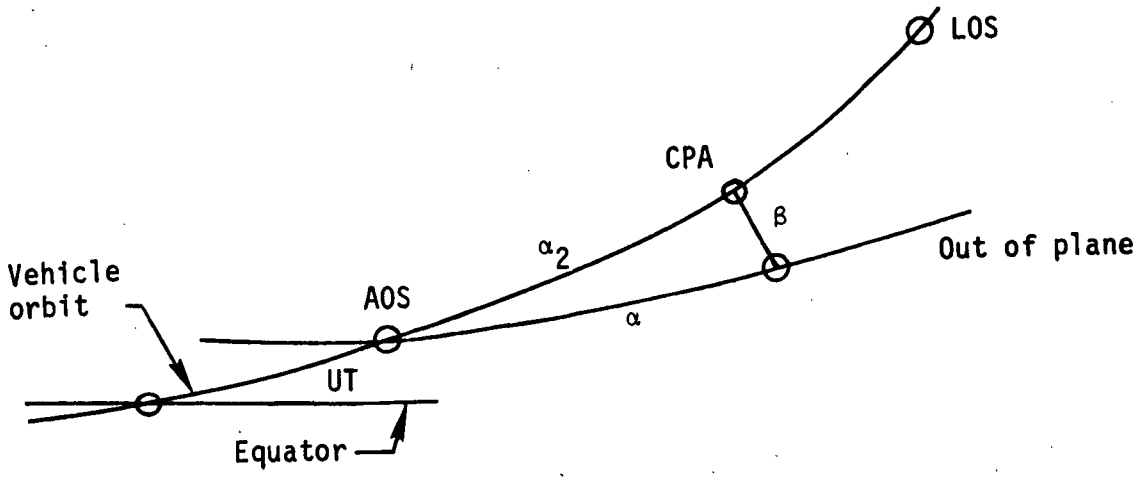


Figure 5.1-1.- Ground site geometry at AOS.



$$\cos \beta = [\, 1 - \sin^2 \beta \,]^{1/2}$$

$$\sin \beta = \vec{H} \cdot \vec{S}$$

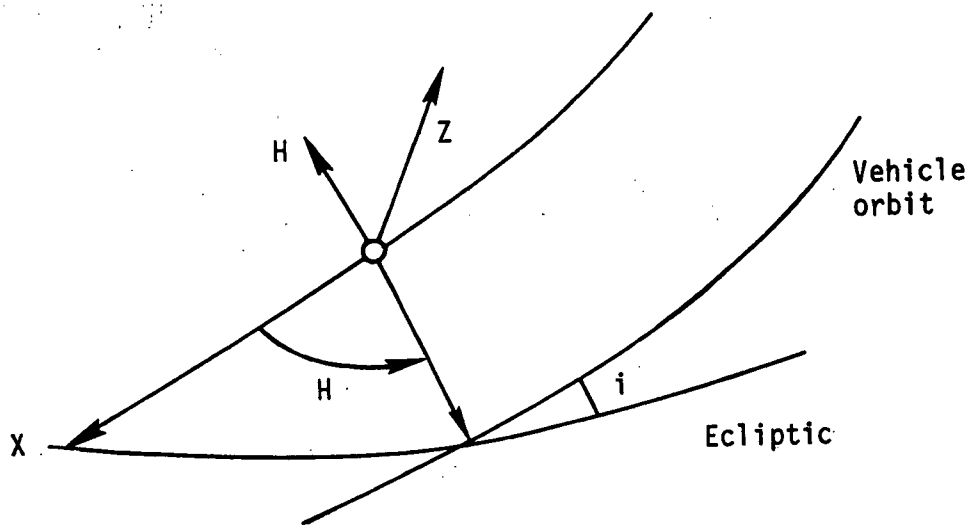


Figure 5.1-2.- CPA/AOS geometry.

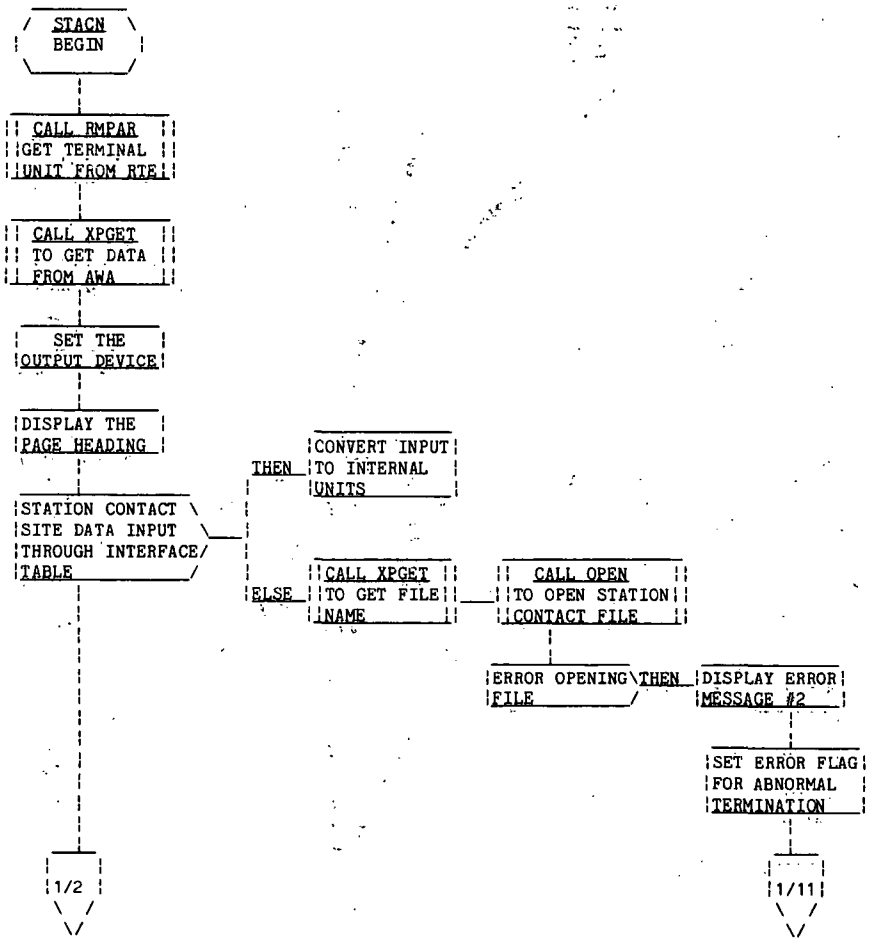


Figure 5.1-3.- STACN functional logic flow.

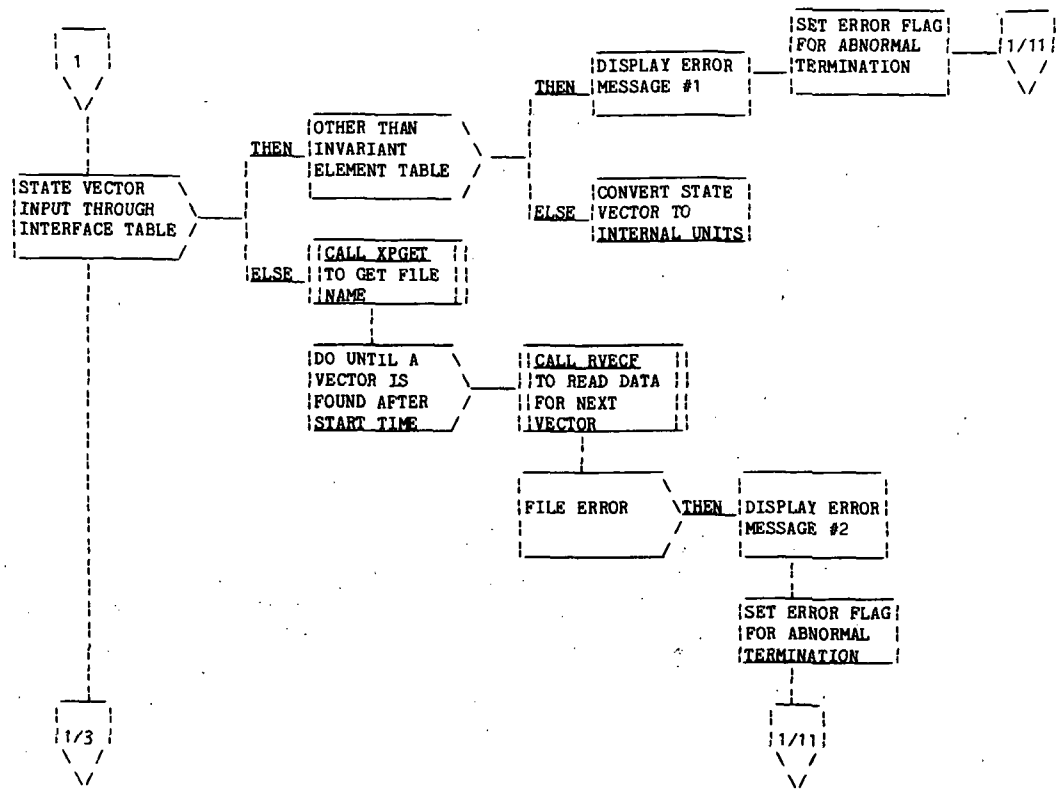


Figure 5.1-3.- Continued.

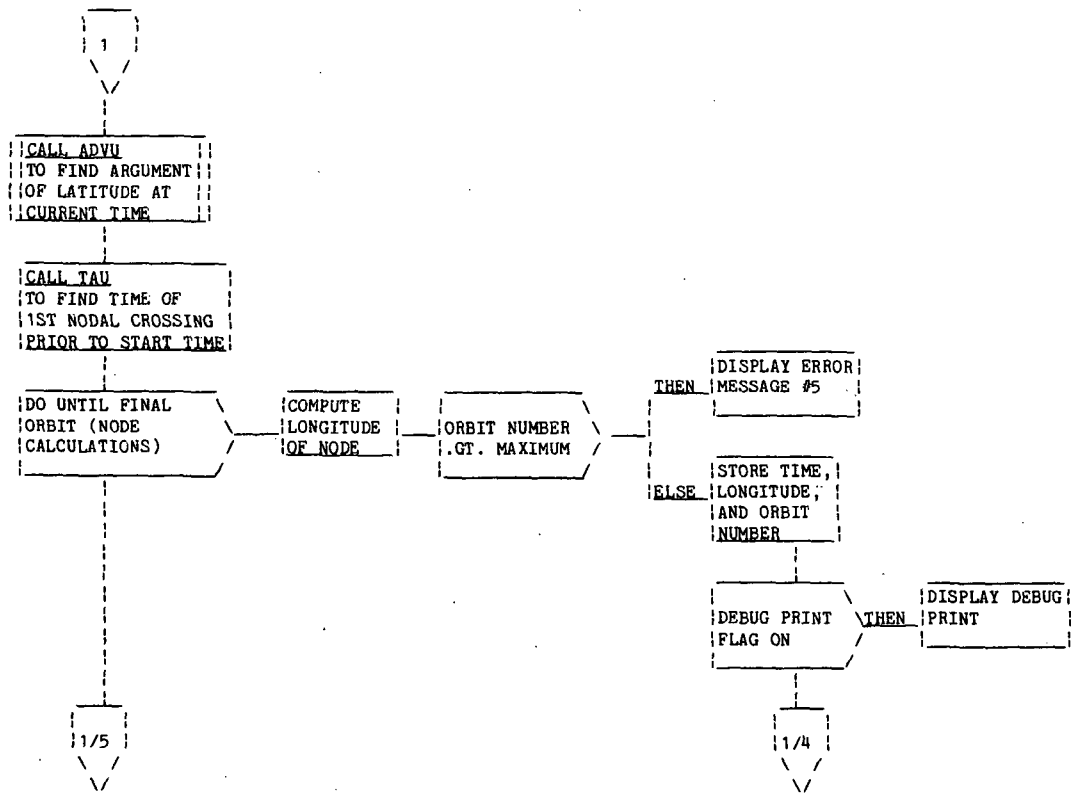


Figure 5.1-3.- Continued.

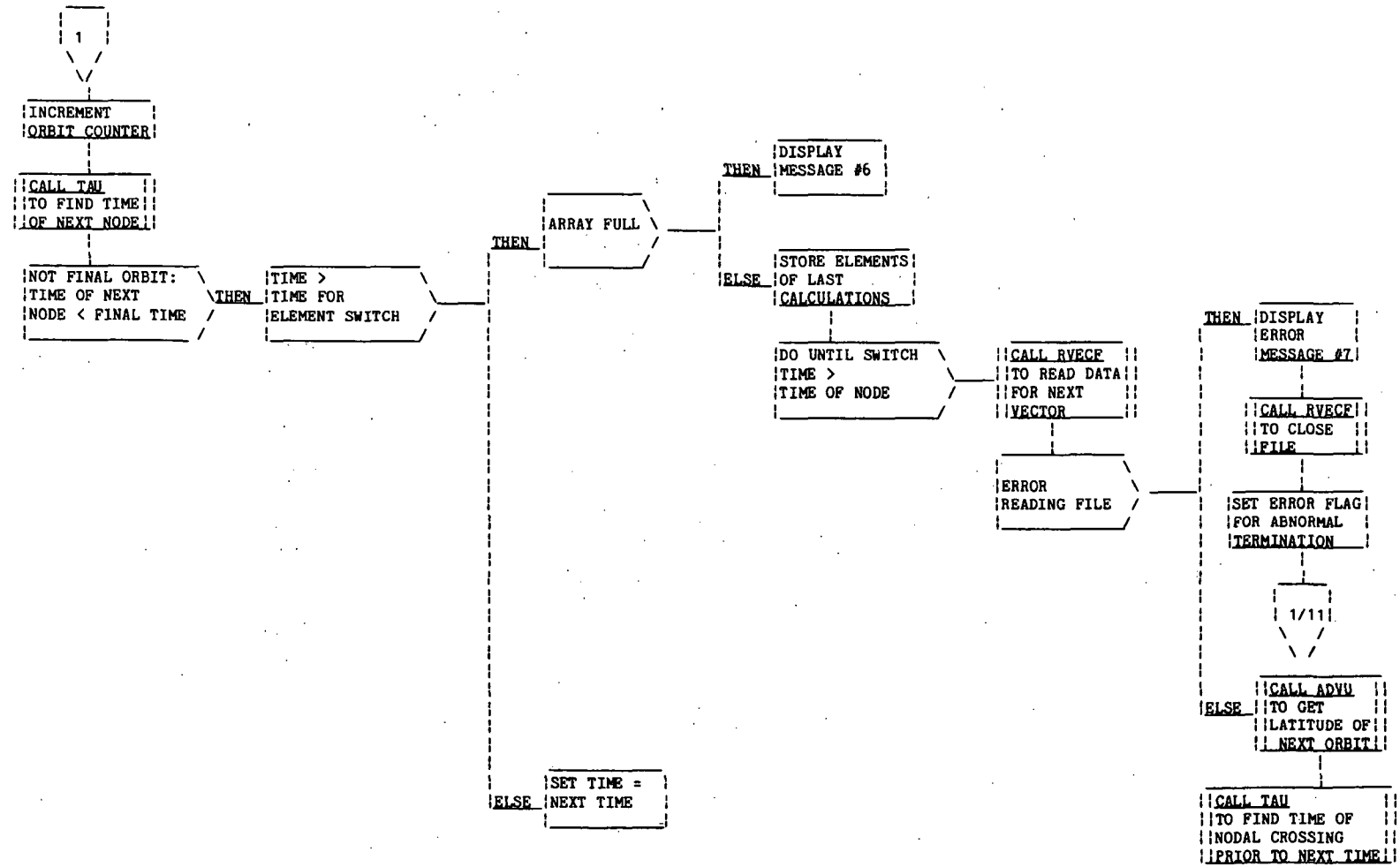


Figure 5.1-3.- Continued.

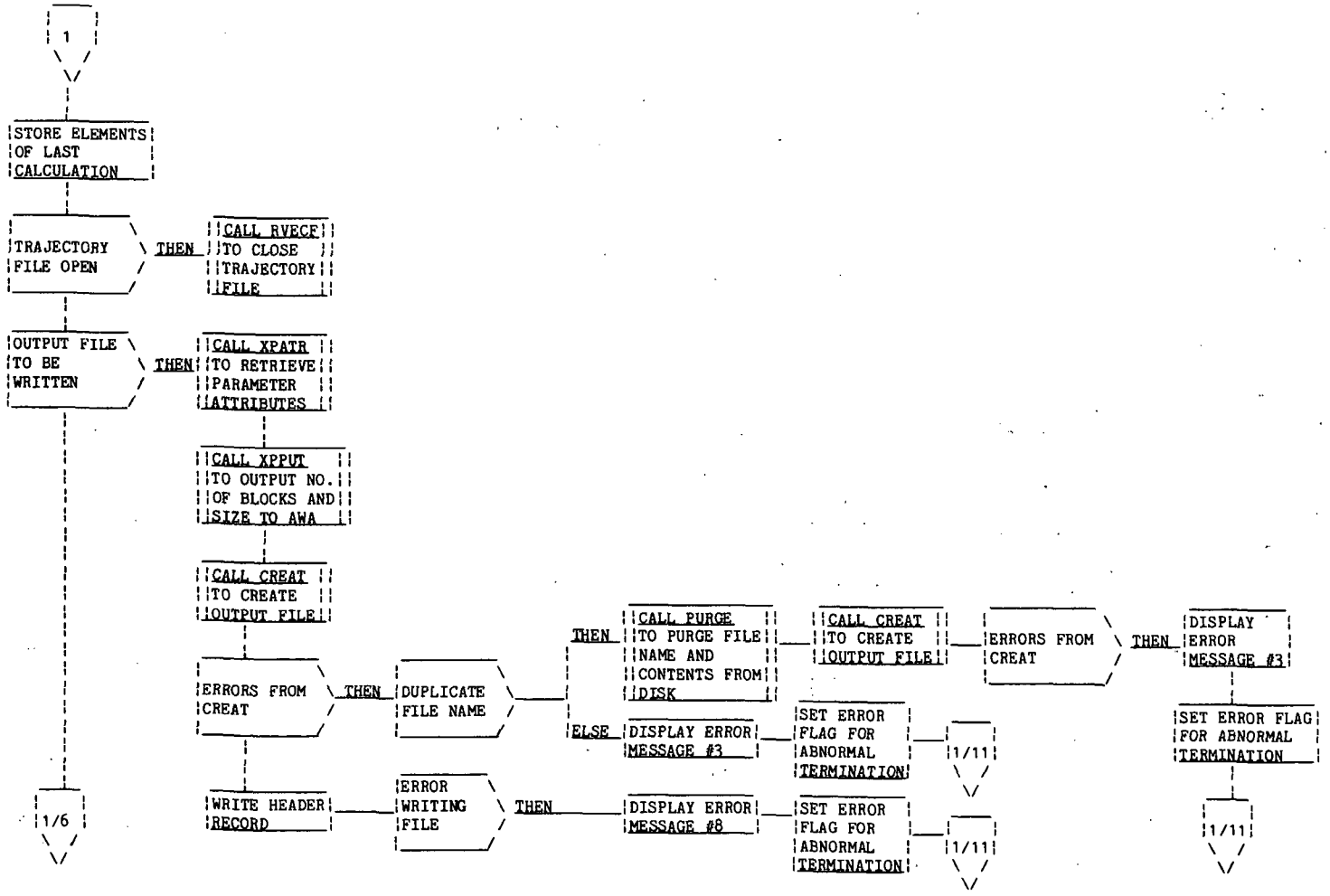


Figure 5.1-3.- Continued

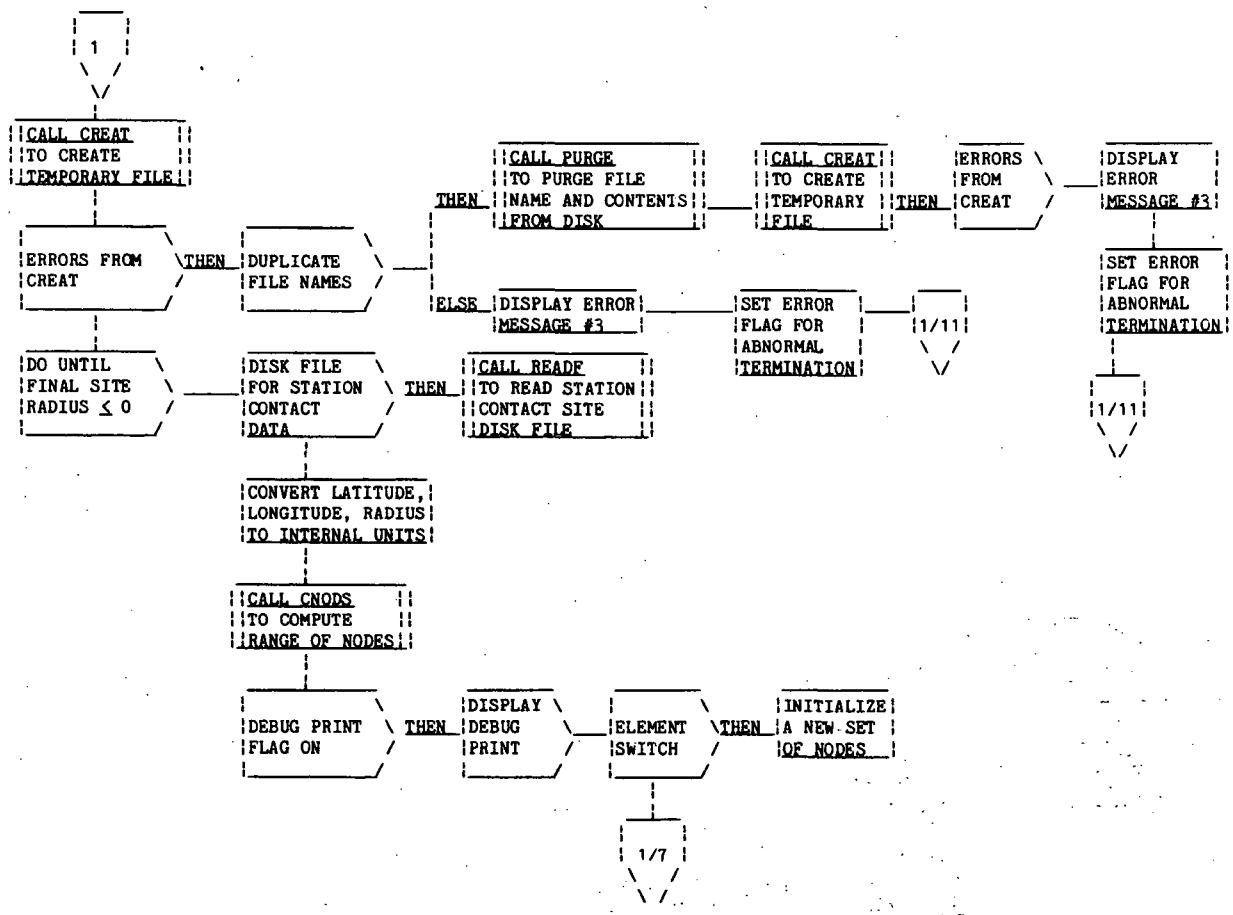


Figure 5.1-3.- Continued.

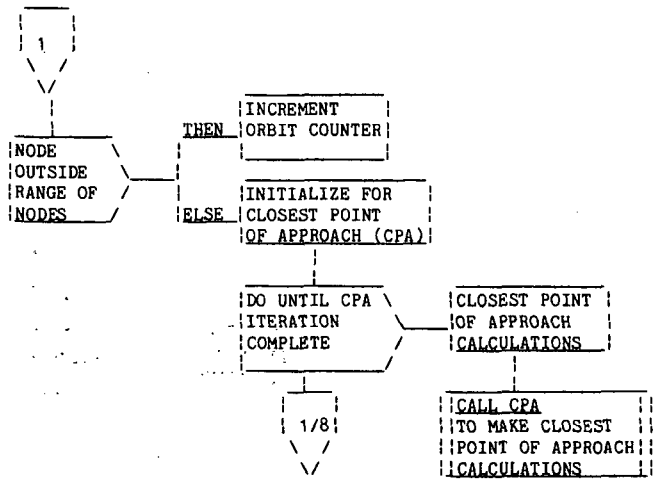


Figure 5.1-3.- Continued.

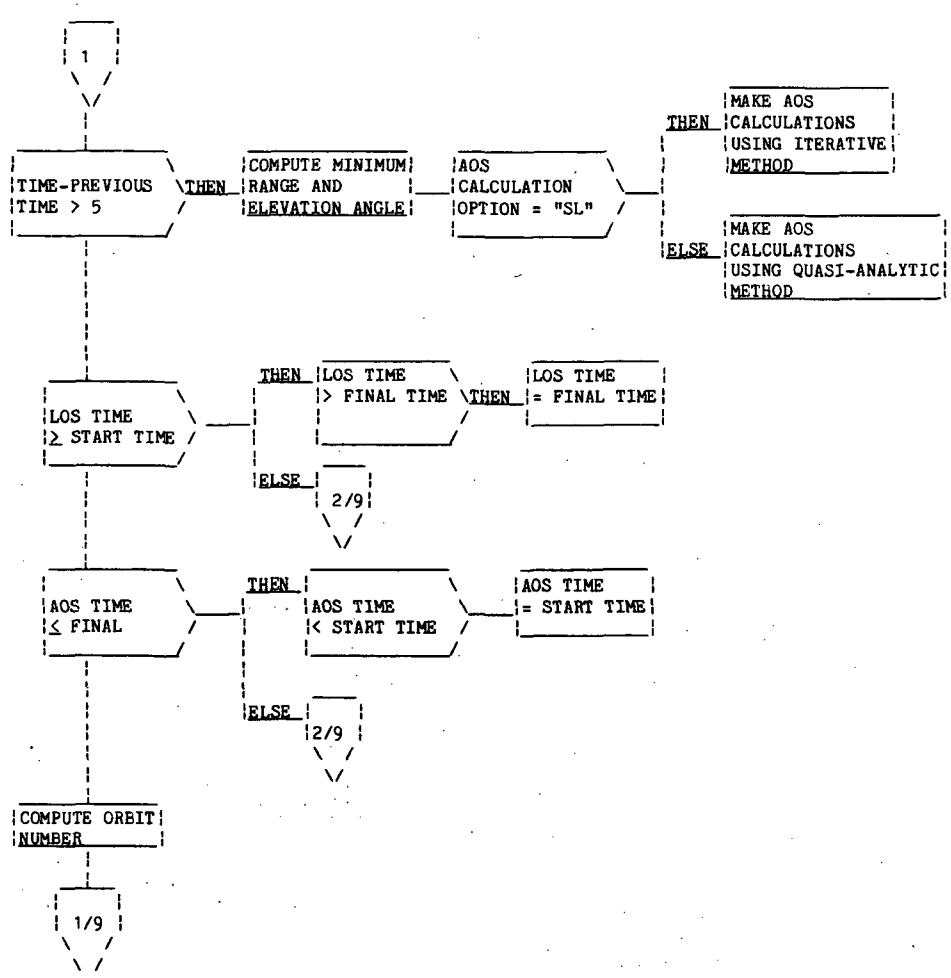


Figure 5.1-3.- Continued.

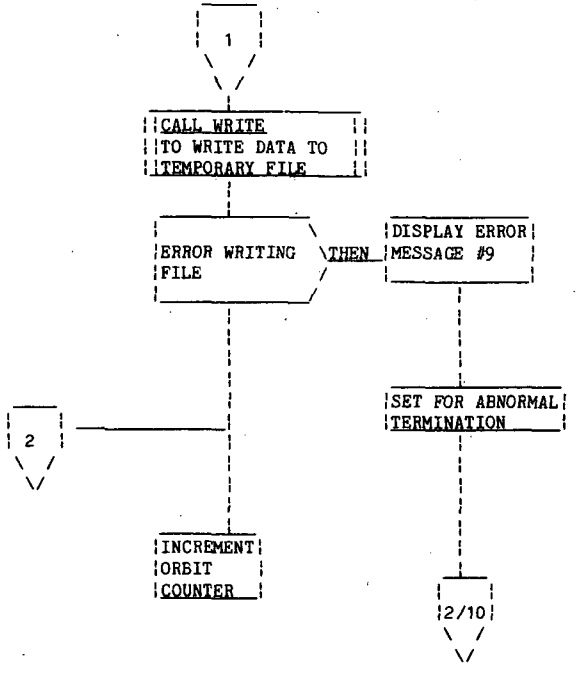


Figure 5.1-3.- Continued.

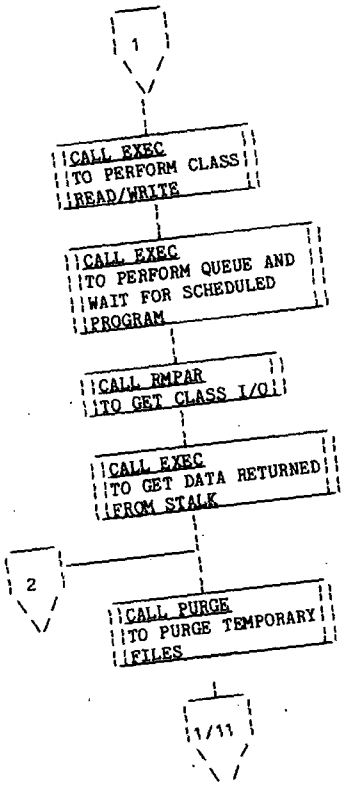


Figure 5.1-3.- Continued.

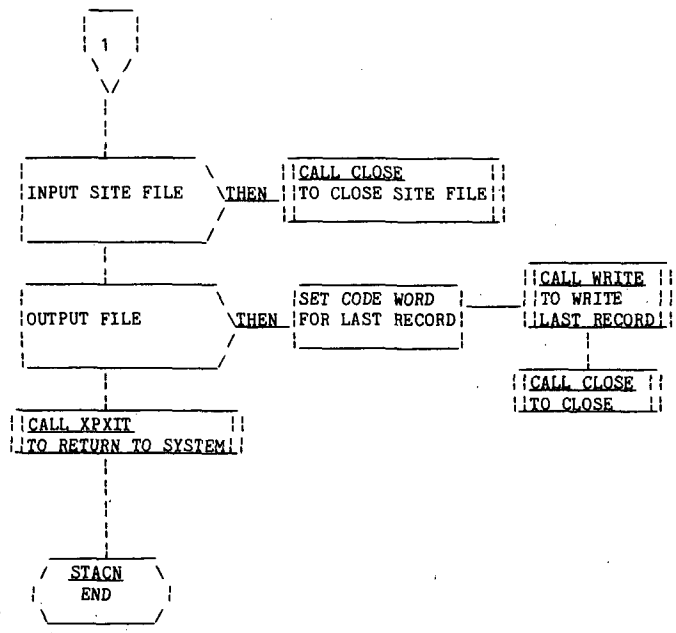


Figure 5.1-3.- Concluded.

5.2 ROUTINE NAME - STALK

5.2.1 Purpose

The routine STALK, station contact linked program, serves as the main routine of the program scheduled by the station contact processor (STACN). The routine calls the subroutine SAOST to sort by acquisition of signal time (AOS) the data on the scratch disk file. It calls the subroutine DWOUT to display the sorted data and optionally to write the data to an output disk file.

5.2.2 Functional Description

The routine STALK calls the RTE-III routine RMPAR to obtain the logical unit of the terminal and the class I/O. A call is made to the RTE-III routine EXEC to obtain the parameters passed from the scheduling routine STACN. The subroutine SAOST is called to sort the scratch disk file. On returning from SAOST, the error flag is checked. If an error condition exists, the program transfers to the exit portion of the program. The subroutine DWOUT is called to display the sorted data and as an option to write the data to an output disk file. After returning from DWOUT, the exit portion of the program is executed.

The exit portion of the program checks the value of the error flag. If an error condition exists, the return code is set to the abnormal condition of -32768. If no error condition exists, the return code is set to zero. Control is returned to the scheduling program, STACN, by calling the RTE-III routine EXEC and the FDS-1 utility routine XPXIT.

5.2.3 Assumptions and Limitations

None.

5.2.4 Method

None.

5.2.5 Routine Input/Output Variables

Table 5.2-I provides the description of the input/output variables contained in the linkage with the scheduling routine STACN.

5.2.6 Functional Logic Flow

The functional logic flow for STALK is provided in figure 5.2-1.

5.2.7 Diagnostics and Debug

None.

5.2.8 Special Comments

The STACN processor (including the sort and display routines) exceeds the HP21MX 16-page allowable program size. The sort and display routines were taken from the processor and included in the scheduled program STALK in order to expedite completion of the STACN processor.

5.2.9 References

None.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine STALK

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IDCB1	--	Intg	I	--	SAM	IDCB1	Data control block for scratch file 1.
IDCB2	--	Intg	I	--	SAM	IDCB2	Data control block for scratch file 2.
IBUFO	--	Intg	I	--	SAM	IBUFO	Data control block for output file.
IRECO	--	Intg	I	--	SAM	IRECO	Array containing data for a station contact.
NAMTM1	--	Intg	I	--	SAM	NAMTM1	Six-character file name of scratch file 1.
NAMTM2	--	Intg	I	--	SAM	NAMTM2	Six-character file name of scratch file 2.
LUOO	--	Intg	I	--	SAM	LUOO	Logical unit of display device. May be different from terminal.
OUTFLG	--	Intg	I	--	SAM	OUTFLG	Flag that indicates whether or not output quantities are to be written to disk file.
PGETS	--	Intg	I	--	SAM	PGETS	GET start time as integer values of hour, minute, and second for display page heading.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.2-I.- Concluded

Routine STALK

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
PGETF	--	Intg	I	--	SAM	PGETF	GET final time as integer values of hour, minute, and second for display page heading.
NAMVEC	--	Intg	I	--	SAM	NAMVEC	Array containing name of input trajectory file or all blanks for display page heading.
NAMSIT	--	Intg	I	--	SAM	NAMSIT	Array containing name of input site file or all blanks for display page heading.
GMTR	--	Real	I	hours	SAM	GMTR	Reference Greenwich mean time.
MLINE	--	Intg	I	lines/ page	SAM	MLINE	Maximum lines per display page.
NRNO	--	Intg	I	--	SAM	NRNO	Total number of station contacts.
CFA	--	Real	I	--	SAM	CFA	Conversion factor for angles - external to internal units.
AMILE	--	Real	I	ft/ n. mi.	SAM	AMILE	Feet per nautical mile.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

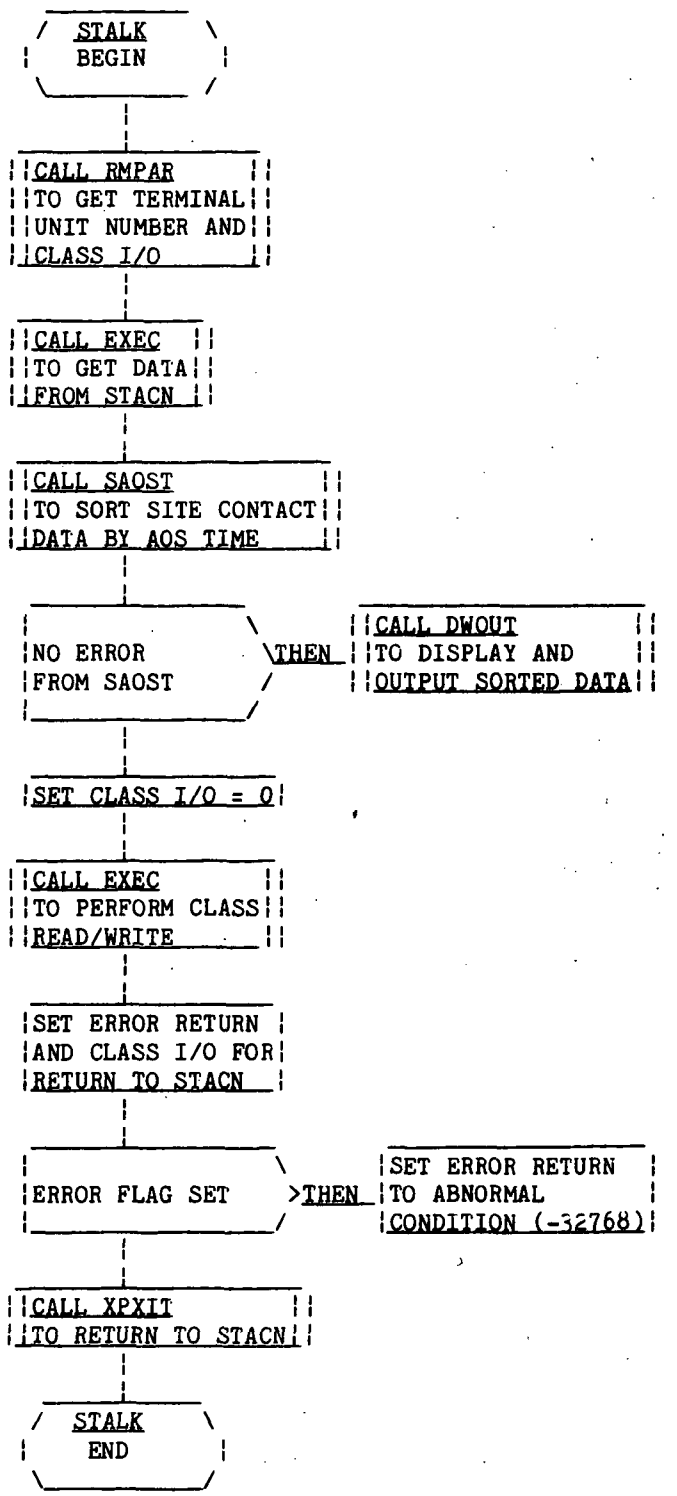


Figure 5.2-1.- STALK functional logic flow.

5.3 ROUTINE NAME - DWOUT

5.3.1 Purpose

The display and write output (DWOUT) subroutine reads the temporary disk file and displays the data on the user's terminal. It writes the data to the output disk file when requested.

5.3.2 Functional Description

A check is made to determine whether or not all the data to be displayed can be held in core. If all data to be displayed can be held in core, the large buffer area used by the sort routine (SAOST) contains the data, in sorted order, that are to be displayed. These data are transferred to 16-word blocks and are displayed on the user's terminal. When an option is set, the data are written to an output file in 16-integer-word records.

If all data to be displayed cannot be held in core, the scratch file containing the sorted acquisition of signal (AOS) times and their file locations are read into a large buffer area. When the buffer is filled, or all data contained on the scratch file have been read, the file locations (record number and block number) are used to read the file containing the computed quantities in the order of the sorted AOS times. The data are displayed as they are read. Before each line is displayed, the count of the number of lines displayed is incremented and a check is made to determine if the number of lines displayed will exceed the maximum lines desirable for the terminal. When the maximum line count is reached, execution of the program pauses, the user is prompted and given the opportunity to make a hard copy of the display before continuing with execution. Execution is resumed by entering a blank character and a carriage return. When the input option OUTFLG is set to "YE", the data are written to the output file in 16-integer-word records. This procedure is repeated until all the sorted data have been read and displayed.

5.3.3 Assumptions and Limitations

None.

5.3.4 Method

None

5.3.5 Routine Input/Output Variables

Table 5.3-I contains the description of the input/output variables for subroutine DWOUT. The calling sequence is

```
CALL DWOUT(IDCB1,IDCB2,IDCB,IERR,IRECO,RECO,IBUFR,
          BUFR,LUO,LU,OUTFLG,NRNO,PGETS,PGETF,NAMVEC,NAMSIT,
          GMTR,CFA,AMILE,MLINE)
```

5.3.6 Functional Logic Flow

Figure 5.3-1 provides the functional logic flow of subroutine DWOUT.

5.3.7 Diagnostics and Debug

There are two error messages that may be displayed by subroutine DWOUT. The message

```
*STACN*SCRATCH FILE ERROR = IIII
```

is displayed when an error occurs while accessing either of the two scratch files. The message

```
*STACN*FILE WRITE ERROR = IIII
```

is displayed when an error occurs while writing the output file. When an error occurs, execution of the subroutine is terminated and control is returned to the calling program with the error flag remaining set as it was returned from the FMP routine.

5.3.8 Special Comments

None.

5.3.9 References

None.

TABLE 5.3-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine DWOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IDCB1	--	Intg	I	--	A	IDCB1	Data control block for scratch file 1.
IDCB2	--	Intg	I	--	A	IDCB2	Data control block for scratch file 2.
IDCB	--	Intg	I	--	A	IBUFO	Data control block for output file.
IERR	--	Intg	O	--	A	IERR	Error code returned from FMP routines.
IRECO	--	Intg	I/O	--	A	IRECO	Array written to output file for each contact.
RECO	--	Real	I/O	--	A	RECO	Array equivalenced to IRECO.
IBUFR	--	Intg	I	--	A	IBUFR	Array containing sorted AOS times.
BUFR	--	Real	I	--	A	BUFR	Array equivalenced to BUFR.
LUO	--	Intg	I	--	A	LUO	Logical unit of display device. May be different from terminal.
LU	--	Intg	I	--	A	LU	Logical unit of terminal.
OUTFLG	--	Intg	I	--	A	OUTFLG	Flag that indicates whether or not output quantities are to be written to disk file.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.3-I.- Continued

Routine DWOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
NRNO	--	Intg	I	--	A	NRNO	Total number of station contacts.
PGETS	--	Intg	I/O	hr/min/ sec	A	PGETS	GET start time as integer values of hour, minute, and second for display page heading.
PGETF	--	Intg	I/O	hr/min/ sec	A	PGETF	GET final time as integer values of hour, minute, and second for display page heading.
NAMVEC	--	Intg	I/O	--	A	NAMVEC	Array containing name of input trajectory file or all blanks for display page heading.
NAMSIT	--	Intg	I/O	--	A	NAMSIT	Array containing name of input site file or all blanks for display page heading.
GMTR	--	Real	I	hours	A	GMTR	Reference Greenwich mean time.
CFA	--	Real	I	--	A	CFA	Conversion factor for angles - external to internal units.
AMILE	--	Real	I	ft/ n. mi.	A	AMILE	Feet per nautical mile.
MLINE	--	Intg	I	lines/ page	A	MLINE	Maximum lines per display page.
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.3-I.- Continued

Routine DWOUT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ACQAZ	--	Real	O	deg	T	--	Azimuth at AOS.
DT	--	Real	O	min	T	--	[Time at LOS - time at AOS.]
ELMAX	--	Real	O	deg	T	--	Maximum elevation.
IAOSH	--	Intg	O	hr	T	--	Hours at AOS.
IAOSM	--	Intg	O	min	T	--	Minutes at AOS.
IAOSR	--	Intg	O	n. mi.	T	--	Range at AOS.
IAOSS	--	Intg	O	sec	T	--	Seconds at AOS.
IRECO(1)	--	2CH	O	--	T	--	Site name (characters 1,2).
IRECO(2)	--	2CH	O	--	T	--	Site name (characters 3,4).
IRECO(3)	--	2CH	O	--	T	--	Site name (characters 5,6).
IRECO(4)	--	Intg	O	--	T	--	Orbit number.
LOSH	--	Intg	O	--	T	--	Hours at LOS.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.3-I.- Concluded

Routine DWOUT

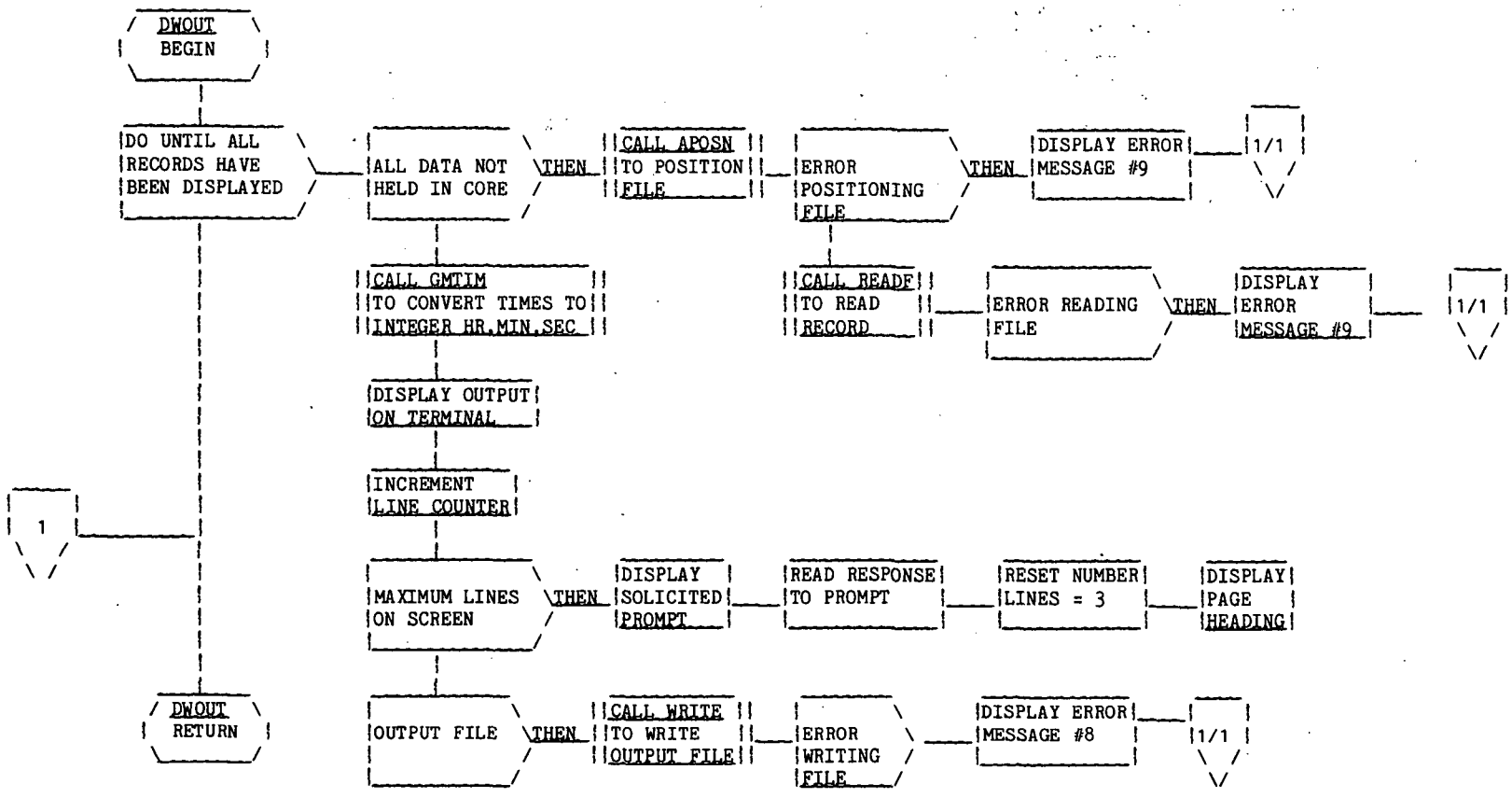
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LOSM	--	Intg	O	--	T	--	Minutes at LOS.
LOSS	--	Intg	O	--	T	--	Seconds at LOS.
MINR	--	Intg	O	--	T	--	Minimum range.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



SPACM-56

Figure 5.3-1.- DWOUT functional logic flow.

5.4 ROUTINE NAME - SAOST

5.4.1 Purpose

The sort on acquisition of signal (SAOST) subroutine sorts in chronological order by time of acquisition of signal (AOS) the computed values that are to be displayed and written to the output disk file.

5.4.2 Functional Description

A check is made to determine if all the data to be sorted can be held in core. If all data can be held in core, the Hewlett-Packard file management (FMP) routine READF is called to read all data records into an array. Once all data to be sorted are read into the array, the data are arranged in order of ascending AOS time. After each call to READF, the error flag is checked. If the error flag is set, control is returned to the calling routine.

When there are more data to be sorted than can be held in core, the procedure is to read each 16-word data record and retain in a large buffer area the AOS time and the file location (record number and block number) of the data associated with the AOS time. When the large buffer area is filled, the buffer containing the AOS times and file locations is written to a scratch file. This procedure is continued until all AOS times and their file locations have been read and saved.

The scratch file containing the AOS times and the file locations of the data are positioned to the first record. The large buffer area is divided into two buffers of equal size or half-arrays. Data from the initial locations of the scratch file are read into the first half-array. The FMP routine LOCF is called to obtain the file location of the next record, and this file location is saved. The next data contained on the scratch file is read into the second half-array, and data contained in the two half-arrays are sorted by ascending AOS time. When the sort is completed, the second half of the sorted array is written to the scratch file at the saved file location. Another half-array of data are read from the scratch file into the second half-array. Again the data in the second half-array are sorted with the previously sorted first half-array, and the sorted second half-array is written to the scratch file. This procedure is continued until all half-arrays except the last have been used as the first half-array and sorted with all other half-arrays.

After each call to an FMP routine, the error flag is checked. If the error flag is set, control is returned to the calling routine. When the sort is completed, control is returned to the calling routine.

5.4.3 Assumptions and Limitations

None.

5.4.4 Method

A modified bubble sort is used to sort the station contact data in order of chronological AOS times. A discussion of the bubble sort is given in reference 1. The bubble sort is an interchange sort where the items being sorted are interchanged in the array in which they reside in core. In the standard bubble sort, the first two values in the array are compared. If the second value is less than the first, the two are interchanged; if the first value is smaller than the second, no interchange is made. The second and third values are compared next. If the third value is less than the second, the two are interchanged or the third number is "bubbled" to the second position. This value is then compared with the first value and "bubbled" up in the array as far as it will go. The value in the next position is used as the starting position for the next series of comparisons. In the variation of the bubble sort used in this routine, the first value is set as the minimum value and compared with the second value. If the second value is less than the minimum, it is set as the minimum and the position or index of the new minimum is saved. The minimum is compared with the third value in the array and the process of comparing and setting the minimum with its index is continued throughout the array. At the end of the pass through the array, the first value in the array is interchanged with the value in the array at the saved index position. The comparison process starts again with the second value as the initial minimum and comparisons are made with the remaining (n-1) values. The sort is completed and the values are in ascending numeric order when (n-1) initial values have been compared.

5.4.5 Routine Input/Output Variables

Table 5.4-I provides the description of the input/output variables. The calling sequence is

```
CALL SAOST(IDCB1,IDCB2,IBUFR,BUFR,IRECO,NAMTM2,NRNO,IERR,LU)
```

5.4.6 Functional Logic Flow

Figure 5.4-1 provides the functional logic flow of subroutine SAOST.

5.4.7 Diagnostics and Debug

One error message may be displayed by subroutine SAOST. The message

```
"*STACN*SCRATCH FILE ERROR ="IIII
```

is displayed when an error occurs while accessing either of the two scratch files. When an error occurs, execution of the subroutine is terminated and control is returned to the calling program with the error flag remaining set as it was returned from the FMP routine.

5.4.8 Special Comments

None.

5.4.9 References

1. Martin, W. A.: Sorting. ACM Computing Surveys, Vol. 3, No. 4, December 1971.

TABLE 5.4-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine SAOST

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IDCB1	--	Intg	I/O	--	A	IDCB1	Data control block for scratch file 1
IDCB2	--	Intg	I/O	--	A	IDCB2	Data control block for scratch file 2
IBUFR	--	Intg	I/O	--	A	IBUFR	Array for sorting AOS times
BUFR	--	Intg	I/O	--	A	BUFR	Array equivalenced to IBUFR
IRECO	--	Intg	I	--	A	IRECO	Array written to scratch file for each contact
NAMTM2	--	Intg	I	--	A	NAMTM2	Six-character file name of scratch file 2
NRNO	--	Intg	I	--	A	NRNO	Total number of station contacts
IERR	--	Intg	O	--	A	IERR	Error code returned from FMP routines
LU	--	Intg	I	--	A	LU	Logical unit of terminal
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

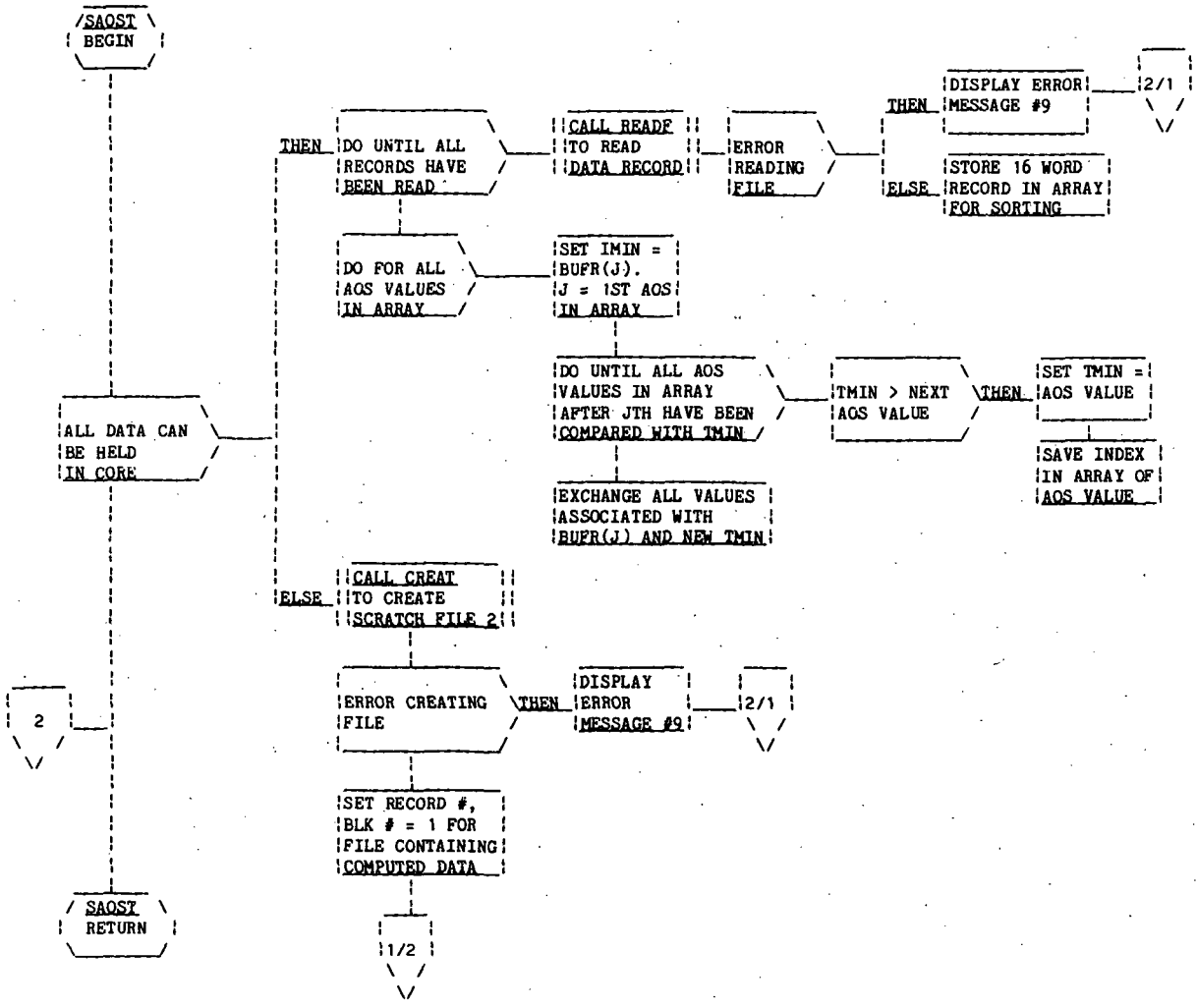


Figure 5.4-1.- SAOST functional logic flow.

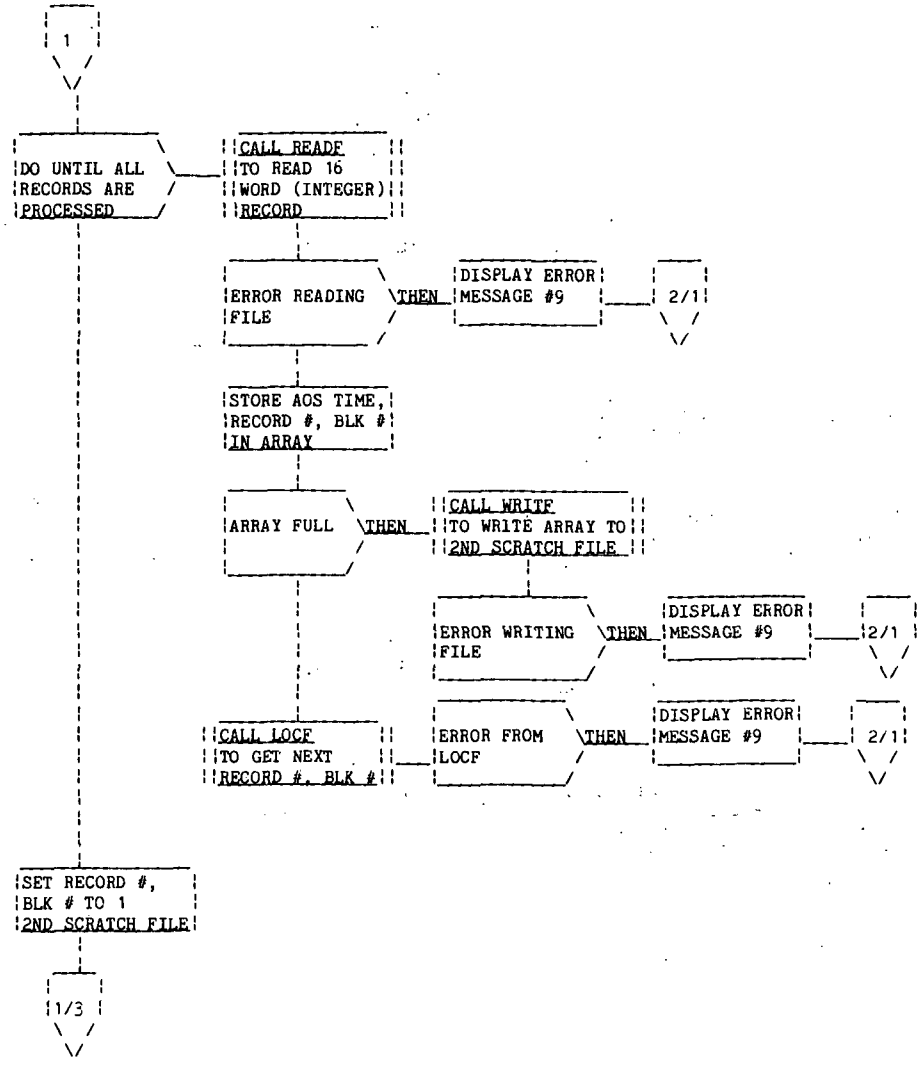


Figure 5.4-1.- Continued.

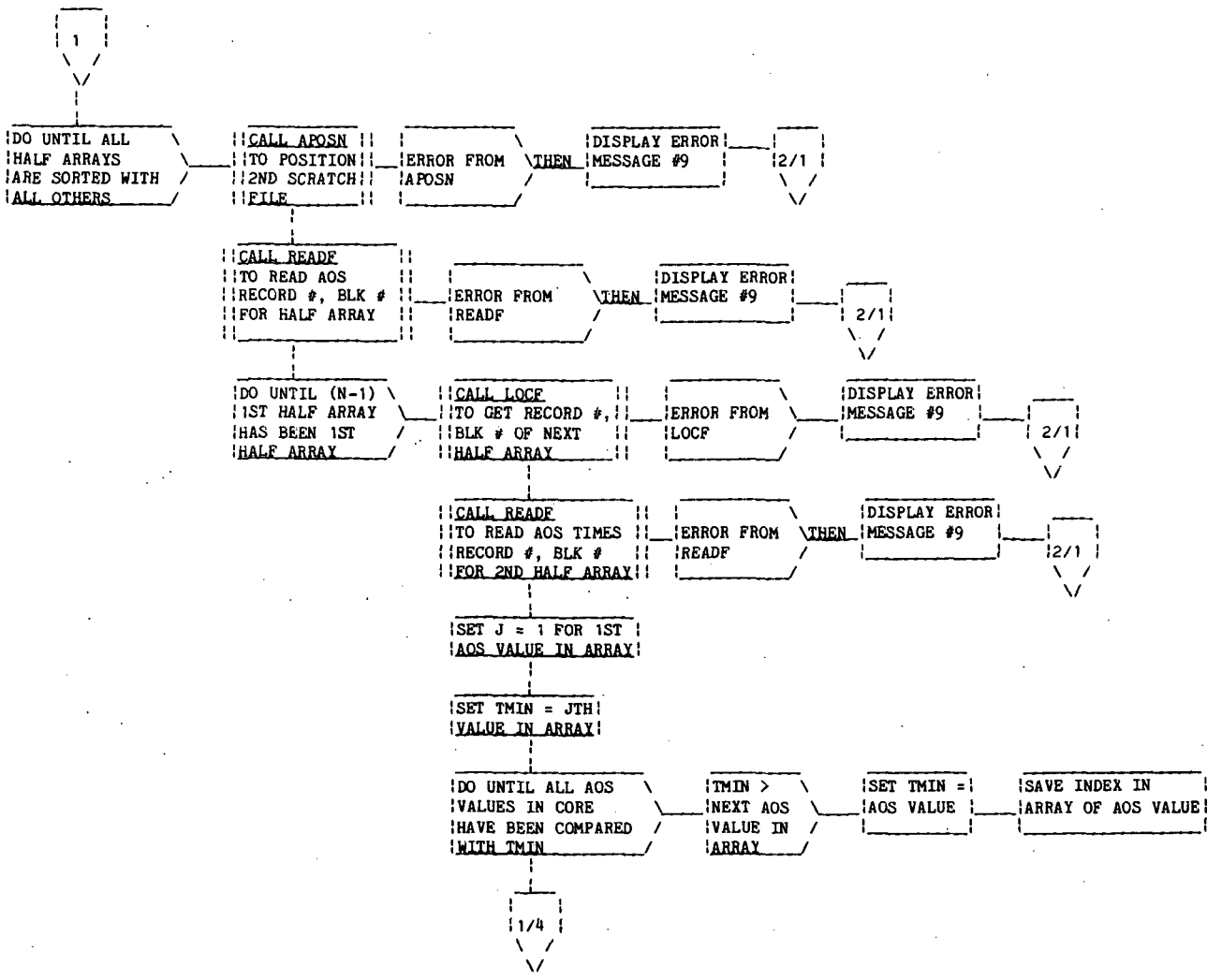


Figure 5.4-1.- Continued.

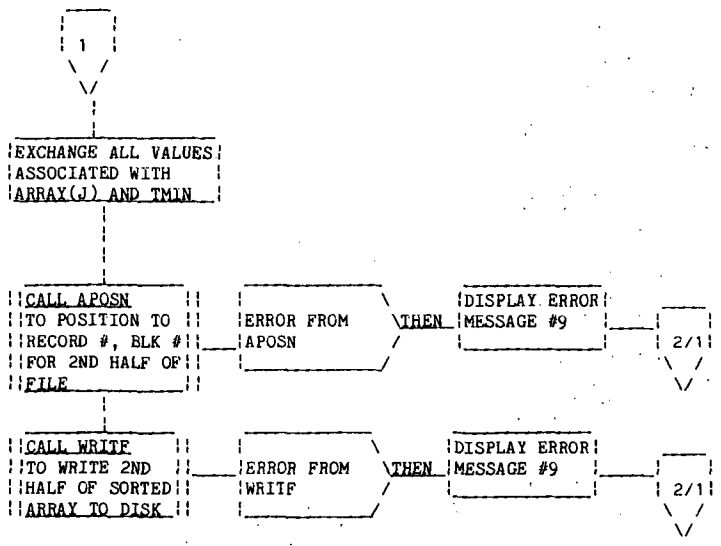


Figure 5.4-1.- Concluded.

5.5 ROUTINE NAME - AZAOS

5.5.1 Purpose

The azimuth at acquisition of signal (AZAOS) routine computes the azimuth at acquisition of signal (AOS) for both the "slow" and "fast" AOS computational methods. The subroutine was written to avoid repeating the same block of code in two places in the main routine STACN.

5.5.2 Functional Description

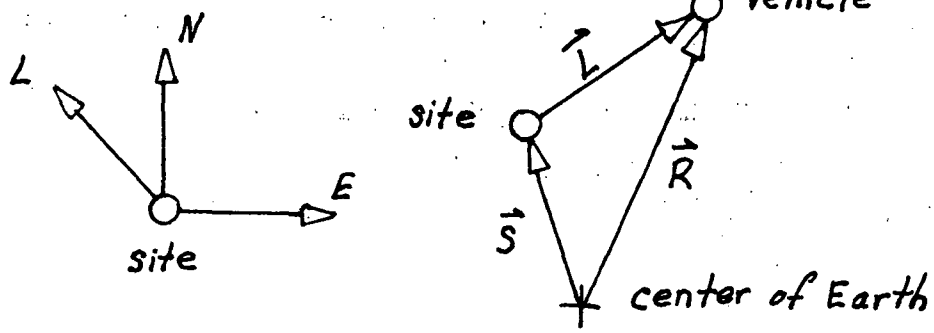
None.

5.5.3 Assumptions and Limitations

None.

5.5.4 Method

Azimuth may be defined in a number of ways. The subroutine AZAOS computes the azimuth from the north point eastward from 0° to 360°.



$$\vec{L} = \begin{bmatrix} R_{vehicle}\ R_1 - S_1 \\ R_{vehicle}\ R_2 - S_2 \\ R_{vehicle}\ R_3 - S_3 \end{bmatrix}^T, \quad \vec{R} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}^T, \quad \text{and} \quad \vec{S} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix}^T$$

$\vec{R}$, $\vec{S}$, and $R_{vehicle}$ are defined in section 5.1.4. The $\vec{L}$ vector is unitized by calling the FDS-1 utility routine UNIT.

$$\vec{E} = \begin{bmatrix} -S_2 \\ S_1 \\ 0 \end{bmatrix}^T \quad \text{and} \quad \vec{N} = \vec{S} \times \vec{E}$$

The X and Y components of the azimuth are computed.

$$AZIX = \vec{L} \cdot \vec{N}$$

$$AZIY = \vec{L} \cdot \vec{E}$$

The azimuth, AZAS, is computed as follows:

$$AZAS = \tan^{-1}(AZIX/AZIY)$$

Table 5.5-I provides a list of mathematical symbols versus program symbols.

5.5.5 Routine Input/Output Variables

Table 5.5-II provides a list of the input/output variables. The calling sequence is

```
CALL AZAOS(RAD,RVEC,SVEC,TWOPI,AZAS)
```

5.5.6 Functional Logic Flow

None.

5.5.7 Diagnostics and Debug

None.

5.5.8 Special Comments

None.

5.5.9 References

None.

TABLE 5.5-I.- MATH SYMBOLS VERSUS CODE SYMBOLS
[AZIMUTH CALCULATIONS]

Math symbol	Code symbol
AZAS	AZAS
AZIX	AZIX
AZIY	AZIY
E	EVEC
L	L
N	NVEC
$R_{vehicle}$	RAD
R	RVEC
S	SVEC
2Π	TWOPI

TABLE 5.5-II.- ROUTINE INPUT/OUTPUT VARIABLES

Routine AZAOS

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
RAD	—	Real	I	--	A	RAD	Radius of ascending node
RVEC	--	Real	I	--	A	RVEC	Unit vector of vehicle
SVEC	—	Real	I	--	A	SVEC	Unit vector ground site
TWOPI	2π	Real	I	--	A	TWOPI	2π
AZAS	—	Real	O	--	A	AZAS	Azimuth at acquisition of signal

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

5.6 ROUTINE NAME - CPA

5.6.1 Purpose

The closest point of approach (CPA) computes the closest point of approach of an orbiting vehicle to a ground site. The subroutine was written to avoid repeating the same block of code in two places in the main routine STACN.

5.6.2 Functional Description

None.

5.6.3 Assumptions and Limitations

None.

5.6.4 Method

The method used in the subroutine is a portion of the method used for the CPA computation in processor LOPT. (Refer to the "Closest Point of Approach" section in LOPT.)

5.6.5 Routine Input/Output Variables

Table 5.6-I provides a list of the input/output variables. The calling sequence is

```
CALL CPA(H,SVEC,UCPA,SINB)
```

5.6.6 Functional Logic Flow

None.

5.6.7 Diagnostics and Debug

None.

5.6.8 Special Comments

None.

5.6.9 References

None.

TABLE 5.6-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine CPA

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
H	[H]	Real	I	--	A	H	Orbital angular momentum
SVEC	[S]	Real	I	--	A	SVEC	Unit vector in direction of ground site
UCPA	U_{CPA}	Real	O	--	A	UCPA	Argument of latitude at CPA
SINB	Sinβ	Real	O	--	A	SINB	H · S

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

SUMMARY TABLE PRINT PROCESSOR (STP)

1.0 PURPOSE

STP is a Summary Table Print utility processor whose purpose is to read a Summary Table Array (STA) of the user's Active Work Area (AWA) and print out all parameters with their values and units in the format shown in table 4-IV.

2.0 FUNCTIONAL DESCRIPTION

The interface table for the STP utility processor contains only one parameter, the input parameter SUMTAB. SUMTAB is a summary table standard data structure as described in figure 2-1 and stored as a data element in the user's AWA. STP obtains the summary table identified by the input SUMTAB, formats the data, and displays it as:

INDEX VARIABLE NAME = VALUE UNITS

e.g., 14 TIME = 24.0 HRS

3.0 ASSUMPTIONS AND LIMITATIONS

a. The only acceptable response for SUMTAB is the name of a summary table; literal data cannot be supplied.

b. The summary table name supplied through the interface table cannot be subscripted; i.e., only the whole summary table can be printed.

c. The display generated by this processor is not paged.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The interface table for the STP utility processor is provided in table 4-I.

b. Interface table data array definitions - The interface table data array definition for the STP utility processor is provided in table 4-II.

c. Interface table data file definitions - None.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definition table - The processor display format for the STP utility processor is provided in table 4-III and the display parameter definition table for STP is shown in table 4-IV.

f. Processor message table - The message table for the STP utility processor is provided in table 4-V.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VI.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR STP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
SUMTAB	AWA	Free	O	1200	(8,150)	--	Input summary table

NOTES	CLASS	TYPE			USE
		Free	2CH	72CH	I = Input
	AWA	Intg	6CH	Mix	O = Output
	Disk	Real	18CH	Symb	I/O = Input/Output
		Dubl	36CH		

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR STP

Array name	Index location	Default value	Definition
SUMTAB	(1,1) . . . (8,150)	—	Input summary table, see figure 2-1 for the format of a summary table. Each column is an individual parameter with the first parameter being an error indicator.

TABLE 4-III.- PROCESSOR DISPLAY FORMAT

PROCESSOR STP

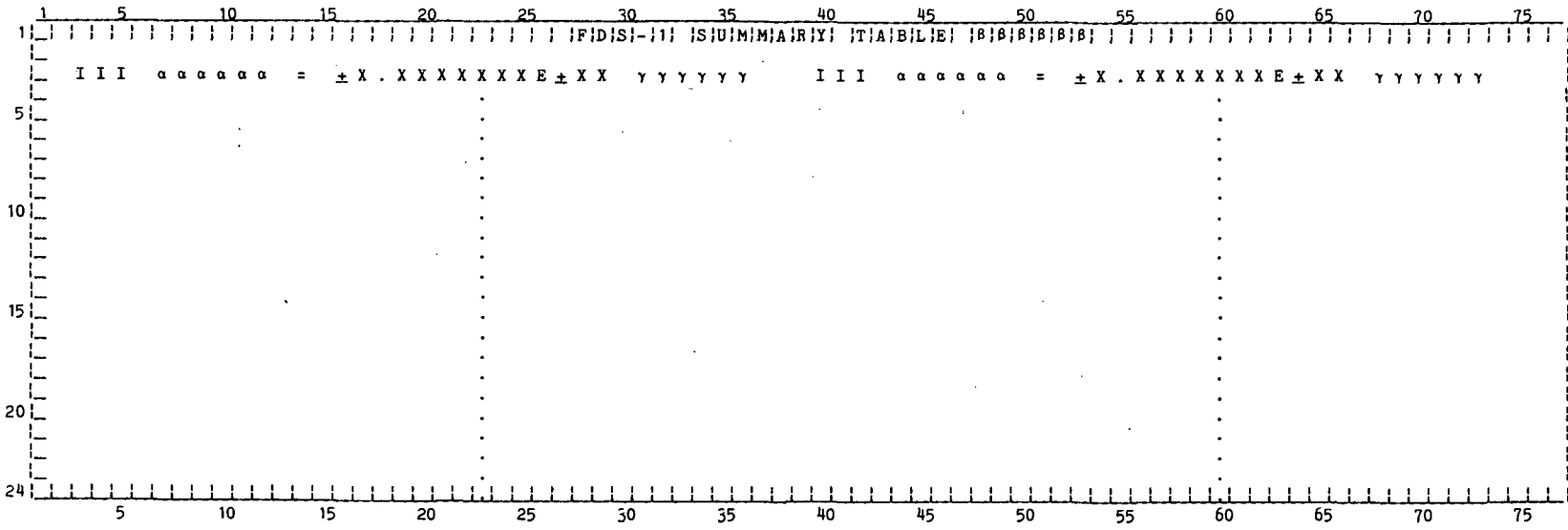


TABLE 4-IV.- DISPLAY PARAMETER DEFINITION TABLE

PROCESSOR STP

FDS-1 SUMMARY TABLE ßßßßßß

Display parameter label	Parameter definition
ßßßßßß	Six-character alphanumeric summary table name
III	Parameter index
αααααα	Parameter name
±XXXXXXE±XX	Parameter value
γγγγγγ	Parameter units value

TABLE 4-V.- PROCESSOR MESSAGE TABLE

PROCESSOR STP

MSG no.	Message ID block	Message text block and explanation
1	*STP*	ABORT, SUMTAB RESPONSE REQUIRES A NAME NOT LITERAL DATA Meaning: STP aborted because the user supplied literal data, not an alphanumeric name to SUMTAB prompt, upon completing the interface table for STP. Severity: Processor STP aborted due to irrecoverable error. Action required by user: Run the Interface Table Editor for STP and supply an alphanumeric name corresponding to the summary table to be displayed. Then rerun processor STP using the newly created interface table for STP.
2	*STP*	ABORT, SUMMARY TABLE ßßßßßß IS NOT TYPED FREE IN AWA Meaning: The name (ßßßßßß) of the AWA data element supplied in STP's interface table is not typed free and may not be a summary table. Severity: Processor STP aborted due to irrecoverable error. Action required by user: User should check the name and make sure it is a summary table.
3	*STP*	ABORT, SUMMARY TABLE ßßßßßß MATRIX COLUMN LENGTH NOT EQUAL 8 Meaning: The name (ßßßßßß) of the AWA data element supplied in STP's interface table does not have a matrix column length = 8 and may not be a summary table. Severity: Processor STP aborted due to irrecoverable error. Action required by user: User should check the name and make sure it is a summary table.
4	*STP*	ABORT, SUMMARY TABLE ßßßßßß CAN NOT BE INDEXED Meaning: Message to the user that STP aborted because the user tried to index into the summary table array by specifying a name and subscript to SUMTAB prompt; e.g., LOPTAB(15) instead of LOPTAB. Severity: Processor STP aborted due to irrecoverable error. Action required by user: Run the Interface Table Editor for STP and supply only an alphanumeric name not a name and subscript to SUMTAB prompt. Then rerun processor STP using the newly created interface table for STP.

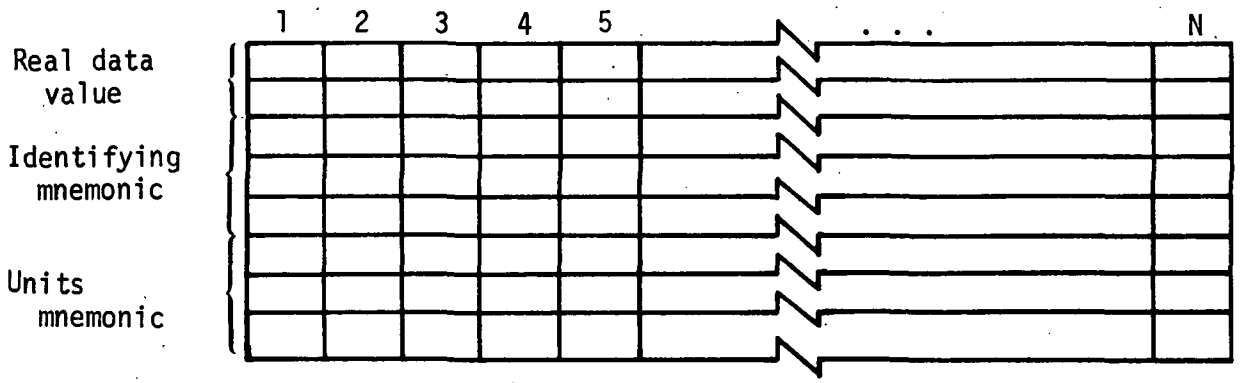
TABLE 4-VI.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR STP

Processor name	Processor abstract prompt (maximum 256 characters)
STP	Summary Table Print (STP) utility processor reads a summary table array from the user's AWA and prints all parameters, their value, and units in a formatted display.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
SUMTAB	An input parameter supplying the name of the summary table to be displayed. The format of the summary table data element is contained in the FDS-1 standards. The maximum size of the array is (8,150).

SUMTAB(8,N) = An 8 by N integer array which is typed "free" in a processor's interface table.



N = Number of summary table entries for the processor.

8 = The number of words per summary table entry.

Where: Words 1 and 2 of each entry are the real data value.

Words 3, 4, and 5 of each entry are the identifying mnemonic (up to six characters) for that entry.

Words 6, 7, and 8 of each entry are the units mnemonic (up to six characters) for that entry.

Note: For N = 1 the entry is a reserved error flag for summary table output processing. The contents for the eight words of this entry will be:

Words 1 and 2, value of the error flag (real number)

Words 3, 4, and 5, "ERRORØ"

Words 6, 7, and 8, "ØØØØØØ"

Figure 2-1.- Summary table format.

5.0 PROCESSOR ROUTINES

5.1 MAIN PROGRAM - STP

5.1.1 Purpose

STP is the main program for the STP processor and performs all its functions. The main purpose of STP is to read a Summary Table Array (STA) out of the user's Active Work Area (AWA) and print all parameters, indices, units, and values in a formatted display as shown in table 4-IV.

5.1.2 Functional Description

The STP utility processor provides the FDS-1 user with the capability to print the STA parameters in a formatted form. Initially STP calls the RTE executive service routine, RMPAR, to obtain the logical unit number of the user's terminal and set IERROR, STP's error flag, to zero. Next, a call is made to the FDS-1 attribute retrieval routine, XPATR, to obtain the attributes of the STA that was specified in the interface table. The attributes obtained are the NAME, TYPE, IDIM, and DSPTYP for the specified STA. These attributes are checked for validity and, if any are invalid, an appropriate error message is displayed, the error flag is set, and the processor aborts with an error return to FDS. If all attributes are valid, then the data are reformatted and displayed.

The actual size of the summary table (to be retrieved from the AWA) may vary from an (8,1) array to an (8,150) array; (8,150) is the maximum size because summary tables are stored as AWA data elements and data elements cannot exceed 1200 words. Since the size of the summary table to be displayed is variable (and not known at routine build time), an internal buffer size of (8,150) or some method of buffering the data from the AWA is required. The method used in this routine is a buffering technique with the internal buffer size set at (8,50). Thus, for the maximum size summary table (8,150), three AWA accesses are required to retrieve and display the entire table. Currently, most processor summary tables contain less than 50 parameters; therefore, in the majority of cases only one AWA access will be required to retrieve the summary table. The internal buffer size of (8,50) was chosen because it is large enough to contain most processor summary tables and also divides evenly into the maximum size of (8,150).

To begin the retrieval and display process, the size (amount of data) to be retrieved from the AWA is initialized to a maximuim (i.e., the internal buffer size (8,50)), and the displacement into the summary table array is initialized to zero. This initialization is done by setting BSIZE = 400 and DISP = 0. Next, the number of AWA accesses (NUMREQ) required to retrieve the entire summary table is computed by dividing the SIZE of the summary table to be retrieved by the size (400) of the internal buffer, (SIZE was one of the attributes obtained in the XPATR call).

Once the initialization is complete, and the number of AWA accesses determined, the actual retrieval and display of the summary table is accomplished in a large

loop where the index on the loop goes from 1 to NUMREQ. The first step within the loop is to determine if this is the last pass through the loop; if it is the last pass, then the size (BSIZE) of the data to be retrieved on this pass is recomputed to be equal to the data remaining in the AWA data element. This ensures that the request to the XPGTI routine will not attempt to read past the end of the AWA data element array (such an attempt would result in an error message from XPGTI and processor abort).

The second step in the loop is to initialize the internal buffer to zero. Then a call is made to the XPGTI routine to retrieve BSIZE words of data from the summary table starting at displacement (DISP) from the beginning of the summary table. The retrieved data are stored in the internal array SUMTAB.

The third step in the loop is to initialize the display index (INDEX) that will be printed along with the summary table parameters. The displacement (DISP) for the next retrieval of data from the summary table is also preset at this point.

The fourth and final step in the loop is an inner loop that reformats and prints the data contained in SUMTAB. SUMTAB is the internal buffer dimensioned (8,50) that contains the retrieved summary table; it is a mixture of real and Hollerith data. For each summary table parameter, it contains the real value of the parameter, a six-character Hollerith name for the parameter, and a six-character mnemonic that defines the units for the parameter. For the purpose of implementing the display logic, the SUMTAB array is divided into the three arrays: (1) VNAME, which contains the six-character name for each parameter; (2) UNITS, which contains the six-character units mnemonic for each parameter; and (3) VALUE, which contains the real value for each parameter. Each parameter value, name, and units mnemonic is moved from SUMTAB into the VALUE, VNAME, and UNITS arrays, respectively, and then immediately written to the output device. The write to the output device is performed immediately rather than moving all parameters from SUMTAB into VALUE, VNAME, and UNITS first and then printing. The reason for this approach is that it is thought that some speed and efficiency will be gained by having the next parameter being moved from SUMTAB while the output of the previous one is still in progress. Since both parameters are displayed in the same format (but at different times), two calls are made to the same write statement per line. Because of the fact that in FORTRAN IV a carriage return/line feed is issued after executing a write statement, flag K is initialized to 0 and alternated from 0 to 1 to determine when to suppress the carriage return/line feed. Essentially, the first write command for any given line requires a carriage return/line feed suppression, but the second write command does not. Once all the data in SUMTAB have been reformatted and displayed, the inner loop is concluded.

Since this was also the final step in the outer loop, the controlling index is compared with NUMREQ. Once this loop is completed, the normal return flag is set and XPXIT terminates processor STP with a normal return to FDS.

5.1.3 Assumptions and Limitations

See Assumptions and Limitations in section 3.0.

5.1.4 Method

See Functional Description in section 5.1.2.

5.1.5 Routine Input/Output Variables

The STP input/output variables are presented in table 5.1-I.

5.1.6 Functional Logic Flow

The functional logic flow for STP is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

None.

5.1.8 Special Comments

STP will display a summary table exactly as it appears in the AWA: i.e., nulls are printed as nulls, not as blanks.

5.1.9 References

1. Flight Design System-1, System Design Document, Standards. Vol. VI, JSC IN 77-FM-18, Rev. 1, January 1978.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine STP

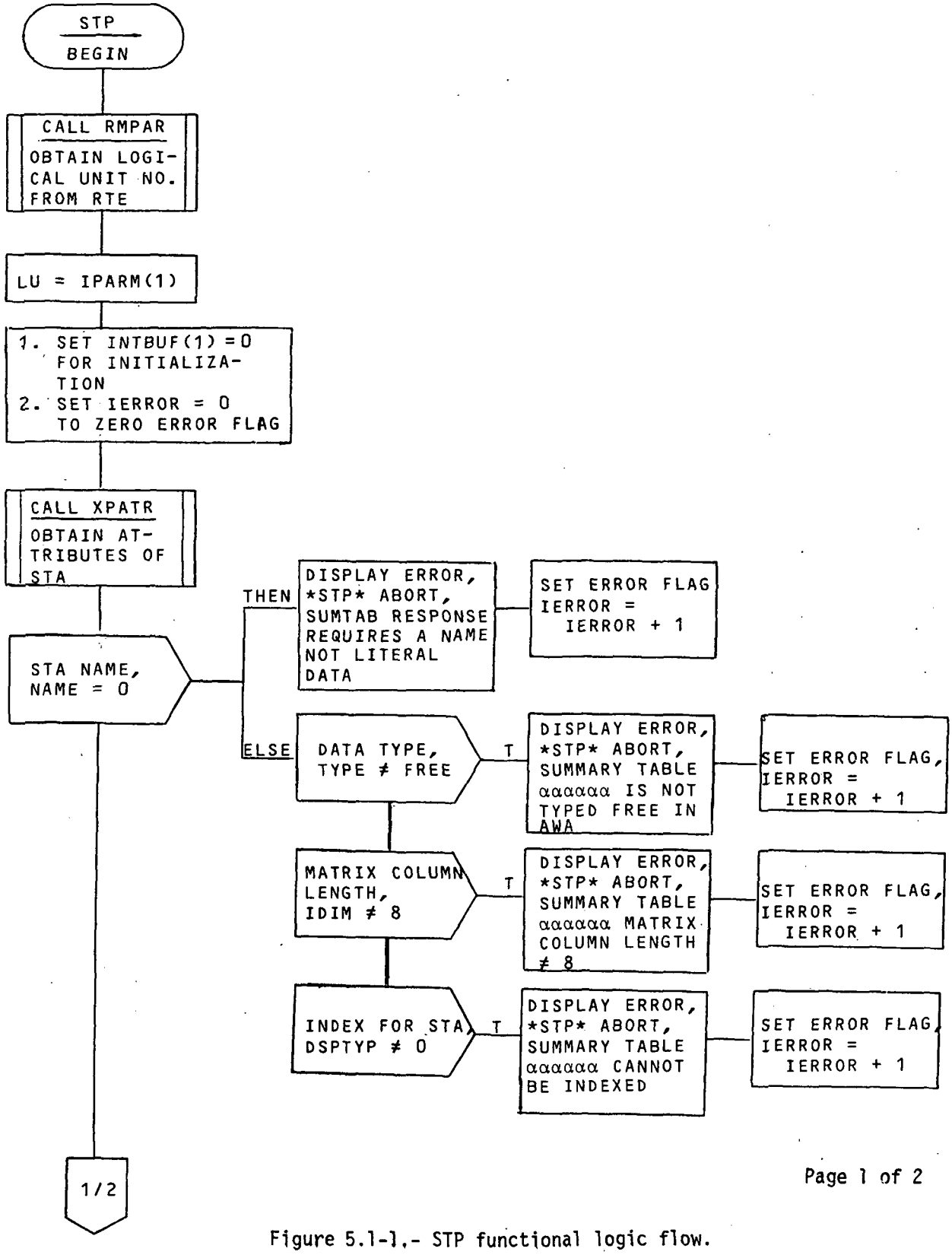
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
SUMTAB	—	Free	I	--	IT	SUMTAB	Input summary table

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

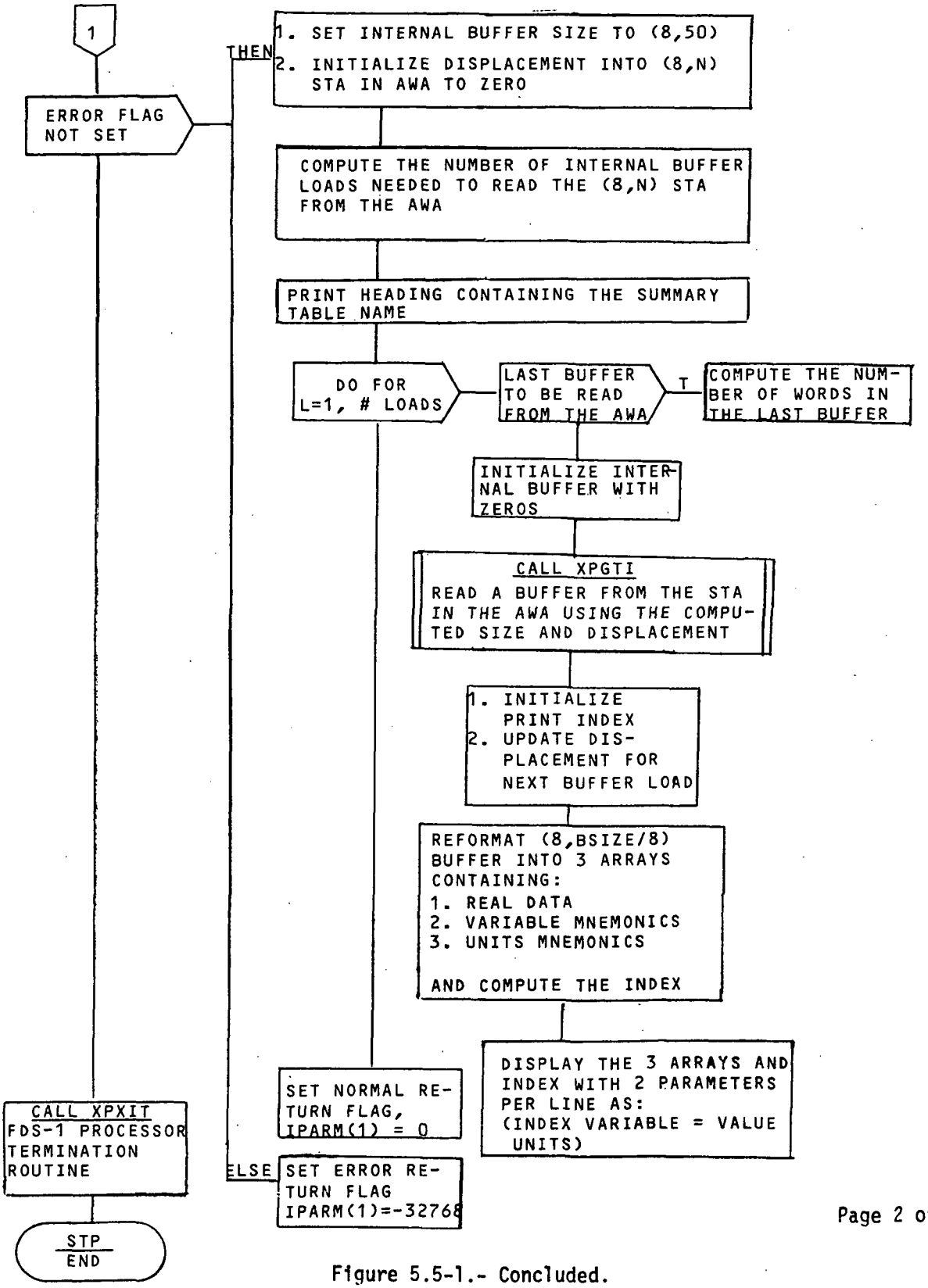
USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 2

Figure 5.1-1.- STP functional logic flow.



Page 2 of 2

Figure 5.5-1.- Concluded.

STATE VECTOR UNITS CONVERSION PROCESSOR (SVUCP)

1.0 PURPOSE

The State Vector Units Conversion Processor (SVUCP) has the capability to accept a position/velocity state vector expressed in one set of units and convert the state to a different set specified by the user.

2.0 FUNCTIONAL DESCRIPTION

The SVUCP takes an input state vector from the users' active work area (AWA) and determines (via an input parameter) the input units for that state vector. The units for the output state are also specified by an input parameter. The input/output state vector arrays are the standard position/velocity state vectors defined in section 7.3 of JSC IN 78-FM-60, volume I. The input/output element set may be any of the 15 element sets defined in table 7.3-VI of JSC IN 78-FM-60, volume I, except the KS element set. The processor converts from the units associated with the input state to the user-specified output units, and stores the resulting state in the output state vector parameter.

3.0 ASSUMPTIONS AND LIMITATIONS

The only acceptable units are given in table 7.1-I of JSC IN 78-FM-60, volume I. If a user selects the "USER SPECIFIED" option for units, he must first execute the PHYDM processor or have an acceptable !SESCN data element in his AWA. The KS element set for state vectors input/output is not supported by this processor.

4.0 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The processor input and output parameters are presented in table 4-I.

b. Interface table data array definitions - The array parameters in the Interface Table are defined in table 4-II.

c. Interface table data file definitions - None.

d. Processor solicited (prompted) inputs - None.

e. Processor displays and display parameter definition table - None.

f. Processor message table - The processor message table is presented in table 4-III.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-IV.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR SVUCP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
SESCON	AWA	Free	I	90	90	!SESCN	Session constants array.
GLOCON	AWA	Free	I	180	180	!!GLCN	Global constants array, master data base element.
ANGUNI	AWA	6CH	I	3	1		Parameter specifying the input state units for angles = "DEG"; Degrees* = "RAD"; Radians = "Ø"; User-specified external unit choice
ANGUNO	AWA	6CH	O	3	1		Parameter specifying the output states units for angles = "DEG"; Degrees* = "RAD"; Radians = "Ø"; User-specified external unit choice
DSTUNI	AWA	6CH	I	3	1		Parameter specifying the input units for distance = "FT"; Feet* = "M"; Meters = "NM"; Nautical miles = "ER"; Earth Radii = "KM"; Kilometers = "Ø"; User specified external unit choice

NOTES

CLASS: AWA, Disk

TYPE: Free, Intg, Real, Dubl, 2CH, 6CH, 18CH, 36CH, 72CH, Mix, Symb

USE: I = Input; O = Output; I/O = Input/Output

*Default external unit choice

TABLE 4-I.- Continued

PROCESSOR SVUCP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
DSTUNO	AWA	6CH	O	3	1		Parameter specifying the output units for distance = "FT"; Feet* = "M"; Meters = "NM"; Nautical miles = "ER"; Earth Radii = "KM"; Kilometers = "Ƃ"; User specified external units choice
TIMUNI	AWA	6CH	I	3	1		Parameter specifying the input units for time = "SEC"; Seconds* = "MIN"; Minutes = "HR"; Hours = "DAYS"; Days = "Ƃ"; User specified external unit choice
TIMUNO	AWA	6CH	O	3	1		Parameter specifying the output units for time = "SEC"; Seconds* = "MIN"; Minutes = "HR"; Hours = "DAYS"; Days = "Ƃ"; User specified external unit choice

NOTES	CLASS	TYPE			USE	*Default external unit choice
	AWA	Free	2CH	72CH	I = Input	
	Disk	Intg	6CH	Mix	O = Output	
		Real	18CH	Symb	I/O = Input/Output	
		Dubl	36CH			

TABLE 4-I.- Continued

PROCESSOR SVUCP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
MASUNI	AWA	6CH	I	3	1		Parameter specifying the input units for mass = "LBM"; Pounds mass* = "KG"; Kilograms = "SLG"; Slugs = "Ø"; User specified external unit choice
MASUNO	AWA	6CH	O	3	1		Parameter specifying the output units for mass = "LBM"; Pounds mass* = "KG"; Kilograms = "SLG"; Slugs = "Ø"; User specified external unit choice
LENUNI	AWA	6CH	I	3	1		Parameter specifying the input units for length = "FTL"; Feet* = "ML"; Meters = "INL"; Inches = "Ø"; User specified external unit choice
LENUNO	AWA	6CH	O	3	1		Parameter specifying the output units for length = "FTL"; Feet* = "ML"; Meters = "INL"; Inches = "Ø"; User specified external unit choice

NOTES

CLASS: AWA, Disk

TYPE: Free 2CH 72CH; Intg 6CH Mix; Real 18CH Symb; Dubl 36CH

USE: I = Input; O = Output; I/O = Input/Output

*Default external unit choice

TABLE 4-I.- Concluded

PROCESSOR SVUCP

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Values stored in default interface table	Definition
VELUNI	AWA	6CH	I	3	1		Parameter specifying the input units for velocity = "FT/S"; Feet/second* = "M/S; Meters/second = "b"; User specified external unit choice
VELUNO	AWA	6CH	O	3	1		Parameter specifying the output units for velocity = "FT/S"; Feet/second* = "M/S"; Meters/second = "b"; User specified external unit choice
SVIN	AWA	R	I	30	15		Parameter specifying the output state vector
SVOUT	AWA	R	O	30	15		Parameter specifying the output state vector

NOTES	CLASS	TYPE			USE	*Default external unit choice
	AWA	Free	2CH	72CH	I = Input	
	Disk	Intg	6CH	Mix	O = Output	
		Real	18CH	Symb	I/O = Input/Output	
		Dubl	36CH			

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine PTP

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IPARM	--	Intg	I/O	--	--	--	System parameter
LU	--	Intg	O	--	--	--	Logical unit number of user's terminal
INTBUF	--	Intg	O	--	--	--	Interface table header buffer
MRBUFF	--	Intg	O	--	--	--	Manager request buffer
INUM	--	Intg	O	--	--	--	Interface table access array
CLASS	--	6CH	I	--	IT	CLASS	Class of phase table
TYPE	--	2CH	I	--	IT	TYPE	Type of phase table
INDEX	--	Intg	I	--	IT	INDEX	Vector number to start the display
NUMBER	--	Intg	I	--	IT	NUMBER	Number of vectors to be displayed
PROMPT	--	6CH	I	--	IT	PROMPT	Mode of prompting user
PROCON	--	Free	I	--	IT	PROCON	Processor constants array
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 4-III.- PROCESSOR MESSAGE TABLE

PROCESSOR SVUCP

MSG no.	Message ID block	Message text block and explanation
1	*SVUCP*	"ßßßß IS NOT A VALID INPUT FOR PARAMETER ßßßßßß" Meaning: ßßßß is the user response to parameter keyword ßßßßßß Severity: Processor terminates Action required by user: Use Interface Table Editor to correct input
2	*SVUCP*	"THE KS STATE VECTOR ELEMENT SET IS NOT SUPPORTED" Meaning: The input state vector contains a KS element set. The KS element set is not supported by this processor. Severity: Processor terminates. Action required by user: Examine the input state vector to determine if it is a state vector. Use PSV processor.
3	*SVUCP*	DEKOD ERROR IER = $n_4n_3n_2n_1$ Meaning: The input state referenced by parameter SVIN does not contain a state vector with an FDS-1-supported reference axis and/or element set. The referenced data may not be a state vector. The value of IER indicates vector type code error with the following meaning: n_1 = 1 reference axis error (J> 4) n_2 = 1 element type error (K> 2) n_3 = 1 element set error (N invalid) n_4 = 1 reference axis/element set combination error n_1 or n_2 or n_3 or n_4 = 0 no error Severity: Processor terminates Action required by user: Use LSV to verify or correct the referenced data element.

TABLE 4-IV.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR SVUCP

Processor name	Processor abstract prompt (maximum 256 characters)
SVUCP	The state vector units conversion processor will convert the units of an input state vector to a different set of units as specified by the user.

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
ANGUNI	Parameter specifying the input state units for angles = "DEG"; Degrees* = "RAD"; Radians = "Ø"; User-specified external unit choice
ANGUNO	Parameter specifying the output states units for angles = "DEG"; Degrees* = "RAD"; Radians = "Ø"; User-specified external unit choice
DSTUNI	Parameter specifying the input units for distance = "FT"; Feet* = "M"; Meters = "NM"; Nautical miles = "ER"; Earth radii = "KM"; Kilometers = "Ø"; User-specified external unit choice
DSTUNO	Parameter specifying the output units for distance = "FT"; Feet* = "M"; Meters = "NM"; Nautical miles = "ER": Earth radii = "KM"; Kilometers = "Ø"; User-specified external units choice

*Default external unit choice

TABLE 4-IV.- Continued

PROCESSOR SVUCP

Processor name	Processor abstract prompt (maximum 256 characters)
Parameter keyword name	**Parameter definition prompt (maximum 256 characters)**
TIMUNI	Parameter specifying the input units for time = "SEC"; Seconds* = "MIN"; Minutes = "HR"; Hours = "DAYS"; Days = "%"; User-specified external unit choice
TIMUNO	Parameter specifying the output units for time = "SEC"; Seconds* = "MIN"; Minutes = "HR"; Hours = "DAYS"; Days = "%"; User-specified external unit choice
VELUNI	Parameter specifying the input units for velocity = "FT/S"; Feet/second* = "M/S"; Meters/second = "%"; User-specified external unit choice
VELUNO	Parameter specifying the output units for velocity = "FT/S"; Feet/second* = "M/S"; Meters/second = "%"; User-specified external unit choice

*Default external unit choice

TABLE 4-IV.- Continued

PROCESSOR SVUCP

Processor name	Processor abstract prompt (maximum 256 characters)

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
MASUNI	Parameter specifying the input units for mass = "LBM"; Pounds mass* = "KG"; Kilograms = "SLG"; Slugs = "B"; User-specified external unit choice
MASUNO	Parameter specifying the output units for mass = "LBM"; Pounds mass* = "KG"; Kilograms = "SLG"; Slugs = "B"; User-specified external unit choice
LENUNI	Parameter specifying the input units for length = "FTL"; Feet* = "ML"; Meters = "INL"; Inches = "B"; User-specified external unit choice
LENUNO	Parameter specifying the output units for length = "FTL"; Feet* = "ML"; Meters = "INL"; Inches = "B"; User-specified external unit choice
SVIN	Parameter specifying the output state vector

*Default external unit choice

TABLE 4-IV.- Concluded

PROCESSOR SVUCP

Processor name	Processor abstract prompt (maximum 256 characters)

Parameter keyword name	Parameter definition prompt (maximum 256 characters)
SVOUT	Parameter specifying the output state vector

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM SVUCP

5.1.1 Purpose

The State Vector Units Conversion Program (SVUCP) has the capability to accept a position/velocity state vector expressed in one set of units and convert the state to a different set of units as specified by the user.

5.1.2 Functional Description

The SVUCP takes an input state vector from the users' active work area (AWA) and determines (via an input parameter in the position/velocity state vector word), the input element set. The input/output element set may be any of the 15 element sets defined in table 1.2-VI of the SDD FDS-1 Standards document (vol. VI). If an invalid element set is input, an appropriate message is displayed and the program is exited. Each input/output unit label is examined to determine if the numeric should be obtained from !SESCN or !!GLCN. If the input label is blank, then the numeric will be obtained from !SESCN. If the label is nonblank, then XRCPR is called to check the validity of the label and obtain the numeric from !!GLCN. If an invalid label is found, an appropriate message is displayed. After all the input/output labels have been examined, a test is made to see if any invalid labels were encountered; if so, the processor is terminated. The conversion factors are then computed by dividing the input numeric by the output numeric. Based on the element set that the state is expressed in, the input state is multiplied by the appropriate conversion factor. The mass and area are also converted to the proper units. XPPUT is called and the output state is stored in the AWA. XPXIT is called and the processor is exited.

5.1.3 Assumptions and Limitations

The only acceptable units are given in table 1.2-I of the FDS Standards document. If a user selects the "USER SPECIFIED" option for units, he must first execute the PHYDM processor or have an acceptable !SESCN data element in his AWA. The KS element set for state vectors input/output is not supported by this processor.

5.1.4 Method

The outputs units are computed using the equation

SVOUT = SVIN*(SCIN/SCUT)

where

SVOUT = Output state vector

SVIN = Input state vector

SCIN = Conversion factor from input units to internal units

SCUT = Conversion factor from output units to internal units

5.1.5 Routine Input/Output Variables

The SVUCP input/output variables are presented in table 5.1-I.

5.1.6 Functional Logic Flow

The functional logic flow for SVUCP is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

Two diagnostic messages are provided by the processor. They are

a. "$\overset{++++}{oooo}$ IS NOT A VALID INPUT PARAMETER $\overset{++++++}{oooooo}$." The user should use the Interface Table Editor to correct the input. Table 4-III presents a detailed description of these messages.

b. "THE KS STATE VECTOR ELEMENT SET IS NOT SUPPORTED." The user should use the PSV vector to determine if it is a valid state vector.

5.1.8 Special Comments

This processor uses the DEKOD, XPXIT, and XPPUT utility routines and the XRCPR function.

5.1.9 References

None.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine SVUCP

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ANGUNI	--	6CH	I	--	IT	ANGUNI	Parameter specifying the input state units for angles
ANGUNO	--	6CH	O	--	IT	ANGUNO	Parameter specifying the output state units for angles
DSTUNI	--	6CH	I	--	IT	DSTUNI	Parameter specifying the input state units for distance
DSTUNO	--	6CH	O	--	IT	DSTUNO	Parameter specifying the output state units for distance
TIMUNI	--	6CH	I	--	IT	TIMUNI	Parameter specifying the input state units for time
TIMUNO	--	6CH	O	--	IT	TIMUNO	Parameter specifying the output state units for time
MASUNI	--	6CH	I	--	IT	MASUNI	Parameter specifying the input state units for mass
MASUNO	--	6CH	O	--	IT	MASUNO	Parameter specifying the output state units for mass
LENUNI	--	6CH	I	--	IT	LENUNI	Parameter specifying the input state units for length
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-I.- Concluded

Routine SVUCP

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LENUNO	—	6CH	O	--	IT	LENUNO	Parameter specifying the output state units for length
VELUNI	—	6CH	I	--	IT	VELUNI	Parameter specifying the input state units for velocity
VELUNO	—	6CH	O	--	IT	VELUNO	Parameter specifying the output state units for velocity
AREAI	—	6CH	I	--	IT	AREAI	Parameter specifying the input state units for area
AREAO	—	6CH	O	--	IT	AREAO	Parameter specifying the output state units for area
SVIN	$\bar{R}$	Real	I	--	IT	SVIN	Parameter specifying the input state vector
SVOUT	$\bar{R}$	Real	O	--	IT	SVOUT	Parameter specifying the output state vector

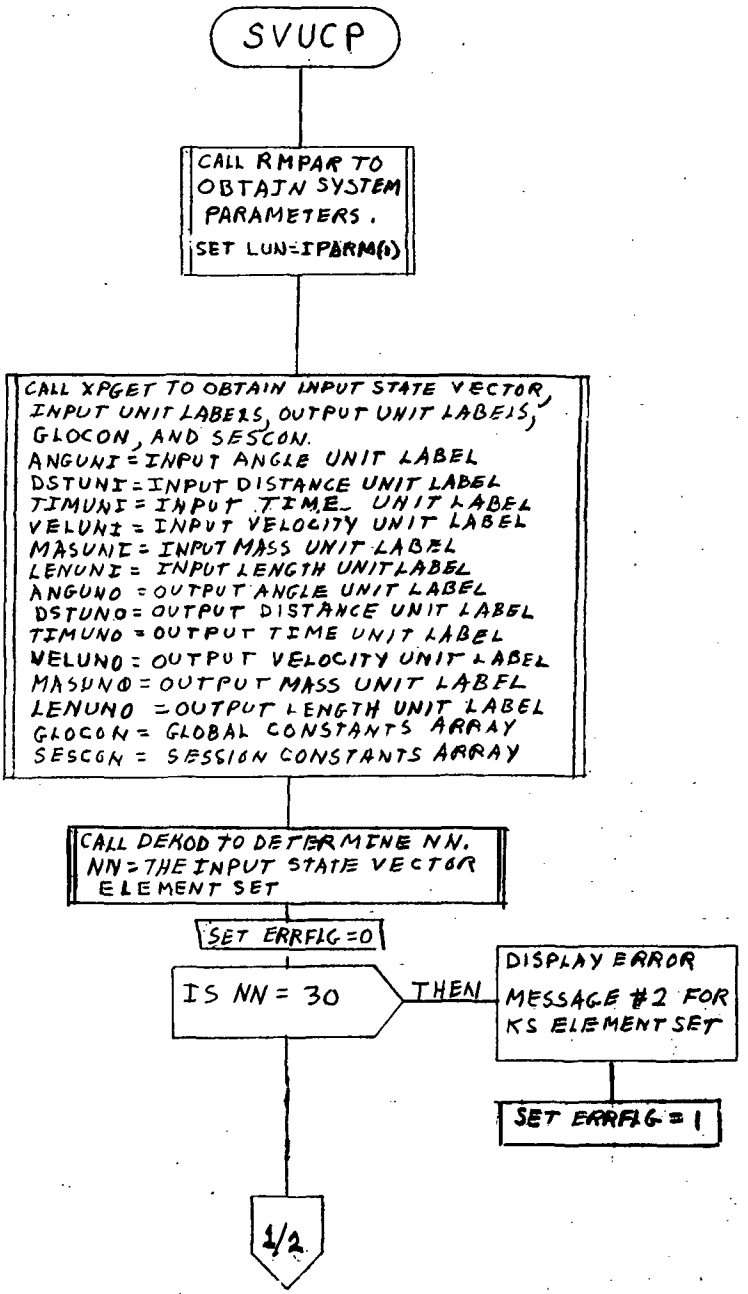
NOTES:

TYPE

Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 3

Figure 5.1-1.- SVUCP functional logic flow.

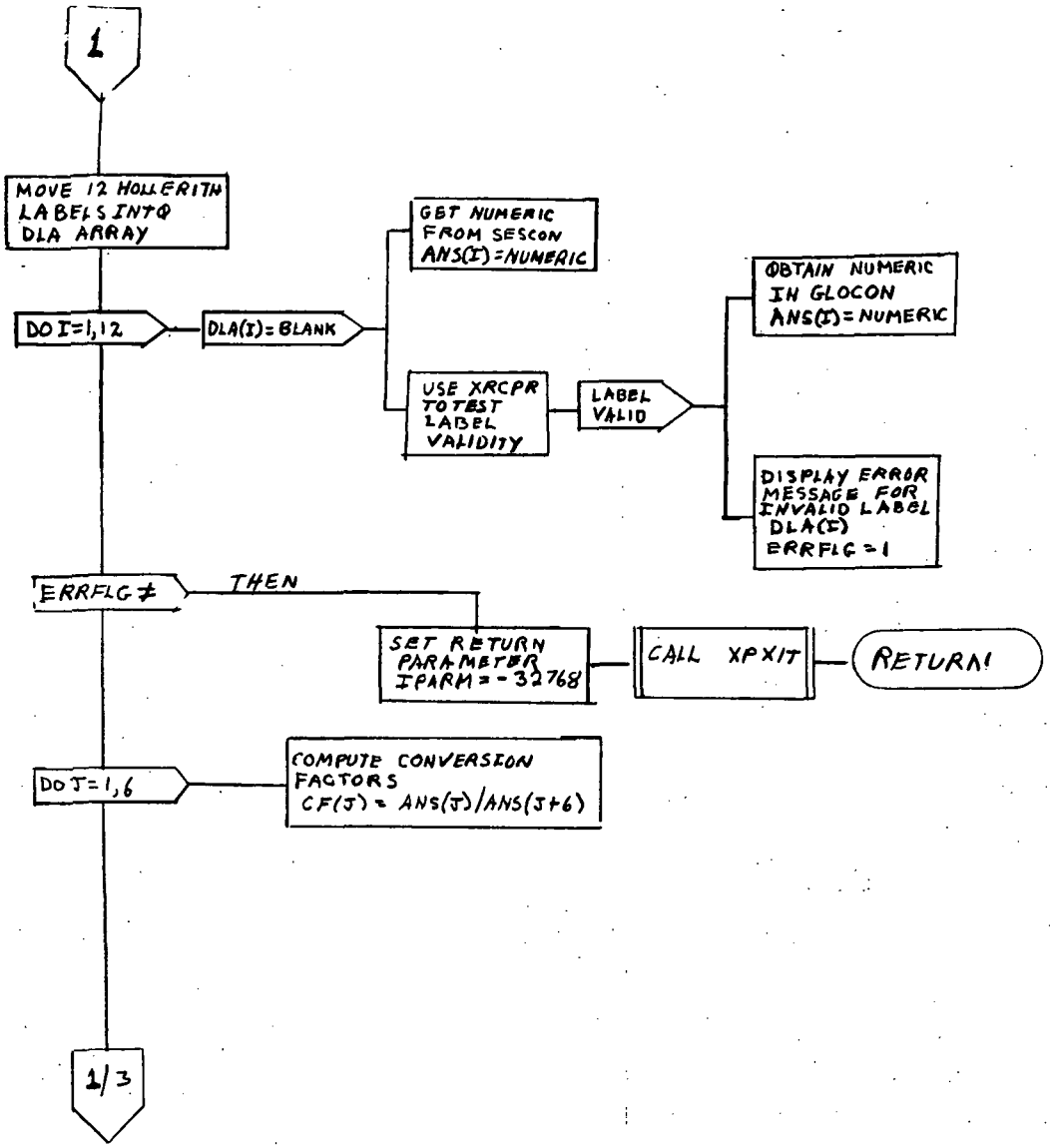


Figure 5.1-1. Continued.

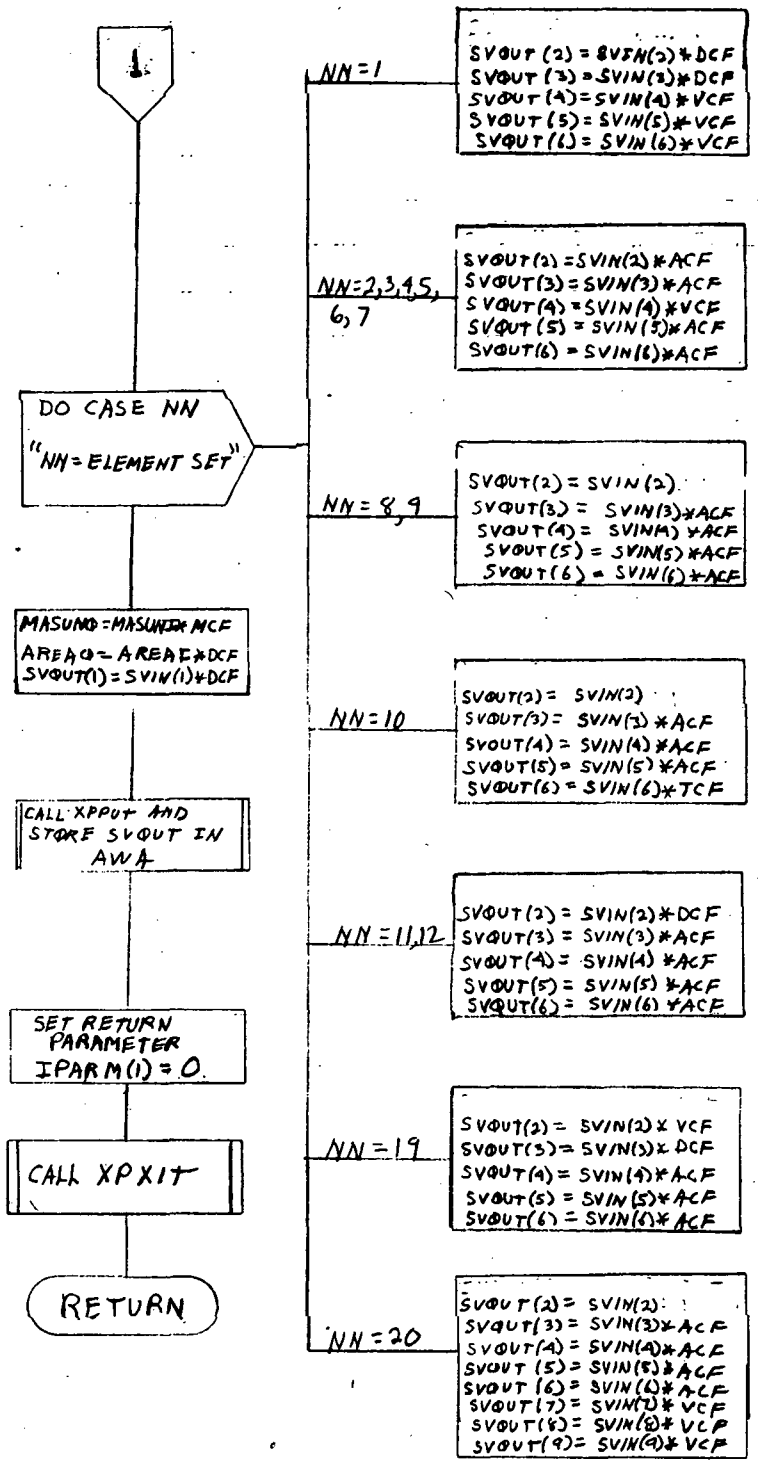


Figure 5.1-1.- Concluded.

STATE VECTOR COORDINATE TRANSFORMATION PROCESSOR (TFSV)

To be supplied

ACTIVITY TIMELINE PROCESSOR (TMLNU)

1.0 PURPOSE

The activity timeline (TMLNU) processor is used to develop, verify, display (in both tabular and bar chart form), and store (in a DRDE file) a mission activities time line.

2.0 FUNCTIONAL DESCRIPTION

TMLNU consists of three separate and distinct phases of operation during its execution. These phases are as follows:

a. Phase 1. Time-line development and storage. The time-line development information is in the form of an activity number, a start time, and a stop time. Some activities may be of a cyclic nature and, in these cases, the period of the cycle and ontime of the activity during the cycle are also required.

   The processor determines which set of logic is necessary to process the information through the response given to a solicited prompt by the processor. The processor will then prompt for the necessary information to be entered. When the information has been entered, the processor will store it in internal arrays for use by the other two phases.

b. Phase 2. Time-line verification for conflicts. There are some activities that must be scheduled at the same time, and there are several that should not. Other conflict situations exist, and all are stored on a disk file. The processor, during the verification phase, will issue a prompt for the names of the required disk files. After obtaining this information, the processor will read these files and determine if any conflicts exist in the time line that has been developed. Messages will be displayed showing the conflicts that were found.

c. Phase 3. Time-line display. The processor, after obtaining the information from a solicited prompt, will display either a tabular listing or a bar chart plot of the time line being developed. These displays may be in either ground elapsed time (GET) or Greenwich mean time (GMT).

   The order of processor execution may be varied since it uses the set of logic determined by the response to solicited prompts. Termination of the processor is controlled by the user. The processor, when termination is requested, will store the time line in a DRDE file for use by other processors.

3.0 ASSUMPTIONS AND LIMITATIONS

a. It is assumed that the user of the time-line processor is familiar with the activities that occur on a Shuttle mission, and is also familiar with their requirements.

b. The user should be cautioned that the time-line processor is still under development, so there may be slight differences during later executions as modifications occur.

c. The number of time-line entries is limited to 350 because of the size of the internal arrays.

4.0 PROCESSOR INPUT/OUTPUT

The time-line processor communication is through the interface table and the solicited prompts. The primary communication during the execution is through the solicited prompts issued by the processor.

a. Processor interface table - The user, through the variable OLDFIL, enters the name of an existing DRDE TIMELINE file. A four-character name must be entered even though there is no file. It is through the variable USEOLD that the user specifies whether or not the file exists. The user specifies the name of the new TIMELINE file by entering a four-character name in the variable NEWF. Three variables from the !SESCN array, as well as a set of processor constants from the PROCON array, are input through the processor interface table. Complete definitions of processor interface table parameters are provided in table 4-I.

b. Interface table data array definitions - The definition of the interface table data array appearing in the TMLNU processor is provided in table 4-II.

c. Interface table data file definitions - The format of the file produced by the TMLNU processor is provided in table 4-III.

d. Processor solicited (prompted) inputs - The processor solicited prompts, their meaning, and the valid responses, are provided in table 4-IV. All literal responses given by the user do not have to be enclosed in quote marks for this processor.

   When the processor requests an activity number, it is expecting an integer value between 1 and 9998. The time entries are to be in the form of

hrs, min, sec
24:24:24

To indicate the absence of an hour, minute, or second value, the user will enter only the colon for that position. For example, to indicate no hours, the entry is

: : 24:24

The user may indicate times prior to lift-off by preceding the time with a minus sign. The user also has the option of entering times in GET or GMT reference bases.

e. Processor displays and display parameter definition table - The format and contents of each display produced by the processor are provided in table 4-V.

The user interface of the 'TMLNU' processor is hierarchial in structure. The uppermost level is represented by a single prompt requesting the option or function to be performed. The second hierarchial level contains all of the functional capability of the processor. Any prompt lower than the second level is dependent on the specific function being performed. Descriptions of the prompts and the expected responses are given below. The responses will be given inside single quotes and will use the following symbols: 'AAA' for alphanumeric characters, 'NNN' for numerical values. All values of time are input and output in either of two forms: HHH:MM:SS or DDD:HH:MM:SS. Either of these forms are applicable to the time formats; however, the particular type desired must be specified by the user. The GET form (HHH:MM:SS) is set at the time of execution and, for convenience, will be the form used in this write-up. Some prompts are established with various optional parameters that are dependent on the function being performed. For these prompts, the optional parameters will be included in brackets ([ ]).

Highest level prompt:

OPTION?: 'AA' or '?'

(expected) - A two-character mnemonic representing the desired option.

(invalid) - Causes processor message no. 23

('?') - Provides table 4-V(a) to user

All remaining prompts are a function of the specific option chosen and will be listed in association with the various options.

**** OPTION 'SB' ***** (Schedule time-line activities from event times)

EVENTS FILE?: 'AAAAAA' or 'EXIT'

(expected) - A six-character name representing the file containing the scheduling criteria to be used in building the time line.

(invalid) - Causes processor message no. 10

('?') - No extended prompt available

('EXIT') - Terminates option and returns to 'OPTION?' prompt
or
(' ') - Terminates option and returns to 'OPTION?' prompt

From the file named in the above prompt, the processor will derive a set of event narratives with which to prompt the user. Associated with each of the event narratives will be a clarification of the time(s) to be supplied. A heading is provided prior to the first event narrative being listed. The processor will remain in this loop until all of the events listed in the file are processed. At that time, the processor will return to the 'OPTION?' prompt. Should a problem be encountered while the processor is trying to read the scheduling criteria data, processor message no. 15 will be sent to the user and the processor will return to the 'OPTION?' prompt.

EVENT TIME(S)

AAAAA (START)?: 'HHH:MM:SS'
or
AAAAA (START/STOP)?: 'HHH:MM:SS HHH:MM:SS'

(expected) - One or two time values (as described above) associated with the event listed. The time(s) represent the event time itself or the start and stop times for events having a duration.

(invalid) - Causes processor message no. 3, 11, or 12

('?') - Provides table 4-V(k) to user

(' ') - Causes processor message no. 13

***** OPTION 'SP' ***** (Schedule previously stored sequence of activities)
FLIGHT PHASE FILE?: 'AAAAAA'

(expected) - A six-character name representing the file containing the previously stored sequence of activities.

(invalid) - Causes processor message no. 10

('?') - No extended prompt available

('EXIT') - Terminates option and returns to 'OPTION?' prompt
or
(' ') - Terminates option and returns to 'OPTION?' prompt

BIAS TIME?: 'HHH:MM:SS'

(expected) - The time reference point for the sequence to start.

(invalid) - Causes processor message no. 3 or 4

('?') - Provides table 4-V(k) to user

(' ') - Defaults the time value to zero.

If a problem is encountered while trying to read the stored time line, processor message no. 21 will be sent to the user and the processor will return to the 'OPTION?' prompt.

***** OPTION 'SA' ***** (Schedule specific flight activity blocks)

The heading listed below is provided prior to the processor prompting the user. The processor will remain in this option until terminated by the commands listed below.

SCHEDULE ACTIVITY: REQUIRES INPUT DATA TO BE -
ACTIVITY START STOP PERIOD ONTIME (EACH SEPARATED BY BLANKS)

The prompt is a colon (:)

: 'NNNN HHH:MM:SS HHH:MM:SS [HHH:MM:SS HHH:MM:SS]'

(expected) - An activity block number, its scheduled start and stop times and, if cyclic, its period and ontime during each cycle.

(invalid) - Causes processor message no. 3, 4, 5, 6, 7, 8, 9, or 14

('?') - Provides table 4-V(i) to user

(' ') - Terminates option and returns to 'OPTION?' prompt. A value of zero for the activity block number will also terminate the option.

***** OPTION 'DI' ***** (Delete specified activity within given interval)

The heading listed below is provided prior to the processor prompting the user. The processor will remain in this option until terminated by the commands listed below.

DELETE ACTIVITY: REQUIRES INPUT DATA TO BE -
ACTIVITY START STOP PERIOD ONTIME (EACH SEPARATED BY BLANKS)

The prompt is a colon (:)

: 'NNNN [HHH:MM:SS HHH:MM:SS [HHH:MM:SS HHH:MM:SS]]'

(expected) - An activity block number and, if desired, the interval in which the activity is to be deleted. If no interval is specified, all occurrences will be deleted.

(invalid) - Causes processor message no. 3, 4, 5, or 6

('?') - Provides table 4-V(i) to user

(' ') - Terminates option and returns to 'OPTION?' prompt. A value of zero for the activity block number will also terminate the option.

If the specified activity does not exist in the interval specified, processor message no. 2 will be sent to the user.

***** OPTION 'DA' ***** (Delete specified activity)

The heading listed below is provided prior to the processor prompting the user. The processor will remain in this option until terminated by the commands listed below.

DELETE ACTIVITY: REQUIRES INPUT DATA TO BE -
ACTIVITY START STOP PERIOD ONTIME (EACH SEPARATED BY BLANKS)

The prompt is a colon (:)

: 'NNNN [HHH:MM:SS HHH:MM:SS [HHH:MM:SS HHH:MM:SS]]'

(expected) - An activity block number, its start and stop times. If the activity is cyclic and more than one entry is to be deleted, the period and ontime can be specified.

(invalid) - Causes processor message no. 3, 4, 5, 6, 7, 8, or 9

('?') - Provides table 4-V(i) to user

(' ') - Terminates option and returns to 'OPTION?' prompt. A value of zero for the activity block number will also terminate the option.

If the specified activity does not exist at the specified times, processor message no. 2 will be sent to the user.

***** OPTION 'MA' ***** (Move specified activity)

The heading listed below is provided prior to the processor prompting the user. The processor will remain in this option until terminated by the commands listed below.

MOVE ACTIVITY: REQUIRES INPUT DATA TO BE -
ACTIVITY START NEW START STOP NEW STOP (EACH SEPARATED BY BLANKS)

The prompt is a colon (:)

: 'NNNN [HHH:MM:SS] [HHH:MM:SS] [HHH:MM:SS] [HHH:MM:SS]'

(expected) - An activity block number, its start and new start time, or its stop and new stop time, or both.

(invalid) - Causes processor message no. 3, 4, 5, or 6

('?') - Provides table 4-V(i) to user

(' ') - Terminates option and returns to 'OPTION?' prompt. A value of zero for the activity block number will also terminate the option.

If the specified activity does not exist at the specified times, processor message no. 2 will be sent to the user.

***** OPTION 'CT' ***** (Change specified time value to a new value)

The heading listed below is provided prior to the processor prompting the user. The processor will remain in this option until terminated by the commands listed below.

CHANGE TIMES: REQUIRES INPUT TO BE -
TIME NEW TIME (SEPARATED BY A BLANK)

The prompt is a colon (:)

: HHH:MM:SS HHH:MM:SS

(expected) - A time value and a new time value

(invalid) - Causes processor message no. 3, 19, or 20

('?') - Provides table 4-V(k) to user

(' ') - Terminates option and returns to 'OPTION?' prompt. A value of zero for the time value will also terminate the option.

If the specified time value does not exist, processor message no. 2 will be sent to the user.

***** OPTION 'GMT' ***** (Set form of time to DDD:HH:MM:SS)

There is no further prompting in this option. The format for time values will become DDD:HH:MM:SS. This form can be used for Greenwich mean time (GMT) or can be used with ground/mission elapsed time (GET/MET). The time reference is set by the 'BASTM' processor.

***** OPTION 'GET' ***** (Set form of time to HHH:MM:SS)

There is no further prompting in this option. The format for time values will become HHH:MM:SS. This form is used for ground/mission elapsed time (GET/MET) only. The time reference is always zero at launch.

***** OPTION 'LN' ***** (List activity numbers and names)

There is no further prompting in this option. This option will cause the processor to provide table 4-V(d). If, however, while trying to read the file names a problem is encountered, processor message no. 18 will be sent to the user and the processor will return to the 'OPTION?' prompt.

***** OPTION 'LO' *****(Locate specified activity)

ACTIVITY NUMBER?: 'NNNN'

(expected) - Activity number

(invalid) - Causes processor message no. 3

('?') - Provides table 4-V(i) to user

(' ') - Terminates option and returns to 'OPTION' prompt. A value of zero for the activity number will also terminate the option.

This function will cause the processor to generate a table in the same format as table 4-V(e), or will cause processor message no. 24 to be sent to the user.

***** OPTION 'PL' ***** (Provides temporal presentation of time line)

DURATION OF PLOT (HRS): 'HHH:MM:SS'

(expected) - The number of hours for which the plot is to generate. Although the processor will read a full unit of time, including minutes and seconds, only the hours portion will be utilized.

(invalid) - Causes processor message no. 3

('?') - Provides table 4-V(k) to user

(' ') - Defaults the duration of the plot to 12 hours.

START TIME?: 'HHH:MM:SS'

(expected) - The earliest time to be included on the plot on the time line. Although the processor will read a full unit of time, including minutes and seconds, the value will be truncated to only the hours value.

(invalid) - Causes processor message no. 3

('?') - Provides table 4-V(k) to user

(' ') - Defaults the start time to zero

This function will cause the processor to generate table 4-V(g), based on the values obtained in the above prompts. Should there not be enough room on the plot for all of the activities, processor message no. 22 will be sent to the user.

***** OPTION 'TB' ***** (Provide tabular presentation of time line)

START STOP?: '[HHH:MM:SS] [HHH:MM:SS]'

(expected) - Either the start time or the stop time (or both) of the desired interval of the time line to be included on the table.

(invalid) - Causes processor message no. 3

('?') - Provides table 4-V(k) to user

(' ') - Blank value for the start time; defaults the value to the earliest time in the time line. Blank value for the stop time defaults the value to the lastest time in the time line.

This function causes the processor to generate table 4-V(e).

***** OPTION 'CD' ***** (Provides conflict detection for activities)

ACTIVITY DATA FILE?: 'AAAAAA'

(expected) - The name of the file containing the activity block data to be used in evaluating the time line.

(invalid) - Causes processor message no. 10

('?') - No extended prompts are available

('EXIT') - Terminates option and returns to "OPTION?' prompt
or
(' ') - Terminates option and returns to 'OPTION?' prompt

This function will use the data in the named file to perform various tests on the activities in the time line. If a problem is encountered while trying to read the activity data, processor message no. 16 will be sent to the user and the processor will return to the 'OPTION?' prompt. If any activity scheduling problems are detected, the processor will provide the appropriate messages from table 4-V(m).

CONFLICTS DATA FILE?: 'AAAAAA'

(expected) - The name of the file containing the scheduling conflicts criteria.

(invalid) - Causes processor message no. 10

('?') - No extend prompts are available

('EXIT') - Terminates option and returns to 'OPTION?' prompt
or
(' ') - Terminates option and returns to 'OPTION?' prompt

This function will use the above named file to perform additional tests on the activities in the time line. If a problem is encountered while trying to read the conflict criteria, processor message no. 17 will be sent to the user and the processor will return to the 'OPTION?' prompt. If any activity scheduling problems are detected, the processor will provide the appropriate messages from table 4-V(m).

***** OPTION 'ST' ***** (Store time line)

If time line has been modified or created during session, the processor will store the time line in the file specified in the interface table. Otherwise, the processor will send processor message no. 1 and return to the 'OPTION?' prompt.

***** OPTION 'EX' ***** (Terminate processor)

The processor will determine if the existing time line has been modified or created. If the time line is in this status and has not been stored, the processor will store the time line in the file specified in the interface table. The processor will then terminate.

f. Processor message table - The messages that may be displayed on the user's terminal during execution are provided in table 4-VI.

g. Interface table extended prompts - The processor extended prompts for each interface table parameter keyword are provided in table 4-VII.

TABLE 4-I.- PROCESSOR INTERFACE TABLE

PROCESSOR TMLNU

Parameter keyword name	Class	Type	Use	Size	Array dimension (I,J)	Value stored in default interface table	Definition
OLDFIL	Disk	Intg	I	--	--		Name of users' old TIMELINE file
NEWFIL	Disk	Intg	O	--	--		Name of users' new TIMELINE file
PROCON	AWA	Intg	I	3	3	16,20,33	Logical unit numbers for SYSTEM and DRDE files, and maximum block size for DRDE files
LYEAR	AWA	Intg	I	1	1	SESCON(3)	Launch year
LDAY	AWA	Intg	I	1	1	SESCON(6)	Launch day
LHOUR	AWA	Real	I	2	1	SESCON(11)	Launch hour (decimal)
USEOLD	AWA	2CH	I	1	1		Flag denoting the existence of a previous DRDE TIMELINE file ("YE" or "NO")

NOTES

CLASS
AWA
Disk

TYPE
Free 2CH 72CH
Intg 6CH Mix
Real 18CH Symb
Dubl 36CH

USE
I = Input
O = Output
I/O = Input/Output

TABLE 4-II.- INTERFACE TABLE DATA ARRAY DEFINITIONS

PROCESSOR TMLNU

Array name	Index location	Default value	Definition
PROCON	1	16	Logical unit number for SYSTEM files
	2	20	Logical unit number for DRDE files
	3	15	Number of blocks to allocate for TIMELINE file

TABLE 4-III.- INTERFACE TABLE DATA FILE DEFINITIONS

PROCESSOR TMLNU

DRDE DATA FILE /XXXXC

Record number	Integer word allocations	Content and definition
1 [a]	1-3	"TMLNU" - Name of the processor that created the file
	4-6	NEWFIL - Interface table variable name through which the file was created
	7-9	"TMLNU" - Name of the processor that last updated the file
	10-12	NEWFIL - Interface table variable name through which the file was changed
2-n	1	IACT(I) - Activity number
	2-3	TIM(I,1) - Activity start time (decimal)
	4-5	TIM(I,2) - Activity stop time (decimal)

TABLE 4-IV.- PROCESSOR SOLICITED (PROMPTED) INPUTS

PROCESSOR TMLNU

Prompt	Meaning	Valid responses
OPTION? p:	Enter the desired option	"?" - Display the list of available user options. "$" - Terminate the processor immediately - do not store the TIMELINE file. "SB" - Display Base Mission Phase Scheduling prompts. "LN" - Display a tabular list of the activity numbers and mnemonic names. "SP" - Display Standard Phase Scheduling prompts. "LO" - Locate a specific activity in the time line. "SA" - Display the prompts to reschedule individual activities. "CT" - Display the prompts to change the start or stop time of an activity. "GE" - Use ground elapsed time (GET) as the time reference. "GM" - Use Greenwich mean time (GMT) as the time reference. "PL" - Display the plot of the time line being developed. "DI" - Display the prompts to delete all occurrences of an activity within a specified time interval. "TB" - Display a table of the time line being developed. "DA" - Display the prompts necessary to delete a specific activity at a specific time. "CD" - Process the time line for possible conflicts. "MA" - Display the prompts necessary to move an activity from one position in the time line to another. "ST" - Store the time line on the DRDE file named in the interface table. "EX" - Normal exit from the processor.

TABLE 4-IV.- Continued

PROCESSOR TMLNU

Prompt	Meaning	Valid responses
STORE? p:	Does the user wish to store this time line?	"YE" - Yes, store the time line. "%" - Terminate the processor immediately - do not store the time line. NOTE - Any other response is considered to be a NO response.
ACTIVITY NUMBER?	Enter the desired activity number.	Any positive integer from 1 to 9998. ZERO is used to exit from the option using this prompt. A negative integer will cause the processor to reprompt.
p:	Enter the data that are described by the heading provided.	The input requested will be described in the heading provided by the processor prior to this prompt. A "?" entry will display the information necessary to complete the request. A "%" entry will terminate the processor immediately.
EVENTS FILE? p:	Enter the name of the file that is to be used.	Any six-character file name. The file must be on the cartridge reference in PROCON(1).
(START/STOP)	Start and stop times will be required.	Any integer time values (60 or less minutes, and 60 or less seconds) when the p: prompt is received.
(START)	Only a start time will be required.	Any integer time values (60 or less minutes, and 60 or less seconds) when the p: prompt is received.
DURATION OF PLOT (HRS)	Enter the time period desired.	Any positive integer within the range of the time line
START TIME	Enter the desired starting time for the plot.	Any time value less than the end time of the time line when the p: prompt is received.

TABLE 4-IV.- Concluded

PROCESSOR TMLNU

Prompt	Meaning	Valid responses
FLIGHT PHASE FILE? p:	Enter the name of the file desired.	Any six-character file name. The file must be on LU 16. A "%" entry will terminate the processor immediately. A "ØØ" or "EX" entry will cause a normal exit from the processor.
BIAS TIME?	Enter a time value.	The amount that the flight phase file times are to be changed for the present time line when the p: prompt is received.
ACTIVITY DATA FILE? p:	Enter the name of the file desired.	The six-character name of the activity data file. The file must be on LU 16. A "%" entry will terminate the processor immediately. A "ØØ" or "EX" entry will cause a normal exit from the processor.
DUPLICATE FILE NAME - ENTER A 4-CHAR NAME	Enter the name requested.	Any four-character name. A "%" entry will terminate the processor immediately. A "ØØ" or "EX" entry will cause a normal exit from the processor.
CONFLICT DATA FILE? p:	Enter the name of the file desired.	The six-character file name of the conflicts detection file. This file must be on LU 16. A "%" entry will terminate the processor immediately. A "ØØ" or "EX" entry will cause a normal exit from the processor.

TABLE 4-V.- PROCESSOR DISPLAY AND DISPLAY PARAMETER DEFINITIONS

(a) Options and input codes

PROCESSOR TMLNU

```
OPTIONS AND INPUT CODES

- SCHEDULE -                    - MISCELLANEOUS -

BASE MISSION PHASE: SB      LIST ACTIVITY NAMES: LN
  STANDARD PHASE: SP          LOCATE ACTIVITY: LO
      ACTIVITY: SA              CHANGE TIMES: CT
                                GET REFERENCE: GE
- DELETE -                      GMT REFERENCE: GM
                                TIMELINE PLOT: PL
WITHIN INTERVAL: DI             TIMELINE TABLE: TB
      ACTIVITY: DA          CONFLICT DETECTION: CD

- MOVE -                     - EXIT/STORE -

MOVE ACTIVITY: MA                       STORE: ST
                                  EXIT/STORE: EX
```

TABLE 4-V.- Continued

(b) Display parameter definition table for the options and input codes display

PROCESSOR TMLNU

OPTIONS AND INPUT CODES	
Display parameter label	Parameter Definition
	No parameters.

TABLE 4-V.- Continued

(c) Activity block names

PROCESSOR TMLNU

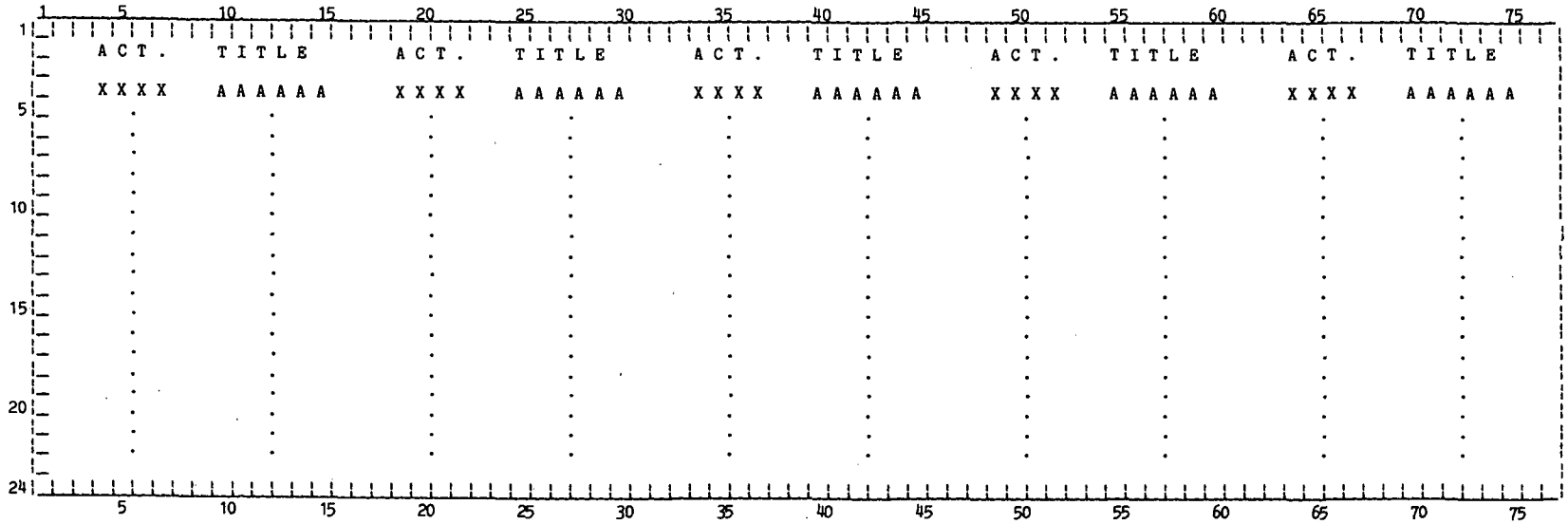


TABLE 4-V.- Continued

(d) Display parameter definition table for the activity block names display

PROCESSOR TMLNU

ACTIVITY BLOCK NAMES	
Display parameter label	Parameter Definition
ACT.	Activity number as read from the file ABIDS.
TITLE	Activity title (six-character mnemonics) as read from the file ABIDS.

TABLE 4-V.- Continued

(e) Time line table

PROCESSOR TMLNU

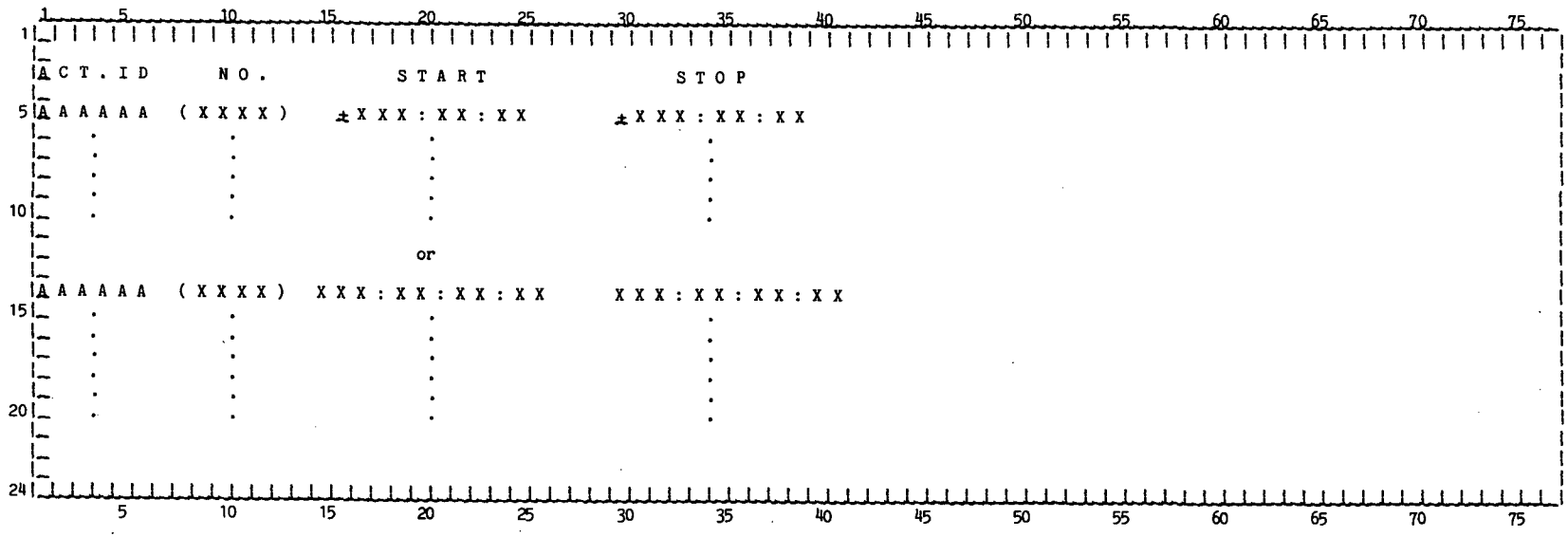


TABLE 4-V.- Continued

(f) Display parameter definition table for the time line table display

PROCESSOR TMLNU

TIMELINE TABLE	
Display parameter label	Parameter Definition
ACT.ID	Activity title (six-character mnemonics) as read from the file "ABIDS" using the activity number from the time line being developed as a reference.
NO.	Activity number from the user's time line.
START	Start time of the activity from the user's time line in GET or GMT (user specified). XXX:XX:XX (GET) XXX:XX:XX:XX (GMT)
STOP	Stop time of the activity from the user's time line in GET or GMT (user specified). XXX:XX:XX (GET) XXX:XX:XX:XXX (GMT)

TABLE 4-V.- Continued

(g) Time line plot

PROCESSOR TMLNU

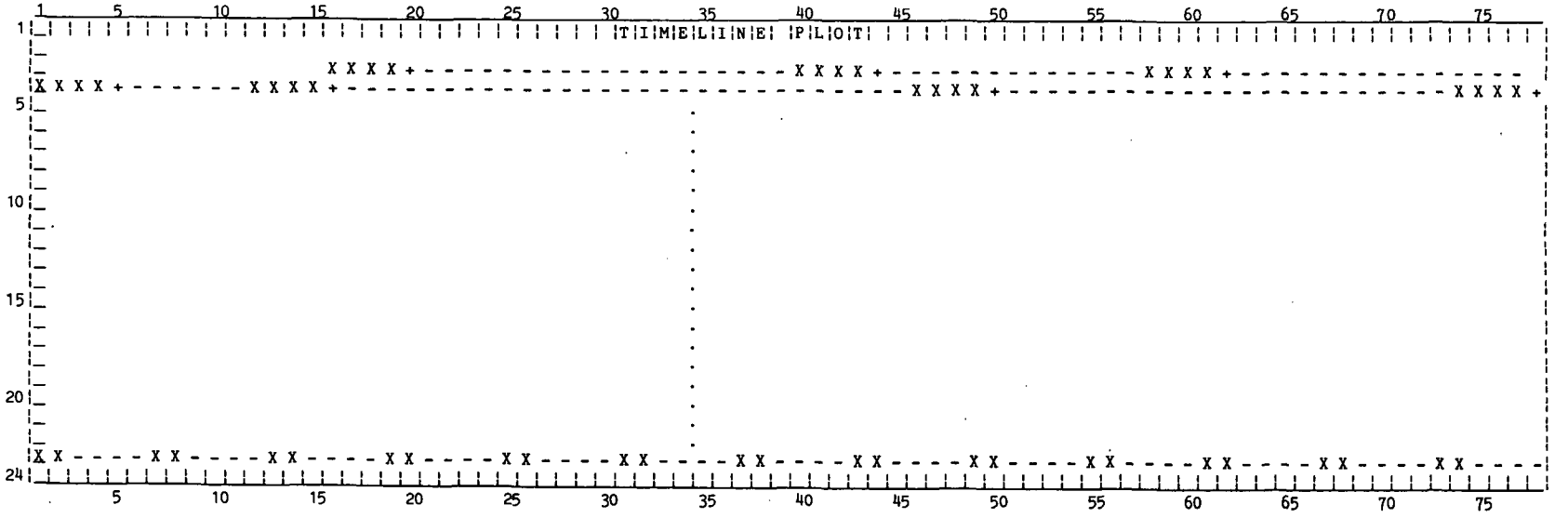


TABLE 4-V.- Continued

(h) Display parameter definition table for the time line plot

PROCESSOR TMLNU

TIMELINE PLOT	
Display parameter label	Parameter Definition
	The XXXX field represents the activity number from the user's time line that starts at the time shown in the XX field at the bottom of the plot.

TABLE 4-V.- Continued

(i) Parameter list and time format

PROCESSOR TMLNU

```
INFORMATION EXPECTED INCLUDES THE FOLLOWING AND IS IN THE FORM OF:
ACTIVITY? TIME1 TIME2 TIME3 TIME4

WHERE TIME3 AND TIME4 ARE OPTIONAL DEPENDING UPON FUNCTION
TIME IS EXPECTED IN THE FORM OF HHH:MM:SS

                                    OR

                                DDD:HH:MM:SS

ZEROS DO NOT HAVE TO BE SUPPLIED (I.E. ::15)
```

TABLE 4-V.- Continued

(j) Display parameter definition table for the parameter list and time format display

PROCESSOR TMLNU

PARAMETER LIST AND TIME FORMAT	
Display parameter label	Parameter Definition
	Informational display in response to a '?' entry by the user when an activity number and times are required for ground elapsed time (GET) or Greenwich mean time (GMT). HHH:MM:SS (GET) DDD:HH:MM:SS (GMT) Time is user specified.

TABLE 4-V.- Continued

(k) Time format

PROCESSOR TMLNU

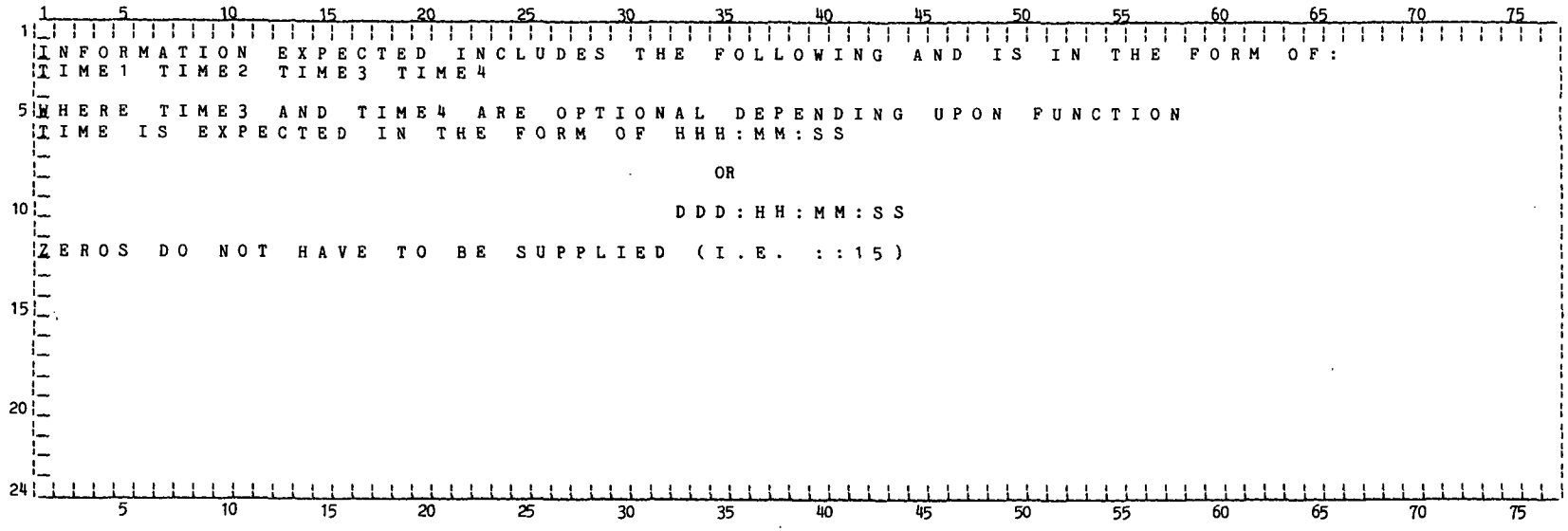


TABLE 4-V.- Continued

(1) Display parameter definition table for the time format display

PROCESSOR TMLNU

TIME FORMAT	
Display parameter label	Parameter Definition
	Informational display in response to a '?' entry by the user when only times are required for ground elapsed time (GET) or Greenwich mean time (GMT). HHH:MM:SS (GET) DDD:HH:MM:SS (GMT) Time is user specified.

TABLE 4-V.- Continued

(m) Possible conflict detection messages

PROCESSOR TMLNU

```
ACTIVITY XXXX AT ±XXX:XX:XX IS NOT IN THE DATA BASE
ACTIVITY XXXX AT ±XXX:XX:XX IS 'ON' FOR LESS THAN ITS FIXED DURATION
ACTIVITY XXXX SCHEDULED 'ON' AT ±XXX:XX:XX WAS ALREADY 'ON'
ACTIVITY XXXX SHOULD COINCIDE WITH XXXX AT ±XXX:XX:XX
ACTIVITY XXXX SHOULD NOT COINCIDE WITH XXXX AT ±XXX:XX:XX
ACTIVITY XXXX AT ±XXX:XX:XX WAS NOT SCHEDULED SOON ENOUGH FOLLOWING XXXX
ACTIVITY XXXX AT ±XXX:XX:XX WAS SCHEDULED TOO SOON FOLLOWING XXXX
COULDN'T FIND XXXX
END OF CONFLICT DETECTION

                                    OR

ACTIVITY XXXX AT XXX:XX:XX:XX IS NOT IN THE DATA BASE
ACTIVITY XXXX AT XXX:XX:XX:XX IS 'ON' FOR LESS THAN ITS FIXED DURATION
ACTIVITY XXXX SCHEDULED 'ON' AT XXX:XX:XX:XX WAS ALREADY 'ON'
ACTIVITY XXXX SHOULD COINCIDE WITH XXXX AT XXX:XX:XX:XX
ACTIVITY XXXX SHOULD NOT COINCIDE WITH XXXX AT XXX:XX:XX:XX
ACTIVITY XXXX AT XXX:XX:XX:XX WAS NOT SCHEDULED SOON ENOUGH FOLLOWING XXXX
ACTIVITY XXXX AT XXX:XX:XX:XX WAS SCHEDULED TOO SOON FOLLOWING XXXX
COULDN'T FIND XXXX
END OF CONFLICT DETECTION
COULDN'T FIND XXXX
```

TABLE 4-V.- Concluded

(n) Display parameter definition table for the possible conflict detection messages display

PROCESSOR TMLNU

POSSIBLE CONFLICT DETECTION MESSAGES	
Display parameter label	Parameter definition
	The XXXX field represents the activity numbers from the user's TIMELINE file while the ±XXX:XX:XX field is the ground elapsed time (GET) or the XXX:XX:XX:XX field is the Greenwich mean time (GMT) from the user's time line. The display represents a list of possible conflict detection messages that may occur. Time is user specified.

TABLE 4-VI.- PROCESSOR MESSAGE TABLE

PROCESSOR TMLNU

MSG no.	Message ID block	Message text block and explanation
1	*TMLNU*	NO CHANGES HAVE BEEN MADE TO TIMELINE, STORE NOT NECESSARY Meaning: The time line has not been changed. Severity: Information. Action required by user: None.
2	*TMLNU*	NOT FOUND Meaning: The activity named was not found in the time line. Severity: Information. Action required by user: None.
3	*TMLNU*	INVALID CHARACTER 'A' Meaning: The character displayed appeared in the input string and is inappropriate. Severity: Warning. Action required by user: Reenter information when reprompted.
4	*TMLNU*	'START TIME' NOT SUPPLIED Meaning: Required start time not entered. Severity: Warning. Action required by user: Enter start time when reprompted.
5	*TMLNU*	'STOP TIME' NOT SUPPLIED Meaning: Required stop time not entered. Severity: Warning. Action required by user: Enter stop time when reprompted.
6	*TMLNU*	'STOP TIME' SMALLER THAN 'START TIME' Meaning: Times are incorrect. Severity: Warning. Action required by user: Reenter start and stop times when reprompted.

TABLE 4-VI.- Continued

PROCESSOR TMLNU

MSG no.	Message ID block	Message text block and explanation
7	*TMLNU*	'PERIOD' NOT SUPPLIED Meaning: Required period not entered. Severity: Warning. Action required by user: Enter the period for a cyclic activity when reprompted.
8	*TMLNU*	'ON TIME' NOT SUPPLIED Meaning: Required on-time not entered. Severity: Warning. Action required by user: Enter ONTIME for a cyclic activity when reprompted.
9	*TMLNU*	'ON TIME' GREATER THAN 'PERIOD' Meaning: Times incorrect. Severity: Warning. Action required by user: Reenter period and on-time when reprompted.
10	*TMLNU*	COULDN'T OPEN FILE Meaning: The processor could not open the file whose name was just entered. Severity: Warning. Action required by user: Verify spelling of the file name. If incorrect, reenter when prompted; otherwise, exit the option.
11	*TMLNU*	NEED ONLY ONE 'TIME' Meaning: Too many time values entered. Severity: Warning. Action required by user: Enter one time value when reprompted.
12	*TMLNU*	NEED TWO 'TIMES', ONLY ONE WAS SUPPLIED Meaning: Incorrect times entered. Severity: Warning. Action required by user: Enter the time values when reprompted.

TABLE 4-VI.- Continued

PROCESSOR TMLNU

MSG no.	Message ID block	Message text block and explanation
13	*TMLNU*	ABOVE EVENT WILL NOT BE INCLUDED Meaning: Activity blocks associated with named event will not be scheduled in time line. Severity: Warning. Action required by user: Enter event manually with "SA" option if necessary.
14	*TMLNU*	ARRAY FULL Meaning: Internal array limits exceeded. Severity: Warning. Action required by user: Stop entering activities.
15	*TMLNU*	SCHEDULING CRITERIA FILE PROBLEM Meaning: A read error has occurred on the file. Severity: Option terminates. Action required by user: None.
16	*TMLNU*	ACTIVITY DATA FILE PROBLEM, XXXX Meaning: An error has occurred with the file. The error type is XXXX. Severity: Warning. Action required by user: Exit option or enter file name when prompted.
17	*TMLNU*	CONFLICT DATA FILE PROBLEM, XXXX Meaning: An error has occurred with the file. The error type is XXXX. Severity: Warning. Action required by user: Exit option or enter file name when prompted.
18	*TMLNU*	ACTIVITY 'ID' FILE PROBLEM, XXXX Meaning: An error has occurred with the file. The error type is XXXX. Severity: Warning. Action required by user: Do not select this option until the error has been corrected.

TABLE 4-VI.- Concluded

PROCESSOR TMLNU

MSG no.	Message ID block	Message text block and explanation
19	*TMLNU*	'TIME' NOT GIVEN! Meaning: Old time not entered for a change time ("CT") option. Severity: Warning. Action required by user: Reenter time values.
20	*TMLNU*	'NEW TIME' NOT GIVEN! Meaning: New time not entered for a change time ("CT") option. Severity: Warning. Action required by user: Reenter time values.
21	*TMLNU*	FILE PROBLEM XXXX IN AAAAAA Meaning: An error, XXXX, has occurred in file AAAAAA. Severity: Terminal. Action required by user: None.
22	*TMLNU*	NO ROOM FOR XXXX AT ±XXX.XX Meaning: There is insufficient room to list this activity in the plot. Severity: Warning. Action required by user: None.
23	*TMLNU*	'AA' NOT DEFINED Meaning: Option code requested does not exist. Severity: Information. Action required by user: Enter desired option code.
24	*TMLNU*	NO OCCURRENCES FOUND Meaning: Requested activity could not be found. Severity: Information. Action required by user: None.

TABLE 4-VII.- INTERFACE TABLE EXTENDED PROMPTS

PROCESSOR TMLNU

Processor name	Processor abstract prompt (maximum 256 characters)
TMLNU	TMLNU is used to build a mission activities time line, or to retrieve an existing TIMELINE file and then modify it by adding, deleting, or moving activities. A tabular list, or a bar chart and the conflicts found, may be displayed.
Parameter keyword name	**Parameter definition prompt (maximum 256 characters)**
OLDFIL	Name of the user's input TIMELINE file
NEWFIL	Name of the file that the user selects to store the time line being developed
PROCON	Processor constants: LU for data files, LU for DRDE files, and blocks in the DRDE file.
LYEAR	Launch year from the SESCON array (SESCON(3))
LDAY	Launch day from the SESCON array (SESCON(6))
LHOUR	Launch hour (decimal) from the SESCON array (SESCON(11))
USEOLD	Flag denoting that the old file name does or does not exist; either "YE" or "NO".

5.0 PROCESSOR ROUTINES

5.1 ROUTINE NAME - MAIN PROGRAM TMLNU

5.1.1 Purpose

The routine TMLNU serves as the main program of the TMLNU processor and provides the executive control for the flow of execution and the loading of processor segments. TMLNU also controls the interactive conversation with the user and passes the responses to control the execution.

5.1.2 Functional Description

The routine TMLNU commences with a call to the RTE executive routine RMPAR to obtain the system parameter array IPARM. The variable LU, used to specify the terminal logical unit number, is equated to the first value in the IPARM array. An informational header that specifies the version of the processor being used is then displayed.

Individual variables are initialized, including the time array (TIM) and activity number array (IACT). The TIM and IACT arrays are used to store the start and stop times with the associated activity number for each activity entered into the time line. IPARM(1) is then converted to an FDS-1 termination flag and set to zero to indicate normal conditions.

The number of days in the year (specified by the user through the interface table variable LYEAR) is determined through the use of the FORTRAN library remainder routine MOD. This enables the processor to detect a leap year.

If the user has specified that a previously developed time line is to be input (through the Hollerith interface table variable USEOLD), the segment TFILU is brought into memory by a call to the RTE executive routine LDSEG. The segment TFILU is used to store or retrieve time lines to or from DRDE files. Otherwise, the processor bypasses this logic.

The processor then enters a looping structure that will continue until the option selected indicates the desire to terminate execution, or until an error from which there is no recovery occurs.

If the file update flag IUPDT is not equal to the saved update flag JUPDT and the array counter NT is greater than or equal to 2, the processor subroutine ORDER is called to sort the time line in ascending order based on start times. If the error flag ISTP is true, an error exit is made and the processor is abnormally terminated. Otherwise, the update flags are equated and the processor displays a prompt to the user requesting the desired option response.

The user's response is then passed to determine the processors action. If the user has entered a percent sign (%), the processor will exit to the percent sign exit point and processing will be terminated. If the response by the user was

a question mark (?), the processor will display a list of the available options and input codes on the CRT, and the user will be reprompted for an option response.

If the response is other than a percent sign (%) or a question mark (?), an array of the input codes IOPT is searched until a match is found. When a match is found to the user's response, the index I of the IOPT array is used as the entry argument for a case structure. Otherwise, if no match is found, a message is displayed to the user stating that the response is not defined in the option list and the user is reprompted for an input option code.

When a match to the input response is made, the case structure (using the array index I to determine the desired case) is entered. The following actions are executed for each value of I.

I=1; user response was "SB", which indicates a base mission phase. A call is made to the RTE executive routine LDSEG to load the segment BASEU into memory. This segment then builds, or adds to a time line, various activities that are keyed to event times. Upon completion of the segment, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=2; user response was "SP", which indicates a standard phase. A call is made to the RTE executive routine to load the segment TFILU into memory. This segment then retrieves a stored time line and places it into the TIM and JACT arrays located in COMMON. Upon completion of the segments, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=3; user response was "SA", which indicates a schedule activity phase. A call is made to the processor subroutine SCHED. This subroutine provides the capability to schedule individual activities into the time line being developed. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=4; user response was "DI", which indicates a delete within interval phase. A call is made to the RTE executive routine LDSEG to load the segment TDELU into memory. This segment provides the capability to delete all occurrences of a given activity within a specified interval of time. Upon completion of the segment, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=5; user response was "DA", which indicates a delete activity phase. A call is made to the RTE executive routine LDSEG to load the segment TDELU into memory. This segment provides the capability to delete activities one at a time. Upon completion of the segment, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=6; user response was "MA", which indicates a move activity phase. A call is made to the RTE executive routine LDSEG to load the segment TMOVU into memory. This segment provides the capability to move an activity from one point in the

time line to another by altering the start time, stop time, or both, for a specified activity. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=7; user response was "LN", which indicates a list activity names phase. A call is made to the RTE executive routine LDSEG to load the segment TABLU into memory. This segment will then produce a listing of the activity numbers and the associated mnemonic for all valid activities. This information comes from a system resident disk file. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=8; user response was "LO", which indicates a locate activity phase. The processor subroutine LOCAT is called and is used to locate and list all occurrences of an activity or activities. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=9; user response was "CT", which indicates a change times phase. A call is made to the RTE executive routine LDSEG to load the segment TCHGU into memory. This segment is used to change all occurrences of a specified time in the time line from one value to another. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=10; user response was "GE", which indicates a ground elapsed time reference phase. The logical flag IGMT is set to false, which allows the displays from the processor and the user time value inputs to be in ground elapsed time. Control is then returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=11; user response was "GM", which indicates a Greenwich mean time reference phase. The logical flag IGMT is set to true, which allows the displays from the processor and the user time value inputs to be in Greenwich mean time. Control is then returned to the top of the DOUNTIL loop and the user is again prompted for an option response.

I=12; user response was "PL", which indicates a time-line plot phase. A call is made to the RTE executive routine LDSEG to load the segment TPLTU into memory. This segment provides a time oriented bar chart of the time-line activities. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=13; user response was "TB", which indicates a time-line table phase. A call is made to the RTE executive routine LDSEG to load the segment TABLU into memory. This segment then displays a table of the time line being developed. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=14; user response was "CD", which indicates a conflict detection phase. A call is made to the RTE executive routine LDSEG to load the segment CNFKU into memory. This segment then determines if there is any conflict between the activities scheduled in the time line being developed. Upon completion, control is returned to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=15; user response was "ST", which indicates a store time line phase. If no changes have been made to the TIMELINE file (indicated by IUPDT equal to zero), a message is displayed to the user stating that no changes have been made. Control then returns to the top of the DOUNTIL loop and the user is again prompted for an option selection. Otherwise, a call to the RTE executive routine LDSEG is made to load the segment TFILU into memory. The time line is then stored. Control returns to the top of the DOUNTIL loop and the user is again prompted for an option selection.

I=16; user response was "EX", which indicates an exit/store phase. If the TIMELINE file has not been altered (indicated by IUPDT equal to zero), control is passed to the normal exit routine and the processor is terminated. Otherwise, the user is prompted to determine if it is desired to store the time line being developed. If the response is YES, control passes to case 15, the file is stored, and an exit to the normal termination routine is made. A percent sign (%) response to the prompt causes control to pass to the percent sign termination routine; a NO response will cause control to pass to the normal termination routine.

For any abnormal condition that may exist, control is passed to the abnormal termination routine where IPARM(1) is set to -32768, and a call is made to the FDS utility routine XPXIT to terminate all input/output.

When a percent sign is entered in response to a prompt, control is passed to a percent sign termination routine where IPARM(1) is set to 8, and a call to the FDS utility routine XPXIT is made to terminate all input/output.

Normal termination routine sets IPARM(1) to zero, and a call to the FDS utility routine XPXJT is made to terminate all input/output.

5.1.3 Assumptions and Limitations

It is assumed that the user will not have more than 350 activities in the time line. This limit is imposed by the size of the internal arrays used to store the data entered until they can be stored to a disk file.

5.1.4 Method

The processor functions during execution in an interactive mode with the user. The responses given by the user to processor prompts determines the logic flow of the processor.

5.1.5 Routine Input/Output Variables

The input/output variables for the TMLNU routine are presented in table 5.1-I.

5.1.6 Functional Logic Flow

A functional logic flow is not provided. The functional level PDL for TMLNU is presented in figure 5.1-1.

5.1.7 Diagnostics and Debug

All user-solicited input is checked for validity. Invalid or undefined responses result in warning messages and reprompting by the processor.

5.1.8 Special Comments

None.

5.1.9 References

None.

TABLE 5.1-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine TMLNU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
HREF	--	Real	I/O	--	IT	--	Reference hour used to bias the "GET" to establish "GMT"
IACT	--	Intg	O	--	C	--	Time line activity number array
IDOY	--	Intg	O	--	C	--	Number of days in the year
IDREF	--	Intg	I/O	--	IT	--	Reference day used to bias the "GET" to establish "GMT"
IGMT	--	Logical	O	--	C	--	Greenwich mean time reference flag
INTBUF	--	Intg	I/O	--	C	--	FDS-1 input buffer
IPARM	--	Intg	I/O	--	C	--	System and FDS-1 parameters
ISTP	--	Logical	I	--	C	--	Flag to test "STOP EXECUTION?"
IUPDT	--	Intg	I	--	C	--	Indicates whether or not the TIMELINE file has been modified
IYEAR	--	Intg	I	--	IT	--	User-specified year of launch
JOPT	--	Intg	O	--	C		ASCII code of user-specified option
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.1-I.- Concluded

Routine TMLNU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU	--	Intg	O	--	C	--	User's logical unit number
NT	--	Intg	I	--	C	--	Indexing indicator of last record in the time-line array
PROCON	--	Intg	I	--	C	--	Logical unit number for SYSTEM and DRDE files, and the maximum block size for DRDE files.
TIM	--	Real	O	--	C	--	Start and stop times of the time line
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

```
1*
1 BEGIN TMLNU
2     CALL RMPAR FOR SYSTEM PARAMETERS
2     SET LU NUMBER TO SYSTEM PARAMETER ONE
2     WRITE PROCESSOR HEADING
2     INITIALIZE INDIVIDUAL PARAMETERS
2*
2     DOFOR I = 1,350,1
3        INITIALIZE ARRAYS
2     ENDDO
2*
2     SET FDS VARIABLES
2     CALL XPGET FOR INTERFACE PARAMETERS
2*
2     DAYS IN YEAR = 365
2     IF(MOD(YEAR,4) = 0)
2     THEN
3        DAYS IN YEAR = 366
2     ENDIF
2*
2     IF(USE OLD FILE = TRUE)
2     THEN
3        CALL XPGET FOR FILE NAME
3        CALL LDSEG TO LOAD INPUT TIMELINE
3        BRANCH TO -LBL30-
2     ELSE
3        BRANCH TO -LBL30-
2     ENDIF
2*
2 -LBL20-
3     WRITE OPTION LIST
2*
2 -LBL30-
2     IF(UPDATE NOT = SAVED UPDATE AND ARRAY COUNTER IS
2        GREATER THAN 2)
2     THEN
3        PERFORM ORDER
2     ENDIF
2*
2 -LBL40-
2     IF(STOP FLAG = TRUE)
2     THEN
3        EXIT TO -EX9000-
2     ELSE
```

Figure 5.1-1.- TMLNU functional level PDL.

```
2       SAVE UPDATE FLAG
3       WRITE OPTION PROMPT
3       READ USERS OPTION RESPONSE
4       IF(OPTION = A % SIGN)
4       THEN
5          EXIT TO -ER8050-
4       ELSE
5          IF(OPTION = A ? MARK)
5          THEN
6             BRANCH TO -LBL20-
5          ELSE
6             DOUNTIL(OPTION FOUND OR LIMIT REACHED)
7                IF(OPTION = OPTION LIST NAME)
7                THEN
8                   SET I = TO OPTION LIST NUMBER
8                   BRANCH TO -LBL60-
7                ELSE
8                   WRITE ERROR MESSAGE
8                   BRANCH TO -LBL30-
7                ENDIF
6             ENDDO
5          ENDIF
4       ENDIF
3    ENDIF
2*
2 -LBL60-
2    DOCASE I (-LBL70-,-LBL80-,-LBL90-,-LBL100-,-LBL110-,
2              -LBL120-,-LBL130-,-LBL140-,-LBL150-,
2              -LBL160-,-LBL170-,-LBL180-,-LBL190-,
2              -LBL200-)
2*
3    -LBL70-
4       CALL LDSEG FOR EVENTS SCHEDULING SEGMENT
4       BRANCH TO -LBL30-
2*
3    -LBL80-
4       CALL LDSEG FOR ADD A TIMELINE SEGMENT
4       BRANCH TO -LBL30-
2*
3    -LBL90-
4       CALL LDSEG FOR SCHEDULING ACTIVITY SEGMENT
4       BRANCH TO -LBL30-
2*
3    -LBL100-
4       CALL LDSEG FOR DELETE SEGMENT
4       BRANCH TO -LBL30-
```

Figure 5.1-1.- Continued.

```
2*
3     -LBL110-
4        CALL LDSEG FOR MOVE SEGMENT
4        BRANCH TO -LBL30-
2*
3     -LBL120-
4        CALL LDSEG FOR TABLE SEGMENT
4        BRANCH TO -LBL40-
2*
3     -LBL130
4        PERFORM LOCAT
4        BRANCH TO -LBL30-
2*
3     -LBL140-
4        CALL LDSEG FOR CHANGE TIME SEGMENT
4        BRANCH TO -LBL30-
2*
3     -LBL150-
4        SET GREENWICH MEAN TIME FLAG TO FALSE
4        IF(OPTION = "GM")
4        THEN
5           SET GREENWICH MEAN TIME FLAG TO TRUE
4        ENDIF
4        BRANCH TO -LBL40-
2*
3     -LBL160-
4        CALL LDSEG FOR PLOT SEGMENT
4        BRANCH TO -LBL40-
2*
3     -LBL170-
4        CALL LDSEG FOR TABLE SEGMENT
4        BRANCH TO -LBL40-
2*
3     -LBL180-
4        CALL LDSEG FOR CONFLICT SEGMENT
4        BRANCH TO -LBL40-
2*
3     -LBL190-
4        IF(UPDATE NOT = 0)
4        THEN
5           BRANCH TO -LBL195-
4        ELSE
5           WRITE NO CHANGES HAVE BEEN MADE MESSAGE
4        ENDIF
4        BRANCH TO -LBL40-
```

Figure 5.1-1.- Continued.

```
2*
3     -LBL195-
4        CALL LDSEG FOR SEGMENT TO STORE TIMELINE
4        IF(OPTION = "ST")
4        THEN
5           BRANCH TO -LBL40-
4        ELSE
3     -LBL200-
4           IF(UPDATE = 0)
4           THEN
5              EXIT TO -EX9000-
4           ELSE
5              WRITE STORE PROMPT
5              READ USER RESPONSE
5              IF(RESPONSE = A % SIGN)
5              THEN
6                 EXIT TO -ER8050-
5              ELSE
6                 IF(RESPONSE = "YE")
6                 THEN
7                    BRANCH TO -LBL190-
6                 ELSE
7                    BRANCH TO -EX9000-
6                 ENDIF
5              ENDIF
4           ENDIF
3        ENDIF
2     ENDCASE
2*
2  -ER8000-
3     SET IPARM ONE TO ABNORMAL TERMINATION VALUE
3     BRANCH TO -EX9010-
2*
2  -ER8050-
3     SET IPARM ONE TO % SIGN VALUE
3     BRANCH TO -EX9010-
2*
2  -EX9000-
3     SET IPARM ONE TO NORMAL TERMINATION VALUE
2*
2  -EX9010-
2     CALL XPXIT
1  END TMLNU
1*
```

Figure 5.1-1.- Concluded.

5.2 ROUTINE NAME - TFILU

5.2.1 Purpose

The segment TFILU provides the capability to retrieve and store time lines to and from disk files. The main routine of TFILU controls the flow of execution by determining which subroutine to execute.

5.2.2 Functional Description

The main routine of segment TFILU performs a test on the common variable JOPT, which is the option specified by the user to determine the logic flow. If JOPT is equal to "ST" or "EX", the subroutine STORE will be called. Otherwise, the subroutine FLNPT will be called. Upon completion of either subroutine, a call is made to the RTE executive routine RETRN to set the return linkage, and a call to TNLNU (the calling routine) is made to return control to the point of the segment call.

5.2.3 Assumptions and Limitations

None.

5.2.4 Method

None.

5.2.5 Routine Input/Output Variables

The input/output variables for the main routine of TFILU is given in table 5.2-I.

5.2.6 Functional Logic Flow

The functional level PDL for TFILU is prsented in figure 5.2-1.

5.2.7 Diagnostics and Debug

None.

5.2.8 Special Comments

None.

5.2.9 References

None.

TABLE 5.2-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine TFILU

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
JOPT	—	2CH	I	--	C	--	ASCII code of the user-specified option.

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

```
1*
1 BEGIN TFILU
2     IF(OPTION = "ST" OR "EX"
2     THEN
3         PERFORM STORE
3         EXIT TO -EX9000-
2     ELSE
3         PERFORM FLNPT
3         EXIT TO -EX9000-
2     ENDIF
2 -EX9000-
2     CALL RETRN
2     CALL TMLNU MAIN PROGRAM
1 END TFILU
1*
```

Figure 5.2-1.- TFILU functional level PDL.

5.3 ROUTINE NAME - FLNPT

5.3.1 Purpose

The file input subroutine (FLNPT) reads into internal arrays a previously stored TIMELINE file that is a DRDE file or a standard phase file from system resources. Provisions are made for adjusting the times in the standard phase file to the time frame of the time line being developed.

5.3.2 Functional Description

Subroutine FLNPT determines if the input file is a previously developed time line or a standard phase file by testing the common variable JOPT for the value "ZZ". If the value "ZZ" is found, the variable FLNAME is equated to the first three positions of the interface table variable OLDFIL, which contains the name and characteristics of the DRDE file. A call is then made to the file manager service routine OPEN to open the file for input. If there is no error return from the file manager, the file is then positioned to the first data record through a call to the file manager service routine POSNT. If there is no error return from the file manager, the processor enters a DOUNTIL loop to read in the file data from the old time line. While in the loop, a call to the file manager service routine READF is made to input a file record. The input variable IBUF is used to hold the activity number in IBUF(1), the start time in IBUF(2), and the stop time in IBUF(4). IBUF(2) and IBUF(4) are equivalenced to the variables STIME and ETIME, respectively. The variables IBUF(1), STIME, and ETIME are equated to the indexed positions in the IACT and TIM arrays. The loop continues until a record containing the predetermined end record values is read, an end of file error occurs, a READF error occurs, or the limits of the internal arrays IACT and TIM are reached.

Upon normal termination of the DOUNTIL loop, the input file is closed through a call to the RTE file manager routine CLOSE. The number of entries in the internal arrays IACT and TIM are set to the common variable NT, and a normal return to the calling routine is made.

Should an error, other than an end of file, be returned by the file manager in the calls to OPEN, POSNT, or READF, an appropriate error message is displayed, the logical flag ISTP is set to true, IPARM(1) is set to the abnormal termination value of -32768, and a return to the calling routine is made.

If the JOPT variable was not equal to the test value "ZZ", the user is prompted for the name of the standard phase file. The user response is checked for the presence of a percent sign, blanks, or the option "EX". If the response was a percent sign, the logical flag ISTP is set to true, IPARM(1) is set to the percent sign exit value of 8, and a return to the calling program is made. The response of blanks or "EX" indicates a desire to terminate the subroutine. In this case, IPARM(1) is set to the normal value of zero and a return to the calling routine is made.

When the response is a file name, a call to the file manager service routine OPEN is made. If there is no error return from the file manager, the file is positioned to the first data record through a call to the file manager service routine RWNDF.

If there is no error return from the file manager, the variable BTIM, which holds the bias time to apply to the standard phase file, is set to zero. The update flag IUPDT is incremented, and a call to the subroutine RDTIM is made. A test on the logical flag ISTP is then made. If ISTP is true, IPARM(1) is set to the abnormal termination value of -32768, and a return to the calling routine is made. Otherwise, the processor enters a DOUNTIL loop to read in the data from the standard phase file. While in the loop, a call to the file manager service routine READF is made to input a file record. The input variable IBUF is used to hold the activity number in IBUF(1), the start time in IBUF(2), and the stop time in IBUF(4). IBUF(2) and IBUF(4) are equivalenced to the variables STIME and ETIME, respectively. The variables IBUF(1), STIME, and ETIME are equated to the indexed positions in the IACT and TIM arrays. The loop continues until a record containing the predetermined end record values is read, and end-of-file error occurs, a READF error occurs, or the limits of the internal arrays IACT and TIM are reached.

Upon normal termination of the DOUNTIL loop, the input file is closed through a call to the RTE file manager routine CLOSE. The number of entries in the internal arrays JACT and TIM is set to the common variable NT, and a normal return to the calling routine is made.

Should an error, other than an end of file, be returned by the file manager in the calls to OPEN, RWNDF, or READF, an appropriate error message is displayed, the logical flag ISTP is set to true, IPARM(1) is set to the abnormal termination value of -32768, and a return to the calling routine is made.

5.3.3 Assumptions and Limitations

None.

5.3.4 Method

None.

5.3.5 Routine Input/Output Variables

The input/output variables for the subroutine FLNPT are presented in table 5.3-I.

5.3.6 Functional Logic Flow

A functional logic flow is not presented. The functional level PDL for FLNPT is presented in figure 5.3-1.

5.3.7 *Diagnostics and Debug*

Subroutine FLNPT displays error messages for the inability to open a file, and for other errors returned by the file manager during execution of file positioning, file rewinding, or file reading.

5.3.8 Special Comments

None.

5.3.9 References

None.

TABLE 5.3-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine FLNPT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IACT	—	Intg	O	—	C	--	Activity number array
IFILE	--	Intg	O	—	C	--	Data control block in file manager calls
IPARM	--	Intg	O	--	C	--	Routine status information
ISTP	--	Logical	I/O	—	C	--	Flag to stop processor execution
IUPDT	--	Intg	O	--	C	--	Indicator of modification to the processor
JOPT	—	Intg	I	--	C		User-specified option of execution control
LU	—	Intg	I	—	C		Logical unit number of user's terminal
IBUF	--	Intg	I	--	F		Input record from disk file
NT	—	Intg	O	--	C	--	Pointer to the last TIM array entry
OLDFIL	--	Intg	I	—	C	--	Hollerith name of the old TIMELINE file
PROCON	--	Intg	I	--	C	--	Logical unit numbers for SYSTEM and DRDE files, and maximum block size for DRDE
NOTES:		TYPE Free Dubl 18CH Mix Intg 2CH 36CH Char Real 6CH 72CH Bin				USE I = Input O = Output I/O = Input/Output	SOURCE IT = Interface Table T = Terminal A = Calling Argument C = Common F = Disk File SAM = System Available Memory

TABLE 5.3-I.- Concluded

Routine FLNPT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
TIM	—	Real	O	—	C	—	Array containing start and stop times for all activities

NOTES:

TYPE

Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

```
1*
1 BEGIN FLNPT
2     SET COUNTERS
2     IF(USER OPTION  = "ZZ"
2     THEN
3        SET OLDFIL NAME = TO NAME
3        CALL OPEN FOR FILE ACCESS
3        IF(NO ERROR)
3        THEN
4           CALL POSNT TO POSITION AT SECOND RECORD
4           IF(NO ERROR)
4           THEN
5*
5              DOUNTIL(LAST RECORD VALUE READ,END OF FILE,OR LIMIT)
6                 CALL READF FOR A RECORD
6                 IF(NO ERROR)
6                 THEN
7                    SET INTERNAL ARRAYS TO RECORD VALUES
6                 ELSE
7                    EXIT TO -ER400-
6                 ENDIF
5              ENDDO
4*
4              CALL CLOSE FILE
4              SET COUNTER FOR NUMBER OF ARRAY ENTRIES
4              EXIT TO -EX9000-
3           ELSE
5              EXIT TO -ER400-
4           ENDIF
3        ELSE
4           EXIT TO -ER500-
3        ENDIF
2     ELSE
3        WRITE PROMPT FOR FLIGHT PHASE FILE NAME
3        READ RESPONSE
3        IF(RESPONSE NOT = A % SIGN)
3        THEN
4           IF(RESPONSE NOT = A BLANK OR "EX")
4           THEN
5              CALL OPEN FOR FILE ACCESS
5              IF(NO ERROR)
5              THEN
6                 CALL RWNDF TO POSITION FILE AT FIRST RECORD
6                 IF(NO ERROR)
6                 THEN
7                    DOUNTIL(NO BIAS TIME INPUT ERRORS OR
7                            STOP FLAG = TRUE)
8                       INCREMENT UPDATE COUNTER
8                       WRITE BIAS TIME PROMPT
8                       PERFORM RDTIM TO READ USER RESPONSE
7                    ENDDO
```

Figure 5.3-1.- FLNPT functional level PDL.

```
6*
7                    DOUNTIL(LAST RECORD VALUE READ,
7                            END OF FILE,OR LIMIT)
8                       CALL READF FOR A RECORD
8                       IF(NO ERROR)
8                       THEN
9                          SET INTERNAL ARRAYS TO RECORD VALUES
8                       ELSE
9                          EXIT TO -ER400-
8                       ENDIF
7                    ENDDO
7*
7                    CALL CLOSE FILE
7                    SET COUNTER FOR NUMBER OF ARRAY ENTRIES
7                    EXIT TO -EX9000-
6                 ELSE
7                    EXIT TO -ER400-
6                 ENDIF
5              ELSE
6                 EXIT TO -ER500-
5              ENDIF
4           ELSE
5              EXIT TO -EX9000-
4           ENDIF
3        ELSE
4           EXIT TO -ER8050-
3        ENDIF
2     ENDIF
2*
2 -ER400-
3     TERMINATE PROCESSOR FOR READ OR POSITION ERROR
3     WRITE ERROR MESSAGE
3     CALL CLOSE FILE
3     BRANCH TO -ER8000-
2*
2 -ER500-
3     TERMINATE PROCESSOR FOR OPEN ERROR
3     WRITE ERROR MESSAGE
3     CALL CLOSE FILE
3     BRANCH TO -ER8000-
2*
2 -ER8000-
3     SET STOP FLAG = TRUE
3     SET IPARM ONE TO ABNORMAL TERMINATION VALUE
3     BRANCH TO -EX9999-
2*
```

Figure 5.3-1.- Continued.

```
2 -ER8050-
3     SET STOP FLAG TO TRUE
3     SET IPARM ONE TO % SIGN ENTRY VALUE
3     BRANCH TO -EX9999-
2*
2 -EX9000-
3     SET IPARM ONE TO NORMAL TERMINATION VALUE
2*
2 -EX9999-
2     RETURN TO TFILU
1 END FLNPT
1*
```

Figure 5.3-1.- Concluded.

5.4 ROUTINE NAME - STORE

5.4.1 Purpose

The store file to disk subroutine (STORE) allows the user to store the developed time line as a DRDE file.

5.4.2 Functional Description

The TIMELINE file is sorted into ascending chronological order, based on the start time of the activity through a call to the subroutine ORDER. A call is then made to the FDS parameter retrieval routine XPATR to retrieve the interface variable NEWFIL, which is the name of the file to be used for output. A test is then made to determine if the file specified in the common interface table variable OLDFIL is the same as the file in the interface table variable NEWFIL.

If the file names are the same, the file is then opened for output through a call to the RTE file manager routine OPEN. If there is no error returned from the file manager, the processor sets the program name into the variable PROGNM and enters a DO loop to fill the names in the DRDE header record. Upon completion of the DO loop, the header record is written as the first record on a type 2 file through a call to the RTE file manager routine WRITF. If there is no error return from the file manager, the processor then enters a DO loop. While in the DO loop, the output variable IBUF is equated to the corresponding indexed positions of the IACT and TIM arrays. The record is output to file through a call to the file manager routine WRITF. The DO loop is terminated when the limit in the common variable NT, which specifies the number of entries in the time line, is reached or a write error occurs.

If no errors in writing the file have occurred, the output variable IBUF is set to values that indicate the last record in the file. These values are 9999 for IBUF(1) and 999999. for IBUF(2) and IBUF(4). The last record is written to the output file through a call to the file manager routine WRITF. If no errors are returned from the file manager, the update counter IUPDT is set to zero, the file is closed by a call to the file manager routine CLOSE, and the subroutine terminates normally and returns to the calling routine.

Should an error be returned by the file manager from the calls to OPEN or WRITF, an appropriate message is displayed, the logical flag ISTP is set to true, IPARM(1) is set to the abnormal termination value of -32768, and a return to the calling routine is made.

If the test of the file names shows that they are different, the processor will then create a file for output using the name specified in the interface table variable NEWFIL. Variables are set describing the file characteristics into the variable ISIZE. A call is then made to the FDS parameter storage routine to insert the file characteristics into the interface table. A call to the file manager routine CREAT is made to create the output file.

If there is no error return from the file manager, the processor sets the program name into the variable PROGNM, and enters a DO loop to fill the names in the DRDE header record. Upon completion of the DO loop, the header record is written as the first record on a type 2 file through a call to the RTE file manager routine WRITF. If there is no error return from the file manager, the processor then enters a DO loop. While in the DO loop, the output variable IBUF is equated to the corresponding indexed positions of the IACT and TIM arrays. The record is output to file through a call to the file manager routine WRITF. The DO loop is terminated when the limit in the common variable NT, which specifies the number of entries in the time line, is reached or a write error occurs.

If no errors in writing the file have occurred, the output variable IBUF is set to values that indicate the last record in the file. These values are 9999 for IBUF(1), and 999999. for IBUF(2) and IBUF(4). The last record is written to the output file through a call to the file manager routine WRITF. If no errors are returned from the file manager, the update counter IUPDT is set to zero, the file is closed by a call to the file manager routine CLOSE, and the subroutine terminates normally and returns to the calling routine.

Should an error be returned by the file manager from the calls to CREAT or WRITF, an appropriate message is displayed, the logical flag ISTP is set to true, IPARM(1) is set to the abnormal termination value of -32768, and a return to the calling routine is made.

5.4.3 Assumptions and Limitations

None.

5.4.4 Method

None.

5.4.5 Routine Input/Output Variables

The input/output variables for the subroutine STORE are presented in table 5.4-I.

5.4.6 Functional Logic Flow

A functional logic flow is not provided. The functional level PDL for STORE is presented in figure 5.4-1.

5.4.7 Diagnostics and Debug

Subroutine STORE displays error messages for the inability to open or create a file and for write errors returned by the file manager during execution.

5.4.8 Special Comments

None.

5.4.9 References

None.

TABLE 5.4-I.- ROUTINE INPUT/OUTPUT VARIABLES

Routine STORE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IFILE	--	Intg	I/O	--	C	--	Data control block needed in file manager calls.
IPARM	--	Intg	O	--	C	--	Routine status information.
ISTP	--	Logical	O	--	C	--	Flag to terminate execution.
IUPDT	--	Intg	O	--	C	--	Counter of updates made to the TIMELINE file.
LU	--	Intg	O	--	C	--	Logical unit number of user's terminal.
NEWFIL	--	6CH	I	--	C	--	Hollerith array containing new file name.
NT	--	Intg	I	--	C	--	Index of the number of time line entries.
OLDFIL	--	6CH	I	--	C	--	Hollerith array containing old file name.
PROCON	--	Intg	I	--	C	--	Processor constants, cartridge reference for SYSTEM and DRDE files, and the number of blocks in the output file.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 5.4-I.- Concluded

Routine STORE

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
TIM	—	Real	I	—	C	—	Array containing start and stop times of time line activities.

NOTES:	TYPE				USE	SOURCE
	Free	Dubl	18CH	Mix	I = Input	IT = Interface Table
	Intg	2CH	36CH	Char	O = Output	T = Terminal
	Real	6CH	72CH	Bin	I/O = Input/Output	A = Calling Argument
						C = Common
						F = Disk File
						SAM = System Available Memory

```
1*
1*
1 BEGIN STORE
2     PERFORM ORDER
2     CALL XPATR FOR NEW FILE NAME
2*
2     IF(OLD FILE FLAG = TRUE)
2     THEN
3        IF(OLD FILE NAME = NEW FILE NAME)
3        THEN
4           SET OLD FILE NAME FOR OPEN CALL
4           CALL OPEN FOR FILE ACCESS
4           IF(NO ERROR)
4           THEN
5              SET PROGRAM NAME FOR HEADER RECORD
5*
6              DOFOR I = 1,3,1
7                 SET HEADER RECORD
6              ENDDO
5*
5              CALL WRITE FOR HEADER OUTPUT
5              IF(NO ERROR)
5              THEN
5*
6                 DOUNTIL(LAST RECORD WRITTEN)
7                    SET OUTPUT RECORD TO ARRAY VALUES
7                    CALL WRITE FILE FOR RECORD OUTPUT
7                    IF(NO ERROR)
7                    THEN
8                       CONTINUE DOUNTIL LOOP
7                    ELSE
8                       EXIT TO -ER400-
7                    ENDIF
6                 ENDDO
5*
5                 SET OUTPUT RECORD TO LAST RECORD VALUES
5                 CALL WRITE FOR RECORD OUTPUT
5                 IF(NO ERROR)
5                 THEN
6                    SET UPDATE COUNTER
6                    CALL CLOSE FILE
6                    BRANCH TO -EX9000-
5                 ELSE
6                    EXIT TO -ER400-
5                 ENDIF
4              ELSE
4                 EXIT TO -ER400-
3              ENDIF
4           ELSE
5              EXIT TO -ER500-
4           ENDIF
3        ENDIF
```

Figure 5.4-1.- STORE functional level PDL.

```
2     ELSE
3        SET FDS PARAMETERS FOR FILE CREATION
3        CALL XPPUT TO SET FILE CHARACTERISTICS
3        SET NEW FILE NAME FOR CREATE CALL
3        CALL CREAT TO CREATE DRDE FILE
3        IF(NO ERROR)
3        THEN
4           SET PROGRAM NAME FOR HEADER RECORD
4*
5           DOFOR I = 1,3,1
6              SET HEADER RECORD
5           ENDDO
4*
4           CALL WRITE FOR HEADER OUTPUT
4           IF(NO ERROR)
4           THEN
4*
5              DOUNTIL(LAST RECORD WRITTEN)
6                 SET OUTPUT RECORD TO ARRAY VALUES
6                 CALL WRITE FILE FOR RECORD OUTPUT
6                 IF(NO ERROR)
6                 THEN
7                    CONTINUE DOUNTIL LOOP
6                 ELSE
7                    EXIT TO -ER400-
6                 ENDIF
5              ENDDO
4*
4              SET OUTPUT RECORD TO LAST RECORD VALUES
4              CALL WRITE FOR RECORD OUTPUT
4              IF(NO ERROR)
4              THEN
5                 SET UPDATE COUNTER
5                 CALL CLOSE FILE
5                 EXIT TO -EX9000-
4              ELSE
5                 EXIT TO -ER400-
4              ENDIF
4           ELSE
5              EXIT TO -ER400-
4           ENDIF
3        ELSE
4           EXIT TO -ER400-
3        ENDIF
2     ENDIF
2*
2  -ER400-
3     WRITE ERROR MESSAGE
3     CALL CLOSE FILE
3     EXIT TO -ER8000-
```

Figure 5.4-1.- Continued.

```
2*
2 -ER500-
3    WRITE ERROR MESSAGE
3    CALL CLOSE FILE
3    EXIT TO -ER8000-
2*
2 -ER8000-
3    SET STOP FLAG = TRUE
3    SET IPARM ONE TO ABNORMAL TERMINATION VALUE
3    EXIT TO -EX9999-
2*
2 -EX9000-
3    SET IPARM ONE TO NORMAL TERMINATION VALUE
2*
2 -EX9999-
2    RETURN TO TFILU
1 END STORE
1*
```

Figure 5.4-1.- Concluded.

☆ U.S. GOVERNMENT PRINTING OFFICE: 1979—671-099/2487

DISTRIBUTION FOR JSC IN 77-FM-18, REVISION 1, VOLUME III

JM6/Technical Library (2)
JM61/Center Data Management (3)
FD7/J. Fisher
FM/Chief
FM/E. Davis
FM2/Chief
    Section Heads (4)
    D. Alexander
FM4/Chief
    C. Graves
FM6/Chief
    A. Nolting
    D. Braley
    R. Davis (2)
    E. Fridge
    J. Martin
    G. Martinez
    R. Merriam
    W. Pruett
    W. Reini
    R. Reynolds
    R. Rogan
    G. Roush
    G. Weisskopf
FM8/Chief
FM14/Report Control Files (25)
     S. Cole
     B. Woodland
FM17/Chief
CSC/M30/O. Dial
        R. Herder
IBM/MC56/R. Turner (2)
MDTSCO/B. Brown (10)

Deletions or additions to this distribution must be coordinated through Gloria Martinez/FM6; 483-4491.