28415-H011-R0-00

NASA CR-
160440

DETAILED DESCRIPTION OF THE HP-9825A HFRMP TRAJECTORY PROCESSOR (#TRAJ)

20 NOVEMBER 1979

PREPARED FOR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

LYNDON B. JOHNSON SPACE CENTER

CONTRACT NAS9-14723

SYSTEMS ENGINEERING AND ANALYSIS DEPARTMENT

TRW

DEFENSE AND SPACE SYSTEMS GROUP

HOUSTON, TEXAS

28415-H011-R0-00

DETAILED DESCRIPTION OF THE HP-9825A HFRMP TRAJECTORY PROCESSOR (#TRAJ)

20 NOVEMBER 1979

PREPARED FOR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

LYNDON B. JOHNSON SPACE CENTER

CONTRACT NAS9-14723

SYSTEMS ENGINEERING AND ANALYSIS DEPARTMENT

TRW

DEFENSE AND SPACE SYSTEMS GROUP

HOUSTON, TEXAS

DETAILED DESCRIPTION OF THE HP-9825A HFRMP TRAJECTORY PROCESSOR (#TRAJ)

20 NOVEMBER 1979

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

LYNDON B. JOHNSON SPACE CENTER

Contract NAS9-14723

Prepared S. M. Kindall

S. M. Kindall
Navigation Analysis Section

Approved by T. P. Kinney

T. P. Kinney
Navigation Analysis Section

S. W. Wilson

S. W. Wilson, Manager
Shuttle/IUS Proximity
Operations Software Project

D. K. Phillips

D. K. Phillips, Manager
Systems Engineering and
Analysis Department

Systems Engineering and Analysis Department

TRW
DEFENSE AND SPACE SYSTEMS GROUP

Houston, Texas

ABSTRACT

This document contains a detailed description of the computer code for the trajectory processor (#TRAJ) of the HP-9825A High Fidelity Relative Motion Program (HFRMP). The #TRAJ processor is a 12-degrees-of-freedom trajectory integrator (6 degrees of freedom for each of two vehicles) which can be used to generate digital and graphical data describing the relative motion of the Space Shuttle Orbiter and a free-flying cylindrical payload. The description of the processor includes a listing of the code, coding standards and conventions, detailed flow charts, and discussions of the computational logic.

CONTENTS

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

CONTENTS (Cont.)

ILLUSTRATIONS

ILLUSTRATIONS (Cont.)

TABLES

NOMENCLATURE

Abbreviations

BL	buttock line (Y coordinate in structural coordinate system)
CG	center of gravity
CPLV	curvilinear payload-centered local vertical coordinates
DAP	digital autopilot
ECI	earth-centered inertial (coordinate system)
HFRMP	High-Fidelity Relative Motion Computer Program
HPL	Hewlett-Packard Language (for the HP-9825A computer)
IMLD	"invariant" mean of launch date orbit elements
MLD	mean of launch date ECI coordinate system
M50	mean of 1950.0 ECI coordinate system
OM50	osculating mean of 1950.0 orbit elements
PBY	payload body-fixed coordinate system
PCON	propellant consumption
PL	payload
PLV	payload-centered local-vertical coordinate system
RCS	reaction control system
RIMP	rotational impulse (integral of torque with respect to time)
RSBY	rectangular Shuttle body-fixed coordinates
RSLV	rectangular Shuttle-centered local vertical coordinates
SBY	Shuttle body-fixed coordinate system
SLV	Shuttle-centered local vertical coordinate system
SRM	solid rocket motor
SS	Space Shuttle (Orbiter)
STA	station (X coordinate in structural coordinate system)
WL	water line (Z coordinate in structural coordinate system)

Symbols

$\bar{A}$	linear acceleration of Shuttle CG
$\bar{a}$	linear acceleration of payload CG
$[I],[I]_B$	Shuttle inertia tensor, referenced to SBY axes
$[I]',[I]_p$	Shuttle inertia tensor, referenced to principal axes of inertia
$[i],[i]_b$	payload inertia tensor, referenced to PBY axes
$[i]',[i]_p$	payload inertia tensor, referenced to principal axes of inertia
$\hat{i},\hat{j},\hat{k}$	unit vector aligned with X,Y,Z coordinate axes (i.e., vector base)
$\bar{L},\bar{\mathcal{L}}$	torque vectors
$\bar{P}_F^{JK}$	position vector drawn from point J to point K, resolved onto axes of system F; i.e., $\bar{P}_F^{JK} = \hat{i}\, P_{F1}^{JK} + \hat{j}\, P_{F2}^{JK} + \hat{k}\, P_{F3}^{JK}$
$\dot{\bar{P}}_F^{JK}$	derivative of $\bar{P}_F^{JK}$ with respect to time in coordinate system F; i.e., $D\bar{P}_F^{JK}/dt = \dot{\bar{P}}_F^{JK} = \hat{i}\, \dot{P}_{F1}^{JK} + \hat{j}\, \dot{P}_{F2}^{JK} + \hat{k}\, \dot{P}_{F3}^{JK}$
$\bar{q}_{FG}$	orientation versor (unit quaternion) defining orientation of coordinate system G with respect to system F
$\tilde{\bar{q}}_{FG}$	conjugate of $\bar{q}_{FG}$; $\tilde{\bar{q}}_{FG} = \bar{q}_{GF}$
$\bar{R}$	geocentric position vector of Shuttle CG
R	$\lvert\bar{R}\rvert$
$\dot{R}$	dR/dt
$\ddot{R}$	d^2R/dt^2
$\bar{r}$	geocentric position vector of payload CG
r	$\lvert\bar{r}\rvert$
$\dot{r}$	dr/dt
$\ddot{r}$	d^2r/dt^2
t	time
$\bar{U},\bar{u}$	unit vectors
$\bar{V}$	geocentric inertial velocity of Shuttle CG

$\bar{v}$	geocentric inertial velocity of payload CG
$[\alpha]$	array of (3) Euler angles
Ω_K	Keplerian angular velocity magnitude of Shuttle CG, $\lvert\bar{R} \times \bar{V}\rvert/R^2$
$\dot{\Omega}_K$	$d\Omega_K/dt$
ω_K	Keplerian angular velocity magnitude of payload CG, $\lvert\bar{r} \times \bar{v}\rvert/r^2$
$\dot{\omega}_K$	$d\omega_K/dt$
$\bar{\omega}_G^{FG}$	angular velocity vector of system G with respect to system F, resolved onto axes of system G
$\dot{\bar{\omega}}_G^{FG}$	$D\bar{\omega}_G^{FG}/dt$

Subscripts

B	Shuttle body-fixed (SBY) coordinate system
b	payload body-fixed (PBY) coordinate system
G	Shuttle-centered local vertical (SLV) coordinate system
g	payload-centered local vertical (PLV) coordinate system
H	Shuttle-centered UVW coordinate system
h	payload-centered UVW coordinate system
I	mean of launch date (MLD) ECI coordinate system
J	mean of 1950.0 (M50) ECI coordinate system
P	principal axes of inertia of Shuttle
p	principal axes of inertia of payload
S	Shuttle sensor coordinate system

Superscripts

A	actual CG of Shuttle
a	actual CG of payload
B	Shuttle body-fixed (SBY) coordinate system
b	payload body-fixed (PBY) coordinate system
G	Shuttle-centered local vertical (SLV) coordinate system
g	payload-centered local vertical (PLV) coordinate system
I	(any) inertial coordinate system
N	nominal CG of Shuttle (STA 1076.7, BL 0, WL 375)
n	"nominal CG" of payload (STA 0, BL 0, WL 0); = center of front face of payload cylinder
S	origin of Shuttle sensor coordinates
T	transpose of matrix
t	payload targetpoint
α	aerodynamic effect
γ	gravitational effect
τ	thrust effect

Product Symbols

o	quaternion product
X	vector cross product
·	vector dot (scalar) product

FOREWORD

This document contains a detailed description of the trajectory processor (#TRAJ) of the High Fidelity Relative Motion Program (HFRMP) Version 03T(1229/ 10NOV79). The HFRMP is coded in Hewlett-Packard Language (HPL) for the HP-9825A Desktop Calculator with memory option 002 (23,228 bytes of read/ write memory) and the following peripheral devices and read-only memory (ROM) modules:

Required Peripheral Devices

HP-9885M	Flexible Disk Drive
HP-9872A	Plotter
HP-9866B	Thermal Line Printer
	or
HP-9871A	Character Impact Printer

Required ROM Modules

HP-98210A	String-Advanced Programming
HP-98211A	Matrix Programming
HP-98217A	9885M Flexible Disk Drive
HP-98216A	9872A Plotter-General I/O-Extended I/O
	or
HP-98215A	9872A Plotter-General I/O

Applicable HPL programming information is contained, along with operating instructions for the calculator and its peripheral devices, in the following HP-9825A manuals published by the Hewlett-Packard Calculator Products Division:

HP Part No.	Title of Manual
09825-90000	Operating and Programming
09825-90020	String Variable Programming
09825-90021	Advanced Programming
09825-90022	Matrix Programming
09825-90024	General I/O Programming
09825-90026	9872A Plotter Programming
09871-90000	9871A Printer...Operating Manual
09885-90000	Disk Programming

A basic understanding of the contents of the cited manuals is required for a full comprehension of this program document.

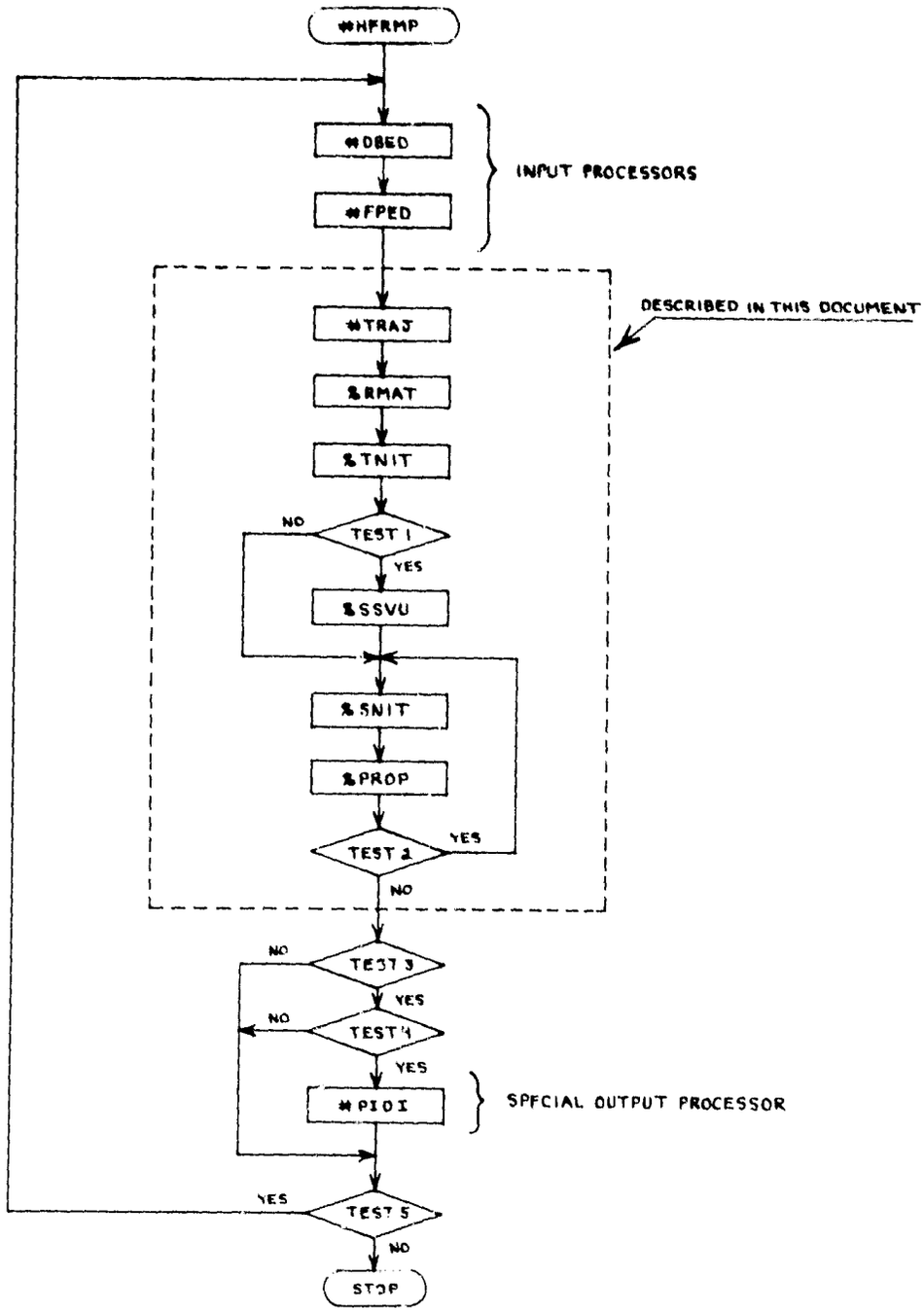
1. INTRODUCTION

The HP-9825A High Fidelity Relative Motion Program (HFRMP) is a disk-resident flight design software system specifically intended for the analysis of Space Transportation System (STS) proximity operations. The mainline logic flow† is illustrated in Figure 1. The code is divided into several primary modules that are designated "processors" and identified as such (according to HFRMP convention) using the pound-sign (#) symbol as the first character in their identification strings (i.e., names). Some processors, being too large to fit in the available read/write memory area, are broken down into subsidiary links which are identified by the use of the percent (%) symbol as the first character in their identification strings. Subsidiary links overlay each other in a shared memory area, as illustrated in Figure 2.

Processors (such as #TRAJ) are called into read/write memory from the program disk by use of the HPL get command, which erases all variables (numerical data) that may have been stored in memory. Information transfer between processors is accomplished by means of disk-resident data files that are stored and retrieved, as necessary, by appropriate commands coded into the processors. This includes user-supplied input data, which can be manipulated (edited/saved/recalled) by the user with the aid of interactive input processors (#DBED and #FPED in Figure 1).

From the preceding remarks it can be seen that every HFRMP processor represents, in a limited sense, a stand-alone program. Therefore it is possible,

†i.e., that which is invoked by a regular production run of the program. Not shown are a number of special processors that are used for program maintenance.



TEST 1 : BODY-FIXED RELATIVE MOTION PLOTS REQUESTED?

TEST 2 : MORE FLIGHT SEGMENTS?

TEST 3 : DID A PAYLOAD SRM BURN OCCUR?

TEST 4 : PLUME IMPINGEMENT DAMAGE ANALYSIS REQUESTED?

TEST 5 : START A NEW RUN?

Figure 1. HFRMP Mainline Logic Flow

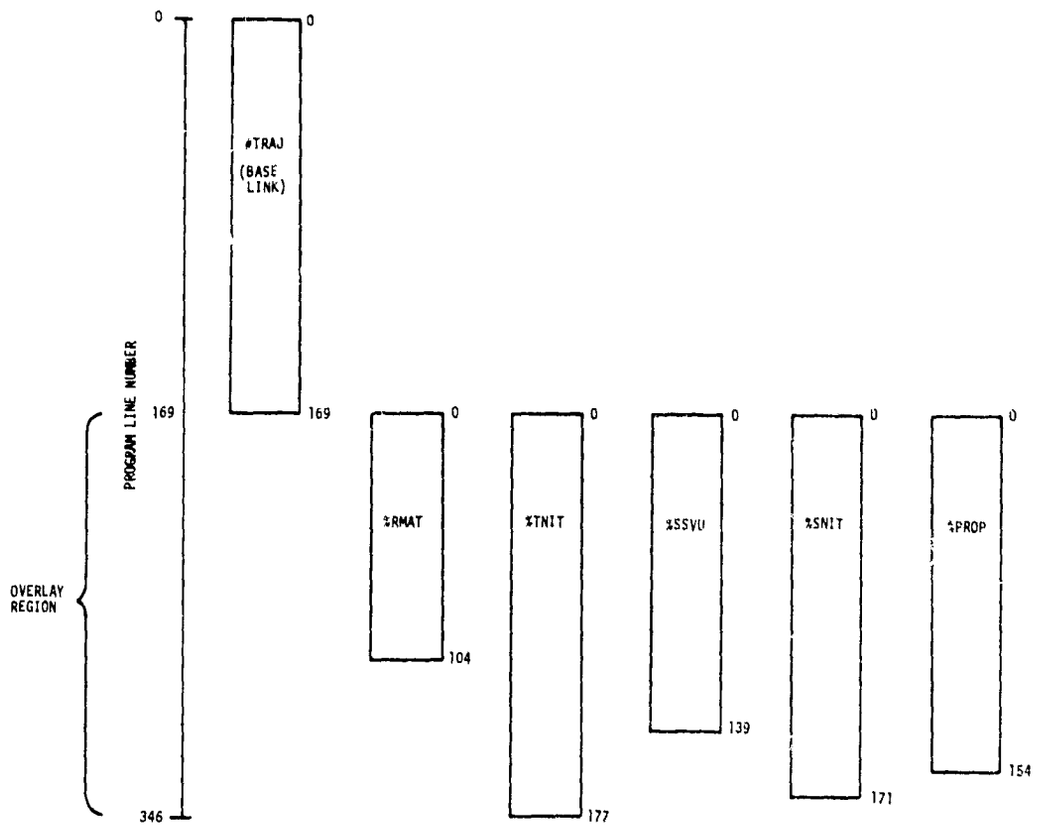


Figure 2. #TRAJ Overlay Structure

for instance, to interrupt the normal HFRMP execution sequence after input processing has been completed and then later to resume execution by commanding get "#TRAJ" from the calculator keyboard and pressing the (RUN) button. By the same token, it is not necessary to execute the HFRMP particle impact damage integrator (#PIDI) immediately after executing #TRAJ. The vehicle ephemeris data that #PIDI requires is stored on the program disk by #TRAJ, where it is preserved indefinitely until written over by a subsequent execution of #TRAJ. Therefore, a user can execute the trajectory processor, shut off the calculator and remove his program disk, and then initiate a particle impact damage analysis of the subject trajectory at any later date simply by inserting his disk, commanding get "#PIDI" from the keyboard, and pressing the (RUN) button.

The remainder of this document will be concerned only with the #TRAJ processor, which contains the HFRMP dynamic models, the trajectory/attitude integration logic, and the basic output computations. Detailed program documentation for the HFRMP input processors and the #PIDI output processor are to be published under separate cover. (Reference 1 contains operating instructions for the HFRMP, with emphasis on input operations. Reference 2 contains a definition of the basic equations used in the #PIDI processor.)

1.1 GENERAL DESCRIPTION OF #TRAJ

The #TRAJ processor is a 12-degrees-of-freedom trajectory integrator (6 degrees of freedom for each of two vehicles) which generates digital and graphical data to describe the relative motion of the Space Shuttle Orbiter and a free-flying payload. These data are obtained by differencing the geocentric states of the individual vehicles, computed to a numerical precision of 12 decimal digits, with respect to an oblate earth whose gravitational model includes the second harmonic coefficient (J_2). The state of the two-vehicle system is computed as a function of time by means of a fourth-order Runge-Kutta numerical integration scheme which uses unit quaternions (versors) to define both the rotational and the translational states of each vehicle. The derivation of the versor-based translational state vector is contained in Reference 3, which is reproduced for the most part (with some changes in coordinate system definition) in Appendix A and Sections 1.2 and 1.3 of this report.

The payload is modeled geometrically as a cylinder whose length and diameter are specified by program input. The Orbiter and the payload are treated as rigid bodies whose individual mass properties (gross weight, moments and products of inertia, and center of gravity location) are specified by program input and are assumed to remain constant† during the HFRMP run. Gravity gradient torque is included in the rotational equations of motion for both vehicles. At the user's option, aerodynamic torque and drag can also be

†An exception is made when integrating the trajectory of an upper stage during a major translational maneuver such as a solid rocket motor burn. In such a case, the payload gross weight is decremented during the burn to reflect the consumption of propellant.

included for either or both vehicles. Aerodynamic forces normal to the relative wind vector are ignored.

The atmosphere is assumed to rotate with the earth, and is modeled as a function of geodetic altitude by a curve fit of the 1962 Standard density profile (Reference 4). The curve fit is valid down to a minimum altitude of approximately 400,000 feet. The density profile can be modified by a program input factor (K_ρ) to account for the major effects of solar activity. Aerodynamic drag and moment coefficients for the Orbiter are computed as curve-fit functions of its attitude with respect to the relative wind vector (References 5-7).

Aerodynamically, the payload is modeled as a flat plate whose size and shape are determined by projecting its cylindrical outline onto a plane normal to the relative wind vector. The payload drag coefficient is assumed to be 2.0, based on its projected frontal area. Aerodynamic effects on the Orbiter and the payload can be modified (or cancelled entirely) by means of input factors ($K\alpha$ and $k\alpha$) which are applied uniformly to all aerodynamic forces and torques that are computed for the specified vehicle. The possible effects of aerodynamic shadowing (of one vehicle by the other) are not accounted for in the internal calculations of #TRAJ.

Several options are provided for defining the initial state of the two-vehicle system. The translational state of the Orbiter can be described either in terms of osculating orbit elements referenced to the Mean of 1950.0 (M50) geocentric equatorial frame, or in terms of invariant orbit elements (Reference 8) measured in the Mean of Launch Date (MLD) equatorial frame. The initial attitude of the Orbiter is defined by pitch, yaw, and roll angles (taken in that order) referenced to the rotating Shuttle-centered local-

vertical (SLV) coordinate system. The Orbiter's angular velocity, measured in terms of rate components about its body axes, can be defined with respect to either the M50 (inertial) frame or the SLV (rotating) frame.

The initial translational state of the payload is defined by rectangular position and velocity components which are measured relative to the Orbiter's center of gravity (CG). At the user's option, these components can be measured in the SLV coordinate system, or in the Shuttle body (SBY) coordinates system. The payload's initial pitch, yaw, and roll angles can be referenced either to the payload-centered local-vertical (PLV) system, or to the SBY system. The payload's angular rate components about its body axes can be defined relative to the M50, the PLV, or the SBY frame.

The initial state of the system can be advanced through up to 40 flight profile segments, each of arbitrary length, which are defined by the user in a prior execution of the #FPED processor.† At the beginning of any segment the user may command the application of an impulsive (i.e., instantaneous) increment to the angular rate of either or both vehicles.* In this regard, the user may specify a particular rate increment (INCR), a desired rate with respect to inertial space (IR), or a desired rate with respect to the local-vertical frame of the vehicle in question (LVR). In all cases, the components of the desired rate or rate increment are measured about the body axes of the

†All input data, including the flight profile definition, are saved in disk files whence they can be recalled (and edited, if necessary) for use in subsequent runs.

*Commanded angular velocity impulses, and the linear velocity impulses which they may induce as a result of RCS translational cross coupling, are the only types of state variable discontinuity that are permitted by #TRAJ. These are allowed only at the beginning of a flight profile segment.

vehicle in question.

After applying the specified angular velocity increment (if any) at the beginning of the flight segment, #TRAJ then (for each vehicle independently of the other)

1. allows the attitude to drift (D) under the influence of inertia and natural torques, or
2. performs inertial rate hold (IRH) control (i.e., maintains a constant angular velocity relative to inertial space), or
3. performs local-vertical rate hold (LVRH) control (i.e., maintains a constant angular velocity relative to the rotating local-vertical frame of the vehicle being controlled)

for the duration of the segment, depending on the user's specifications for that segment.

When the IRH or the LVRH attitude-maintenance option is specified for the Orbiter, a simplified RCS/DAP model (Reference 9) is used to compute average values for the propellant consumption rates and translational cross-coupling accelerations that result from the intermittent thruster firings which are required to apply the necessary control torques. The model takes into account the mass properties of the Orbiter, the electrical width (an integer multiple of the DAP cycle time) and the effective width (the duration of steady-state acceleration) of the RCS thruster pulses, and the width of the attitude deadband about each of the Orbiter's body axes. Deadbands can be changed from segment to segment in the flight profile, as can the selection of primary or vernier thrusters and the mode of cross-coupling compensation. Translational cross-coupling accelerations are integrated along with those produced by gravity, aerodynamics, and commended translational thrust. They are reflected in the output data by the flight path deviations they produce. Propellant

consumption rates are also integrated (but not subtracted from the Orbiter gross weight), and the accumulated expenditures are tabulated, along with other data, at user-specified time intervals. Propellant consumption is broken down according to source (forward, aft left, or aft right tank) and function (translational or rotational control).

When the IRH or the LVRH attitude-maintenance option is specified for the payload, the magnitudes of the necessary control torques are integrated and the accumulated rotational impulse (measured in pound-foot-seconds) in the positive and negative direction about each body axis is printed along with the other digital output data. Since no specific method of implementation is modeled, it is not possible to compute propellant consumption rates or cross-coupling effects that may result from payload attitude control.

Translational thrust acceleration of either or both vehicles can be commanded at the beginning of any flight profile segment. Payload translational thrust is always applied in the direction of the payload's +X body axis and is assumed to be directed through the CG. Once initiated, payload thrust acceleration continues until all of the rocket motor propellant is consumed, as determined by a table of flow rates versus burn time.

Translational acceleration of the Orbiter is initiated by commanding ignition of either or both of the OMS engines (L, R, or L+R) and/or by firing primary RCS thrusters to produce thrust nominally in the positive or negative directions of the Orbiter body axes (+X, -X, +Y, -Y, +Z, or -Z). Once initiated, Orbiter translational acceleration is applied continuously at the nominal steady-state level, throughout the duration of the flight profile segment.

1.2 STATE VECTOR DEFINITIONS

There are many numerical methods which can be used to integrate the differential equations that govern the motion of a rigid body such as the Orbiter or a free-flying payload. A common requirement of such methods is the definition of a state vector[†] along with a set of equations which, given the values of the state variables at any particular time, permit the calculation of the first derivative of every state variable with respect to time.

A fourth-order Runge-Kutta numerical integration technique is used in the #TRAJ processor. However, the state vector definitions and the derivative calculations (which comprise the bulk of the computations involved in state vector propagation) are not influenced directly by the integration logic except in the sense that the integrator determines how frequently the derivatives have to be evaluated. In #TRAJ, the integration logic and the derivative calculations are isolated from each other in separate subroutines, thus making it easy to change the numerical integration technique if that should become desirable.

The purpose of this section of the report is to describe the state variable that have been adopted for use in #TRAJ. The associated derivative calculations, described in terms of external forces and torques that act on the body under consideration, will be covered in Section 1.3. The detailed calculation of these forces and torques will be described in Section 2.3, and the Runge-Kutta integration logic will be described in Section 7.2.

[†]A set of variables that define the position and linear velocity of the body's CG (in the case of translational motion) with respect to a chosen system of reference, and/or the body's attitude and angular velocity (in the case of rotational motion about the CG).

The #TRAJ processor makes use of versors (unit quaternions) as internal state variables to describe the translational as well as the rotational motions of the Orbiter and the payload. (The HFRMP user never sees the state vectors in quaternion form; input and output data are defined in terms of Euler angles.) The computational advantages of using versors to describe rotational motion are well known. Certain advantages also accrue from their use to describe the translational motion of a satellite, as explained in Reference 3. Versors are also used extensively in #TRAJ for coordinate transformations in general, and a working knowledge of their characteristics and rules of manipulation is necessary for a good understanding of the remainder of this report. A careful study of Appendix A is recommended at this point for the reader who is not accustomed to the use of quaternions. Even for those who are familiar with quaternion applications, it probably will be helpful to review the conventions and the system of notation defined in Appendix A. All readers should examine the coordinate system definitions in Appendix B.

1.2.1 Rotational State Vector

The rotational state of the Shuttle at any given time t is defined by the variables $[\bar{\bar{q}}_{IB}, \bar{\omega}_B^{IB}]$, where $\bar{\bar{q}}_{IB}$ is a versor that defines the orientation of the Shuttle body-fixed (SBY) coordinate system B with respect to the geocentric mean-of-launch-date (MLD) equatorial inertial coordinate system I. The vector symbol $\bar{\omega}_B^{IB}$ represents the angular velocity of the Orbiter. The double superscript IB identifies the angular velocity as that of system B with respect to system I, and the subscript B indicates that the vector is resolved into components along the axes of the B system. The rotational state of the payload is similarly defined by the variables $[\bar{\bar{q}}_{Ib}, \bar{\omega}_b^{Ib}]$, whose definitions are the same in all respects except that the lower-case symbol b is used to identify

the body-fixed coordinate system of the payload.

1.2.2 Translational State Vector

The translational state of the Orbiter is defined by the variables $[\bar{q}_{IG}, R, \dot{R}, \Omega_K]$. The symbol $\bar{q}_{IG}$ represents a versor that defines the orientation of the Shuttle local-vertical (SLV) coordinate system G with respect to the inertial system I. The symbol R represents the distance from the center of the earth to the Shuttle CG, and $\dot{R}$ stands for its derivative dR/dt. The symbol Ω_K represents the scalar magnitude of the instantaneous geocentric angular velocity of the Shuttle CG. The subscript K in this case does not designate a coordinate system; it merely identifies the angular velocity in question as that which is associated with Kepler's law of equal areas.

The translational state of the payload is similarly composed of the variables $[\bar{q}_{Ig}, r, \dot{r}, \omega_K]$, which are defined in the same manner as those described in the preceding paragraph except for the use of lower-case symbols to represent the scalar quantities and the coordinate-system designator (g) of the payload local-vertical (PLV) coordinate system.

1.3 BASIC EQUATIONS OF MOTION

The purpose of this section is to define the equations for calculating the derivatives of the #TRAJ internal state variables, assuming the forces and torques acting on the bodies are known. Equations for calculating the forces and torques will be given in Section 2.3.

The only equations that will be defined explicitly in this section are those that apply to the motion of the Shuttle. The payload equations are identical in all important respects, and can be obtained simply by substituting the lower-case symbols b, g, r, $\dot{r}$, and ω_K for their upper-case counterparts B, G, R, $\dot{R}$, and Ω_K.

1.3.1 Rotational Motion

What is required here are the first derivatives of $\overline{\overline{q}}_{IB}$ and $\overline{\omega}_B^{IB}$ with respect to time. The derivative of the orientation versor is equal to one-half of its quaternion product with $\overline{\omega}_B^{IB}$, i.e.,

$$\dot{\overline{\overline{q}}}_{IB} = \left(\overline{\overline{q}}_{IB} \circ \overline{\omega}_B^{IB}\right)/2. \tag{1}$$

The derivative of $\overline{\omega}_B^{IB}$ is given by the equation

$$\dot{\overline{\omega}}_B^{IB} = [I]_B^{-1} \left(\overline{L}_B - \overline{\omega}_B^{IB} \times [I]_B \overline{\omega}_B^{IB}\right), \tag{2}$$

where the vector $\overline{L}_B$ represents the external torque acting on the Shuttle. The symbol $[I]_B$ represents the Shuttle's inertia tensor (a 3x3 matrix whose elements consist of the moments and negative products of inertia, referenced to the B system), and $[I]_B^{-1}$ represents its inverse. For the purpose of computing the product of a matrix and a vector, the vector is treated as a column matrix, e.g.,

$$\bar{\omega}_B^{IB} = \begin{bmatrix} \omega_{B1}^{IB} \\ \omega_{B2}^{IB} \\ \omega_{B3}^{IB} \end{bmatrix}$$

1.3.2 Translational Motion

We seek now to define the first derivatives of the translational state variables $\bar{\bar{q}}_{IG}$, $\dot{R}$, and Ω_K with respect to time. (The derivative of the state variable R is no problem, because it is equal to $\dot{R}$). By definition, the Shuttle's local-vertical coordinate system G rotates in such a manner that

$$\bar{R}_G = - \hat{k} R \tag{3}$$

and

$$(\bar{R}x\bar{V})_G = - \hat{j} |\bar{R}x\bar{V}| = - \hat{j} R^2 \Omega_K \tag{4}$$

at every instant of time, where $\bar{R}$ and $\bar{V}$ represent the geocentric position and inertial velocity vectors of the Shuttle CG.

An expression for the inertial velocity can be obtained by differentiating Equation (3) with respect to time. This yields

$$\bar{V}_G = - \hat{k} \dot{R} - \bar{\omega}_G^{IG} \times \hat{k} R, \tag{5}$$

where the vector

$$\bar{\omega}_G^{IG} = \hat{i} \omega_{G1}^{IG} + \hat{j} \omega_{G2}^{IG} + \hat{k} \omega_{G3}^{IG} \tag{6}$$

(whose components have yet to be evaluated) represents the total angular velocity of coordinate system G with respect to the inertial system I. We

now differentiate Equation (5) with respect to time and obtain the expression

$$\bar{A}_G = - \hat{k}\,\ddot{R} - 2\,\bar{\omega}_G^{IG} \times \hat{k}\,\dot{R}$$

$$- \dot{\bar{\omega}}_G^{IG} \times \hat{k}\,R - \bar{\omega}_G^{IG} \times \left(\bar{\omega}_G^{IG} \times \hat{k}\,R\right) \tag{7}$$

for the linear acceleration of the Shuttle CG, where the vector

$$\dot{\bar{\omega}}_G^{IG} = \hat{i}\,\dot{\omega}_{G1}^{IG} + \hat{j}\,\dot{\omega}_{G2}^{IG} + \hat{k}\,\dot{\omega}_{G3}^{IG} \tag{8}$$

represents the angular acceleration of coordinate system G.

We now proceed to evaluate the components of $\bar{\omega}_G^{IG}$ and $\dot{\bar{\omega}}_G^{IG}$. First, we substitute the expression for $\bar{\omega}_G^{IG}$ from Equation (6) into (5), and expand the vector cross product. This results in the equation

$$\bar{V}_G = - \hat{i}\,R\,\omega_{G2}^{IG} + \hat{j}\,R\,\omega_{G1}^{IG} - \hat{k}\,\dot{R}. \tag{9}$$

Using the expression for $\bar{R}_G$ and $\bar{V}_G$ from Equations (3) and (9), we form the cross product

$$(\bar{R}\times\bar{V})_G = \hat{i}\,R^2\,\omega_{G1}^{IG} + \hat{j}\,R^2\,\omega_{G2}^{IG}. \tag{10}$$

Equating the vector components of Equations (4) and (10), we obtain

$$\omega_{G1}^{IG} = 0 \tag{11}$$

and

$$\omega_{G2}^{IG} = - \Omega_K \tag{12}$$

which lead to

$$\dot{\omega}_{G1}^{IG} = 0 \tag{13}$$

and

$$\dot{\omega}_{G2}^{IG} = - \dot{\Omega}_K \tag{14}$$

Substituting (11), (12), (13), and (14) into Equations (6), (8), and (9), we obtain

$$\bar{\omega}_G^{IG} = - \hat{j}\, \Omega_K + \hat{k}\, \omega_{G3}^{IG}, \tag{15}$$

$$\dot{\bar{\omega}}_G^{IG} = - \hat{j}\, \dot{\Omega}_K + \hat{k}\, \dot{\omega}_{G3}^{IG}, \tag{16}$$

and

$$\bar{V}_G = \hat{i}\, R\, \Omega_K - \hat{k}\, \dot{R}. \tag{17}$$

Using the expressions for $\bar{\omega}_G^{IG}$ and $\dot{\bar{\omega}}_G^{IG}$ from Equations (15) and (16), the cross products appearing in Equation (7) can now be expanded to obtain

$$\bar{A}_G = \hat{i}\, (2\, \dot{R}\, \Omega_K + R\, \dot{\Omega}_K) + \hat{j}\, R\, \Omega_K\, \omega_{G3}^{IG} + \hat{k}\, (R\, \Omega_K^2 - \ddot{R}). \tag{18}$$

From Newton's second law, we have

$$\bar{F}_G = M\, \bar{A}_G, \tag{19}$$

where M represents the mass of the Shuttle and the vector

$$\bar{F}_G = \hat{i}\, F_{G1} + \hat{j}\, F_{G2} + \hat{k}\, F_{G3} \tag{20}$$

represents the external force acting on it. Combining Equations (18), (19), and (20), and equating vector components, we obtain the equations

$$\ddot{R} = R\, \Omega_K^2 - F_{G3}/M \tag{21}$$

$$\dot{\Omega}_K = (F_{G1}/M - 2\,\dot{R}\,\Omega_K)/R \tag{22}$$

and

$$\omega_{G3}^{IG} = F_{G2}/(M\,R\,\Omega_K). \tag{23}$$

Equations (21) and (22) define the derivatives of the translational state variables $\dot{R}$ and Ω_K. The derivative of the remaining state variable $\bar{\bar{q}}_{IG}$ is given by

$$\dot{\bar{\bar{q}}}_{IG} = (\bar{\bar{q}}_{IG} \circ \bar{\omega}_G^{IG})/2, \tag{24}$$

where the angular velocity vector

$$\bar{\omega}_G^{IG} = -\hat{j}\,\Omega_K + \hat{k}\,F_{G2}/(M\,R\,\Omega_K) \tag{25}$$

is obtained by substituting Equation (23) into (15).

Equations (21), (22), (24), and (25) are valid for all states of motion except those where $|\bar{R}x\bar{V}| = 0$, in which case the geocentric flight path would be vertical, and the orientation of the X and Y axes in the local-vertical coordinate system would be indeterminate.

1.4 CONVENTIONS AND CODING STANDARDS

HP-9825 programs are coded in the Hewlett-Packard Language (HPL), which is functionally similar to BASIC but unique in terms of its syntax. To facilitate correlation with the #TRAJ code (Appendix C), some of the syntactical features of HPL have been carried over into the flow charts (Sections 2-7).

For instance, consider a computation which involves forming the product of two simple variables A and B, where the resulting value is to be assigned to a third variable C. In FORTRAN code this operation would be represented by writing C = A * B, and in most flow charts by writing C = AB. In HPL code, and in the flow charts that follow, the same operation would be described by writing AB → C. (Perhaps the most succinct verbalization of the HPL value-assignment symbol is obtained by substituting the phrase "goes to" for the right-running arrow "→".)

Another feature of HPL code and the flow charts, which may be confusing to FORTRAN programmers, is the use of square brackets to enclose the index or indices of a dimensioned variable. The Jth element of the HPL dimensioned variable A[*] is designated by writing A[J], whereas in FORTRAN it would be written A(J). In HPL code, the square brackets are always used to enclose indexing or dimensioning information about an array (dimensioned) variable, and for no other purpose. This is necessary in HPL to distinguish, for instance, the simple (scalar) variable P from the array variable P[*] whose dimensions might have been declared by a statement dim P[5,10], which is analagous to the FORTRAN statement DIMENSION P(5,10). In a FORTRAN program where such a dimension statement exists, a reference to the variable P is an implicit reference to P(1,1), i.e., the first element in the P array. However, this is not the case in HPL, where the simple variable P has no connection

whatever with the array variable P[*] or any of its elements. For instance, the sequence of HPL statements 2 → P; 3 → P[1,1]; P P[1,1] → X would cause the number 6 to be assigned to the variable X.

1.4.1 General Arrangement of Program Files

The first line of code (line 0) in every HFRMP program file (processor or processor link) contains, in the form of a label, the name under which the file is stored on the program disk, followed by the date and time of its most recent revision. The executable part of this line contains the statement gto "RUN", where "RUN" is the label of the line where the main logic flow begins. The next two lines (1 and 2) contain statements that facilitate an automatic listing of the complete program, and which are never executed in a normal production run.

All subprograms are located between line 2 and the "RUN" label, and they are arranged in the following general order:

1. Function subprograms
2. Utility subroutines
3. Special subroutines.

The rationale for this order is to make the program listings as insensitive as possible to corrections or revisions of the logic. Function subprograms and utility subroutines are revised much less frequently than the main (driver) logic in a given link. Therefore the code in the top part of each link tends to remain comparatively stable.

To the greatest extent that is practicable, the code in each subroutine and in the driver logic of each individual link is arranged so that execution control flows from top to bottom, i.e., from smaller line numbers to greater ones.

1.4.2 Branching Conventions

For ease of program maintenance, branching to absolute line numbers is avoided. Relative addressing (e.g., gto + 4) is used whenever practicable; otherwise, the general rule is to branch to a symbolic label (e.g., gto "RUN"). The jmp statement is used only in the rare instances where it is necessary to compute a relative address (line number) at execution time.

To avoid confusion regarding the flow of execution control (and also generally to permit the transfer of date through an argument list), the HPL cll statement is used for branching to a subroutine in lieu of the gsb statement.

1.4.3 Program Labels

No lower-case alphabetic symbols are used in program labels (character strings used to identify branching targets, including the names of subprograms). The purpose of this convention is to provide a contrast between such labels and the commands and standard functions of HPL, all of which are spelled with lower-case alphabetic symbols.

1.4.4 Input/Output Mnemonics

Although numerical option codes are sometimes used internally by #TRAJ, it has been adopted as a policy that HFRMP users should not have to memorize (or look up) obscure numerical codes in order specify logical options in the input data files. Rather, such options are specified by the user in the form of mnemonic character strings which, if necessary, are converted to numerical codes internally.

Mnemonic character strings are also used to identify the HFRMP digital output data. With regard to all input/output mnemonics, a determined effort

has been put forth to make them as meaningful and as consistent as possible.

1.4.5 Data-Register Protocol

Because of the limitations of the HPL syntax, it has been found necessary to adopt a comparatively rigid protocol to govern the use of named variables and of the numbered r-registers. Appendix D contains storage allocation information for the r-registers.

1.4.5.1 Simple Variables

The simple (scalar) variable names, which are limited in HPL to single upper-case alphabetic characters (A thru Z), are used primarily as loop counters and for the temporary storage of intermediate computational results. Eight of the 26 simple variables are designated as volatile registers, which means that they are least rigidly controlled, and that their contents are most susceptible to frequent change. These are the registers identified by the characters H,I,J,K and W,X,Y,Z.

1.4.5.2 r-Registers

Because of the limited number of usable names for variables, and also because array names cannot be passed through the argument lists of HPL subprograms, most of the data that would normally be assigned unique names or stored in individual arrays (e.g., as in a FORTRAN program) are stored instead in the numbered r-registers. The correlation between the r-numbers and the logical symbols, shown in Appendix D, is extremely critical to the understanding of the #TRAJ code.

When there is a need for a utility subprogram to perform a standard computation involving one or more logical arrays (such as vectors and quaternions),

the address of each logical array (i.e., the r-number of its first element) is passed through the argument list (in lieu of an array name, which is not permitted in HPL). For instance, suppose the components of a vector $\overline{R}_G$ reside in the registers r1 through r3, the components of another vector $\overline{V}_G$ reside in r4 through r6, and it is desired to compute the vector $\overline{H}_G = \overline{R}_G \times \overline{V}_G$ and store it in r7 through r9. Symbolically, in the flow charts, this would be written $\overline{R}_G \times \overline{V}_G \rightarrow \overline{H}_G$, or possibly as c*ll* 'CRSP' $(\overline{R}_G,\overline{V}_G,\overline{H}_G)$. In the HPL code, it would appear as c*ll* 'CRSP' (1,4,7), where 'CRSP' is the name of the vector cross-product routine, and the numbers 1,4,7 are the addresses of the vectors involved in the operation.

The first 19 r-registers (r0 through r18) are similar to the simple-variable registers in that they are used mainly to store the intermediate results of array computations. The first ten r-registers (r0 through r9) are analogous to the simple variables H,I,J,K and W,X,Y,Z in that they are designated as volatile, i.e., most frequently used. Preferential use of the lower-numbered registers tends to minimize code-storage requirements, simply because they entail writing out fewer digits (each of which requires one byte of code storage) to identify the registers. That is to say, the statement c*ll* 'CRSP' (101,104,107) requires 6 more bytes of code storage space than the statement c*ll* 'CRSP' (1,4,7).

1.4.5.3 Restrictions on Utility Subprograms

During the execution of an HPL subprogram, the HP-9825 operating system allocates temporary storage (only for the duration of subprogram execution) to numbered p-variables. These variables are numbered sequentially beginning with p0, into which the operating system loads the number of parameters that appear in the argument list of the calling routine's reference to the sub-

program. If there should be three arguments, then at the beginning of subprogram execution p0 would contain the number 3. The registers p1, p2, and p3 would contain the values assigned to the arguments of the subprogram by the calling routine. The subprogram can make use of as many higher-numbered p-registers (p4, p5, etc.) as it may need for temporary storage of intermediate computational results. All of the p-registers are de-allocated (in effect, erased) when execution control is returned to the calling routine.

The operating system also permits the subprogram to reference (get values from or store values into) any of the registers accessible to the main program (this includes the simple variables, array variables, string variables, and r-registers). However, to prevent inadvertent modification of the contents of registers that the calling routine may be using, certain conventions have been adopted that limit the access of utility subprograms to global variables (i.e., any other than the p-variables).

In general, utility (general-purpose) subprograms are not allowed to make literal reference to any r-register. They are allowed to make logical reference to such registers by means of addresses that may be passed to them through their argument lists by the calling routine. For example, the character strings r0 or r25 are not permitted in the code of a utility subprogram. However, the character strings rp1 and r(p3+3) are permitted, where p1 and p3 represent addresses (in this case, the first and third numbers in the argument list) of logical arrays that are passed to the subprogram by the calling routine.

Function (as distinguished from subroutine) subprograms are not allowed to make reference to any simple variable. They are required to use p-variables for any temporary storage they might need. The same restriction applies to

any utility subroutine whose logical argument list contains only scalars and vectors.

Utility subroutines that perform quaternion and matrix operations are permitted to use the volatile simple variables (H,I,J,K and W,X,Y,Z), but no others. Special subroutines such as 'DERVIS' (Section 2.3), whose calculations generally are more complicated than those of utility subroutines, have no such general restrictions on their access to storage registers.

2. BASE LINK (#TRAJ)

The "#TRAJ" program file is the base link of the #TRAJ processor, containing all of the general-purpose subprograms, i.e., all that are used by two or more subsidiary links.

2.1 FUNCTION SUBPROGRAMS

2.1.1 'ANGl'

The function subprogram 'ANGl' converts an input angle to its equivalent value in the range of 0 to 2π.

2.1.1.1 Arguments

p1 = Input angle, measured in radians.

2.1.1.2 Example of Usage

The instruction 'ANGl'$(-3\pi) \rightarrow$ A in the calling routine would cause the value of $+\pi$ to be assigned to the variable A.

2.1.1.3 Computations

The fractional part of $p1/2\pi$ is multiplied by 2π. If the result is negative, 2π is added. A flow chart is shown in Figure 3.

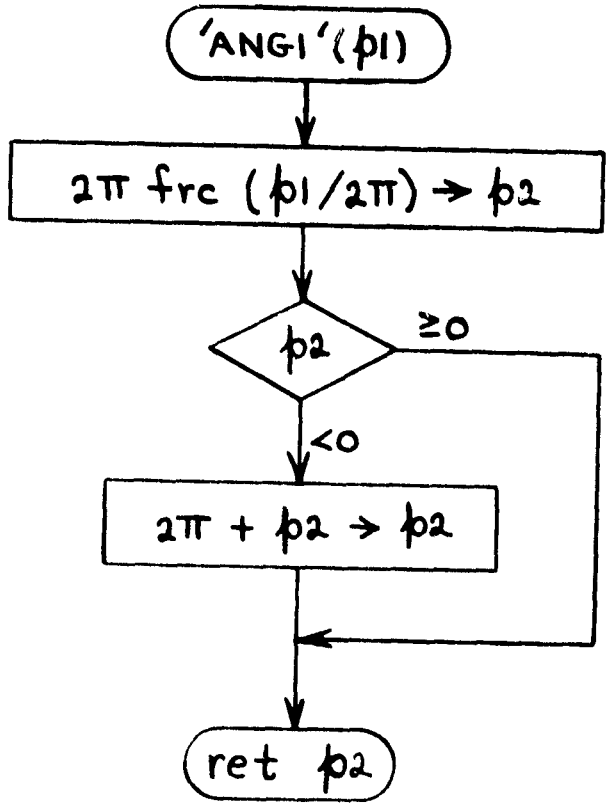


Figure 3. 'ANGI' Function Subprogram Logic Flow

2.1.2 'ANG2'

The function subprogram 'ANG2' converts an input angle to its equivalent value in the range of $-\pi$ to $+\pi$.

2.1.2.1 Arguments

p1 = Input angle, measured in radians.

2.1.2.2 Example of Usage

The instruction 'ANG2'$(-3\pi) \rightarrow$ A in the calling routine would cause the value of $-\pi$ to be assigned to the variable A.

2.1.2.3 Computations

The fractional part of $p1/2\pi$ is multiplied by 2π. If the result is greater than π, 2π is subtracted. A flow chart is shown in Figure 4.

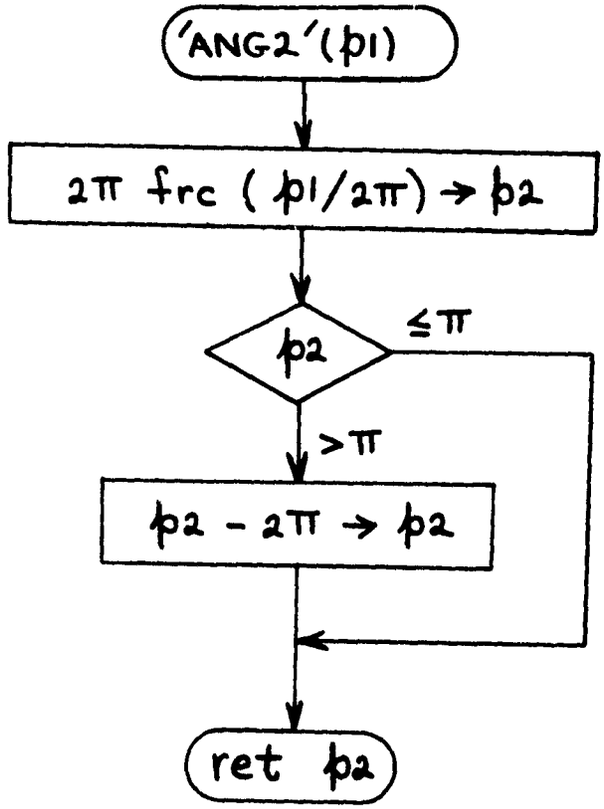


Figure 4. 'ANG2' Function Subprogram Logic Flow

2.1.3 'ATN1'/'ATN2'

This multiple entrypoint function subprogram computes the arctangent of the ratio of two input quantities, whose individual signs determine the quadrant of the output angle. The 'ATN2' entrypoint is analogous to the FORTRAN ATAN2 function, in that the output angle lies in the range of $-\pi$ to $+\pi$. When the 'ATN1' entrypoint is used, the output angle lies in the range of 0 to 2π.

2.1.3.1 Examples of Usage

The instruction 'ATN1'(-1,1) → A would cause the variable A to be assigned the value of $7\pi/4$, while the instruction 'ATN2'(-1,1) → A would cause it to be assigned the value of $-\pi/4$.

2.1.3.2 Computations

Tests are made to avoid division by zero. Depending on whether p1 or p2 has the greater magnitude, a base value of the output angle is computed from the expression $\pi/2$ - atn(p2/abs(p1)) or the expression atn(abs(p1/p2)), where atn and abs represent the HPL arctangent and absolute value functions. Tests on the signs of p1 and p2 then are made so that the base value can be rotated into the proper quadrant. If p1 = p2 = 0, the output angle is set equal to zero. A flow chart is shown in Figure 5.

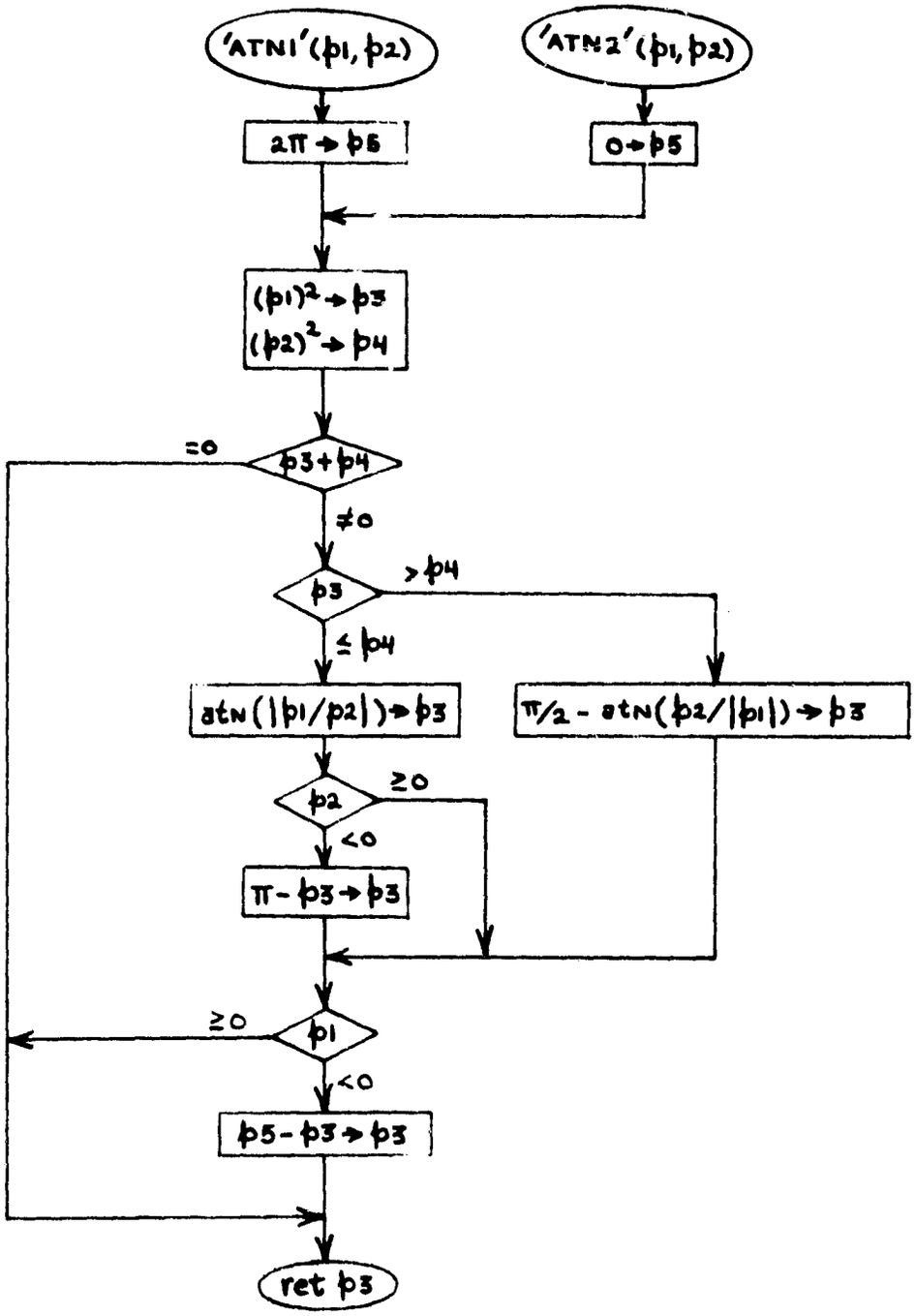


Figure 5. 'ATN1'/'ATN2' Function Subprogram Logic Flow

2.1.4 'HM.S'

The 'HM.S' function converts time from seconds to hours, minutes, and seconds. The output is a single number in the form of HHMM.SS, where HH denotes the hours digits, MM denotes the minutes digits, and SS denotes the seconds digits.

2.1.4.1 Arguments

p1 = Input time, measured in seconds.

2.1.4.2 Example of Usage

The instruction 'HM.S'(36385.8) → H in the calling routine would cause the variable H to be assigned the value 1006.258 (10 hours, 6 minutes, and 25.8 seconds).

2.1.4.3 Computations

See Figure 6.

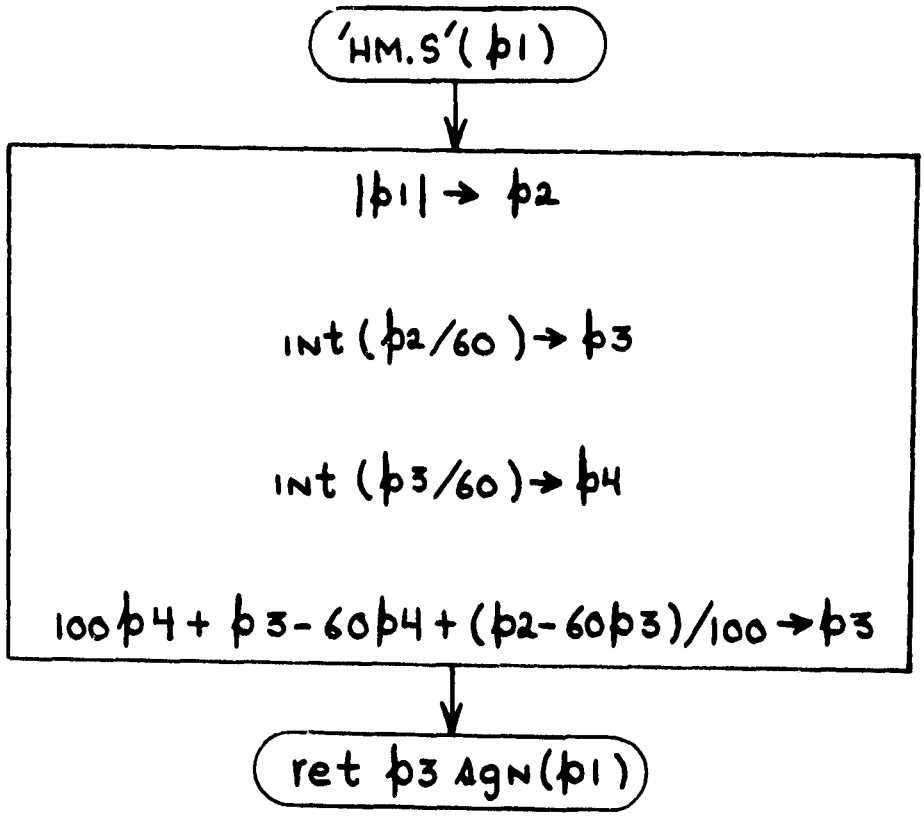


Figure 6. 'HM.S' Function Subprogram Logic Flow

2.1.5 'SECS'

The 'SECS' function is the inverse of the 'HM.S' function described in Section 2.1.4. It converts time from hours, minutes, and seconds (expressed in the HHMM.SS format) into seconds.

2.1.5.1 Arguments

p1 = Input time, in the HHMM.SS format.

2.1.5.2 Example of Usage

The instruction 'SECS'(1006.258) → S in the calling routine would cause the variable S to be assigned the value 36385.8 (the number of seconds in 10 hours, 6 minutes, and 25.8 seconds).

2.1.5.3 Computations

See Figure 7.

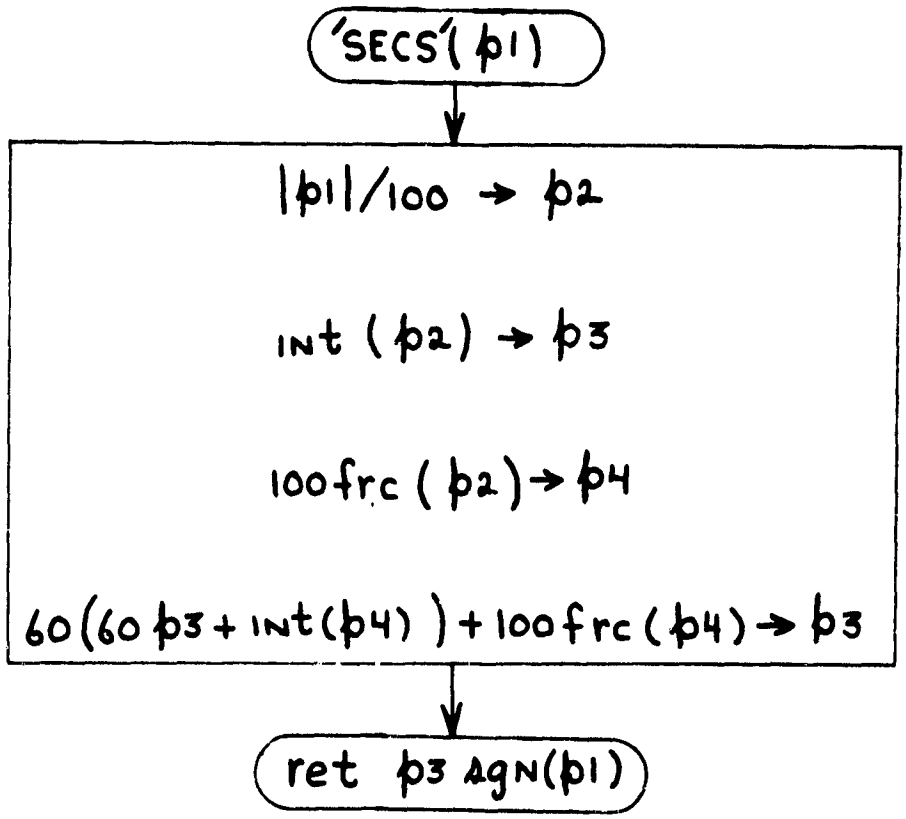


Figure 7. 'SECS' Function Subprogram Logic Flow

2.1.6 'DOTP'

The 'DOTP' function subprogram computes the dot (scalar) product of two vectors.

2.1.6.1 Arguments

p1 = Address of first vector.

p2 = Address of second vector.

2.1.6.2 Example of Usage

If vector $\overline{A}$ resides in registers r1, r2, r3, and vector $\overline{B}$ resides in registers r4, r5, r6, then the instruction 'DOTP'(1,4) → D in the calling routine would cause the value of $\overline{A} \cdot \overline{B}$ to be assigned to the variable D.

2.1.6.3 Computations

See Figure 8.

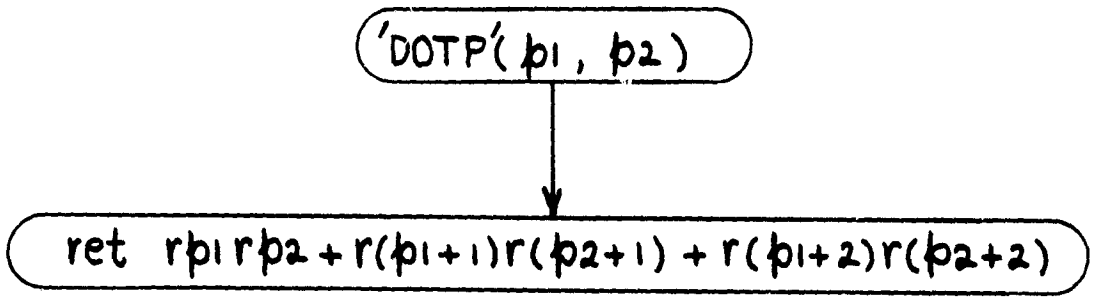


Figure 8. 'DOTP' Function Subprogram Logic Flow

2.2 UTILITY SUBROUTINES

2.2.1 'SXV'

The subroutine 'SXV' calculates the product of a scalar with a vector.

2.2.1.1 Arguments

p1 = Value of scalar.

p2 = Address of input vector.

p3 = Address of output vector.

2.2.1.2 Examples of Usage

The instruction *cℓℓ* 'SXV'(6.8,0,3) in the calling routine would cause the values 6.8 r0, 6.8 r1, 6.8 r2 to be stored in r3, r4, r5. If vector $\overline{A}$ resides in r11, r12, r13, and $\overline{B}$ resides in r7, r8, r9, the instruction *cℓℓ* 'SXV'('DOTP'(11,7),11,7) would cause the value of $(\overline{A} \cdot \overline{B})\ \overline{A}$ to be stored in r7, r8, r9.

2.2.1.3 Computations

See Figure 9.

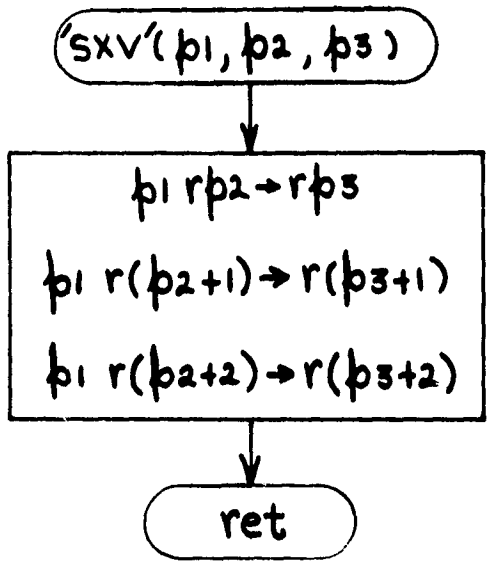


Figure 9. 'SXV' Subroutine Logic Flow

2.2.2 'VADD'/'VSUB'

Depending on which entrypoint is used, this subroutine will calculate either the sum or the difference of two vectors.

2.2.2.1 Arguments

p1 = Address of first input vector.

p2 = Address of second input vector.

p3 = Address of output vector.

2.2.2.2 Example of Usage

If vector $\overline{A}$ resides in r1, r2, r3 and vector $\overline{B}$ resides in r4, r5, r6, the instruction *cll* 'VADD'(1,4,7) in the calling routine would cause the value of $\overline{A} + \overline{B}$ to be stored in r7, r8, r9. The instruction *cll* 'VSUB'(1,4,4) would cause the value of $\overline{A} - \overline{B}$ to be stored in r4, r5, r6. The instruction *cll* 'VADD'(4,4,4) would cause the value of $2\overline{B}$ to replace $\overline{B}$ in r4, r5, r6. The instruction *cll* 'VSUB'(1,1,1) would cause zeros to be stored in r1, r2, r3.

2.2.2.3 Computations

See Figure 10.

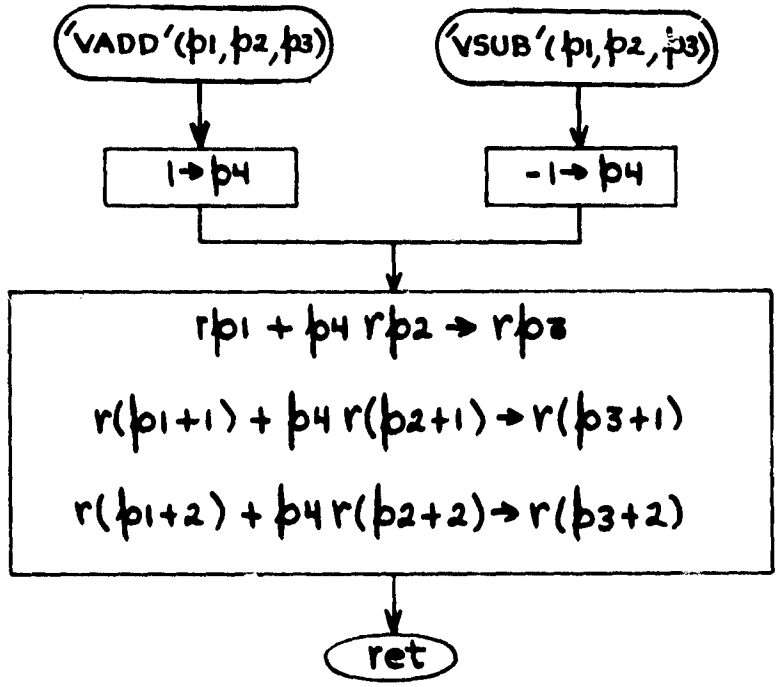


Figure 10. 'VADD'/'VSUB' Subroutine Logic Flow

2.2.3 'CRSP'

The 'CRSP' subroutine computes the cross product of two input vectors.

2.2.3.1 Arguments

p1 = Address of first input vector.

p2 = Address of second input vector.

p3 = Address of output vector.

2.2.3.2 Example of Usage

If the vector $\overline{A}$ resides in r1, r2, r3 and the vector $\overline{B}$ resides in r4, r5, r6, the instruction cℓℓ 'CRSP'(1,4,7) in the calling routine would cause the value of $\overline{A} \times \overline{B}$ to be stored in r7, r8, r9. The instruction cℓℓ 'CRSP'(4,1,4) would cause the value of $\overline{B} \times \overline{A}$ to replace $\overline{B}$ in r4, r5, r6.

2.2.3.3 Computations

See Figure 11.

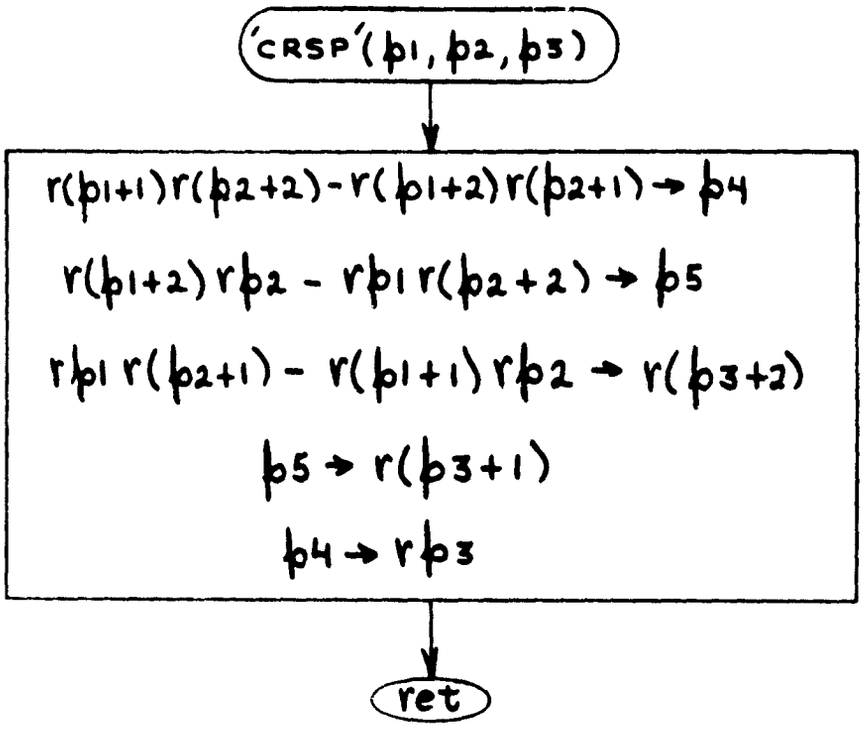


Figure 11. 'CRSP' Subroutine Logic Flow

2.2.4 'ROT'/'IROT'

This multiple entrypoint subroutine is used for transforming the components of a vector between two Cartesian coordinate systems that are angularly displaced from one another by a rotation about their common origin. Let the two coordinate systems be designated F and G, and let the versor (unit quaternion) $\overline{\overline{q}}_{FG}$ define the orientation of G with respect to F (see Appendix A). The 'ROT' entrypoint performs the computation symbolized by $\widetilde{\overline{q}}_{FG} \circ \overline{V}_F \circ \overline{\overline{q}}_{FG} \rightarrow \overline{V}_G$. The symbol $\overline{V}_F$ represents the value of a vector $\overline{V}$ in coordinate system F, and $\overline{V}_G$ represents the value of the same vector in system G. The IROT entrypoint performs the inverse transformation, symbolized by $\overline{\overline{q}}_{FG} \circ \overline{V}_G \circ \widetilde{\overline{q}}_{FG} \rightarrow \overline{V}_F$.

2.2.4.1 Arguments

p1 = Address of input vector.

p2 = Address of versor.

p3 = Address of output vector.

2.2.4.2 Examples of Usage

If the versor $\overline{\overline{q}}_{FG}$ resides in r0, r1, r2, r3 and the vector $\overline{V}_F$ resides in r4, r5, r6, the instruction *cℓℓ* 'ROT'(4,0,7) in the calling routine would cause the vector $\overline{V}_G$ to be stored in r7, r8, r9. A subsequent instruction *cℓℓ* 'IROT'(7,0,7) would cause a copy of the vector $\overline{V}_F$ to replace $\overline{V}_G$ in r7, r8, r9. The instruction *cℓℓ* 'ROT'(4,0,1) would cause $\overline{V}_G$ to be stored in r1, r2, r3 (thus destroying the last 3 components of the versor $\overline{\overline{q}}_{FG}$).

2.2.4.3 Computations

The quaternion operation $\widetilde{\overline{q}}_{FG} \circ \overline{V}_F \circ \overline{\overline{q}}_{FG} \rightarrow \overline{V}_G$ is equivalent to the matrix operation [R] $\overline{V}_F \rightarrow \overline{V}_G$, where

$$[R] = \begin{bmatrix} (HH+II-JJ-KK) & 2(IJ+HK) & 2(IK-HJ) \\ 2(IJ-HK) & (HH-II+JJ-KK) & 2(JK+HI) \\ 2(IK+HJ) & 2(JK-HI) & (HH-II-JJ+KK) \end{bmatrix}, \qquad (26)$$

and where $\overline{\overline{q}}_{FG} = H + \hat{i}\, I + \hat{j}\, J + \hat{k}\, K$. The inverse transformation $\overline{\overline{q}}_{FG} \circ \overline{V}_G \circ \tilde{\tilde{q}}_{FG} \rightarrow \overline{V}_F$ is equivalent to the matrix operation $[R]^T\, \overline{V}_G \rightarrow \overline{V}_F$, where $[R]^T$ is the transpose of [R]. By inspecting Equation (26), it can be seen that $[R]^T$ is obtained simply by reversing the sign of H, as reflected in the flow chart that is shown in Figure 12.

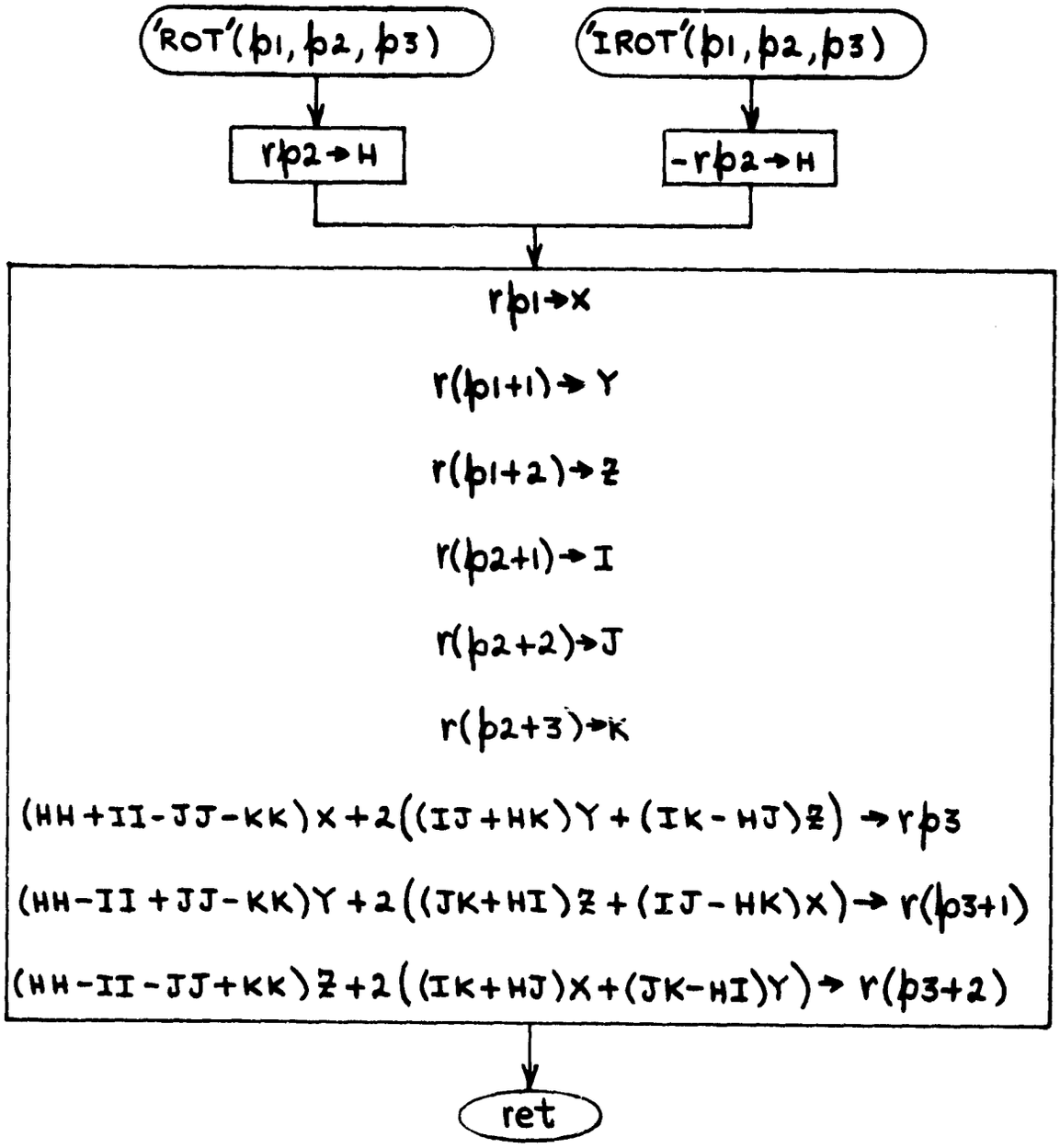


Figure 12. 'ROT'/'IROT' Subroutine Logic Flow

2.2.5 'QDOT'/'QXQ'/'QXQC'/'QCXQ'

This multiple entrypoint subroutine performs a variety of related quaternion operations. As in Section 2.2.4, let $\overline{\overline{q}}_{FG}$ represent a versor (unit quaternion) which defines the orientation of a Cartesian coordinate system G with respect to another system F that has the same origin. Let the vector $\overline{\omega}_G^{FG}$ represent the angular velocity (measured in system G) of system G with respect to system F. The 'QDOT' entrypoint is used to perform the operation symbolized by $(\overline{\overline{q}}_{FG} \circ \overline{\omega}_G^{FG})/2 \rightarrow \dot{\overline{\overline{q}}}_{FG}$, where $\dot{\overline{\overline{q}}}_{FG}$ represents the derivative of $\overline{\overline{q}}_{FG}$ with respect to time.

Let $\overline{\overline{P}}$ and $\overline{\overline{Q}}$ represent any two quaternions (not necessarily versors), and let $\tilde{\overline{\overline{P}}}$ and $\tilde{\overline{\overline{Q}}}$ represent their conjugates. The entrypoints 'QXQ', 'QXQC', and 'QCXQ' (respectively) perform the operations symbolized by $\overline{\overline{P}} \circ \overline{\overline{Q}} \rightarrow \overline{\overline{R}}$, $\overline{\overline{P}} \circ \tilde{\overline{\overline{Q}}} \rightarrow \overline{\overline{R}}$, and $\tilde{\overline{\overline{P}}} \circ \overline{\overline{Q}} \rightarrow \overline{\overline{R}}$.

2.2.5.1 Arguments

'QDOT' Entrypoint:

p1 = Address of the orientation versor $\overline{\overline{q}}_{FG}$.

p2 = Address of the angular velocity vector $\overline{\omega}_G^{FG}$.

p3 = Address of the output derivative $\dot{\overline{\overline{q}}}_{FG}$.

'QXQ'/'QXQC'/'QCXQ' Entrypoints:

p1 = Address of the first input quaternion $\overline{\overline{P}}$.

p2 = Address of the second input quaternion $\overline{\overline{Q}}$.

p3 = Address of the output quaternion $\overline{\overline{R}}$.

2.2.5.2 Examples of Usage

Assume the variable A has the value of 25, that the versor $\overline{\overline{q}}_{FG}$ resides in r50, r51, r52, r53, and that the vector $\overline{\omega}_G^{FG}$ resides in r54, r55, r56. Then

the statement cℓℓ 'QDOT'(A+25,A+29,A+125) will cause the value of $\dot{\overline{\overline{q}}}_{FG}$ to be stored in r150, r151, r152, r153. If the versor $\overline{\overline{q}}_{GH}$ resides in r0, r1, r2, r3, the statement cℓℓ 'QXQ'(A+25,0,4) will cause $\overline{\overline{q}}_{FH}$ to be stored in r4, r5, r6, r7. If this were followed by cℓℓ 'QXQC'(0,4,8), the versor $\overline{\overline{q}}_{GH} \circ \widetilde{\overline{\overline{q}}}_{FH} = \overline{\overline{q}}_{GH} \circ \overline{\overline{q}}_{HF} = \overline{\overline{q}}_{GF}$ would be stored in r8, r9, r10, r11. Then the statement cℓℓ 'QCXQ'(8,0,0) would cause a copy of the versor $\widetilde{\overline{\overline{q}}}_{GF} \circ \overline{\overline{q}}_{GH} = \overline{\overline{q}}_{FG} \circ \overline{\overline{q}}_{GH} = \overline{\overline{q}}_{FH}$ (already residing in r4, r5, r6, r7) to be stored in r0, r1, r2, r3.

2.2.5.3 Computations

The computational sequence depicted by the flow chart in Figure 13 results from a straightforward application of the rules of quaternion algebra that are defined in Appendix A.

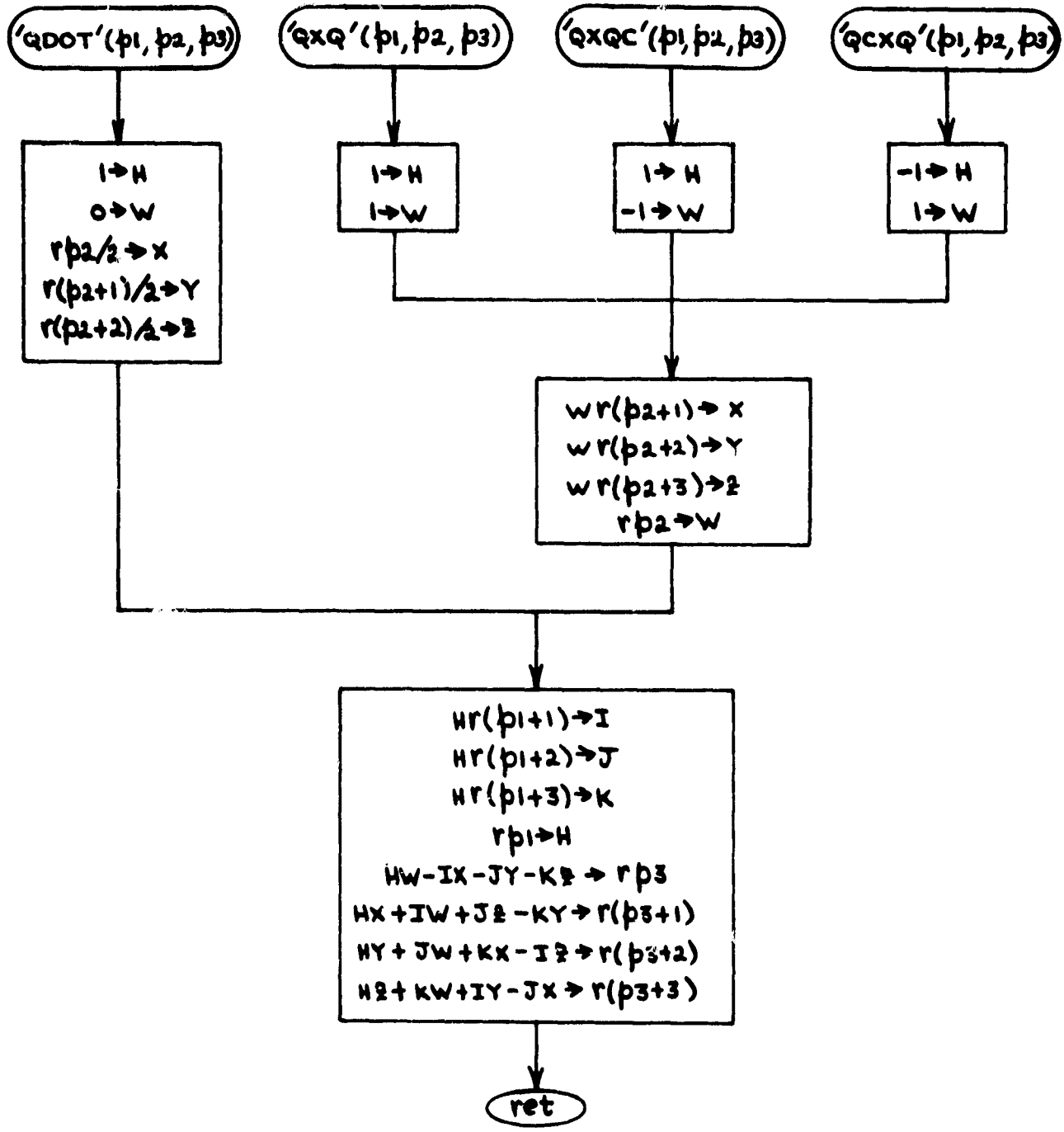


Figure 13. 'QDOT'/'QXQ'/'QXQC'/'QCXQ' Subroutine Logic Flow

2.2.6 'Q231'/'Q213'

The digits 1,2,3 in the 'Q231' and 'Q213' entrypoint names identify coordinate-system rotations about the X,Y,Z axes (respectively) in two different Euler-angle sequences. Generally, rotation about the X axis is designated as roll, rotation about the Y axis is designated as pitch, and rotation about the Z axis is designated as yaw. Given an orientation versor $\overline{\overline{q}}_{FG}$, the 'Q231' entrypoint is used to calculate the values of the pitch, yaw, roll angles (taken in that sequence) that are required to rotate the axes of coordinate system F into coincidence with those of system G. The 'Q213' entrypoint is used to calculate the corresponding Euler angle set for a pitch, roll, yaw sequence.

2.2.6.1 Arguments

p1 = Address of the orientation versor.

p2 = Address of the output array of Euler angles (measured in radians, and stored in the same order as the logical rotation sequence).

2.2.6.2 Example of Usage

If the orientation versor under consideration resides in r0, r1, r2, r3, the statement *cll 'Q231'(0,4)* would cause the first rotation (pitch) angle to be stored in r4, the second rotation (yaw) angle to be stored in r5, and the third rotation (roll) angle to be stored in r6. The statement *cll 'Q213(0,0)* would cause the rotation angles of a pitch, roll, yaw sequence to be stored in r0, r1, r2 (thus wiping out the first three conponents of the orientation versor).

2.2.6.3 Computations

The output of the 'Q231'/'Q213' subroutine is an ordered set of Euler angles

α, β, γ. For a pitch, yaw, roll sequence the Euler angles are related to the orientation versor $\bar{\bar{q}}_{FG} = H + \hat{i}\, I + \hat{j}\, J + \hat{k}\, K$ by the equations (see Reference 10)

$$H = C_\alpha\, C_\beta\, C_\gamma - S_\alpha\, S_\beta\, S_\gamma, \tag{27}$$

$$I = C_\alpha\, C_\beta\, S_\gamma + S_\alpha\, S_\beta\, C_\gamma, \tag{28}$$

$$J = S_\alpha\, C_\beta\, C_\gamma + C_\alpha\, S_\beta\, S_\gamma, \tag{29}$$

and

$$K = C_\alpha\, S_\beta\, C_\gamma - S_\alpha\, C_\beta\, S_\gamma, \tag{30}$$

where $C_\alpha = \cos \tfrac{1}{2}\alpha$, $S_\alpha = \sin \tfrac{1}{2}\alpha$, $C_\beta = \cos \tfrac{1}{2}\beta$, etc. Solving equations (27) - (30) for α, β, γ results in

$$\tfrac{1}{2}(\alpha + \gamma) = \tan^{-1}[(J + I)/(H + K)], \tag{31}$$

$$\tfrac{1}{2}(\alpha - \gamma) = \tan^{-1}[(J - I)/(H - K)], \tag{32}$$

$$\beta = \tan^{-1}[2\,(HK + IJ)/\sqrt{X^2 + Y^2}], \tag{33}$$

where

$$X = H^2 - I^2 + J^2 - K^2, \tag{34}$$

and

$$Y = 2\,(JK - HI). \tag{35}$$

As reflected by the flow chart shown in Figure 14, Equations (31) - (35) are used to compute the Euler angles when the 'Q231' entrypoint is used.

For a pitch, yaw, roll sequence, the orientation versor and the Euler angles are related by the equations

$$H = C_\alpha C_\beta C_\gamma + S_\alpha S_\beta S_\gamma, \tag{36}$$

$$I = C_\alpha S_\beta C_\gamma + S_\alpha C_\beta S_\gamma, \tag{37}$$

$$J = S_\alpha C_\beta C_\gamma - C_\alpha S_\beta S_\gamma, \tag{38}$$

and

$$K = C_\alpha C_\beta S_\gamma - S_\alpha S_\beta C_\gamma. \tag{39}$$

The Euler-angle solutions in this case are

$$\tfrac{1}{2}(\alpha - \gamma) = \tan^{-1}[(J - K)/(H + I)], \tag{40}$$

$$\tfrac{1}{2}(\alpha + \gamma) = \tan^{-1}[(J + K)/(H - I)], \tag{41}$$

$$\beta = \tan^{-1}[2(HI - JK)/\sqrt{X^2 + Y^2}], \tag{42}$$

where

$$X = H^2 - I^2 + J^2 - K^2 \tag{43}$$

and

$$Y = 2(IJ + HK). \tag{44}$$

It can be seen that Equations (31) - (35) become identical with (40) - (44) when

(a) -K is substituted for I,

(b) I is substituted for K, and

(c) the sign of γ is reversed in the first set of equations.

This characteristic of the two Euler sequences allows common logic to be used for the major portion of the computations, as illustrated in Figure 14.

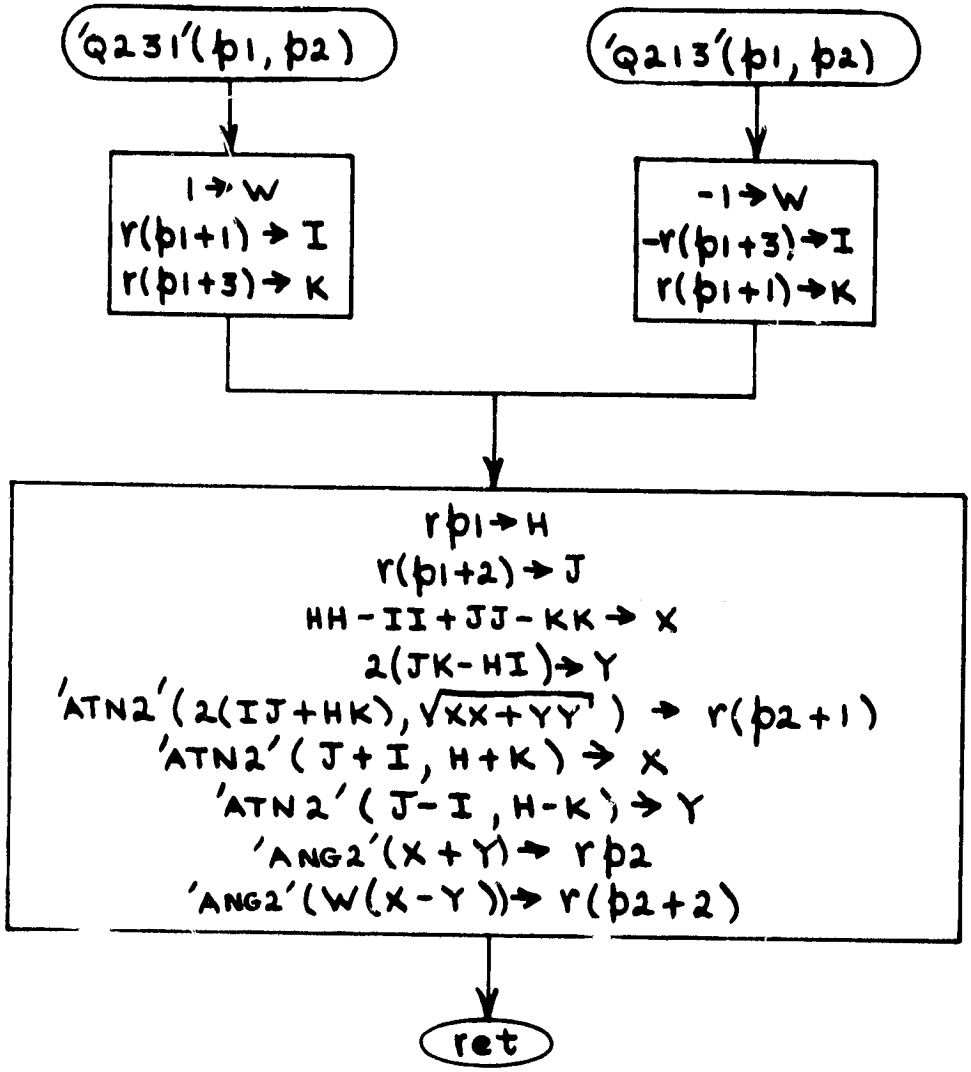


Figure 14. 'Q231'/'Q213' Subroutine Logic Flow

2.2.7 'Q313'

Given an orientation versor $\overline{\overline{q}}_{FG}$, the 'Q313' subroutine calculates a set of Euler angles (α,β,γ) that corresponds to a rotation of coordinate system F into coincidence with system G, assuming F is rotated sequentially about its Z, X, Z axes.

2.2.7.1 Arguments

p1 = Address of the orientation versor.

p2 = Address of the output array of Euler angles (measured in radians, and stored in the same order as the logical rotation sequence).

2.2.7.2 Example of Usage

Suppose that coordinate system F is a geocentric inertial system defined such that the first point in Aries lies on the X_F axis, and the earth's angular momentum vector lies along the Z_F axis. Let G be another geocentric system defined such that the instantaneous position of some satellite lies on the X_G axis, and such that the angular momentum vector of the satellite CG lies along the Z_G axis. Let the versor $\overline{\overline{q}}_{FG}$ reside in r0, r1, r2, r3. Then the statement cℓℓ 'Q313'(0,4) would cause the right ascension of the ascending node of the satellite orbit to be stored in r4, the inclination of the orbit to be stored in r5, and the argument of latitude corresponding to the satellite position to be stored in r6.

2.2.7.3 Computations

The Euler angles are related to the versor $\overline{\overline{q}}_{FG} = H + \hat{i}\, I + \hat{j}\, J + \hat{k}\, K$ by the equations

$$H = \cos \tfrac{1}{2} (\alpha + \gamma) \cos \tfrac{1}{2} \beta, \tag{45}$$

$$I = \cos \tfrac{1}{2} (\alpha - \gamma) \sin \tfrac{1}{2} \beta, \tag{46}$$

$$J = \sin \tfrac{1}{2} (\alpha - \gamma) \sin \tfrac{1}{2} \beta, \tag{47}$$

and

$$K = \sin \tfrac{1}{2} (\alpha + \gamma) \cos \tfrac{1}{2} \beta \tag{48}$$

(see Reference 10). It follows that

$$\tfrac{1}{2}\beta = \tan^{-1} \left(\sqrt{II+JJ}, \sqrt{HH+KK}\right), \tag{49}$$

$$\tfrac{1}{2} (\alpha + \gamma) = \tan^{-1} (K,H), \tag{50}$$

and

$$\tfrac{1}{2} (\alpha - \gamma) = \tan^{-1} (J,I). \tag{51}$$

A flow chart of the 'Q313' logic is shown in Figure 15.

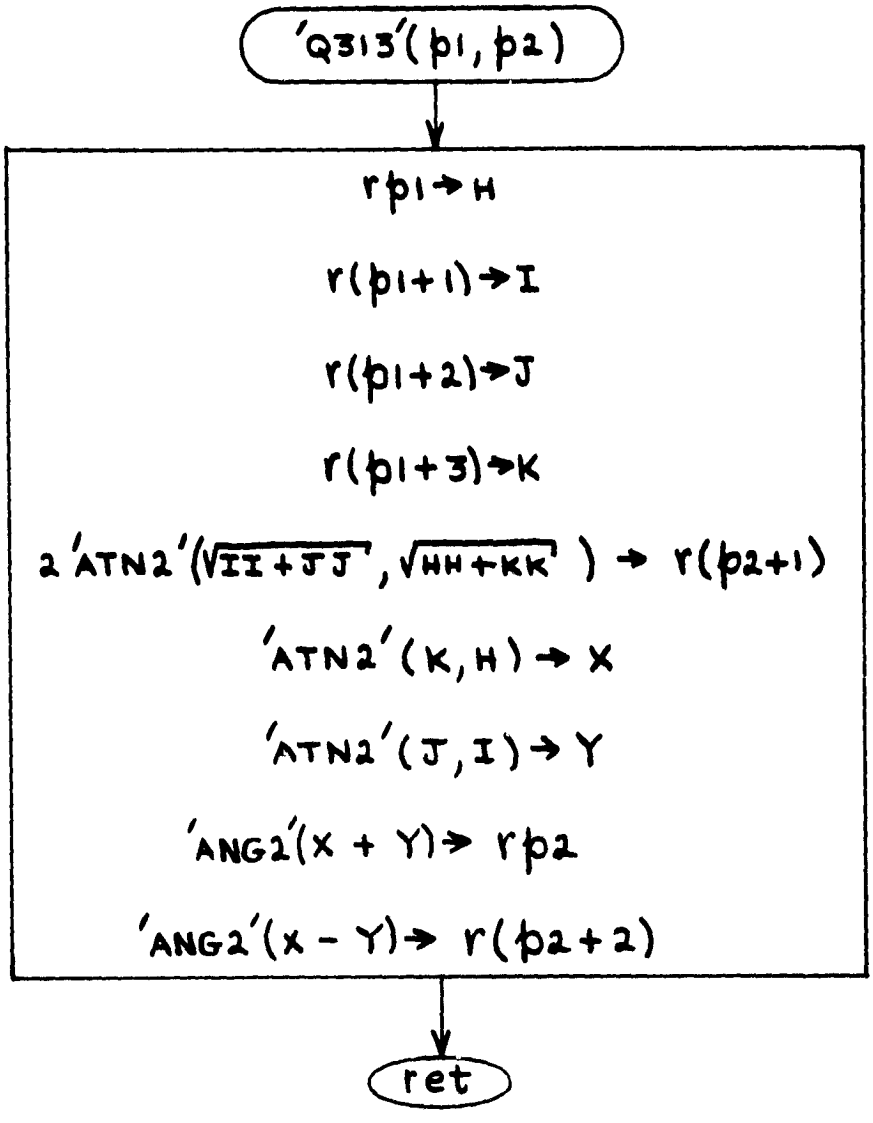


Figure 15. 'Q313' Subroutine Logic Flow

2.2.8 '231Q'/'213Q'

The '231Q'/'213Q' subroutine performs the inverse function of the 'Q231'/'Q213' subroutine described in Section 2.2.6. That is to say, given an ordered set of Euler angles α, β, γ, it calculates the associated orientation versor $\bar{\bar{q}}_{FG}$. The '231Q' entrypoint is used when the rotation sequence is pitch, yaw, roll, and the '213Q' entrypoint is used when the sequence is pitch, roll, yaw.

2.2.8.1 Arguments

p1 = Address of the input array of Euler angles (measured in radians, and stored in the same order as the logical rotation sequence).

p2 = Address of the orientation versor.

2.2.8.2 Example of Usage

If r1, r2, r3 contain the pitch, yaw, and roll angles that define a certain orientation of the Orbiter's body axes B relative to its local-vertical frame G, then the statement *cll* '231Q'(1,0) in the calling routine would cause the orientation versor $\bar{\bar{q}}_{GB}$ to be stored in r0, r1, r2, r3 (thus destroying the input values of the Euler angles, plus whatever value may have been stored in r0 before calling the subroutine). If the variable A has been assigned the value 100, and if r4, r5, r6 contain the pitch, roll, yaw angles that define the orientation of system G with respect to another system F, the statement *cll* '213Q'(4,A+10) would cause the versor $\bar{\bar{q}}_{FG}$ to be stored in r110, r111, r112, r113.

2.2.8.3 Computations

The computational sequence defined by the flow chart in Figure 16 is a straightforward implementation of Equations (27) - (30) and (36) - (39) from

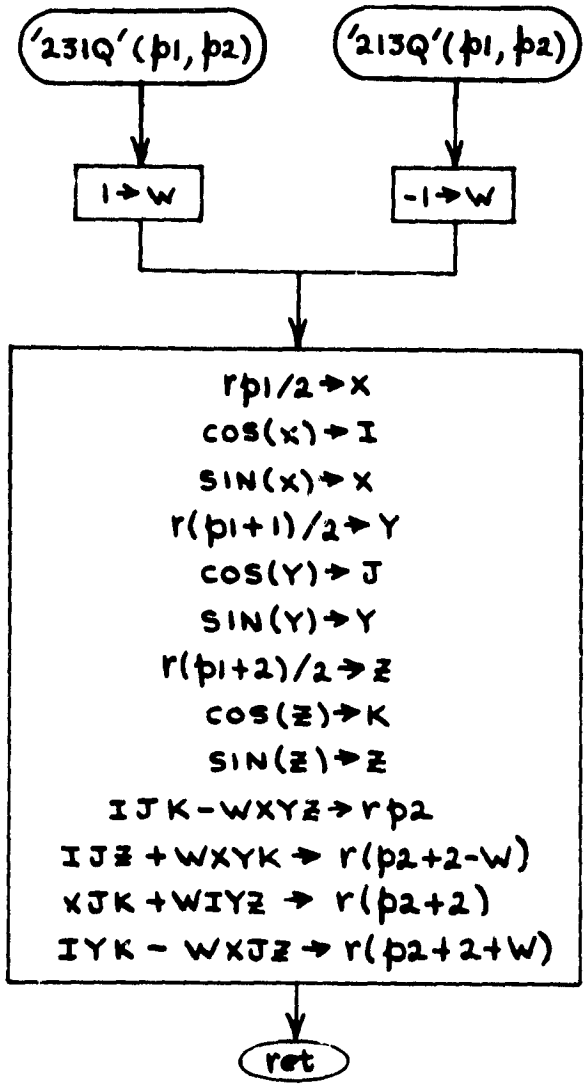


Figure 16. '231Q'/'213Q' Subroutine Logic Flow

2.2.9 '313Q'

The '313Q' subroutine performs the inverse function of the 'Q313' subroutine described in Section 2.2.7. Given an ordered set of angles α, β, γ that defines the orientation of a coordinate system G with respect to another system F in terms of sequential rotations of F about its Z, X, Z axes, the '313Q' subroutine computes the associated versor $\bar{\bar{q}}_{FG}$.

2.2.9.1 Arguments

p1 = Address of the input array of Euler angles (measured in radians, and stored in the same order as the rotation sequence).

p2 = Address of orientation versor.

2.2.9.2 Example of Usage

Let the F and G coordinate systems be defined as in Section 2.2.7. Let the right ascension of the satellite orbit reside in r4, the inclination of the orbit in r5, and the satellite's argument of latitude in r6. Then the statement c*ll* '313Q'(4,0) would cause the orientation versor $\bar{\bar{q}}_{FG}$ to be stored in r0, r1, r2, r3.

2.2.9.3 Computations

The computational sequence defined by the flow chart in Figure 17 is a straightforward implementation of Equations (45) - (48) from Section 2.2.7.

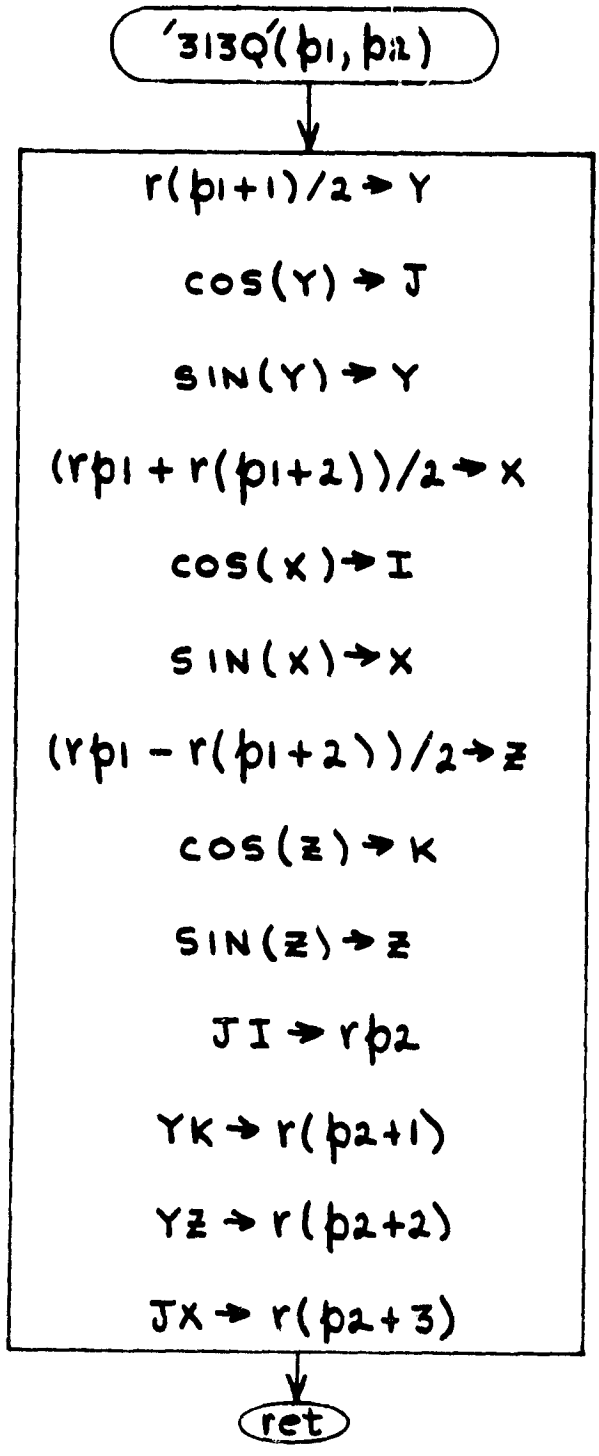


Figure 17. '313Q' Subroutine Logic Flow

2.2.10 'IMATQ'

Given the inverse

$$[R]^{-1} = [R]^T = [T] = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} \tag{52}$$

of the coordinate transformation matrix [R] defined by Equation (26) in Section 2.2.4, the 'IMATQ' subroutine computes the associated orientation versor

$$\overline{\overline{q}}_{FG} = q_0 + \hat{i}\, q_1 + \hat{j}\, q_2 + \hat{k}\, q_3. \tag{53}$$

2.2.10.1 Arguments

p1 = Address of the inverse transformation matrix, stored (columnwise) in the order T_{11}, T_{21}, T_{31}, T_{12}, T_{22}, T_{32}, T_{13}, T_{23}, T_{33}.

p2 = Address of the output versor.

2.2.10.2 Example of Usage

Let F represent an ECI coordinate system, defined such that the first point in Aries lies on the X_F axis, and the earth's angular momentum vector is aligned with the Z_F axis. Let G represent another geocentric coordinate system defined such that the X_G axis is aligned with the geocentric position vector $\overline{R}$ of a satellite, and such that the Z_G axis is aligned with $\overline{R} \times \overline{V}$, where $\overline{V}$ is the geocentric inertial velocity vector of the satellite. Suppose that $\overline{R}$ and $\overline{V}$ are known in terms of their components in F, and that it is desired to evaluate $\overline{\overline{q}}_{FG}$.

The inverse transformation matrix can be expressed as

$$[T] = [\overline{U}_F^{GX} \mid \overline{U}_F^{GY} \mid \overline{U}_F^{GZ}]$$

where $\overline{U}_F^{GX} = \overline{R}_F/|\overline{R}_F|$, $\overline{U}_F^{GZ} = (\overline{R}_F \times \overline{V}_F)/|\overline{R}_F \times \overline{V}_F|$, and $\overline{U}_F^{GY} = \overline{U}_F^{GZ} \times \overline{U}_F^{GX}$. If $\overline{R}_F$ resides in r13, r14, r15, the statement c*ll* 'SXV'(1,√('DOTP'(13,13)),13,0) will cause $\overline{U}_F^{GX}$ to be stored in r0, r1, r2. If $\overline{V}_F$ resides in r16, r17, r18, the statement c*ll* 'CRSP'(13,16,6); c*ll* 'SXV'(1,√('DOTP'(6,6)),6,6) will cause $\overline{U}_F^{GZ}$ to be stored in r6, r7, r8, and the statement c*ll* 'CRSP'(6,0,3) will then cause $\overline{U}_F^{GY}$ to be stored in r3, r4, r5, thus completing the columnwise storage of [T] in r0 through r8. The statement c*ll* 'IMATQ'(0,9) will then cause $\overline{\overline{q}}_{FG}$ to be stored in r9, r10, r11, r12.

2.2.10.3 Computations

By comparing Equations (26) and (52), it can be seen that [T] can be expressed, in terms of the versor components appearing in Equation (53), as

$$[T] = \begin{bmatrix} (q_0^2+q_1^2-q_2^2-q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & (q_0^2-q_1^2+q_2^2-q_3^2) & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & (q_0^2-q_1^2-q_2^2+q_3^2) \end{bmatrix}. \qquad (54)$$

Taking cognizance of the facts that $q_0^2 + q_1^2 + q_2^2 + q_3^2 \equiv 1$, and that one (but only one) of the versor components can be given an arbitrary sign, four different solutions for the values of $q_k (k = 0,1,2,3)$ can be found in terms of the $T_{ij} (i,j = 1,2,3)$. These solutions are given by the equations

$$q_0 = \sqrt{1 + T_{11} + T_{22} + T_{33}}/2 \qquad (54a)$$

$$q_1 = (T_{32} - T_{23})/4\, q_0 \qquad (54b)$$

$$q_2 = (T_{13} - T_{31})/4\, q_0 \tag{54c}$$

$$q_3 = (T_{21} - T_{12})/4\, q_0 \tag{54d}$$

or

$$q_1 = \sqrt{1 + T_{11} - T_{22} - T_{33}}/2 \tag{55a}$$

$$q_2 = (T_{21} + T_{12})/4\, q_1 \tag{55b}$$

$$q_3 = (T_{13} + T_{31})/4\, q_1 \tag{55c}$$

$$q_0 = (T_{32} - T_{23})/4\, q_1 \tag{55d}$$

or

$$q_2 = \sqrt{1 - T_{11} + T_{22} - T_{33}}/2 \tag{56a}$$

$$q_3 = (T_{32} + T_{23})/4\, q_2 \tag{56b}$$

$$q_0 = (T_{13} - T_{31})/4\, q_2 \tag{56c}$$

$$q_1 = (T_{21} + T_{12})/4\, q_2 \tag{56d}$$

or

$$q_3 = \sqrt{1 - T_{11} - T_{22} + T_{33}}/2 \tag{57a}$$

$$q_0 = (T_{21} - T_{12})/4\, q_3 \tag{57b}$$

$$q_1 = (T_{13} + T_{31})/4\, q_3 \tag{57c}$$

$$q_2 = (T_{32} + T_{23})/4\, q_3. \tag{57d}$$

Since any versor component (or as many as three of them) may turn out to be zero, numerical tests are necessary to determine which one of the alternate computational sequences defined by the preceding equations can be used in a given case. It is known from the versor identity $q_0^2 + q_1^2 + q_2^2 + q_3^2 \equiv 1$ that at least one component must have a magnitude as great as ½, and that none can have a magnitude greater than unity. As defined by the flow chart shown in Figure 18, the 'IMATQ' logic tests the magnitude of each versor component in sequence until one is found that has a magnitude at least as great as ½. As soon as a component q_N is found that satisfies the inequality $4\, q_N^2 \geq 1$, it is arbitrarily given a positive sign, and the remainder of the components are calculated by using the equations that contain $4\, q_N$ in the denominator.

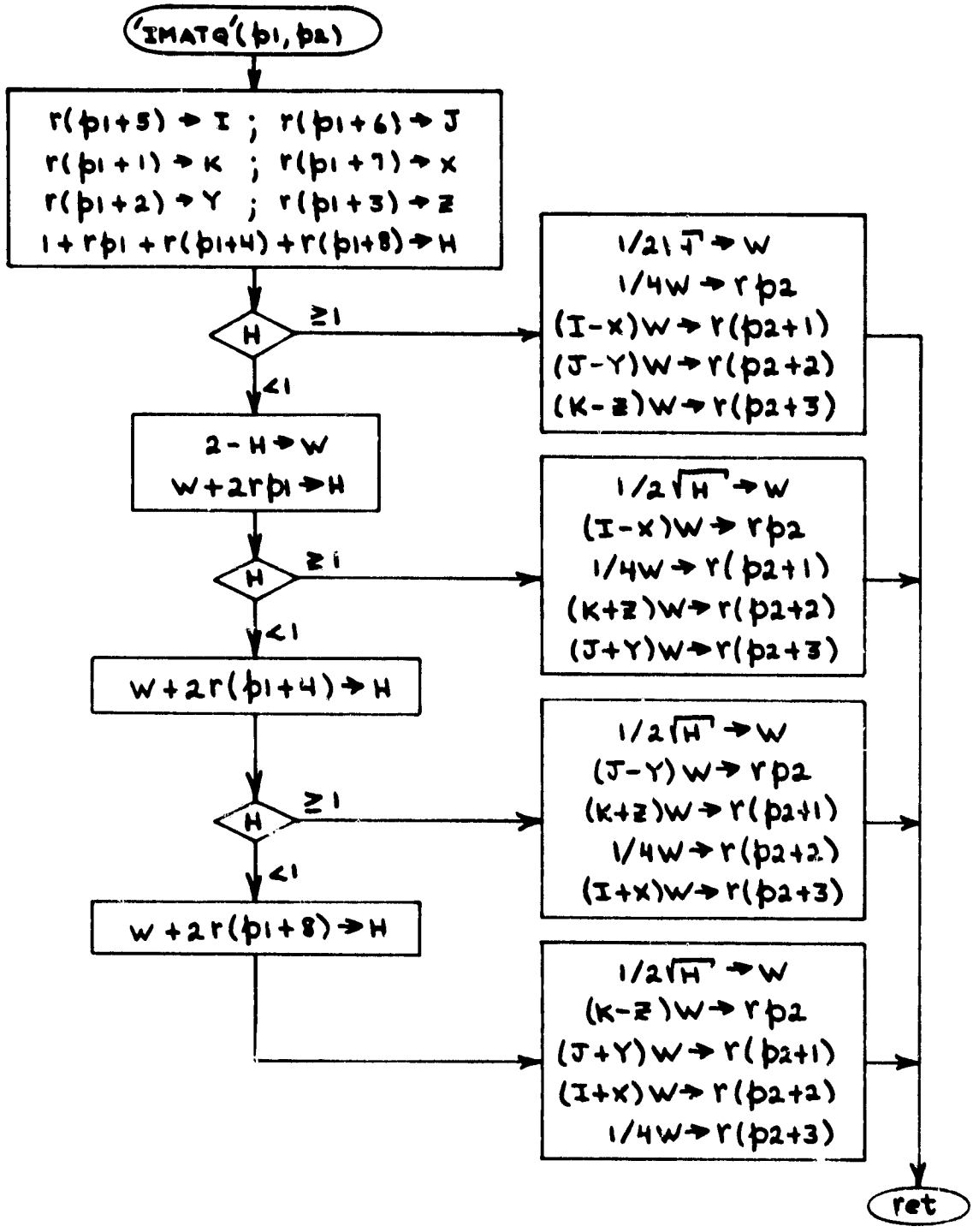


Figure 18. 'IMATQ' Subroutine Logic Flow

2.3 STATE VECTOR DERIVATIVES SUBROUTINE ('DERIVS')

The 'DERIVS' subroutine computes the first derivative (with respect to time) of each quantity contained in an extended array of state variables that includes, in addition to the translational and rotational state vectors defined in Section 1.2, such things as propellant expenditures and control torque integrals. The memory allocation table contained in Appendix D lists the specific quantities that are included in the Shuttle state variable array (registers r25-r49) and the payload state variable array (registers r50-r74).

Containing mathematical models of the orbital flight environment and of all vehicle systems affecting the dynamics of the Shuttle and the payload, the 'DERIVS' subroutine represents the heart of the #TRAJ processor.

2.3.1 Input Data

2.3.1.1 Argument List

The 'DERIVS' argument list consists of a single parameter (pl) which is assigned a value of 0 or 1 by the calling routine according to whether state variable derivatives are to be computed for the Shuttle or the payload.

2.3.1.2 r-Registers

The contents of the following r-registers (see Appendix D) are used as input quantities in the 'DERIVS' calculations:

Shuttle Computations (pl = 0)	Payload Computations (pl = 1)
r23, r24	r23, r24
r25-r39	r50-r64, r74
r75-r86	r100-r111
r91-r99, r124	r116-r124

2.3.1.3 Array Variables

The appropriate 12x12 RCS/DAP response matrix (see Section 3) must reside in the HPL array variable R[*] (represented symbolically as [R] in the following flow charts) before calling 'DERIVS' for Shuttle derivatives (i.e., with pl = 0).

When SRM thrusting is to be simulated, the name of the disk file containing the appropriate SRM thrust table must have been assigned to file no. 3 and the array variable A[*] (see Figure 19d for contents) must have been initialized before calling DERIVS for payload derivatives (i.e., with pl = 1). The contents of the A[*] array (symbolically, [A]) are updated as required in 'DERIVS' by reading new thrust-profile coordinates from file no. 3 as the burn progresses.

2.3.2 Output Data

As indicated in Appendix D , the r-registers have been allocated so that the address of every state derivative can be found simply by adding 100 to the address of the corresponding state variable. Thus, Shuttle state derivatives are stored in r125-r149 when pl = 0, and payload state derivatives are stored in registers r150-r174 when pl = 1. When pl = 0, the contents of r285 (normally zero) will be set equal to 1 if 'DERIVS' finds the Shuttle RCS inadequate to maintain the commanded attitude.

Sometimes the calling routine requires, in addition to the state variable derivatives, the angular velocity vector ($\bar{\omega}_G^{IG}$ or $\bar{\omega}_g^{Ig}$) of the local-vertical coordinate system. This vector resides in the volatile registers r1-r3 when execution control is returned from 'DERIVS' to the calling routine. Another vector sometimes needed by the calling routine is the linear acceleration vector ($\bar{A}_G$ or $\bar{a}_g$), which resides in r7-r9 upon the return from 'DERIVS'.

2.3.3 Computations

The computations performed by 'DERIVS' are defined by the flow chart shown in Figure 19a through 19g. For the most part, the computations are described in terms of logical symbols (as opposed to the HPL variable names and addresses that appear in the code.) The major portion of the computational logic applies equally to Shuttle and to payload calculations. To minimize confusion in following the flow of the logic, Shuttle symbols (e.g., $\overline{A}_G$ rather than $\overline{a}_g$) are used almost exclusively throughout the flow chart, even in those sections of the logic that apply uniquely to the payload. The only exceptions to this general rule occur when there is no Shuttle-related equivalent of a payload-related variable (e.g., ℓ_{1+}, ℓ_{2+}, etc.).

Insofar as practicable, the order of computation in the HPL code (Section C.1) is arranged to follow that shown in the flow chart. Even so, Appendix D, which shows the correlation between logical symbols and r-register numbers, is indispensable to the understanding of the code.

Following is a summary of all the data registers that will or may be modified by an execution of 'DERIVS':

Shuttle Computations (pl = 0)	Payload Computations (pl = 1)
r0-r18	r0-r18
r87-r90	r112-r115
r125-r149	r150-r174
r285	A[5]-A[8]
A-Z	A-Z

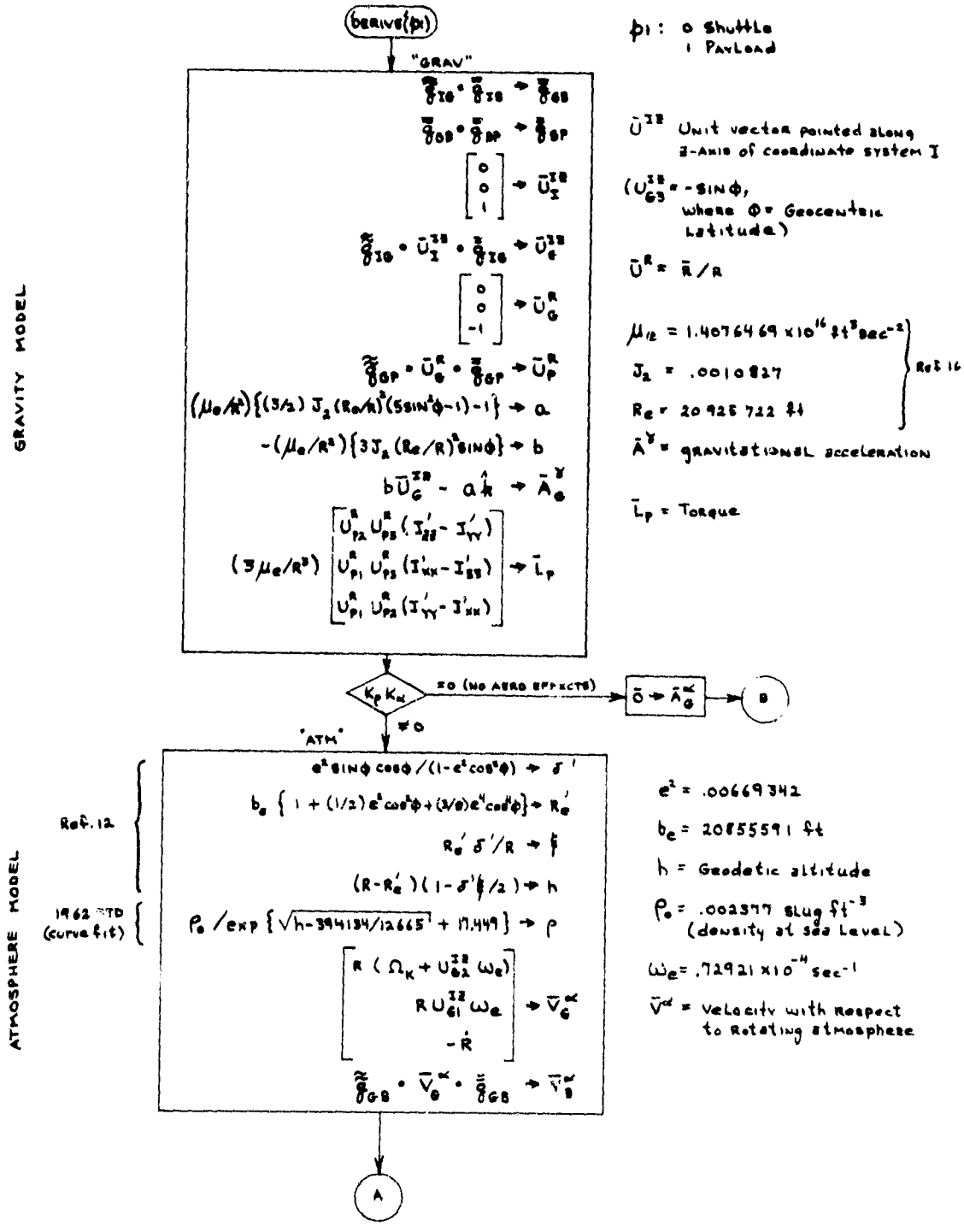


Figure 19a. 'DERIVS' Subroutine Logic Flow

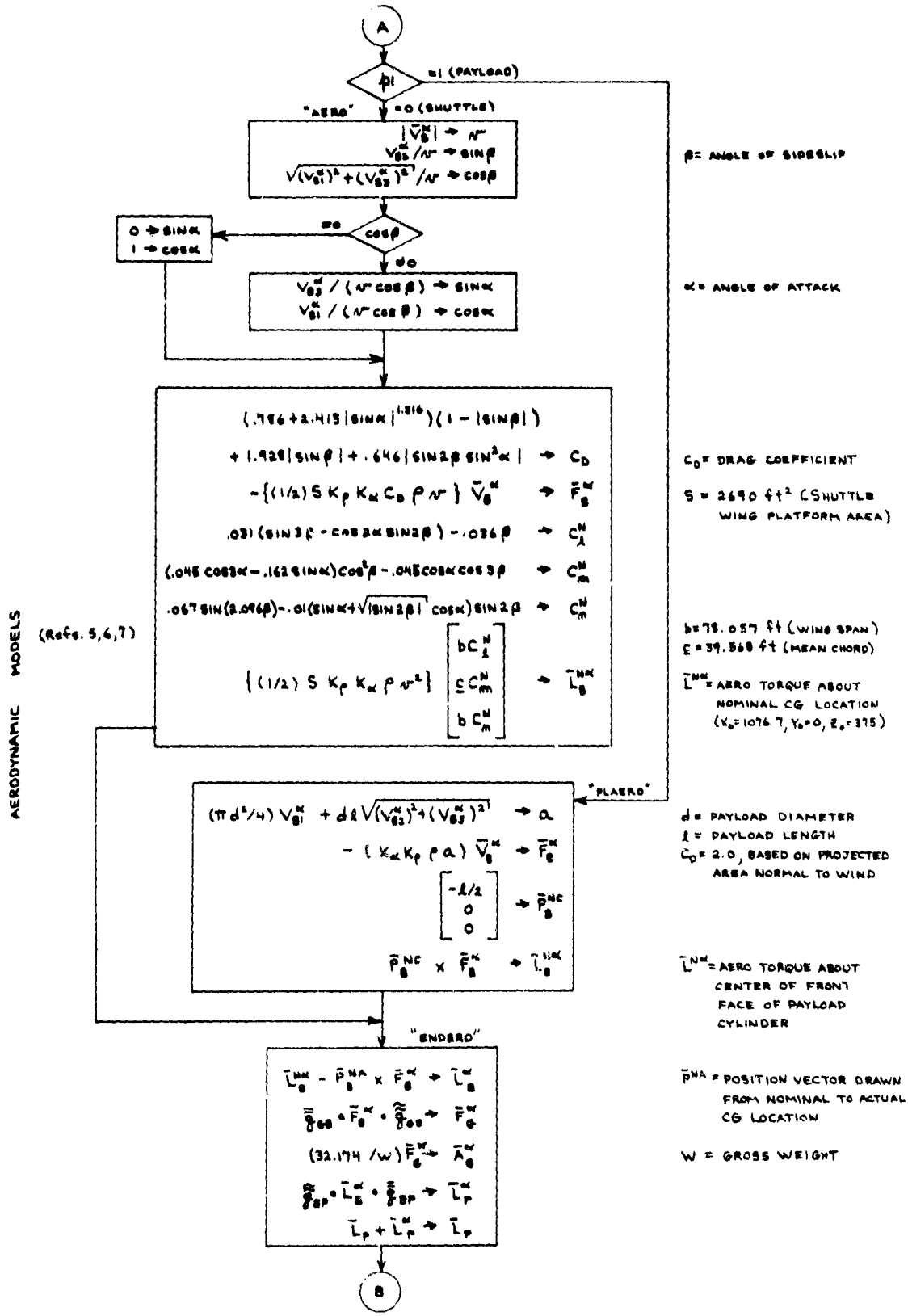


Figure 19b. 'DERIVS' Subroutine Logic Flow (cont.)

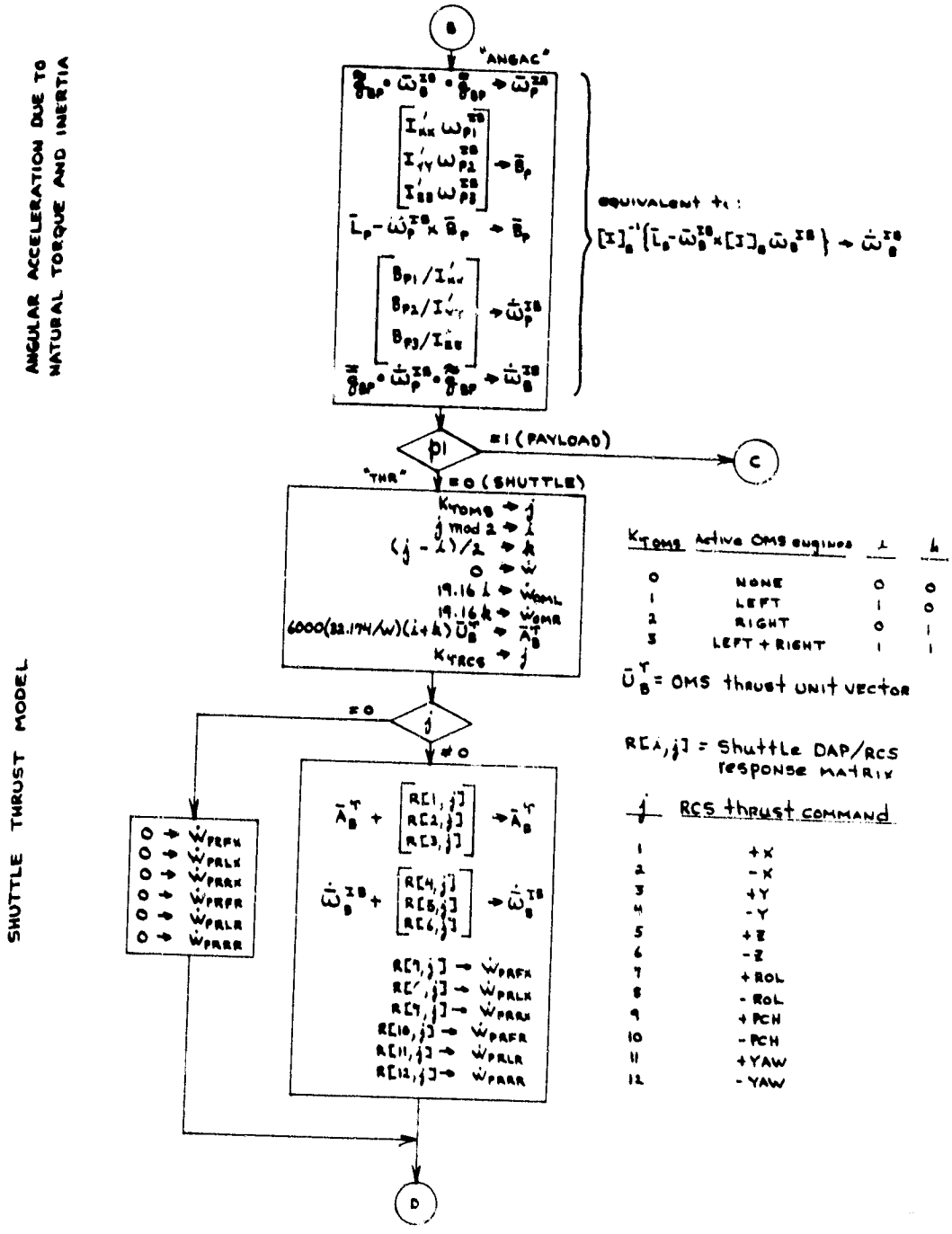


Figure 19c. 'DERIVS' Subroutine Logic Flow (cont.)

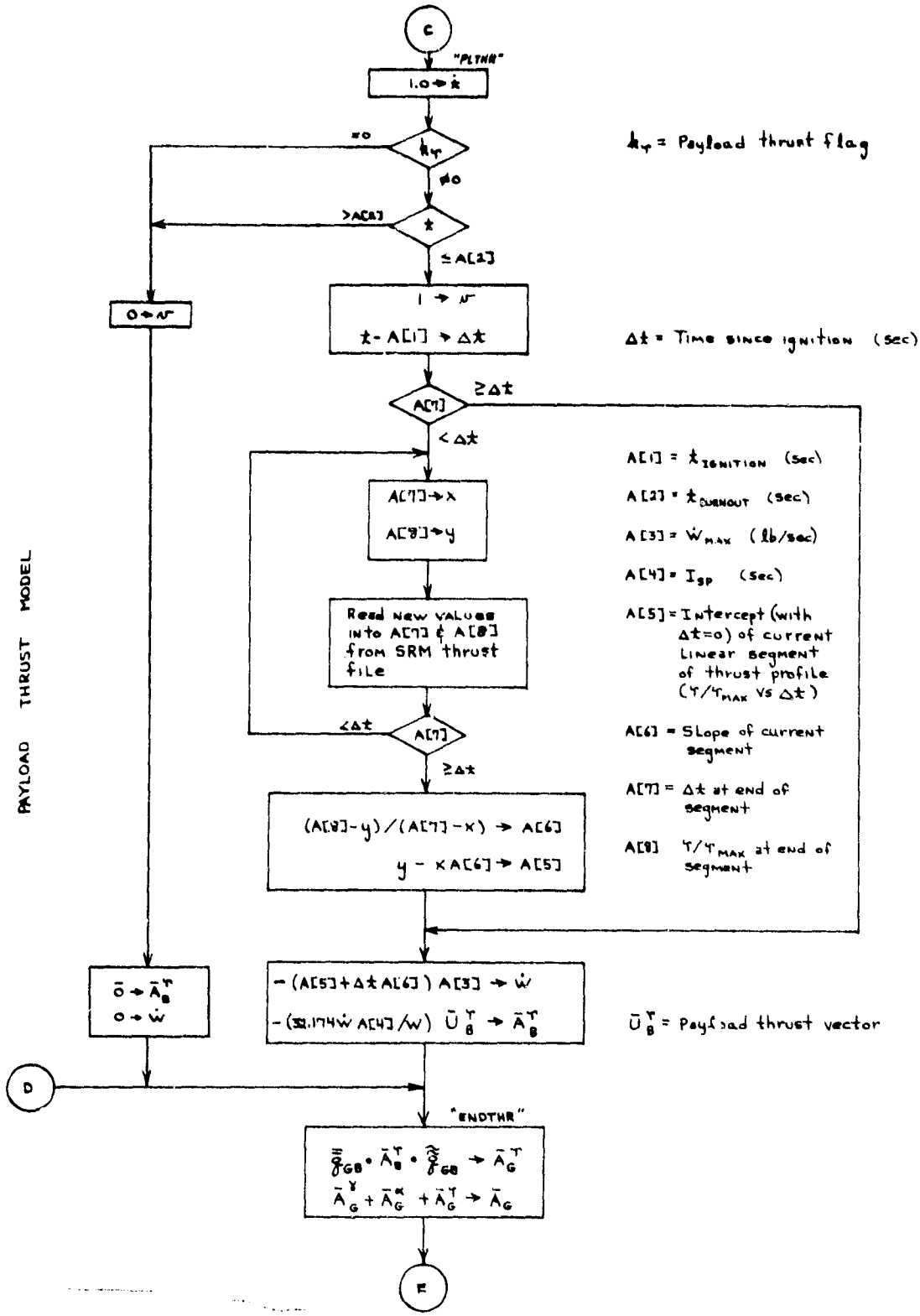


Figure 19d. 'DERIVS' Subroutine Logic Flow (cont.)

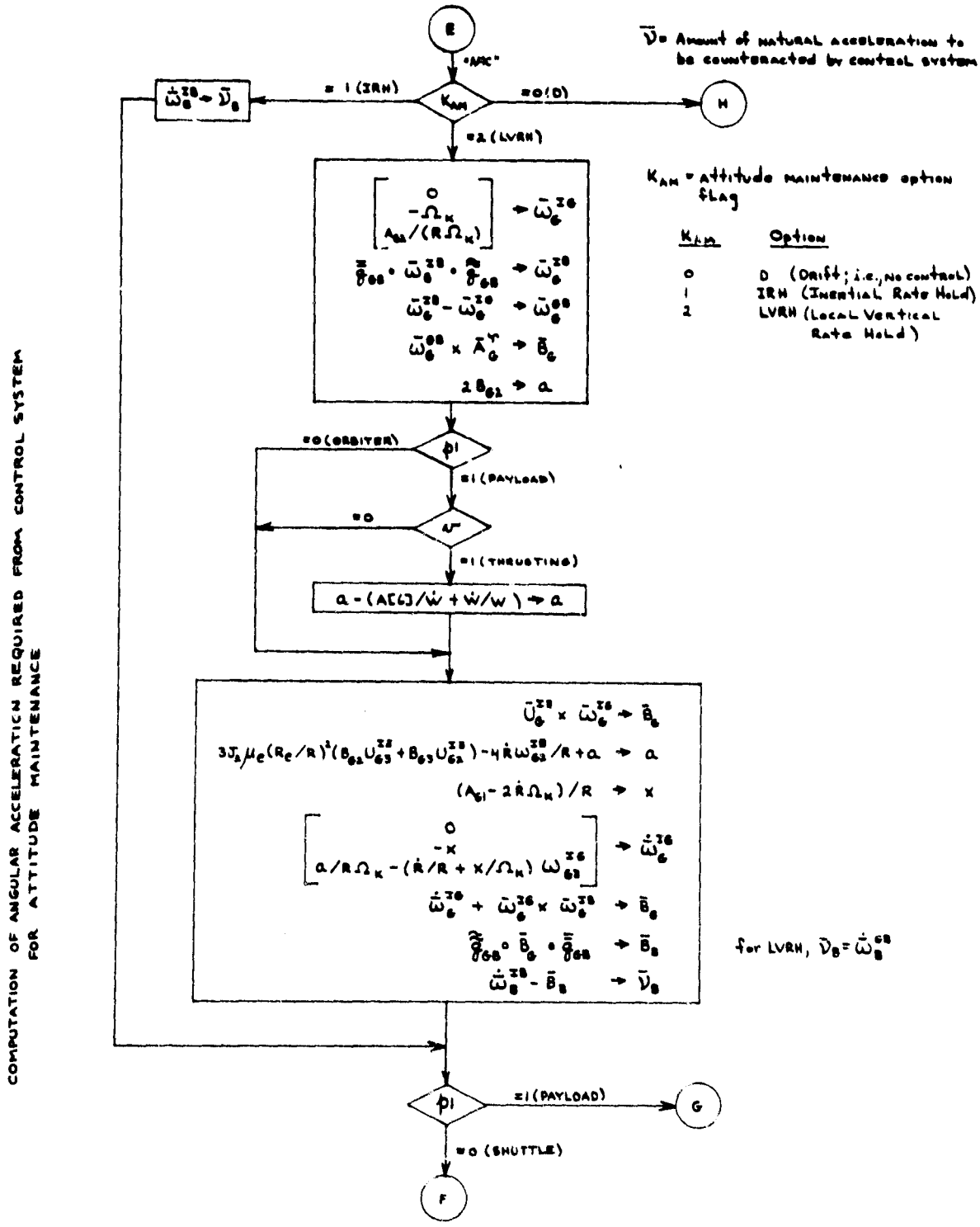


Figure 19e. 'DERIVS' Subroutine Logic Flow (cont.)

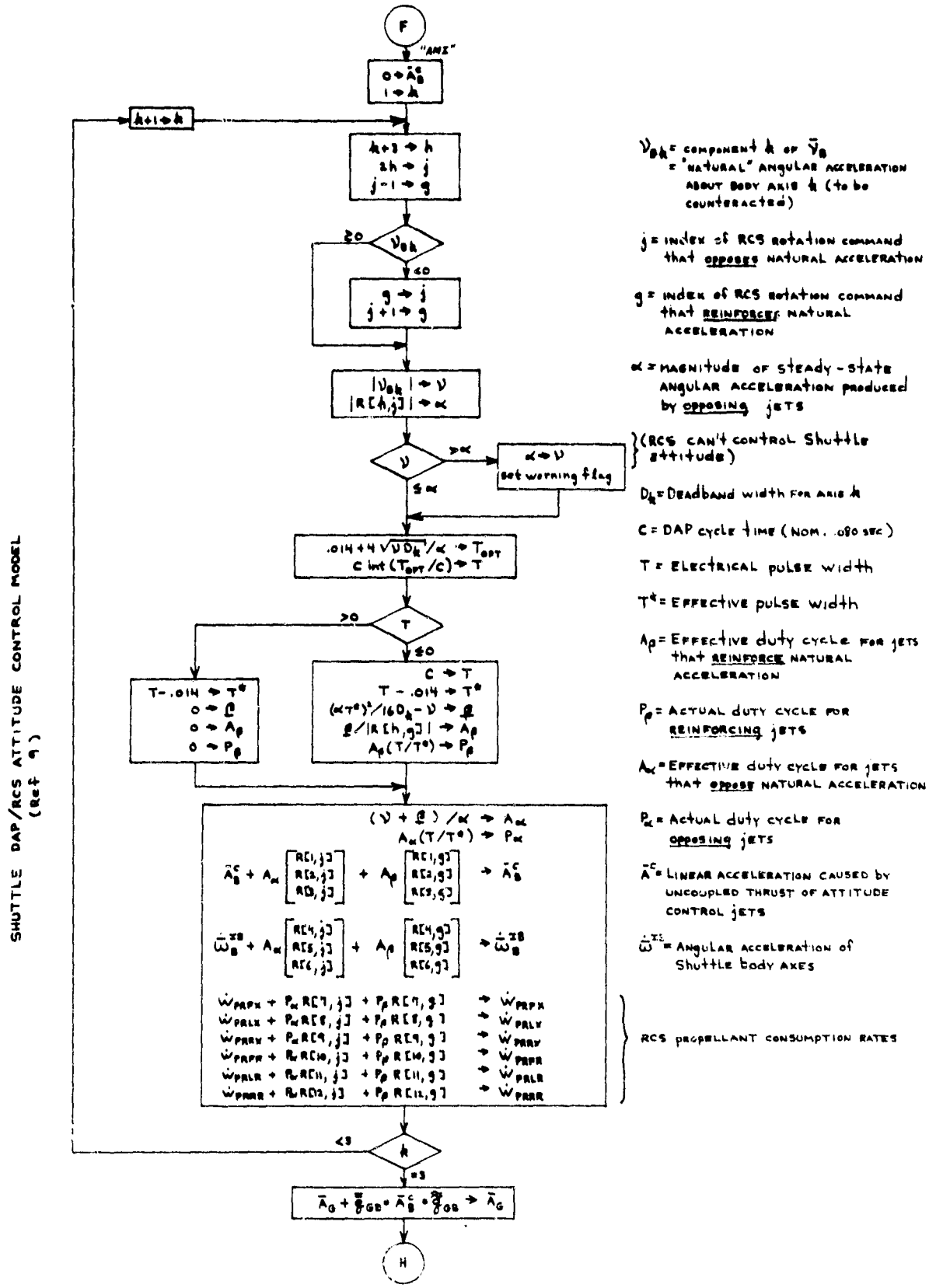


Figure 19f. 'DERIVS' Subroutine Logic Flow (cont.)

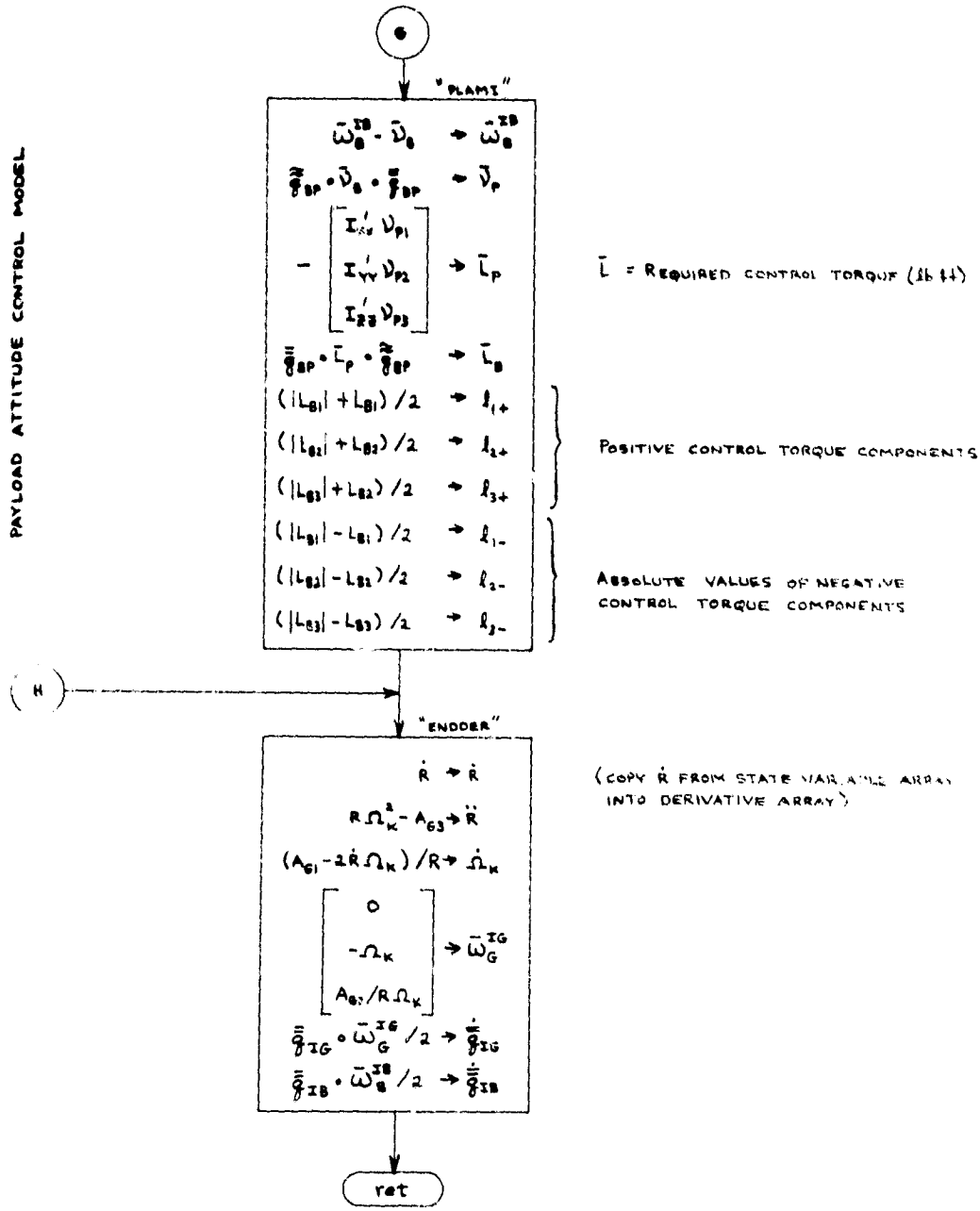


Figure 19g. 'DERIVS' Subroutine Logic Flow (cont.)

2.4 MAIN LOGIC FLOW

The "main logic" of the #TRAJ link consists of nothing more than an instruction that causes the %RMAT link (Section 3) to be appended to #TRAJ.

3. RESPONSE MATRIX COMPUTATION LINK (%RMAT)

The function of the %RMAT link is to compute matrices that describe the response of the Shuttle to DAP/RCS commands, and to store the matrices in designated disk files for subsequent use during trajectory integration.

3.1 INPUT DATA

3.1.1 Jet Force Table

Basic inputs to the response matrix computations include the data shown in Table 1, which reside in the disk files named "$JFT" and "$JFTM". The "$JFTM" file contains the mnemonic identification symbols (shown in the second column of Table 1) for the Orbiter's 44 RCS jets, as defined in Figure 4.3.2.-2 of Reference 11. The "$JFT" file contains the body-fixed thrust components and the station coordinates of the thrust application point (which together define a torque component normal to the thrust line) for each thruster, along with a factor C (in the last column of Table 1) whose product with the thrust vector defines a component of torque parallel to the thrust line. This latter component of torque (parallel to the thrust line) is the result of RCS jet plume impingement on the exterior surfaces of the Orbiter. Table 1 results from combining the basic RCS jet data shown in Table 2 with the plume impingement data shown in Table 3, as explained in Reference 9.

3.1.2 Jet Select Tables

For each flight profile segment, the HFRMP user must specify which one of three basic combinations of thrusters is to be used for attitude and/or translational control of the Shuttle. The available options are designated V (vernier jets), P (primary jets), and PZI (primary jets with +Z thrusters

Table 1. RCS Thruster Data, Disk Files "$JFT" and "$JFTM"

THRUSTER NO.	THRUSTER ID.	F_X (LB)	F_Y (LB)	F_Z (LB)	STA (IN)	BL (IN)	WL (IN)	C (FT)
1	F2F	-879.4	-26.2	119.9	306.72	14.65	392.96	0.0000
2	F3F	-879.5	0.0	122.7	306.72	0.00	394.45	0.0000
3	F1F	-879.4	26.2	119.9	306.72	-14.65	392.96	0.0000
4	F1L	-26.3	873.6	18.2	362.67	-69.50	373.73	0.0000
5	F3L	-21.0	870.3	0.5	364.71	-71.65	359.25	0.0000
6	F2R	-26.3	-873.6	18.2	362.67	69.50	373.73	0.0000
7	F4R	-21.0	-870.3	0.5	364.71	71.65	359.25	0.0000
8	F2U	-32.3	-11.7	874.4	350.93	14.39	413.46	0.0000
9	F3U	-31.9	0.0	873.5	350.92	0.00	414.53	0.0000
10	F1U	-32.3	11.7	874.4	350.93	-14.39	413.45	0.0000
11	F2D	-28.0	-616.4	-639.5	333.84	61.42	356.95	0.0000
12	F1D	-28.0	616.4	-639.5	333.84	-61.42	356.95	0.0000
13	F4D	-24.8	-612.6	-639.4	348.44	66.23	358.44	0.0000
14	F3D	-24.8	612.6	-639.4	348.44	-66.23	358.44	0.0000
15	R3A	856.8	0.0	151.1	1555.29	137.00	473.06	0.0000
16	R1A	856.8	0.0	151.1	1555.29	124.00	473.06	0.0000
17	L3A	856.8	0.0	151.1	1555.29	-137.00	473.06	0.0000
18	L1A	856.8	0.0	151.1	1555.29	-124.00	473.06	0.0000
19	L4L	3.3	868.8	4.8	1515.41	-149.91	452.87	-0.2421
20	L2L	3.3	868.8	4.8	1528.43	-149.91	452.88	-0.2759
21	L3L	3.3	868.8	4.8	1541.45	-149.91	452.88	-0.3098
22	L1L	3.3	868.8	4.8	1554.47	-149.91	452.88	-0.3437
23	R4R	3.3	-868.8	4.8	1515.41	149.91	452.87	0.2421
24	R2R	3.3	-868.8	4.8	1528.43	149.91	452.88	0.2759
25	R3R	3.3	-868.8	4.8	1541.45	149.91	452.88	0.3098
26	R1R	3.3	-868.8	4.8	1554.47	149.91	452.88	0.3437
27	L4U	0.1	76.3	872.2	1516.36	-115.43	481.95	-0.1586
28	L2U	0.1	76.3	872.2	1529.23	-115.43	481.95	-0.0648
29	L1U	0.1	76.3	872.2	1542.10	-115.43	481.95	0.0290
30	R4U	0.1	-76.3	872.2	1516.36	115.43	481.95	0.1586
31	R2U	0.1	-76.3	872.2	1529.23	115.43	481.95	0.0648
32	R1U	0.1	-76.3	872.2	1542.10	115.43	481.95	-0.0290
33	L4D	210.6	318.4	-576.0	1510.34	-104.34	431.12	0.1103
34	L2D	210.6	318.4	-576.0	1526.50	-103.22	434.78	-0.0959
35	L3D	210.6	318.4	-576.0	1542.65	-102.11	438.44	-0.3020
36	R4D	210.6	-318.4	-576.0	1510.34	104.34	431.12	-0.1103
37	R2D	210.6	-318.4	-576.0	1526.50	103.22	434.78	0.0959
38	R3D	210.6	-318.4	-576.0	1542.65	102.11	438.44	0.3020
39	F5R	-0.8	-17.0	-17.6	324.35	59.70	350.12	0.0000
40	F5L	-0.8	17.0	-17.6	324.35	-59.70	350.12	0.0000
41	R5R	0.0	-24.0	-0.6	1565.00	149.87	459.00	0.0000
42	L5L	0.0	24.0	-0.6	1565.00	-149.87	459.00	0.0000
43	R5D	0.0	0.0	-24.0	1565.00	118.00	455.44	0.0000
44	L5D	0.0	0.0	-24.0	1565.00	-118.00	455.44	0.0000

C-2

Table 2. Basic RCS Thrust Data (Without Plume Impingement)

THRUSTER NO.	THRUSTER ID.	F_X (LB)	F_Y (LB)	F_Z (LB)	STA (IN)	BL (IN)	WL (IN)
1	F2F	-879.4	-26.2	119.9	306.72	14.65	392.96
2	F3F	-879.5	0.0	122.7	306.72	0.00	394.45
3	F1F	-879.4	26.2	119.9	306.72	-14.65	392.96
4	F1L	-26.3	873.6	18.2	362.67	-69.50	373.73
5	F3L	-21.0	870.3	0.5	364.71	-71.65	359.25
6	F2R	-26.3	-873.6	18.2	362.67	69.50	373.73
7	F4R	-21.0	-870.3	0.5	364.71	71.65	359.25
8	F2U	-32.3	-11.7	874.4	350.93	14.39	413.46
9	F3U	-31.9	0.0	873.5	350.92	0.00	414.53
10	F1U	-32.3	11.7	874.4	350.93	-14.39	413.46
11	F2D	-28.0	-616.4	-639.5	333.84	61.42	356.95
12	F1D	-28.0	616.4	-639.5	333.84	-61.42	356.95
13	F4D	-24.8	-612.6	-639.4	348.44	66.23	358.44
14	F3D	-24.8	612.6	-639.4	348.44	-66.23	358.44
15	R3A	856.8	0.0	151.1	1555.29	137.00	473.06
16	R1A	856.8	0.0	151.1	1555.29	124.00	473.06
17	L3A	856.8	0.0	151.1	1555.29	-137.00	473.06
18	L1A	856.8	0.0	151.1	1555.29	-124.00	473.06
19	L4L	0.0	870.5	-22.4	1516.00	-149.87	459.00
20	L2L	0.0	870.5	-22.4	1529.00	-149.87	459.00
21	L3L	0.0	870.5	-22.4	1542.00	-149.87	459.00
22	L1L	0.0	870.5	-22.4	1555.00	-149.87	459.00
23	R4R	0.0	-870.5	-22.4	1516.00	149.87	459.00
24	R2R	0.0	-870.5	-22.4	1529.00	149.87	459.00
25	R3R	0.0	-870.5	-22.4	1542.00	149.87	459.00
26	R1R	0.0	-870.5	-22.4	1555.00	149.87	459.00
27	L4U	0.0	0.0	870.0	1516.00	-132.00	480.50
28	L2U	0.0	0.0	870.0	1529.00	-132.00	480.50
29	L1U	0.0	0.0	870.0	1542.00	-132.00	480.50
30	R4U	0.0	0.0	870.0	1516.00	132.00	480.50
31	R2U	0.0	0.0	870.0	1529.00	132.00	480.50
32	R1U	0.0	0.0	870.0	1542.00	132.00	480.50
33	L4D	170.4	291.8	-801.7	1516.00	-111.95	437.40
34	L2D	170.4	291.8	-801.7	1529.00	-111.00	440.00
35	L3D	170.4	291.8	-801.7	1542.00	-110.06	442.60
36	R4D	170.4	-291.8	-801.7	1516.00	111.95	437.40
37	R2D	170.4	-291.8	-801.7	1529.00	111.00	440.00
38	R3D	170.4	-291.8	-801.7	1542.00	110.06	442.60
39	F5R	-0.8	-17.0	-17.6	324.35	69.70	350.12
40	F5L	-0.8	17.0	-17.6	324.35	-69.70	350.12
41	R5R	0.0	-24.0	-0.6	1565.00	149.87	459.00
42	L5L	0.0	24.0	-0.6	1565.00	-149.87	459.00
43	R5D	0.0	0.0	-24.0	1565.00	118.00	455.44
44	L5D	0.0	0.0	-24.0	1565.00	-118.00	455.44

Table 3. Force and Moment Increments Due to Plume Impingement (CG @ STA 1076.7, BL 0, WL 375.0)

THRUSTER ID	Δ FORCE (LB)			Δ MOMENT (FT-LB)		
	F_X	F_Y	F_Z	L_X	L_Y	L_Z
L4L	3.34	-1.66	27.17	-795.30	762.42	144.46
L2L	3.34	-1.66	27.17	-795.30	762.42	144.46
L3L	3.34	-1.66	27.17	-795.30	762.42	144.46
L1L	3.34	-1.66	27.17	-795.30	762.42	144.46
L4U	0.09	76.30	2.21	1859.90	94.21	-2932.80
L2U	0.09	76.30	2.21	1859.90	94.21	-2932.80
L1U	0.09	76.30	2.21	1859.90	94.21	-2932.80
L4D	40.19	26.59	225.66	-2475.50	8469.00	-645.46
L2D	40.19	26.59	225.66	-2475.50	8469.00	-645.46
L3D	40.19	26.59	225.66	-2475.50	8469.00	-645.46
R4R	3.34	1.66	27.17	795.30	762.42	-144.46
R2R	3.34	1.66	27.17	795.30	762.42	-144.46
R3R	3.34	1.66	27.17	795.30	762.42	-144.46
R1R	3.34	1.66	27.17	795.30	762.42	-144.46
R4U	0.09	-76.30	2.21	-1859.90	94.21	2932.80
R2U	0.09	-76.30	2.21	-1859.90	94.21	2932.80
R1U	0.09	-76.30	2.21	-1859.90	94.21	2932.80
R4D	40.19	-26.59	225.66	2475.50	8469.00	645.46
R2D	40.19	-26.59	225.66	2475.50	8469.00	645.46
R3D	40.19	-26.59	225.66	2475.50	8469.00	645.46

(Taken from Reference 13)

inhibited). Corresponding to each of these options is a jet-select table (Tables 4-6) which identifies the particular jet or combination of jets that is to be fired in response to each of the six RCS translational acceleration commands (+X, -X, +Y, -Y, +Z, -Z) and the six rotational acceleration commands (+ROL, -ROL, +PCH, -PCH, +YAW, -YAW). The jet select tables, which reside in the disk files named "$JSV", "$JSP", and "$JSPZI", are not routinely available for modification by the user. However, the HFRMP software system includes a jet-select editing processor that makes it possible to update the tables or to provide additional options with little difficulty.

As indicated in Table 4 by the absence of any jet designations for the execution of translation commands, the V option (vernier jets) can be used only for rotational control. The P option (Table 5) and the PZI option (Table 6) can be used for translational and/or rotational control.

In the PZI option, no jets are fired that would expel propellant directly upward with respect to the Orbiter body. Translational acceleration in the downward direction, if commanded, is achieved (at a comparatively high propellant cost) by firing the +X and -X thrusters simultaneously. The cant angles of the +X and -X jet thrust lines produce a small net acceleration in the +Z (downward) direction. This option normally is used only when the Orbiter is maneuvering in the near vicinity of a payload that must be protected from jet plume impingement.

3.1.3 Shuttle Mass Properties

At %RMAT execution time, the user-defined mass properties of the Shuttle[†]

[†]Which remain constant during HFRMP trajectory integration.

Table 4. Vernier (V) Jet Select Table, Disk File "$JSV"

CMD	THRUSTERS TO BE FIRED							
	1	2	3	4	5	6	7	8
+X								
-X								
+Y								
-Y								
+Z								
-Z								
+ROL	L5D							
-ROL	R5D							
+PCH	F5R	F5L						
-PCH	L5D	R5D						
+YAW	R5R							
-YAW	L5L							

Table 5. Primary (P) Jet Select Table, Disk File "$JSP"

CMD	THRUSTERS TO BE FIRED							
	1	2	3	4	5	6	7	8
+X	R1A	L1A						
-X	F2F	F1F						
+Y	F1L	L4L						
-Y	F2R	R4R						
+Z	F3U	L4U	R4U					
-Z	F1D	F2D	L4D	L2D	R4D	R2D		
+ROL	L4D	R4U						
-ROL	L4U	R4D						
+PCH	F1D	F2D	L4U	R4U				
-PCH	F3U	L4D	R4D					
+YAW	F1L	R4R						
-YAW	F2R	L4L						

Table 6. Primary with +Z Thrusters Inhibited (PZI) Jet Select Table, Disk File "$JSPZI"

CMD	THRUSTERS TO BE FIRED							
	1	2	3	4	5	6	7	8
+X	R1A	L1A						
-X	F2F	F1F						
+Y	F1L	L4L						
-Y	F2R	R4R						
+Z	F2F	F1F	R1A	L1A				
-Z	F1D	F2D	L4D	L2D	R4D	R2D		
+ROL	L4D							
-ROL	R4D							
+PCH	F1D	F2D						
-PCH	L4D	R4D						
+YAW	F1L	R4R						
-YAW	F2R	L4L						

reside in the disk file named "1*", where they are stored by the data-base editing processor #DBED. The "1*" file contains 32 scalar quantities, of which %RMAT makes use of the following:

Item No.	Description		
1	Shuttle gross weight (lb)		
2	I_{XX}	moments of inertia	($slug\text{-}ft^2$)
3	I_{YY}	moments of inertia	($slug\text{-}ft^2$)
4	I_{ZZ}	moments of inertia	($slug\text{-}ft^2$)
5	I_{YZ}	products of inertia	($slug\text{-}ft^2$)
6	I_{ZX}	products of inertia	($slug\text{-}ft^2$)
7	I_{XY}	products of inertia	($slug\text{-}ft^2$)
8	STA	station coordinates of CG (in)	
9	BL	station coordinates of CG (in)	
10	WL	station coordinates of CG (in)	
.			
.			
.			
32	%RMAT execution flag		

Item 32 in the "1*" file is a flag that is tested immediately after entry into %RMAT to determine whether a re-computation of the response matrices is necessary. It is set equal to zero during initialization of the program disk, and thereafter (by the appropriate editing processor) when the contents of "1*", "$JFT", or any jet-select table are changed in any way. It is set equal to 9 by %RMAT upon completion of the computation and storage of the response matrices.

3.2 OUTPUT DATA

The output of the %RMAT link consists of eight 12x12 matrices, each corresponding to a particular jet select table and a particular mode of RCS cross-

coupling compensation, and each stored in a separate disk file as indicated in Table 7. The compensation modes available to the HFRMP user are designated NONE (no compensation), ROT (rotational compensation only), and FULL (rotational and translational compensation). In any given flight profile segment, the HFRMP user may choose any jet-select option in combination with any compensation mode, except V with FULL. Full compensation is impossible with the vernier jets, and the compensation mode is internally defaulted to ROT when such a combination is specified by the user.

Tables 8-10 show typical response matrices for the P jet-select option and increasing degrees of cross-coupling compensation. For purposes of illustration, each matrix was transposed and then partitioned into two 12x6 matrices so it could be printed conveniently on a single page. Each column of the (untransposed) response matrix corresponds to a particular translational or rotational command. Rows 1-3 contain the body-axis components of the steady-state linear acceleration, and rows 4-6 contain the components of angular acceleration. Rows 7-12 contain RCS propellant consumption rates, broken down according to source (forward, aft left, or aft right tanks) and control function (translation or rotation). The computations that produce the results shown in Tables 8-10 are described by the flow chart shown in Figures 20a through 20g, wherein the symbols [U], [R], and [F] represent the uncompensated, rotationally compensated, and fully compensated response matrices.

3.3 COMPUTATIONS

It will be noted that six small rectangles, each enclosing a pair of acceleration components, run diagonally across the upper partition of the matrix shown in Figure 8. The components thus enclosed represent the uncompensated "principal response" of the Shuttle to each of the 12 translation and

Table 7. File Names for Response Matrices

		JET SELECT		
		V	P	PZI
COMPENSATION	NONE	"*V"	"*P"	"*PZI"
	ROT	"*VR"	"*PR"	"*PZIR"
	FULL	——	"*PF"	"*PZIF"

Table 8. Uncompensated Response Matrix† for Jet Select Option P, Disk File "*P"

CMD	LINEAR ACCELERATION (FT/SEC^2)			ANGULAR ACCELERATION (RAD/SEC^2)			
	a_X	a_Y	a_Z	α_X	α_Y	α_Z	
+X	0.27564	0.00000	0.04867	-0.00001	-0.00033	0.00001	1
-X	-0.28291	0.00000	0.03857	-0.00002	-0.00211	-0.00001	2
+Y	-0.00369	0.28028	0.00369	0.00669	-0.00018	0.00365	3
-Y	-0.00369	-0.28028	0.00369	-0.00669	-0.00019	-0.00365	4
+Z	-0.00510	0.00000	0.42111	-0.00007	0.00111	-0.00000	5
-Z	0.12649	-0.00013	-0.57637	0.00017	-0.00064	0.00702	6
+ROL	0.03389	0.03894	0.04764	0.01588	0.00154	-0.00037	7
-ROL	0.03389	-0.03894	0.04764	-0.01588	0.00151	0.00037	8
+PCH	-0.00898	0.00000	0.07487	0.00013	0.02227	0.00000	9
-PCH	0.06262	0.00000	-0.04481	-0.00009	-0.01499	0.00000	10
+YAW	-0.00369	0.00077	0.00369	-0.00306	-0.00019	0.01237	11
-YAW	-0.00369	-0.00077	0.00369	0.00306	-0.00018	-0.01237	12
	1	2	3	4	5	6	

Column Indices

Row Indices

CMD	PROPELLANT CONSUMPTION RATES (LB/SEC)						
	TRANSLATIONAL CONTROL			ROTATIONAL CONTROL			
	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK	
+X	0.00000	3.10710	3.10710	0.00000	0.00000	0.00000	1
-X	6.21420	0.00000	0.00000	0.00000	0.00000	0.00000	2
+Y	3.10710	3.10710	0.00000	0.00000	0.00000	0.00000	3
-Y	3.10710	0.00000	3.10710	0.00000	0.00000	0.00000	4
+Z	3.10710	3.10710	3.10710	0.00000	0.00000	0.00000	5
-Z	6.21420	6.21420	6.21420	0.00000	0.00000	0.00000	6
+ROL	0.00000	0.00000	0.00000	0.00000	3.10710	3.10710	7
-ROL	0.00000	0.00000	0.00000	0.00000	3.10710	3.10710	8
+PCH	0.00000	0.00000	0.00000	6.21420	3.10710	3.10710	9
-PCH	0.00000	0.00000	0.00000	3.10710	3.10710	3.10710	10
+YAW	0.00000	0.00000	0.00000	3.10710	0.00000	3.10710	11
-YAW	0.00000	0.00000	0.00000	3.10710	3.10710	0.00000	12
	7	8	9	10	11	12	

Column Indices

Row Indices

† Transposed and partitioned for purposes of illustration.

Table 9. Rotationally Compensated Response Matrix† for Jet Select Option P, Disk File "*PR"

CMD	LINEAR ACCELERATION (FT/SEC^2)			ANGULAR ACCELERATION (RAD/SEC^2)		
	a_X	a_Y	a_Z	α_X	α_Y	α_Z
+X	0.27554	0.00001	0.04974	0.00000	0.00000	0.00000
-X	-0.28368	0.00003	0.04577	0.00000	0.00000	0.00000
+Y	0.01420	0.26132	0.02634	0.00000	0.00000	0.00000
-Y	0.01426	-0.26130	0.02637	0.00000	0.00000	0.00000
+Z	-0.00027	0.00018	0.41799	0.00000	0.00000	0.00000
-Z	0.12667	-0.00056	-0.57375	0.00000	0.00000	0.00000
+ROL	0.04022	0.03897	0.04316	0.01578	0.00000	0.00000
-ROL	0.04010	-0.03897	0.04324	-0.01579	0.00000	0.00000
+PCH	-0.00870	-0.00032	0.07526	0.00000	0.02228	0.00000
-PCH	0.06280	0.00021	-0.04455	0.00000	-0.01498	0.00000
+YAW	0.00329	0.00828	0.01256	0.00000	0.00000	0.01230
-YAW	0.00330	-0.00827	0.01254	0.00000	0.00000	-0.01230

CMD	PROPELLANT CONSUMPTION RATES (LB/SEC)					
	TRANSLATIONAL CONTROL			ROTATIONAL CONTROL		
	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK
+X	0.00000	3.10710	3.10710	0.09246	0.04949	0.04798
-X	6.21420	0.00000	0.00000	0.59096	0.30190	0.30373
+Y	3.10710	3.10710	0.00000	1.15363	2.62193	1.65885
-Y	3.10710	0.00000	3.10710	1.15908	1.66496	2.62801
+Z	3.10710	3.10710	3.10710	0.23295	0.24650	0.24763
-Z	6.21420	6.21420	6.21420	0.18607	0.13175	0.12678
+ROL	0.00000	0.00000	0.00000	0.41495	3.42754	3.52125
-ROL	0.00000	0.00000	0.00000	0.40939	3.51573	3.42169
+PCH	0.00000	0.00000	0.00000	6.21529	3.13352	3.13243
-PCH	0.00000	0.00000	0.00000	3.10772	3.12437	3.12472
+YAW	0.00000	0.00000	0.00000	3.12941	0.62176	3.72886
-YAW	0.00000	0.00000	0.00000	3.13013	3.72896	0.62188

†Transposed and partitioned for purposes of illustration.

Table 10. Fully Compensated Response Matrix for Jet Select Option P, Disk File "*PF"

CMD	LINEAR ACCELERATION (FT/SEC^2)			ANGULAR ACCELERATION (RAD/SEC^2)		
	a_X	a_Y	a_Z	α_X	α_Y	α_Z
+X	0.28653	0.00000	0.00000	0.00000	0.00000	0.00000
-X	-0.27357	0.00000	0.00000	0.00000	0.00000	0.00000
+Y	0.00000	0.26130	0.00000	0.00000	0.00000	0.00000
-Y	0.00000	-0.26133	0.00000	0.00000	0.00000	0.00000
+Z	0.00000	0.00000	0.41806	0.00000	0.00000	0.00000
-Z	0.00000	0.00000	-0.55325	0.00000	0.00000	0.00000
+ROL	0.00000	0.00000	0.00000	0.01578	0.00000	0.00000
-ROL	0.00000	0.00000	0.00000	-0.01579	0.00000	0.00000
+PCH	0.00000	0.00000	0.00000	0.00000	0.02228	0.00000
-PCH	0.00000	0.00000	0.00000	0.00000	-0.01498	0.00000
+YAW	0.00000	0.00000	0.00000	0.00000	0.00000	0.01230
-YAW	0.00000	0.00000	0.00000	0.00000	0.00000	-0.01230

CMD	PROPELLANT CONSUMPTION RATES (LB/SEC)					
	TRANSLATIONAL CONTROL			ROTATIONAL CONTROL		
	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK	FWD TANK	AFT LEFT TANK	AFT RIGHT TANK
+X	0.53953	3.64647	3.64604	0.10887	0.06149	0.05942
-X	6.71087	0.49632	0.49613	0.60613	0.31305	0.31441
+Y	3.88315	3.42861	0.32151	1.20648	2.65083	1.68763
-Y	3.88537	0.32201	3.42911	1.21211	1.69395	2.65688
+Z	3.10929	3.11017	3.11236	0.23386	0.24773	0.24954
-Z	8.98554	6.22074	6.21420	0.45238	0.27208	0.26589
+ROL	2.26765	0.60628	1.06903	0.72008	3.74737	3.98437
-ROL	2.26587	1.07028	0.60640	0.71352	3.97819	3.74023
+PCH	1.24471	0.96111	0.95643	6.28160	3.17562	3.17245
-PCH	1.87511	0.44562	0.44836	3.27278	3.22396	3.22567
+YAW	0.40820	0.15745	0.25574	3.18539	0.68537	3.82285
-YAW	0.40790	0.25558	0.15707	3.18589	3.82282	0.68510

†Transposed and partitioned for purposes of illustration.

rotation commands. All other components of acceleration represent "extraneous responses", commonly referred to as "cross-coupling effects", that arise from the canting of certain jet centerlines away from the body axes, uneven moment arms, and plume impingement on the exterior surfaces of the Orbiter. The details of the uncompensated response matrix calculations are defined in Figures 20b and 20c.

The process of cross-coupling compensation can be explained in the following general terms. Assume that the jets activated by a given "primary command" P fire continuously for a long period of time Δt. The effects of extraneous accelerations are nullified by intermittent firings of jets activated by "compensating commands". Let the accumulated firing time of the jets activated intermittently by any particular compensating command K be represented by δt_K. The ratio $Y_K = \delta t_K/\Delta t$ is referred to as the "duty cycle" of the (jets activated by the) compensating command.

The uncompensated response to the primary command (a column in the uncompensated response matrix) we represent with the symbol $[U]_P \equiv [(U_{1,P})\ (U_{2,P})\ \ldots\ (U_{12,P})]^T$. The compensated response (a column in the compensated response matrix is defined by the expression.

$$[C]_P = [U]_P + \sum^{K} Y_K\ [U]_K,$$

where the summation includes all the compensating commands that are required to nullify the extraneous accelerations.

Rotational compensation, which involves the nullification of only the extraneous <u>angular</u> accelerations (see Table 9), is carried out in two phases. The rotation commands themselves are rotationally compensated in the first phase (Figure 20d), which is of necessity an iterative process. Successi

compensations introduce new extraneous angular accelerations, whose magnitudes must diminish progressively if convergence is to be realized. Non-convergence of this process implies that (given the input mass properties and the jet-select table under consideration) the attitude of the Shuttle can not be controlled, and an appropriate warning message is output.

The translation commands are rotationally compensated in the second phase of rotational compensation (Figure 20e). This is a non-iterative process that is simplified by use of the results from the first phase.

Full compensation is achieved by nullifying the extraneous linear accelerations that remain in the rotationally compensated matrix. Again, the process is carried out in two phases. Starting with the rotationally compensated matrix as an input (which eliminates the need to worry about compensation commands producing extraneous angular accelerations) the first phase (Figure 20f) is devoted to the translational compensation of the translation commands. This is an iterative process that is essentially identical to that defined in Figure 20d. Likewise, the second phase of translational compensation (Figure 20g) is essentially identical to that shown in Figure 20e.

The computations are repeated for each of the three jet select tables, except that full compensation is not attempted with the vernier jets. Appropriate warning messages (if any) are stored in the desk files along with the response matrices.

The last executable statement in the %RMAT link causes the code of the %TNIT link (Section 4) to be appended to that of the base link (#TRAJ), in the region of computer memory formerly occupied by the %RMAT code. The HPL get command is used for this purpose, which causes all data registers used by %RMAT to be de-allocated (erased) before execution control is handed over to %TNIT.

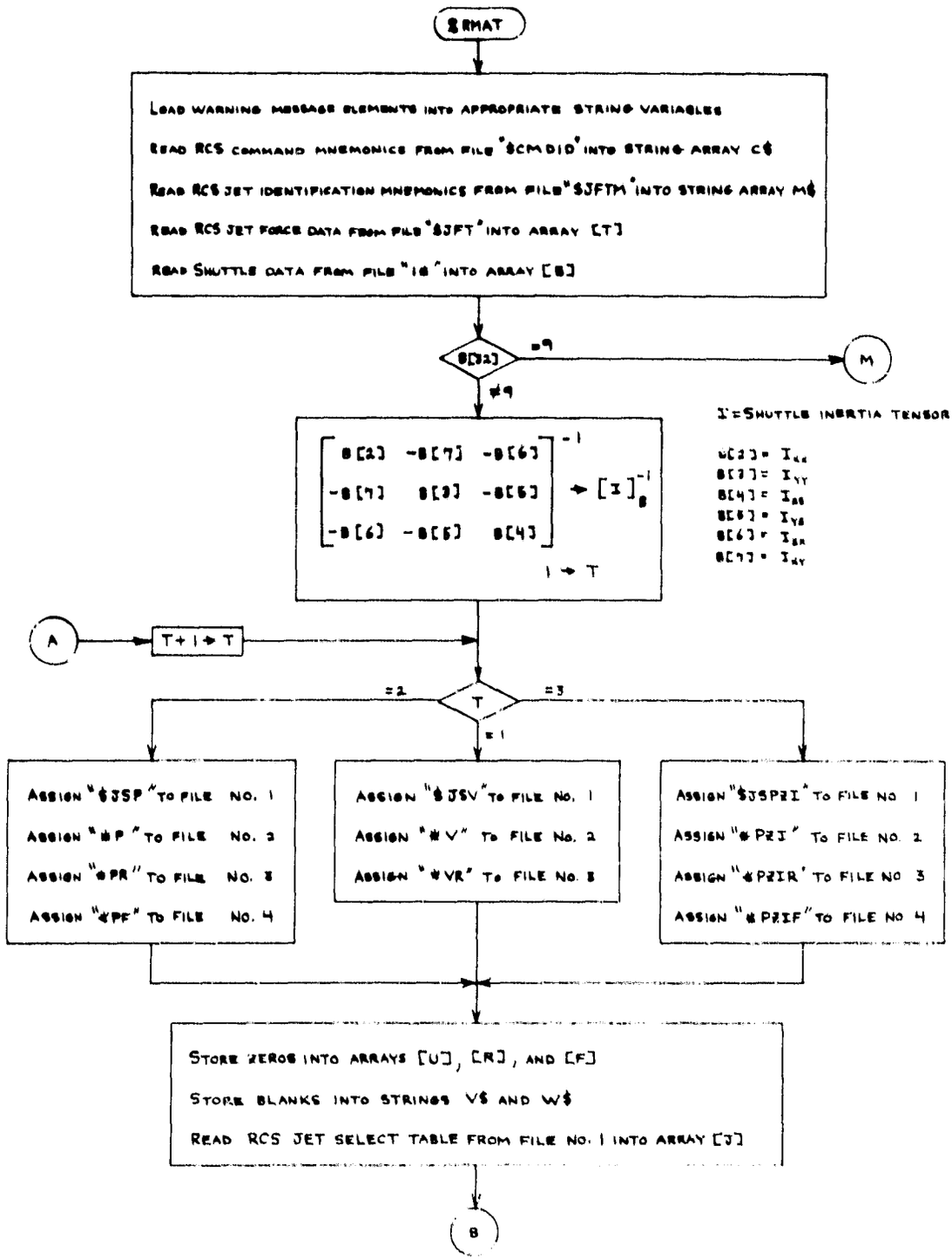


Figure 20a. $RMAT Logic Flow

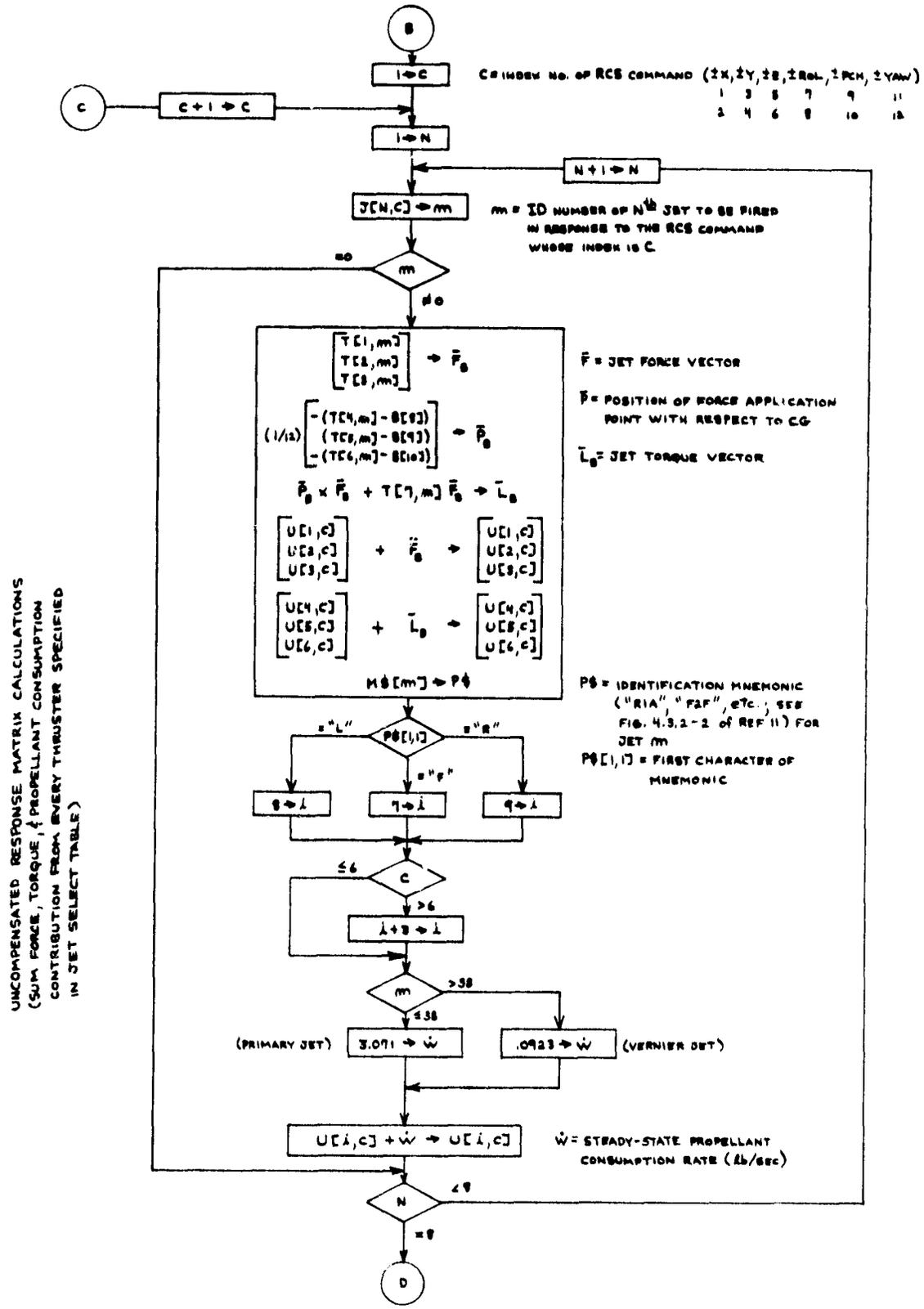


Figure 20b. %RMAT Logic Flow (cont.)

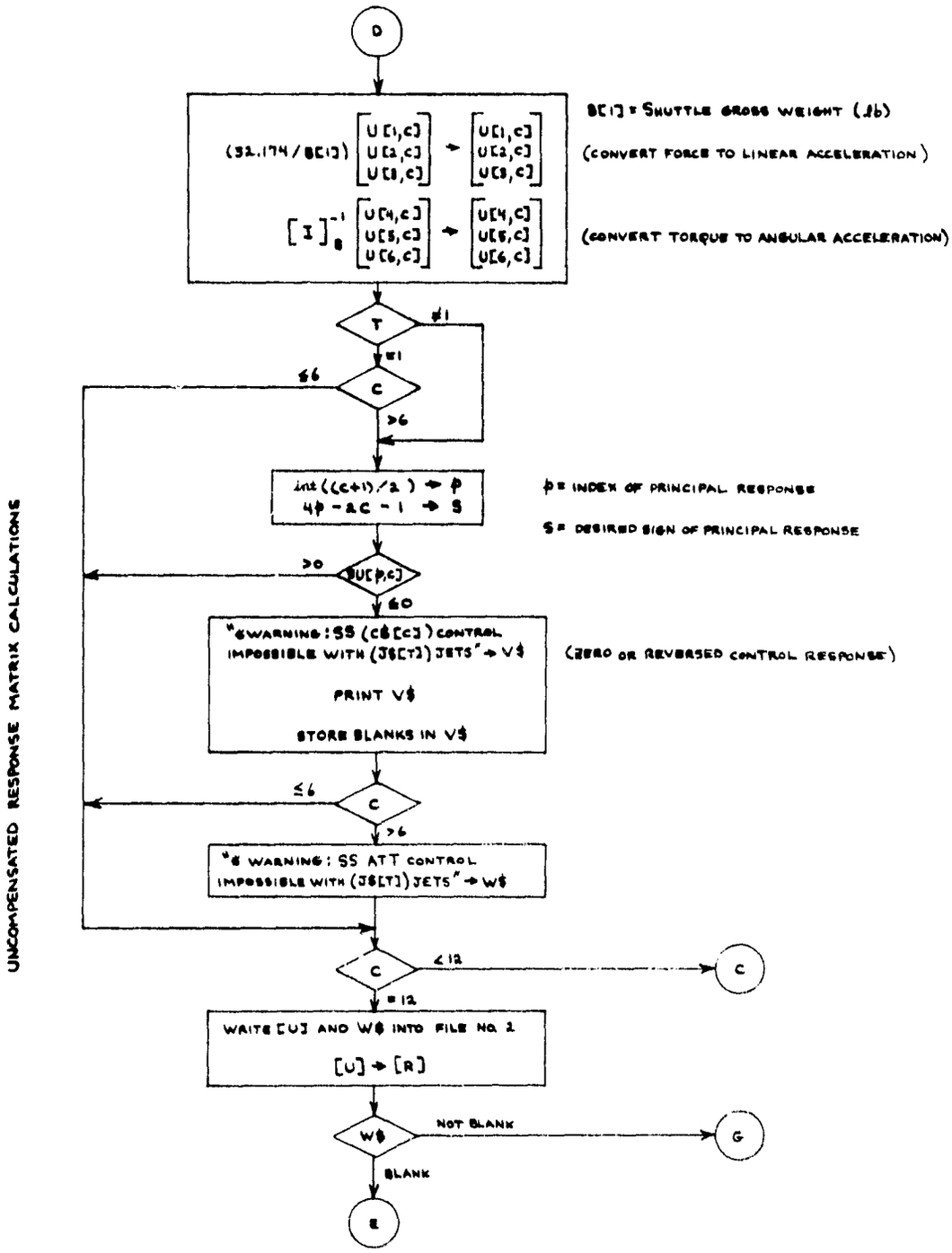


Figure 20c. %RMAT Logic Flow (cont.)

ROTATIONAL COMPENSATION OF ROTATION COMMANDS

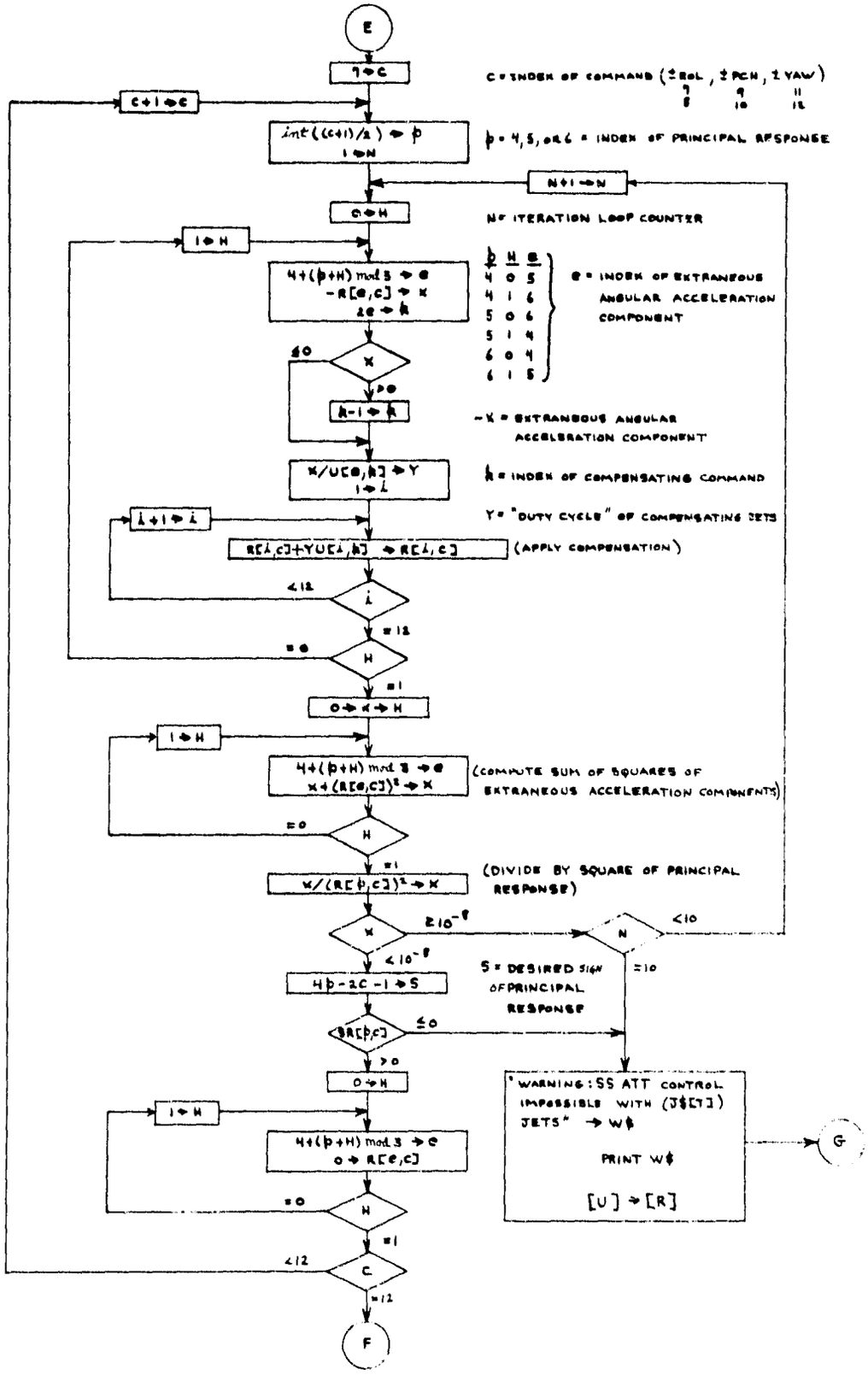


Figure 20d. SRMAT Logic Flow (cont.)

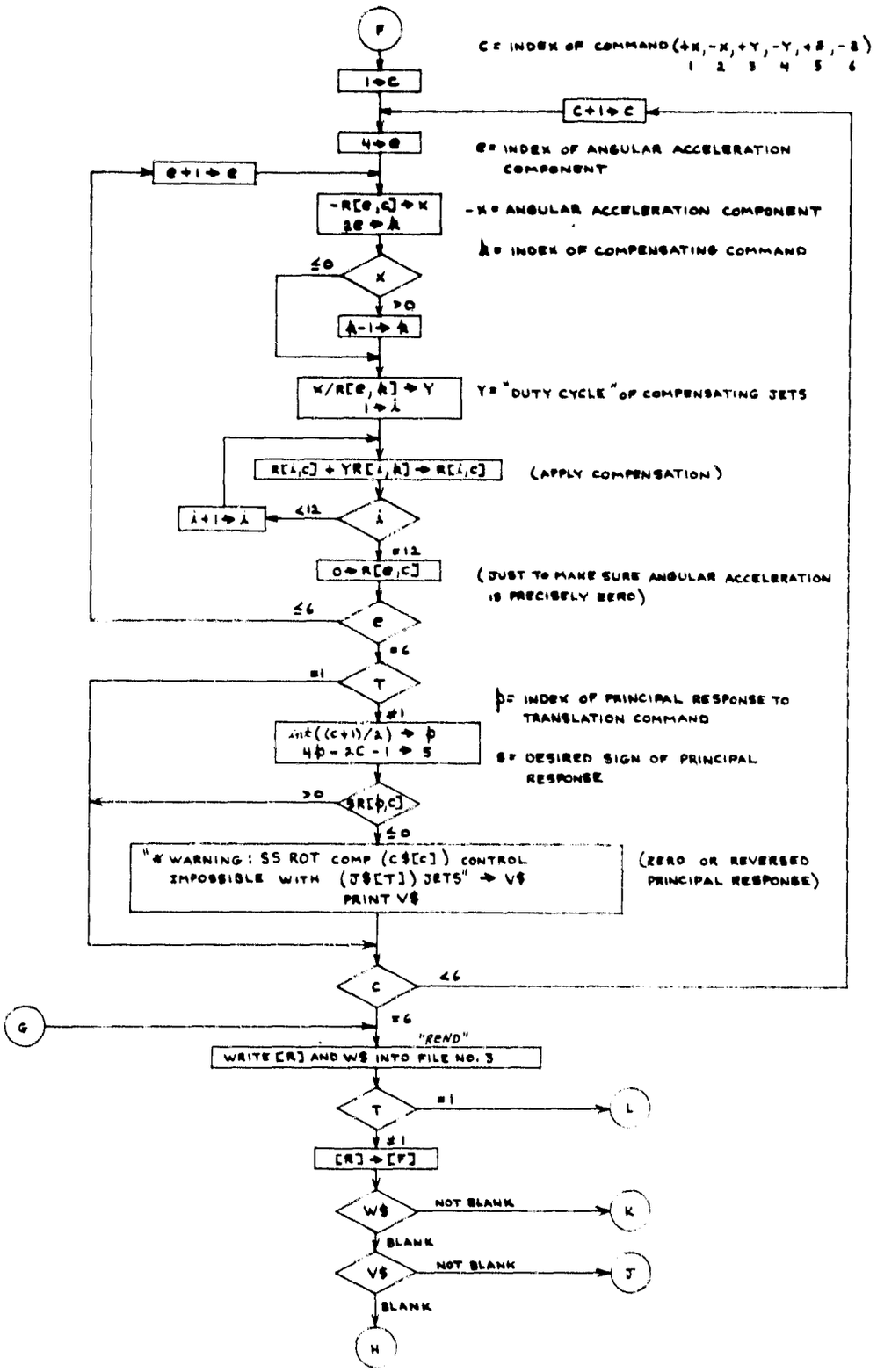


Figure 20e. SRMAT Logic Flow (cont.)

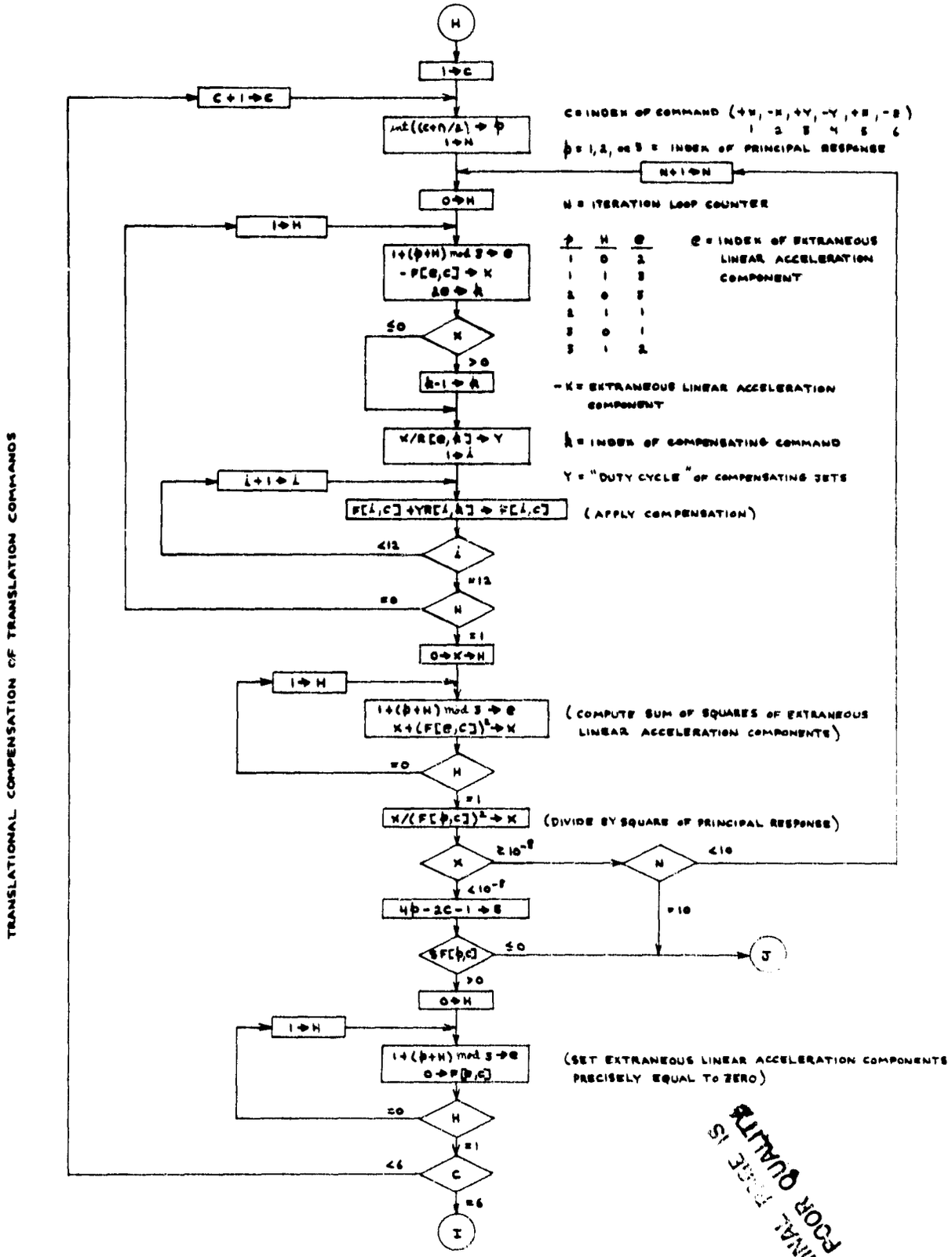


Figure 20f. %RMAT Logic Flow (cont.)

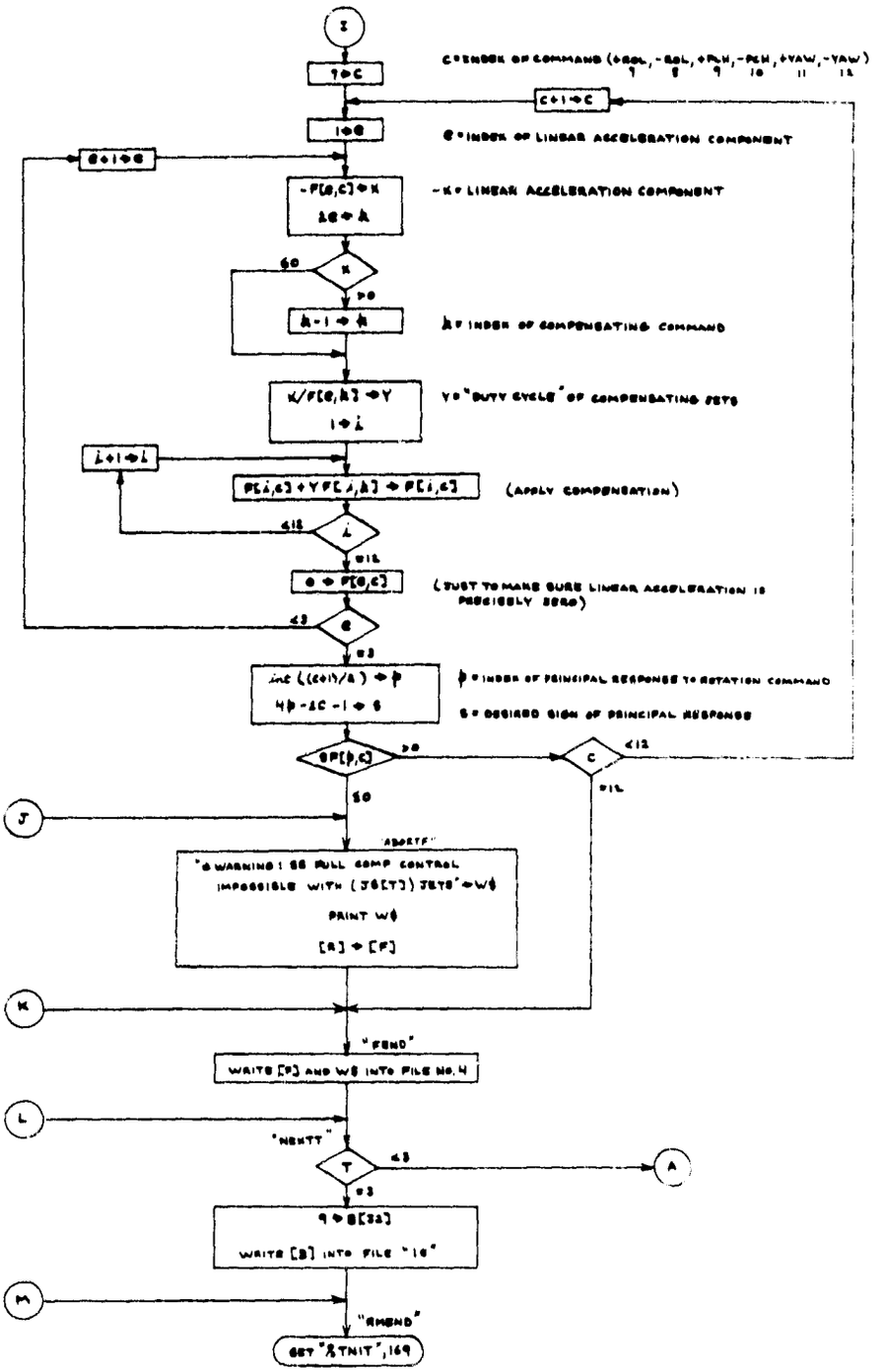


Figure 20g. %RMAT Logic Flow (cont.)

4. TRAJECTORY INITIALIZATION LINK (%TNIT)

The function of the %TNIT link is to initialize calculator memory in preparation for trajectory integration. Elements of the user-supplied data base (Appendix E) are read from appropriate disk files (where they will have been stored by a previous execution of the #DBED processor) and processed as necessary to define the constants and the initial state of the Shuttle/payload system.

4.1 FUNCTION SUBPROGRAMS

4.1.1 'JD'

The 'JD' function subprogram computes the Julian day number corresponding to a given year, month, and day of the Gregorian (civil) calendar.

4.1.1.1 Argument List

p1 = year
p2 = month
p3 = day

} integers

4.1.1.2 Example of Usage

The instruction 'JD'(1980,4,2) → D would cause D to be assigned the value 2444332 (the number of the Julian day commencing at Greenwich noon on 2 April 1980).

4.1.1.3 Computations

Figure 21 is a flow chart of the 'JD' computational procedure, which was taken from Reference 14.

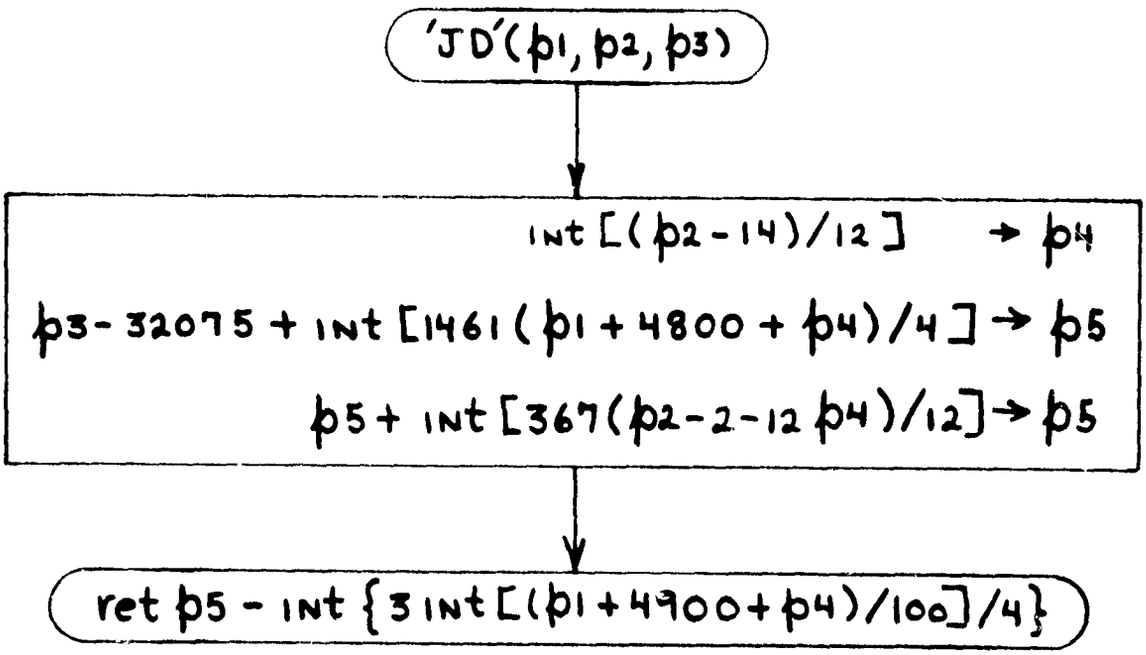


Figure 21. 'JD' Function Subprogram Logic Flow

4.1.2 'PLOTYP'

Given the user-assigned mnemonic symbol which designates the desired type of data plot, 'PLOTYP' returns the appropriate numeric code for internal use.

4.1.2.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'PLOTYP' is executed.

4.1.2.2 Example of Usage

If the character string "CPLV" resides in P$, the instruction 'PLOTYP' → I will cause I to be assigned the value 2.

4.1.2.3 Computations

See Figure 22.

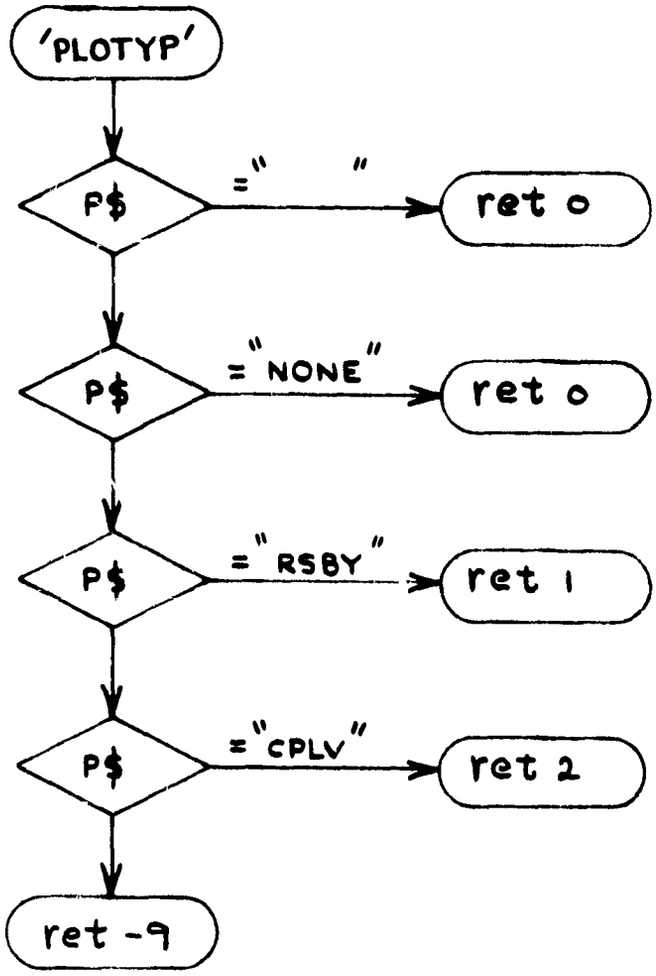


Figure 22. 'PLOTYP' Function Subprogram Logic Flow

4.1.3 'UNIT'

Given the user-assigned mnemonic symbol which designates the unit of distance for data plots, the 'UNIT' subprogram returns the appropriate conversion factor (the number of feet in the designated unit).

4.1.3.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'UNIT' is executed.

4.1.3.2 Example of Usage

The instructions " NMI" → P$; 'UNIT' → C would cause C to be assigned the value 6076.115.

4.1.3.3 Computations

See Figure 23.

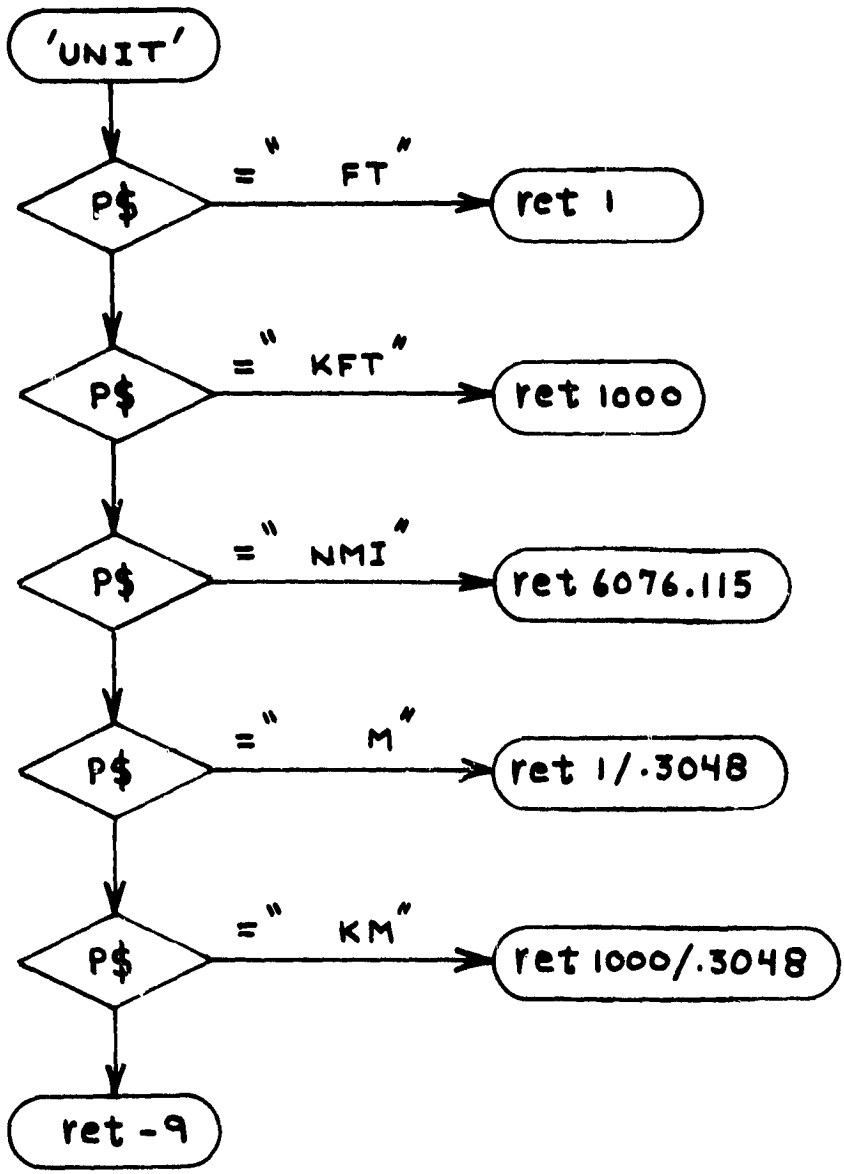


Figure 23. 'UNIT' Function Subprogram Logic Flow

4.1.4 'MONTH'

Given the abbreviated name of a month, the 'MONTH' function subprogram returns the appropriate month number.

4.1.4.1 Argument List

None. The abbreviated month name must be assigned to the string variable P$ before MONTH is executed.

4.1.4.2 Example of Usage

If the character string " AUG" resides in P$, the instruction 'MONTH' → M will cause the number 8 to be assigned to the variable M.

4.1.4.3 Computations

See Figure 24.

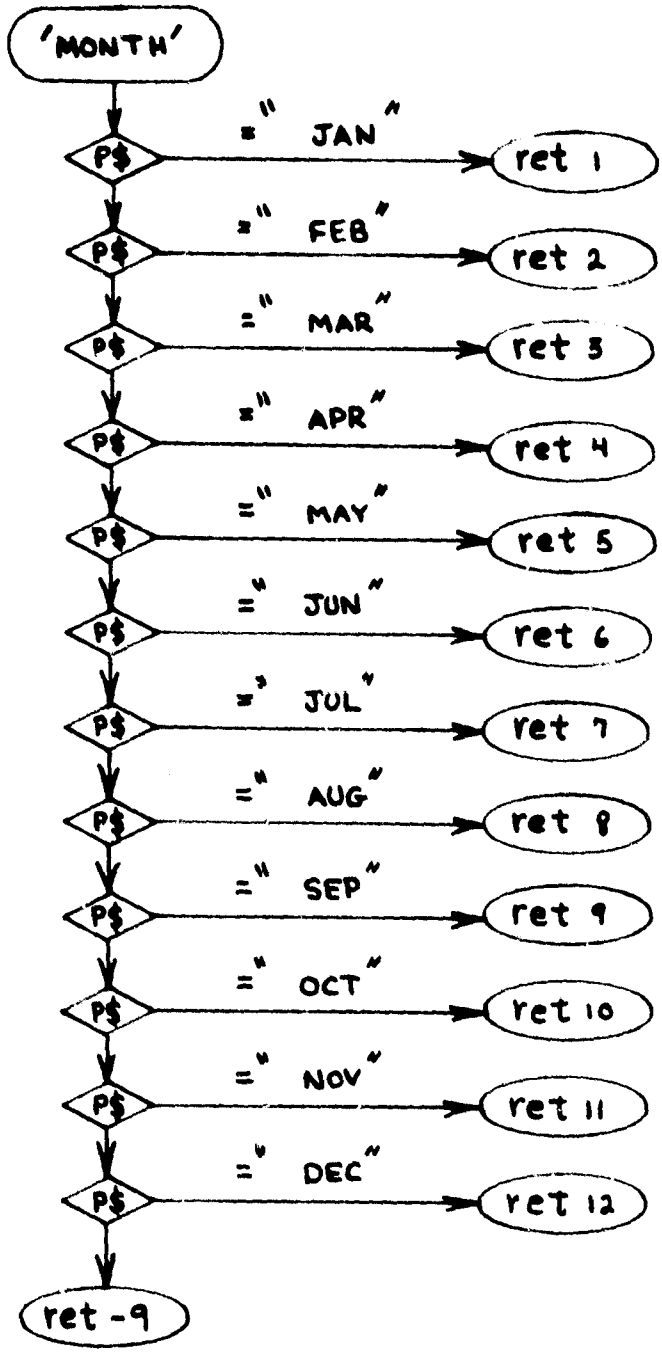


Figure 24. 'MONTH' Function Subprogram Logic Flow

4.1.5 'C'

Given the size of a square matrix stored columnwise in a one-dimensional array, and the row and column indices of one of its elements, the 'C' function returns the relative address (in the one-dimensional array) of the matrix element.

4.1.5.1 Argument List

p1 = Row index of matrix element.

p2 = Column index of matrix element.

p3 = Matrix size (number of rows = number of columns).

4.1.5.2 Example of Usage

Assume the 3x3 matrix [T] is stored columnwise in registers r8 - r16, as shown below:

Relative Address	Register No.	Matrix Element
0	r8	$T_{1,1}$
1	r9	$T_{2,1}$
2	r10	$T_{3,1}$
3	r11	$T_{1,2}$
4	r12	$T_{2,2}$
5	r13	$T_{3,2}$
6	r14	$T_{1,3}$
7	r15	$T_{2,3}$
8	r16	$T_{3,3}$

The instruction r(8 + 'C'(1,2,3)) → A would cause the value of $T_{1,2}$ to be assigned to the variable A.

4.1.5.3 Computations

See Figure 25.

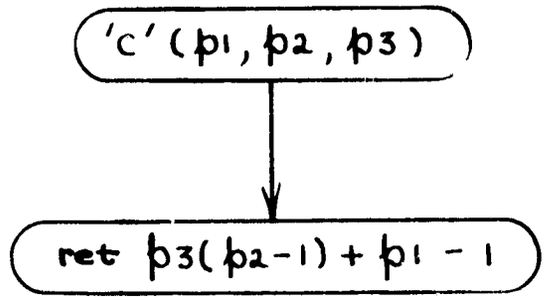


Figure 25 'C' Function Subprogram Logic Flow

4.1.6 'T'

Given the row and column indices of an element in a symmetric matrix whose upper triangular form is stored columnwise in a one-dimensional array, the 'T' function returns the relative address (in the one-dimensional array) of the matrix element.

4.1.6.1 Argument List

p1 = Row index of matrix element.

p2 = Column index of matrix element.

4.1.6.2 Example of Usage

Assume the upper triangular form

$$\begin{bmatrix} I_{1,1} & I_{1,2} & I_{1,3} \\ & I_{2,2} & I_{2,3} \\ & & I_{3,3} \end{bmatrix}$$

of the symmetric 3x3 matrix [I] is stored columnwise in registers r10-r15 as indicated below:

Relative Address	Register No.	Matrix Element
0	r10	$I_{1,1}$
1	r11	$I_{1,2} = I_{2,1}$
2	r12	$I_{2,2}$
3	r13	$I_{1,3} = I_{3,1}$
4	r14	$I_{2,3} = I_{3,2}$
5	r15	$I_{3,3}$

The instructions 10 + 'T'(1,2) → H; rH → X would cause the value of $I_{1,2} = I_{2,1}$ to be assigned to the variable X.

4.1.6.3 Computations

See Figure 26.

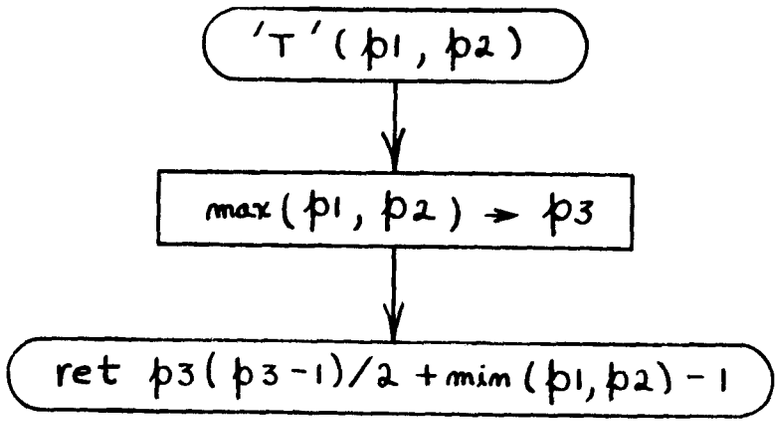


Figure 26 'T' Function Subprogram Logic Flow

4.2 MATRIX DIAGONALIZATION SUBROUTINE ('DIAG')

Given a symmetric matrix [M] of order C, the 'DIAG' subroutine solves the equation

$$[R]\,[M]\,[R]^T = [D]$$

for [R] and [D], where [D] is a diagonal matrix (one whose off-diagonal elements are all zero, within some specified tolerance).

4.2.1 Argument List

p1 = Order (size) of the input matrix [M].

p2 = Address of the input matrix [M], stored columnwise in upper triangular form.

p3 = Address of the diagonalized output matrix [D], stored columnwise in upper triangular form.

p4 = Address of the square (p1 x p1) output matrix [R], stored columnwise.

p5 = Tolerance on the maximum squared value of $D_{i,j}$, where $i \neq j$ (this input is optional; if not supplied by calling routine, it will be calculated by 'DIAG').

4.2.2 Example of Usage

Assume that the upper triangular form of the Shuttle's inertia tensor $[I]_B$ (referenced to the body coordinate system B) is stored columnwise in registers r10 - r15 as illustrated in Section 4.1.5.2. Then the instruction *cℓℓ 'DIAG'(3,10,10,1)* would cause $[I]_B$ to be replaced by $[I]_P$ (the same inertia tensor but now referenced to the coordinate system P, whose axes are the principal axes of inertia). Upon return from 'DIAG', the registers r1 - r9 would contain the 3x3 coordinate transformation matrix [R] that satisfies the

equations

$$[R]\,\bar{A}_P = \bar{A}_B$$

and

$$[R]^{-1}\,\bar{A}_B = [R]^T\,\bar{A}_B = \bar{A}_P,$$

where $\bar{A}$ is any arbitrary vector.

4.2.3 Computations

The method used to find the matrices [D] and [R] is of the type known as the Jacobi method (Reference 15). This consists of a series of matrix rotations of the form

$$[D] = ([R]_1[R]_2 \cdots)\,[M]\,(\cdots [R]_2^T[R]_1^T)$$

where each $[R]_k$ is selected so as to cause one pair of off-diagonal elements to be zero after the k^{th} rotation. An initial trigger level, δ, is computed from

$$\delta = \sqrt{\alpha}/c$$

where α is the sum of the squares of all off-diagonal elements and C is the order of [M]. The square of each off-diagonal element of [M] is compared to δ. When an element is encountered whose square is larger than δ, the matrix [M] is rotated so as to cause that off-diagonal element to be zero.

For example, suppose that the square of the (i,j)th element of [M] is greater than δ, then [M] is rotated by $[R]_1$ to get [M]' where

$$M'[i,j] = M'[j,i] = 0$$

The matrix $[R]_1$ has the form

$$R_1[i,i] = R_1[j,j] = \cos \phi$$

$$R_1[i,j] = - \sin \phi$$

$$R_1[j,i] = \sin \phi$$

and all other elements of $[R]_1$ are identical to the unit matrix [1]. Rotation with $[R]_1$, i.e.,

$$[M]' = [R]_1 \, [M] \, [R]_1^T$$

gives rise to the elements

$$M'[i,i] = M[i,i] \cos^2 \phi + 2M[i,j] \sin \phi \cos \phi + M[j,j] \sin^2 \phi$$

$$M'[j,j] = M[i,i] \sin^2 \phi - 2M[i,j] \sin \phi \cos \phi + M[j,j] \cos^2 \phi$$

$$M'[i,j] = M'[j,i] = (M[j,j] - M[i,i]) \sin \phi \cos \phi$$

$$+ M[i,j] (\cos^2 \phi - \sin^2 \phi)$$

from which it can be seen that

$$M'[i,j] = M'[j,i] = 0$$

provided ϕ is selected such that

$$\tan 2 \phi = 2M[i,j]/(M[i,i] - M[j,j])$$

After this rotation, scan of the off-diagonal elements continues using now, however, [M]'. It is true that any one rotation will cause other off-diagonal elements which were zero to become non-zero, but the trend is to reduce all off-diagonal elements to small numbers.

When the scan of all off-diagonal elements has been completed, the trigger level δ is reduced by dividing it again by C, and the process is repeated

until δ is less than a pre-set tolerance. The programmer may include this tolerance in the argument list or he may let 'DIAG' set the tolerance for him.

This process is shown in detail in Figures 27(a) and 27(b).

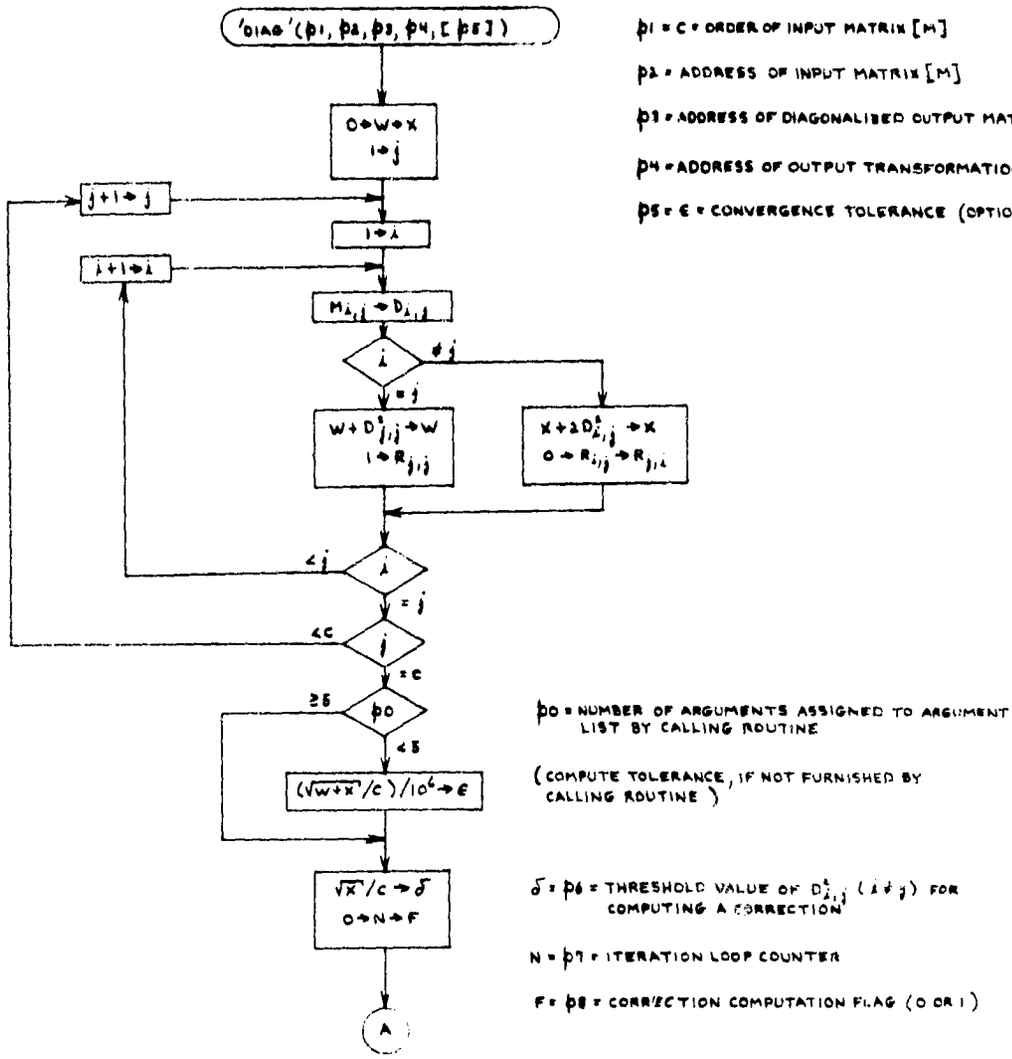


Figure 27a. 'DIAG' Subroutine Logic Flow

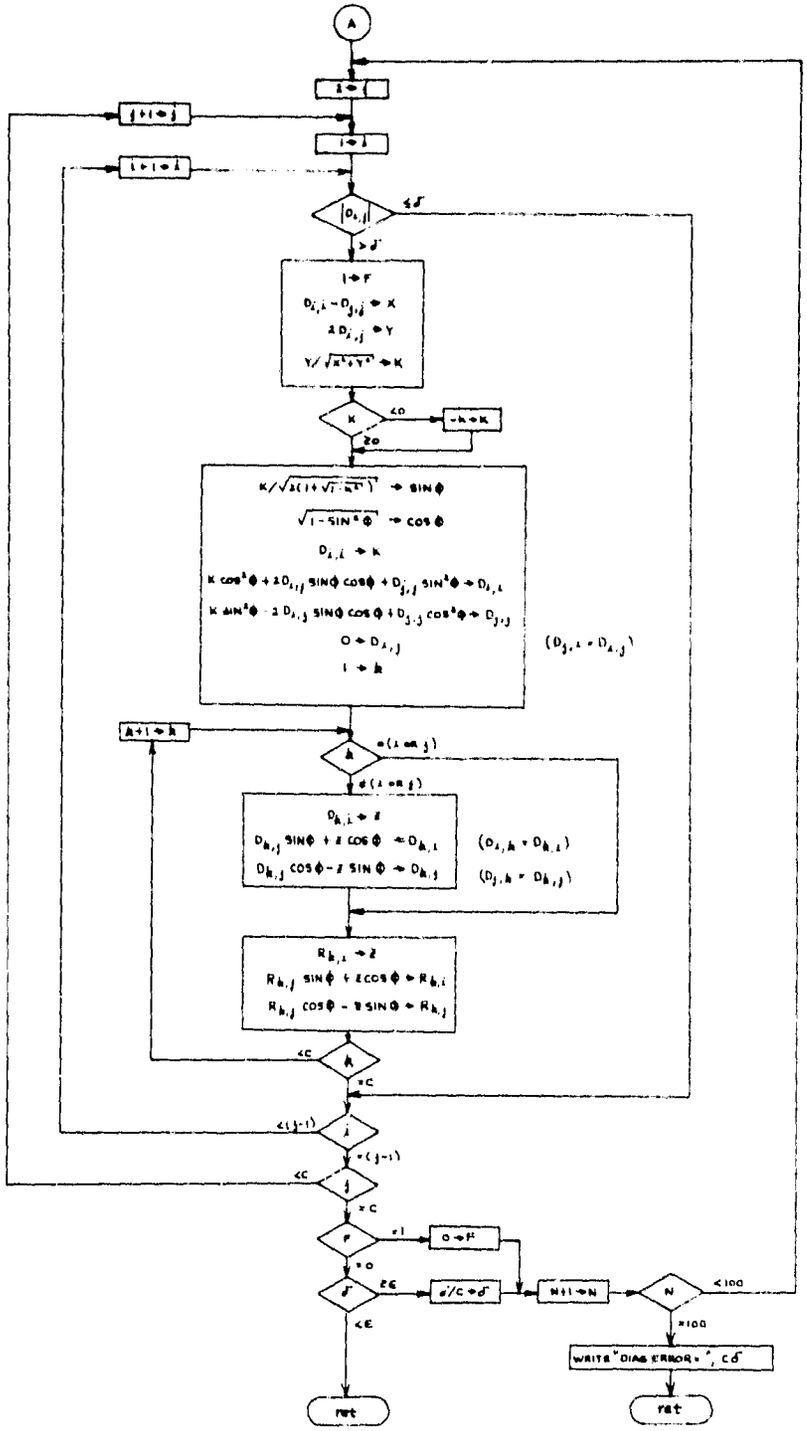
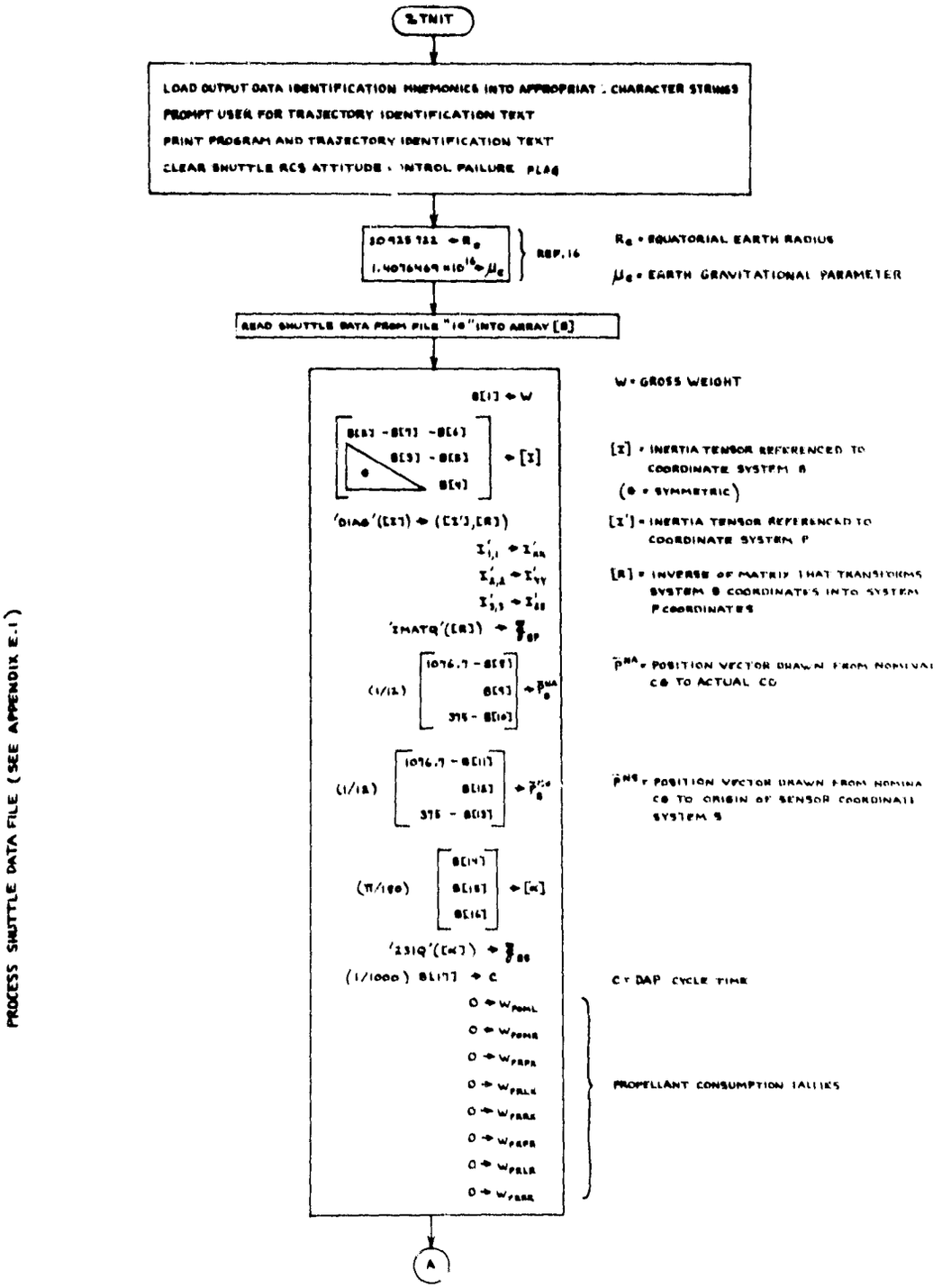


Figure 27b. 'DIAG' Subroutine Logic Flow (cont.)

4.3 MAIN LOGIC

The %TNIT computations are described by the flow chart contained in Figures 28a through 28g. The memory allocation table in Appendix D will have to be consulted for the purpose of correlating the symbols appearing in the flow chart with rhe r-register numbers appearing in the HPL code, which is contained in Appendix C.3.



PROCESS SHUTTLE DATA FILE (SEE APPENDIX E.1)

Figure 28a. %TNIT Logic Flow

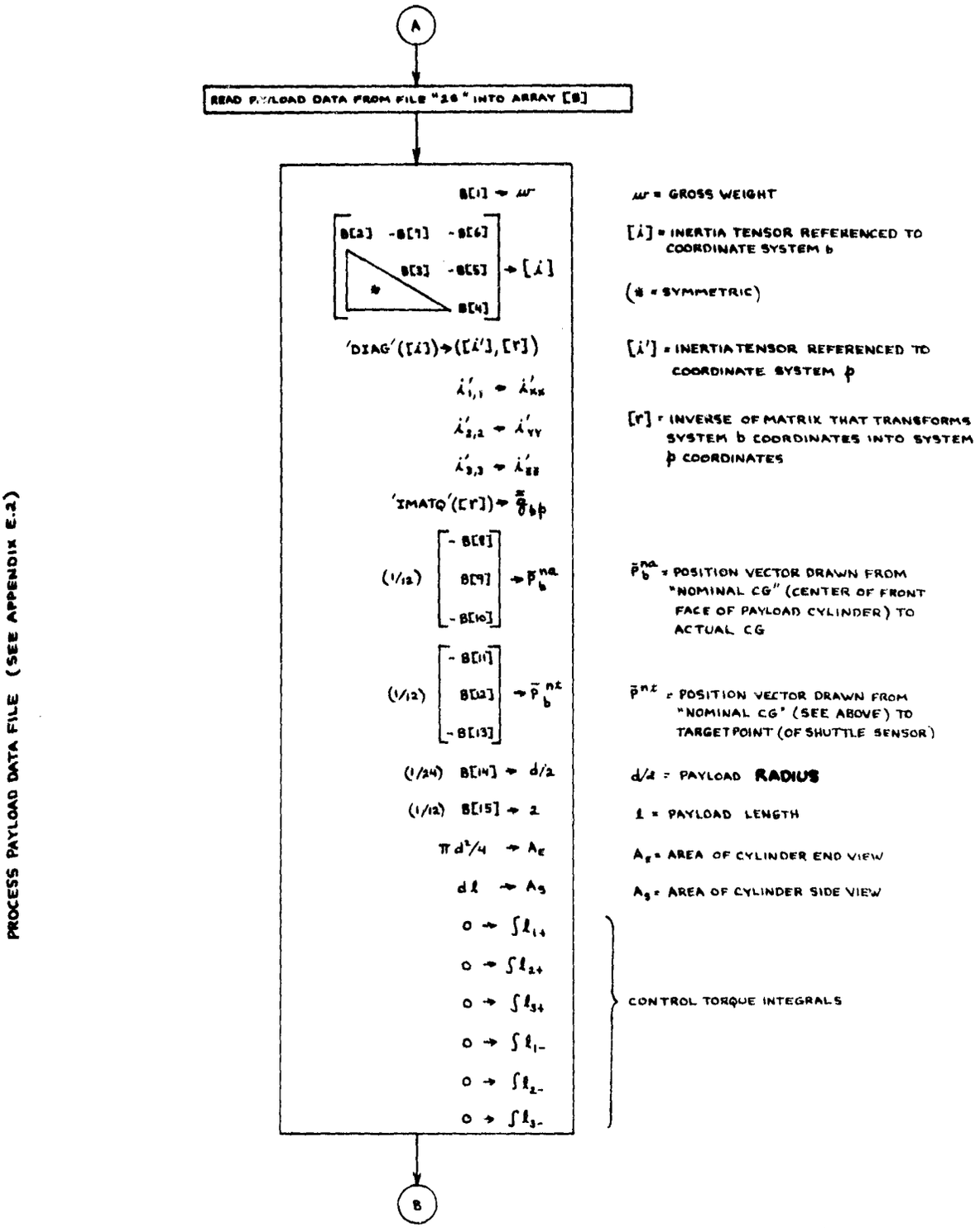


Figure 28b. %TNIT Logic Flow (cont.)

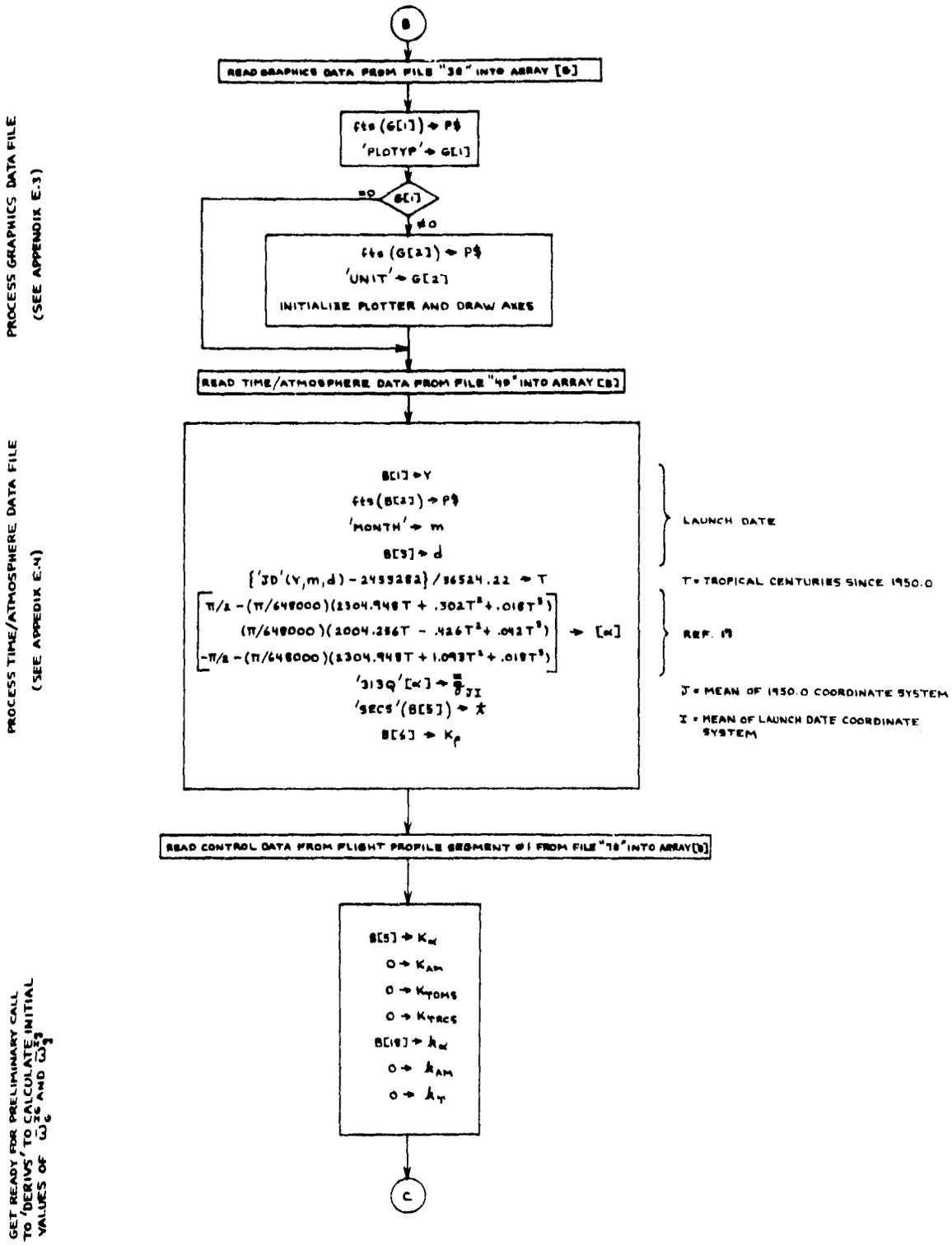


Figure 28c. %TNIT Logic Flow (cont.)

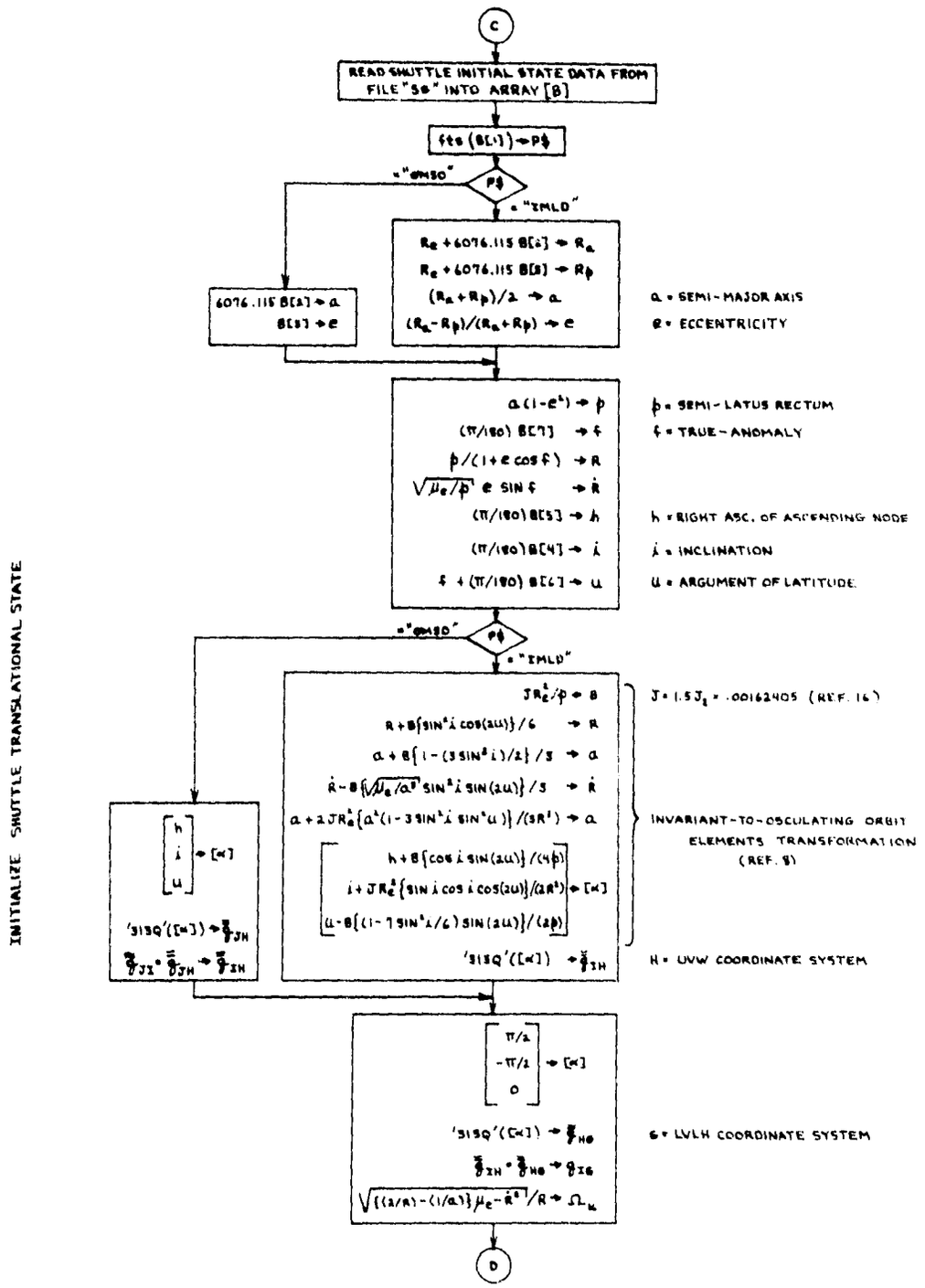


Figure 28d. %TNIT Logic Flow (cont.)

INITIALIZE SHUTTLE ROTATIONAL STATE

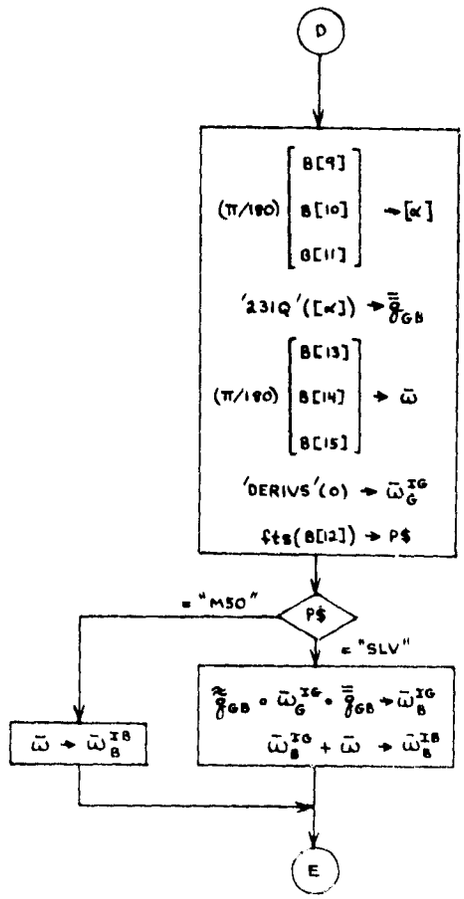


Figure 28e. %TNIT Logic Flow (cont.)

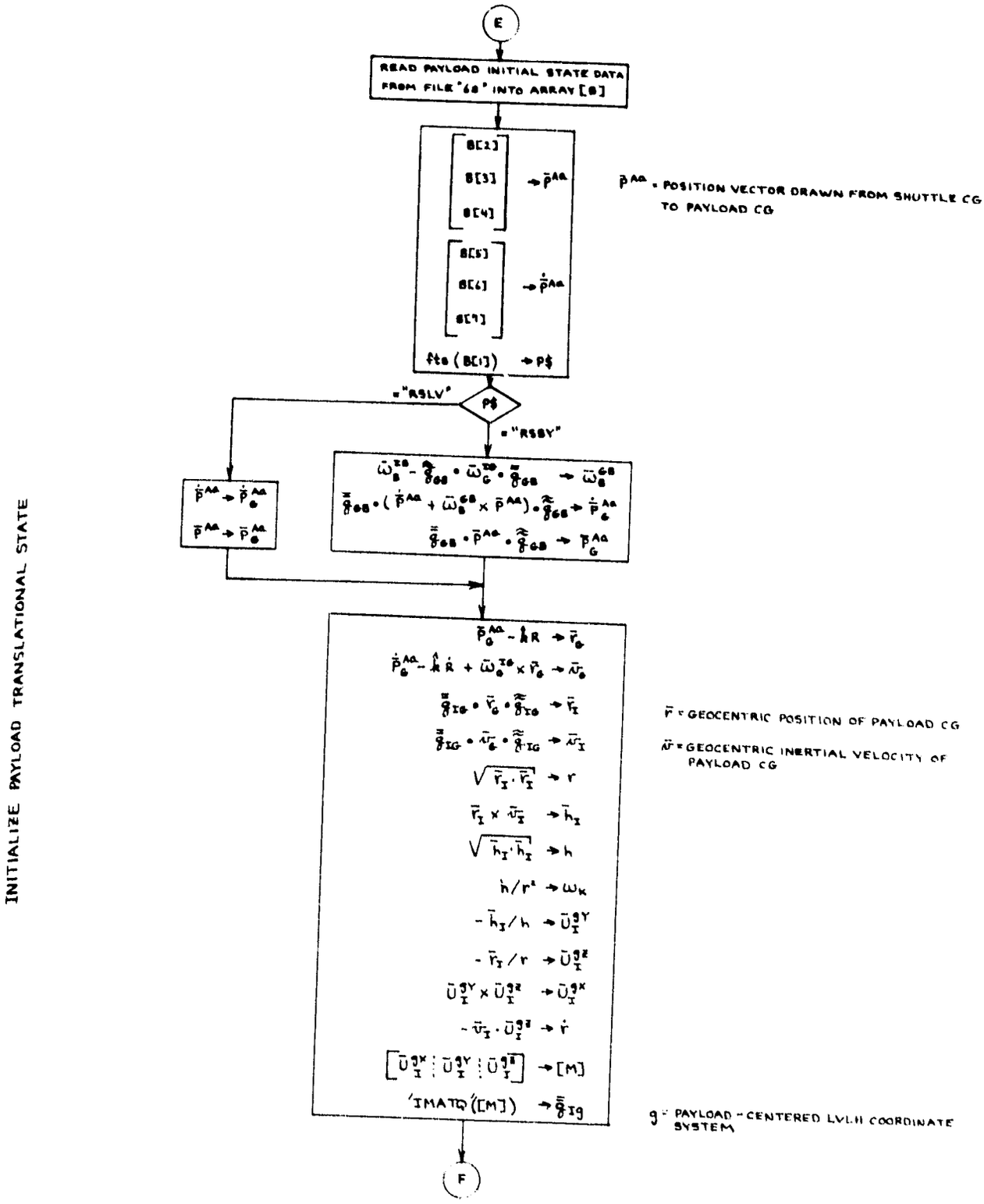


Figure 28f. STNIT Logic Flow (cont.)

INITIALIZE PAYLOAD ROTATIONAL STATE

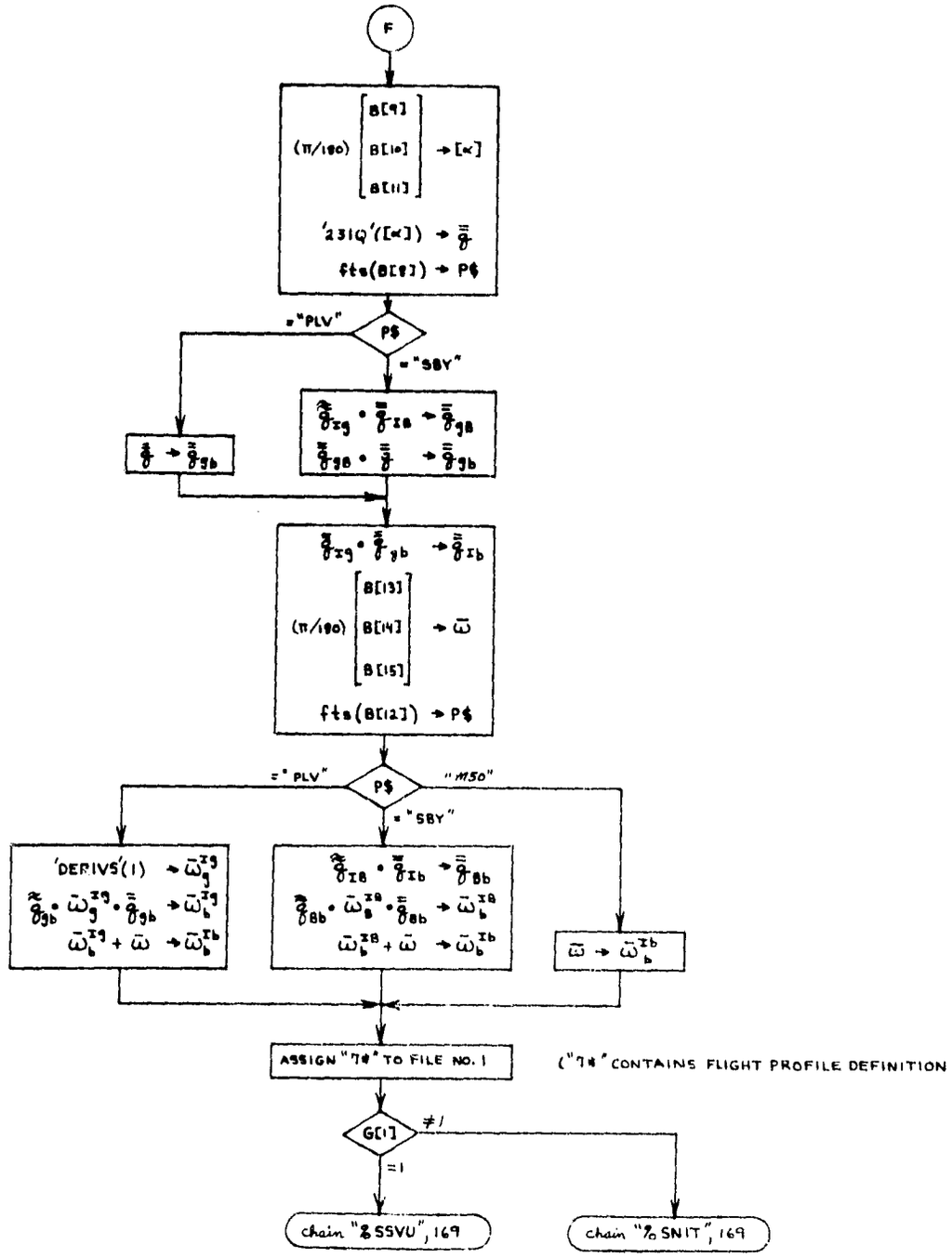


Figure 28g. %TNIT Logic Flow (cont.)

5. ORBITER GEOMETRY LINK (%SSVU)

The function of this link of the #TRAJ processor is to plot an end and/or a side view of the Shuttle Orbiter profile. It is executed only when the user specifies RSBY (Rectangular Shuttle Body-fixed coordinates) for the plot type, in the graphics data file (see Appendix E.3). In addition, it draws dashed lines that represent the nominal limits of visibility (through the overhead and aft windows) from the on-orbit pilot's control position. The code (Appendix C.4) is straightforward, for the most part consisting of HPL plt commands followed by the station coordinates (measured in inches) of the points that define the end and side profiles of the Orbiter. Typical Orbiter profile plots are illustrated in Appendix H.

6. FLIGHT SEGMENT INITIALIZATION LINK (%SNIT)

The %SNIT link is called into memory and executed at the beginning of every flight profile segment. Each time it is executed, it reads a new flight segment definition (see Appendix F) from file number 1, to which the name "7*" has been assigned in the %TNIT link.

After each new flight segment definition is read from disk, appropriate processing of the new input data is performed in preparation for state variable propagation through the ensuing flight segment. The propagation itself (involving numerical integration of the equations of motion) is performed by the %PROP link (Section 7). Upon completion of the segment initialization process, execution control is passed to %PROP by means of a chain instruction that causes the %SNIT code to be replaced by that of %PROP.

Trajectory integration is terminated (i.e., %PROP is not called into memory) when an end-of-file mark is encountered during the attempt to read a new flight segment definition. If a payload solid rocket motor (SRM) burn has been simulated, the Particle Impact Damage Integrator Processor (#PIDI) is called into memory immediately after the termination of trajectory integration. Otherwise, the user is asked whether he wants to start a new run. If the answer is yes ((CONTINUE)), the Data Base Editor Processor (#DBED) is called into memory. If the answer is no, ((0), (CONTINUE)), program execution is terminated by a stp command.

6.1 FUNCTION SUBPROGRAMS

6.1.1 'KAM'

Given the user-supplied mnemonic symbol that designates an attitude-maintenance option, 'KAM' returns the appropriate numeric code for internal use.

6.1.1.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'KAM' is executed.

6.1.1.2 Example of Usage

If the character string "LVRH" resides in P$, the instruction 'KAM' → K will cause K to be assigned the value 2.

6.1.1.3 Computations

See Figure 29.

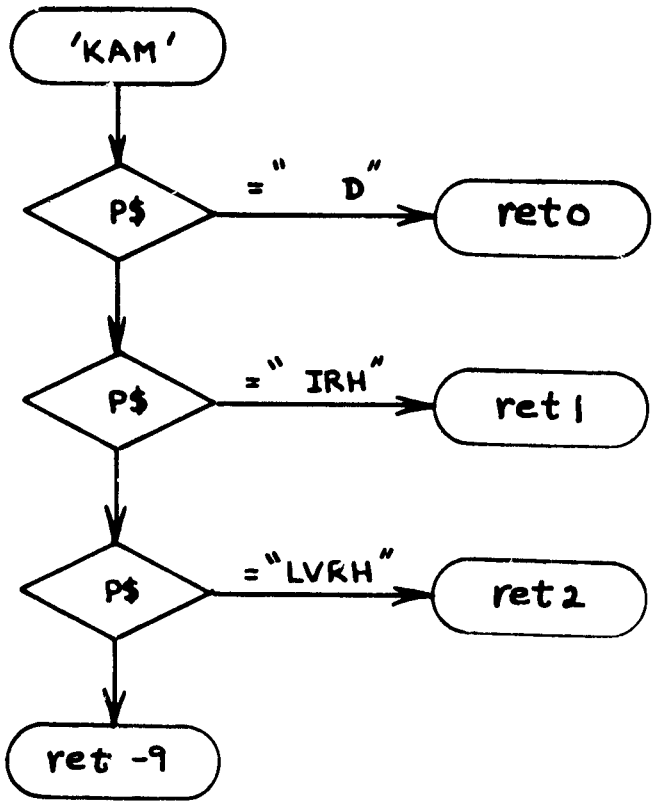


Figure 29 'KAM' Function Subprogram Logic Flow

6.1.2 'JSEL'

Given the mnemonic symbol that designates a Shuttle RCS jet-select table, 'JSEL' returns the appropriate numeric code for internal use.

6.1.2.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'JSEL' is executed.

6.1.2.2 Example of Usage

If the character string " P" resides in P$, the instruction 'JSEL' → J will cause J to be assigned the value 1.

6.1.2.3 Computations

See Figure 30.

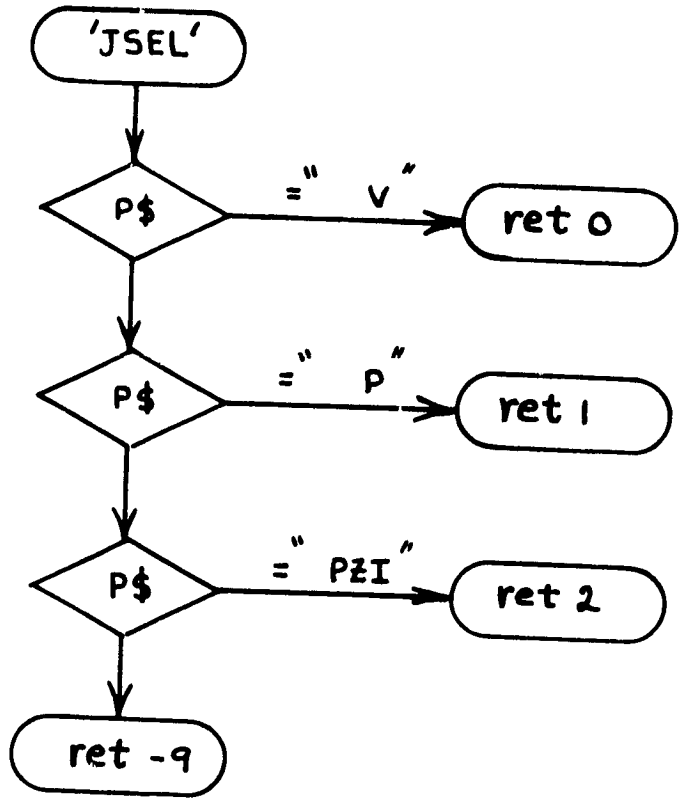


Figure 30 'JSEL' Function Subprogram Logic Flow

6.1.3 'COMP'

Given the mnemonic symbol that designates a Shuttle RCS cross-coupling compensation option, 'COMP' returns the appropriate numeric code for internal use.

6.1.4.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'COMP' is executed.

6.1.3.2 Example of Usage

If the character string "NONE" resides in P$, the instruction 'COMP' → C will cause C to be assigned the value zero (0).

6.1.3.3 Computations

See Figure 31.

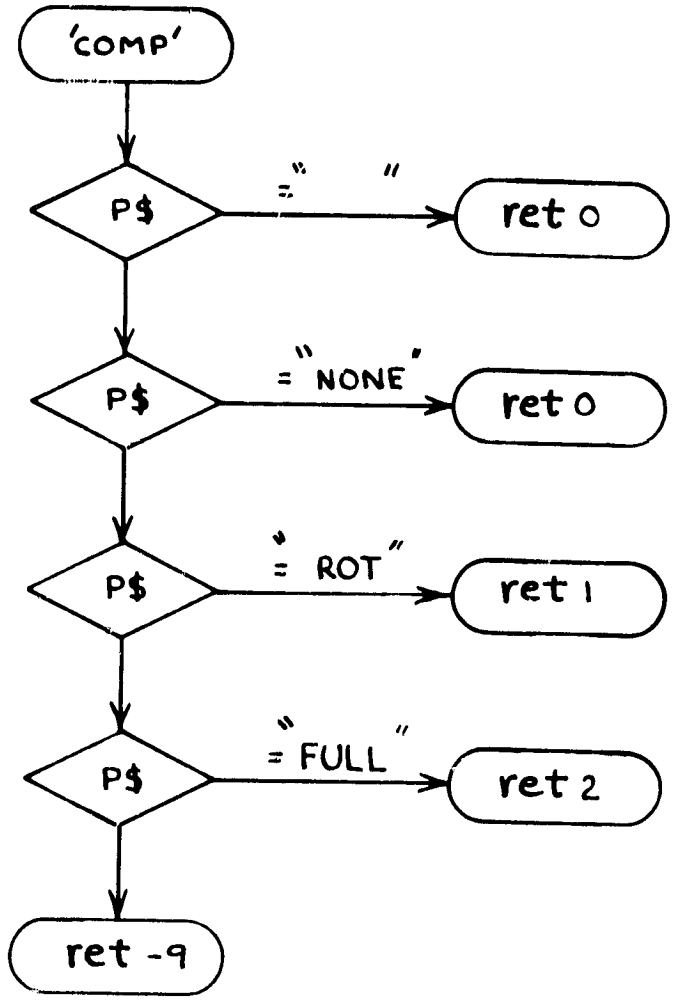


Figure 31 'COMP' Function Subprogram Logic Flow

6.1.4 'KTOMS'

Given the mnemonic symbol that designates which (if any) of the Shuttle OMS engines are to be fired firing a given flight profile segment, 'KTOMS' returns the appropriate numeric code for internal use.

6.1.4.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'KTOMS' is executed.

6.1.4.2 Example of Usage

If the character string " L+R" resides in P$, the instruction 'KTOMS' → K would cause K to be assigned the value 3.

6.1.4.3 Computations

See Figure 32.

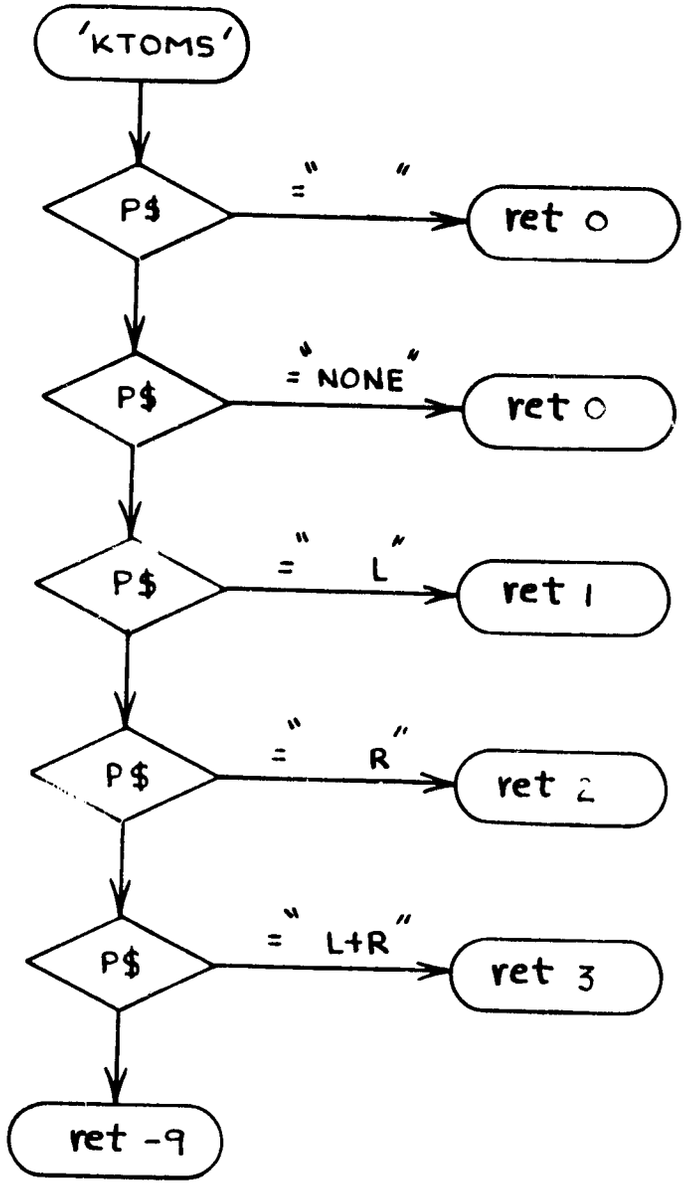


Figure 32 'KTOMS' Function Subprogram Logic Flow

6.1.5 'KTRCS'

Given the mnemonic symbol that designates a Shuttle RCS thrust command, 'KTRCS' returns the appropriate numeric code for internal use.

6.1.5.1 Argument List

None. The mnemonic symbol must be assigned to the string variable P$ before 'KTRCS' is executed.

6.1.5.2 Example of Usage

If the character string "+ROL" resides in P$, the instruction 'KTRCS' → K would cause K to be assigned the value 7.

6.1.5.3 Computations

See Figure 33.

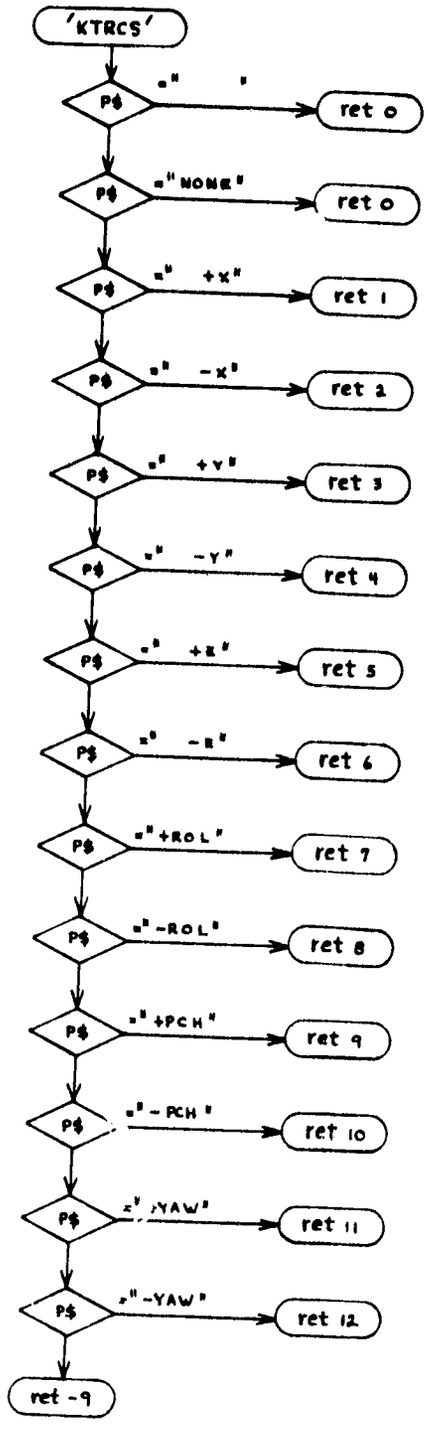


Figure 33. 'KTRCS' Function Subprogram Logic Flow

6.1.6 'DATYP'

HFRMP input data files contain three general types of data: non-integer numbers, integers, and mnemonic symbols (character strings). Given the item number of an entry in a flight segment definition (see Appendix F), 'DATYP' returns a numeric code which identifies the type of data so that it can be printed in the appropriate format. 'DATYP' also loads the appropriate identification text for the data item into the string variable B$, and the description of its unit of measurement into the string variable U$.

6.1.6.1 Argument List

p1 = Item number in the flight segment definition (see Appendix F).

6.1.6.2 Example of Usage

The instruction 'DATYP'(7) → I would cause I to be assigned the value 1, the character string "SS XB RATE OR INCR ....." to be loaded into B$, and the character string "DEG/SEC " to be loaded into U$.

6.1.6.3 Computations

The 'DATYP' code, which is straightforward and has no effect on trajectory computations, can be found in Appendix C.5.

6.2 SEGMENT DEFINITION LISTING SUBROUTINE ('SLIST')

The 'SLIST' subroutine is used to list each flight profile segment definition on the output line printer, immediately after it is read from the disk, in the format that is illustrated in Appendix F. The 'SLIST' code, which is straightforward and has no effect on trajectory computations, can be found in Appendix C.5.

6.3 MAIN LOGIC

The %SNIT computations are described by the flow chart contained in Figures 34a through 34f. The memory allocation table in Appendix D will have to be consulted to correlate the logical symbols appearing in the flow chart with the r-register numbers that appear in the HPL code, which is contained in Appendix C.5.

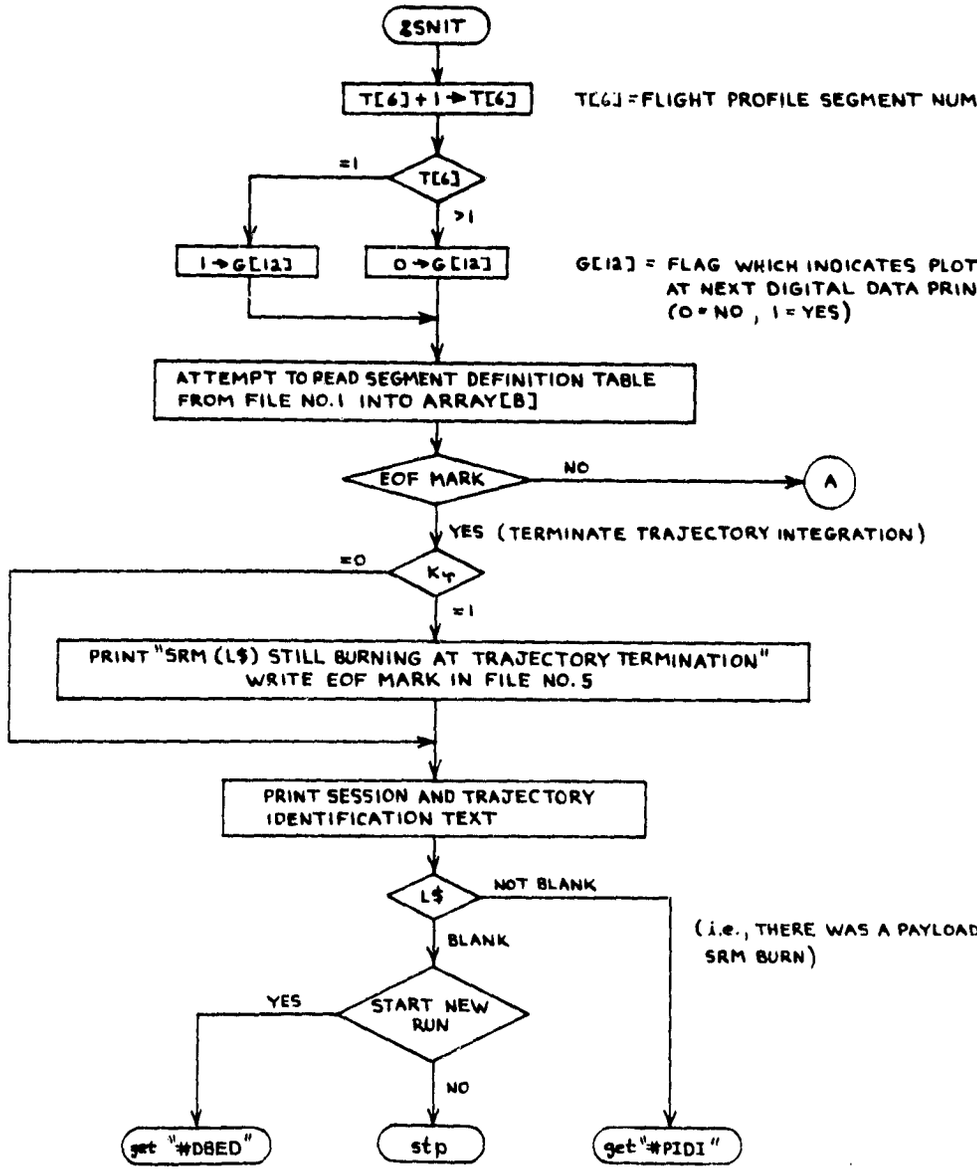


Figure 34a. %SNIT Logic Flow

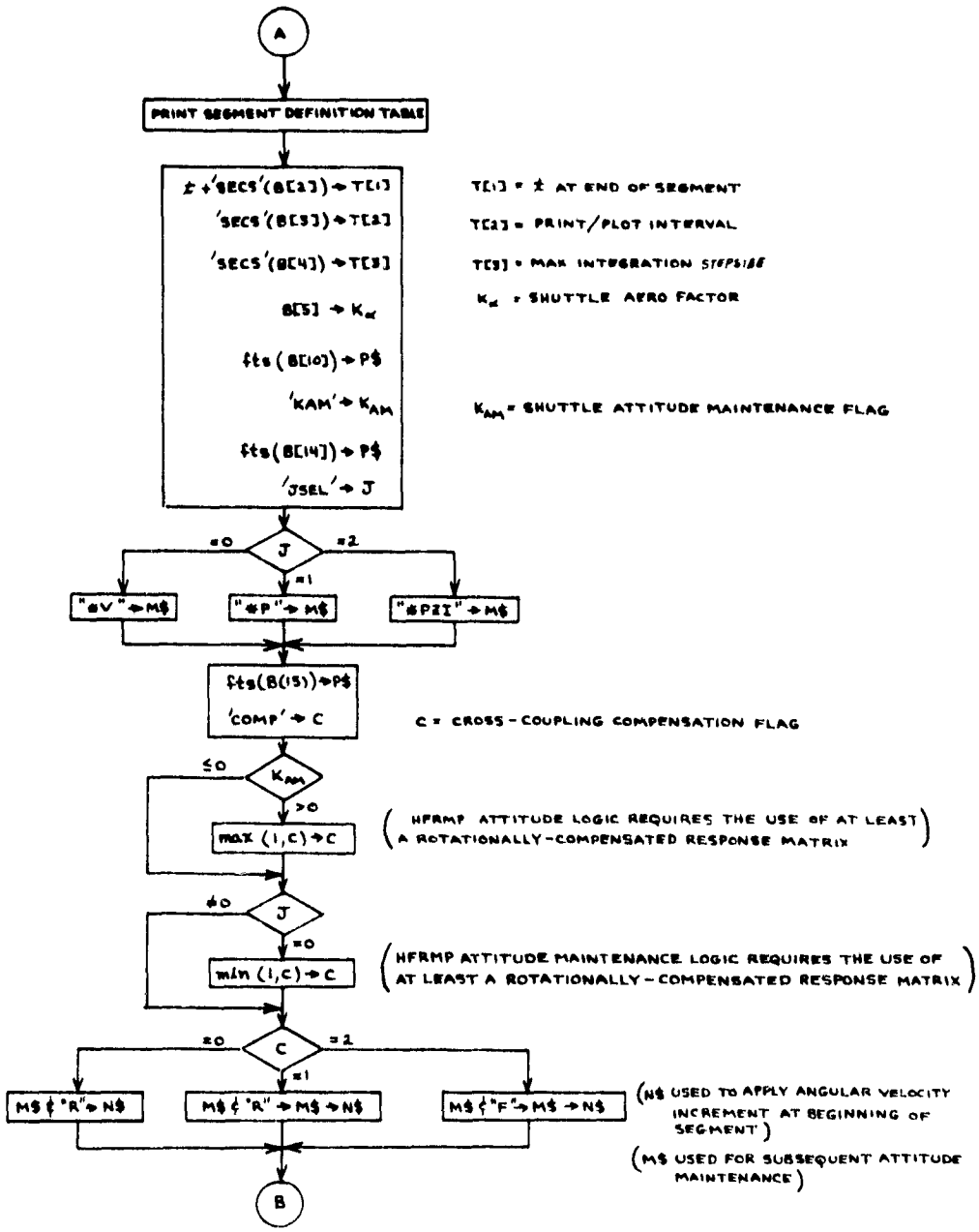


Figure 34b. %SNIT Logic Flow

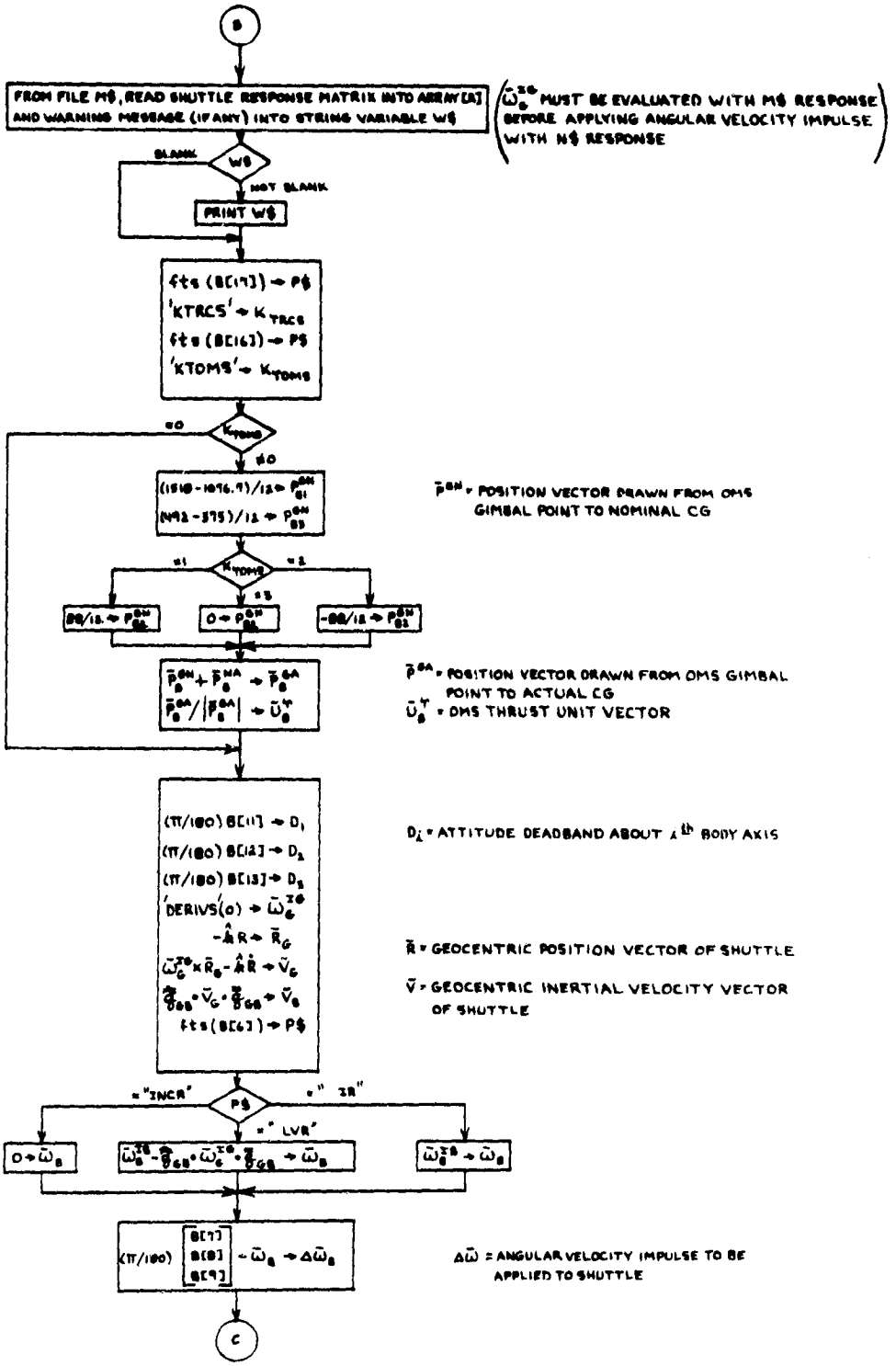


Figure 34c. %SNIT Logic Flow (cont.)

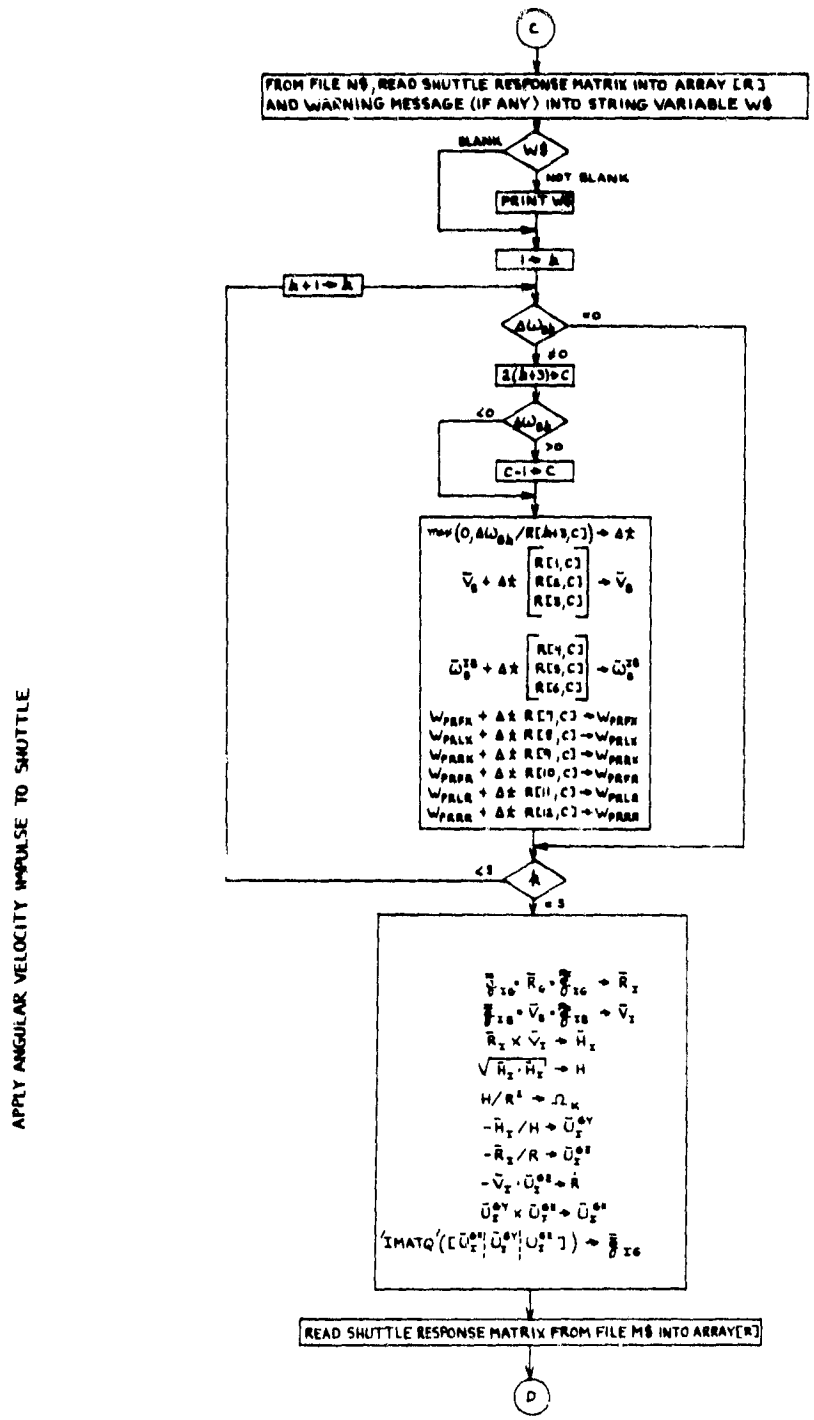


Figure 34d. %SNIT Logic Flow (cont.)

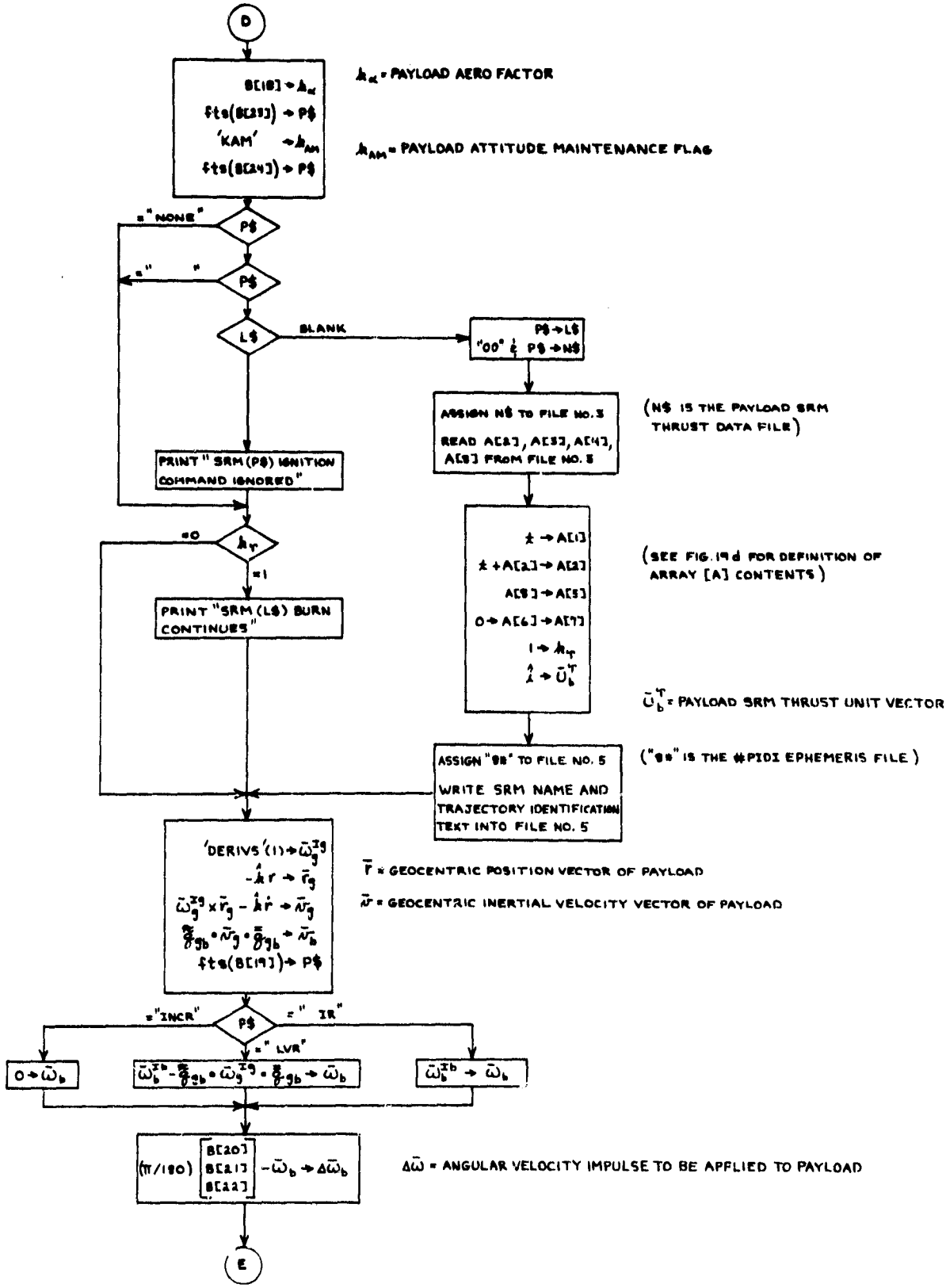


Figure 34e. %SNIT Logic Flow (cont.)

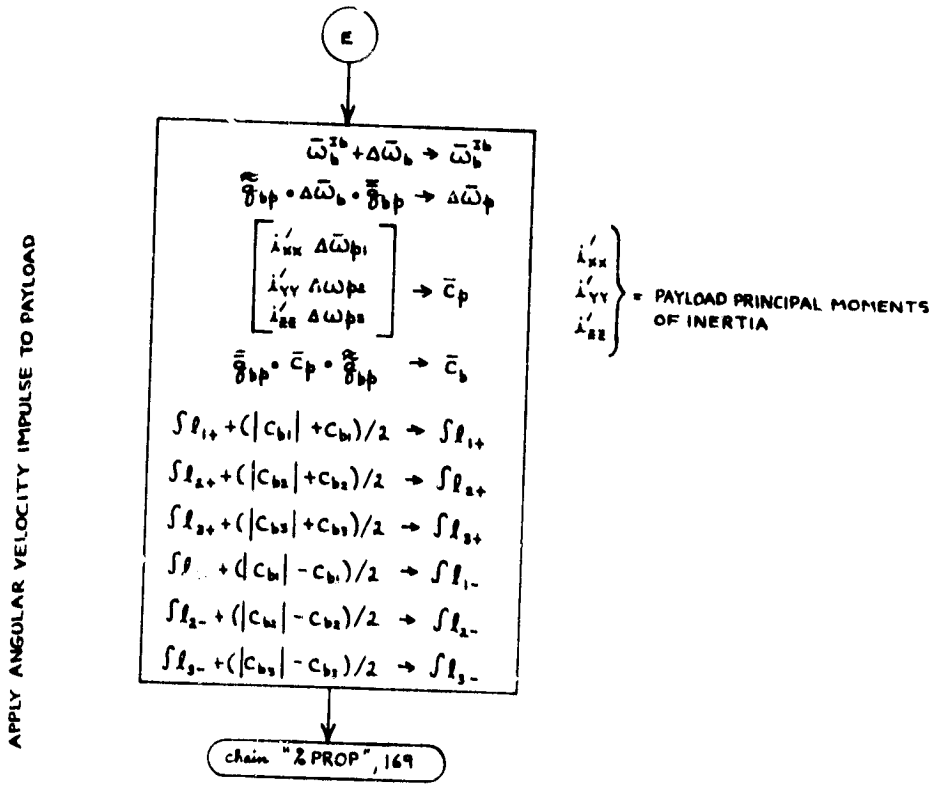


Figure 34f. %SNIT Logic Flow (cont.)

7. STATE PROPAGATION LINK (%PROP)

The function of the %PROP link is to propagate the state of the Shuttle/payload system through a single flight profile segment. Digital output data describing the state of the system are printed at the beginning and the end of the segment, and at user-specified intervals between these two points. If the user has specified "RSBY" (Rectangular Shuttle Body-fixed coordinates) or "CPLV" (Curvilinear Payload Local Vertical coordinates) for the PLOT TYPE (in the graphics data file, Appendix F-3), then the appropriate graphical data are also plotted at each data output point. Upon reaching the end of the flight profile segment, %PROP passes execution control back to %SNIT.

7.1 ROTATED-ELLIPSE PLOTTING SUBROUTINE ('RELIP')

When the user selects the "RSBY" plot type, %PROP draws one or two pictures of the payload (depending on how many views are specified) at each data output point. These pictures represent orthogonal projections of the payload's cylindrical outline into the X-Z and/or the Y-Z planes of the Shuttle's body axes. The pictures are composed of straight lines (corresponding to the sides of the cylinder) and ellipses and elliptical arcs (corresponding to canted views of the circles that represent the ends of the cylinder).

The ellipses and elliptical arcs are drawn by using the HPL ofs instruction to move the origin of plotter coordinates to the appropriate point on the HP-9872A plotting surface, and then by calling 'RELIP' to draw an ellipse (or portion thereof) about the offset origin. The geometry of the ellipse, referenced to the offset origin of plotter coordinates, is shown in Figure 35.

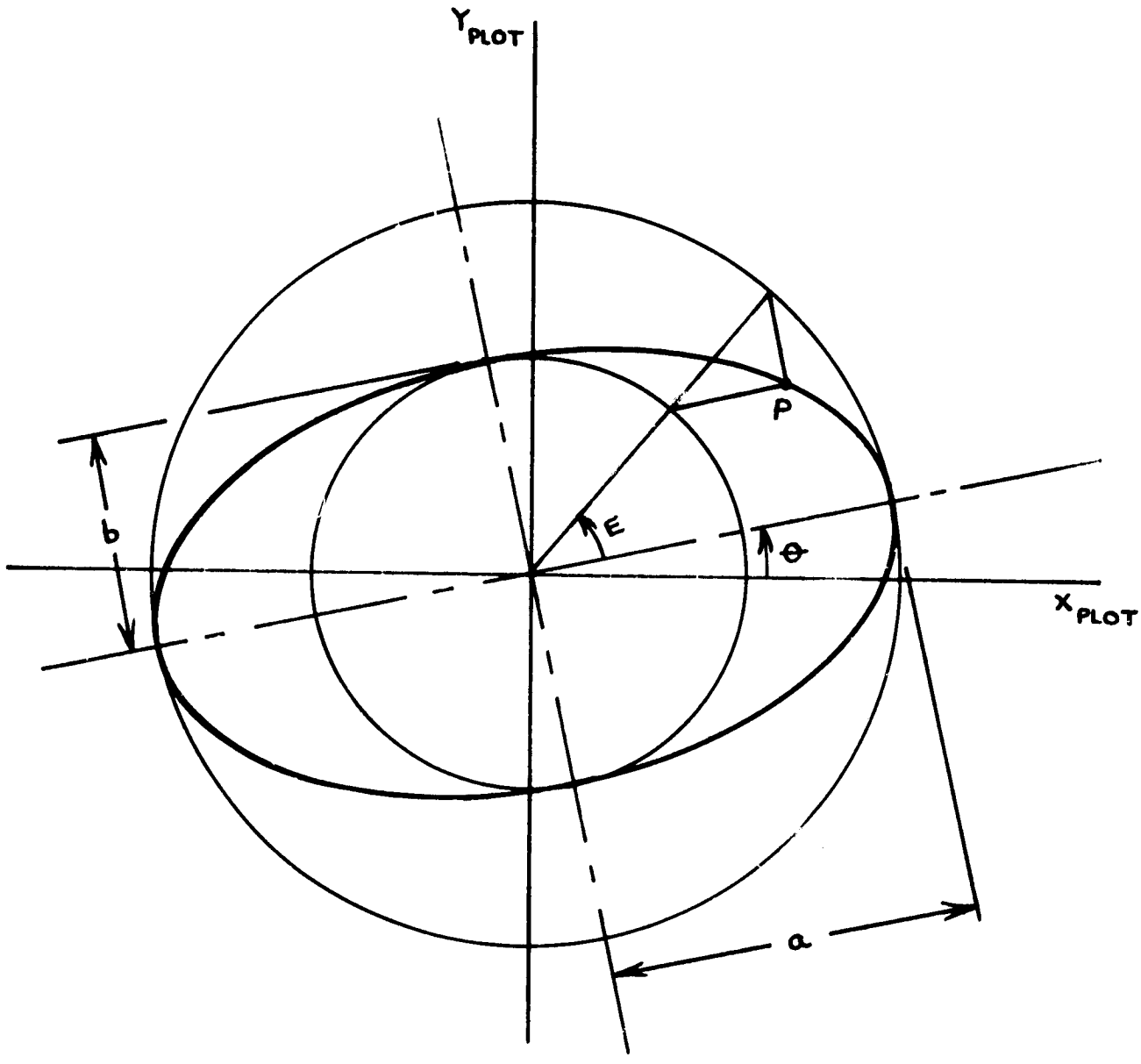


Figure 35. 'RELIP' Ellipse Geometry

7.1.1 Argument List

p1 = a = Semi-major axis of ellipse.

p2 = b = Semi-minor axis of ellipse.

p3 = E_F = Eccentric anomaly of first point on elliptical arc.

p4 = $\Delta E = E_L - E_F$, where E_L = eccentric anomaly of last point on elliptical arc.

p5 = Number of chords to be drawn between the first and last points, for the purpose of approximating the true arc.

p6 = cos θ | , where θ = ellipse rotation angle, measured from X_{PLOT} axis

p7 = sin θ | to semi-major axis.

7.1.2 Examples of Usage

Suppose that it is necessary to draw the upper half of the ellipse shown in Figure 35, centered on the Shuttle body-fixed coordinates $X_B = 20$, $Z_B = -50$. We will assume that the relationship between the HP-9872A plot coordinates (X_{PLOT}, Y_{PLOT}) and the Shuttle coordinates (X_B, Z_B) has already been defined by the execution of an HPL scl instruction. We will further assume that the lengths of the semi-major and semi-minor axes, respectively, reside in registers A and B. The values of cosθ and sinθ we assume to reside in registers C and S, respectively. The necessary instructions for drawing the desired arc are then pen; ofs 20, -50; cℓℓ 'RELIP'(A,B,0,π,N,C,S), where N is the number of chords that are to be used to approximate the true shape of the semi-ellipse.

The 'RELIP' subroutine can be used to draw figures other than elliptical arcs. For instance, the instruction pen; cℓℓ 'RELIP'(R,R,0,2π,40,1,0) would cause a circle of radius R (approximated by 40 chords of equal length) to be drawn about the origin of coordinates. If this were followed by the instructions pen; cℓℓ 'RELIP'(R,R,π/2,2π,3,1,0) an equilateral triangle would then be inscribed within the circle.

7.1.3 Computations

The coordinates of any particular point P, on the ellipse shown in Figure 35, are given by the equations

$$X_{PLOT} = a \cos E \cos\theta - b \sin E \sin\theta$$

and

$$Y_{PLOT} = a \cos E \sin\theta + b \sin E \cos\theta.$$

As indicated by the flow chart in Figure 36, these equations are embedded in a loop and evaluated at regular intervals of E. The elliptical arc is approximated by drawing straight lines (chords) between the points thus defined.

Since a great number of chords may be required to obtain an accurate approximation of the true arc, the trigonometric identities

$$\sin (E + \delta E) = \sin E \cos \delta E + \cos E \sin \delta E$$

and

$$\cos (E + \delta E) = \cos E \cos \delta E - \sin E \sin \delta E$$

are used within the loop. This avoids repetitive references to the sin and cos functions, which could result in excessive execution time requirements.

It should be noted that the 'RELIP' subroutine makes use of the volatile simple variables (H,I,J,K and W,X,Y,Z); therefore, values assigned to those registers by the calling routine will be lost upon execution of 'RELIP'.

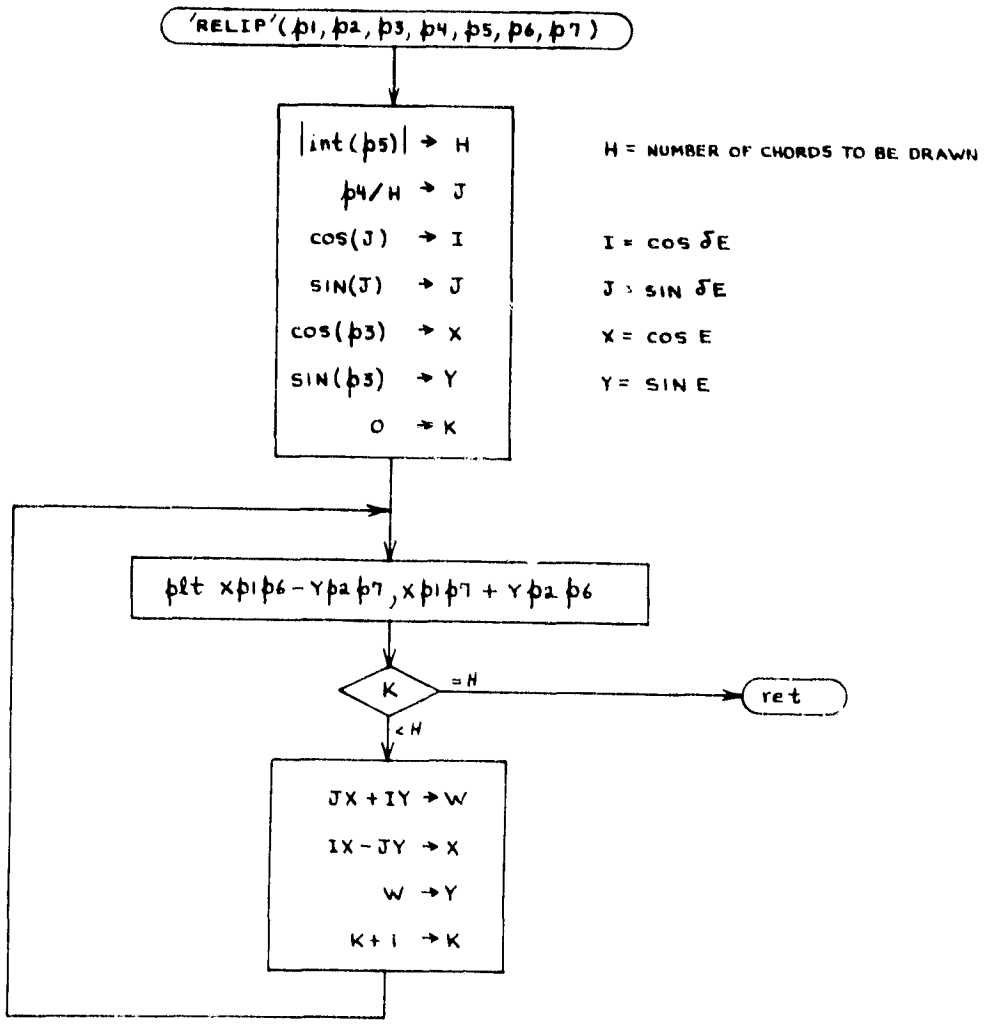


Figure 36. 'RELIP' Subroutine Logic Flow

7.2 FOURTH-ORDER RUNGE-KUTTA INTEGRATION SUBROUTINE ('RK4')

The function of the'RK4' subroutine is to propagate the state of the Shuttle/payload system across one integration time step, using the fourth-order method of Runge-Kutta.

7.2.1 Input Data

7.2.1.1 Argument List

pl = h = Value of time step.

7.2.1.2 Others

The 'RK4' routine calls 'DERIVS'; therefore, all the 'DERIVS' input data listed in Sections 2.3.1.2 and 2.3.1.3 must have been defined before calling 'RK4'.

7.2.2 Output Data

'RK4' updates the contents of the extended array of state variables (r25-r74), which will be referred to symbolically in this section as [X]. It should be noted that time is one of the state variables in the HFRMP, and is integrated just like any other state variable. It should also be noted that, although the contents of the derivatives array (r125-r174) will change as a result of executing 'RK4', the values that reside there upon return from 'RK4' do not represent the true derivatives at the end of the time step. To obtain the true derivatives, it is necessary to call 'DERIVS' again after executing 'RK4'.

7.2.3 Example of Usage

The instruction *cll* 'RK4'(300) would cause the state of the system to be advanced 300 seconds.

7.2.4 Computations

Let $[\dot{X}]$ represent the array of first derivatives (with respect to time) of the state variables in the array $[X]$. Let $[X]_n$ represent the state of the system at time t_n, and $[X]_{n+1}$ represent the state at t_{n+1}, where

$$h = t_{n+1} - t_n$$

is some relatively small time interval. Since $[\dot{X}] = f([X])$, then according to the fourth-order method of Runge-Kutta we can write

$$[X]_{n+1} = [X]_n + ([k]_1 + 2[k]_2 + 2[k]_3 + k[4])/6,$$

where

$$[k]_1 = h\ f([X]_n)$$

$$[k]_2 = h\ f([X]_n + \tfrac{1}{2}\ [k]_1)$$

$$[k]_3 = h\ f([X]_n + \tfrac{1}{2}\ [k]_2)$$

and

$$[k]_4 = h\ f([X]_n + [k]_3).$$

The error introduced into the system state by a single fourth-order RK integration step is on the order of

$$\frac{d^5\ [X]}{d\ t^5}\ \frac{h^5}{5!}\ .$$

As a rule of thumb, it has been found in using the HFRMP that the integration stepsize should always satisfy the relationship

$$h \leq 40 \text{ degrees}/\omega_{max}.$$

The symbol ω_{max} represents the angular velocity magnitude, measured in degrees per second, of whichever one of the four state-variable reference coordinate systems (B,b,G, or g) is rotating most rapidly with respect to inertial space. That is to say, no reference coordinate system should ever be allowed to rotate more than 40 degrees during a single integration step. In some cases it may be necessary to reduce the single-step rotation limit to 20 or even 10 degrees to achieve the desired integration accuracy.

A flow chart of the 'RK4' subroutine is shown in Figure 37. The [Y] array is stored in registers r175-r224, and the [Z] array is stored in registers r225-r274. Each quaternion in the [X] array (only) is normalized, each time that array is updated, by means of the computational sequence

$$\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \rightarrow m$$

$$q_0/m \rightarrow q_0$$

$$q_1/m \rightarrow q_1$$

$$q_2/m \rightarrow q_2$$

$$q_3/m \rightarrow q_3.$$

The quaternion normalization procedure is never applied to the [Y] and the [Z] arrays; to do so would introduce a systematic integration error.

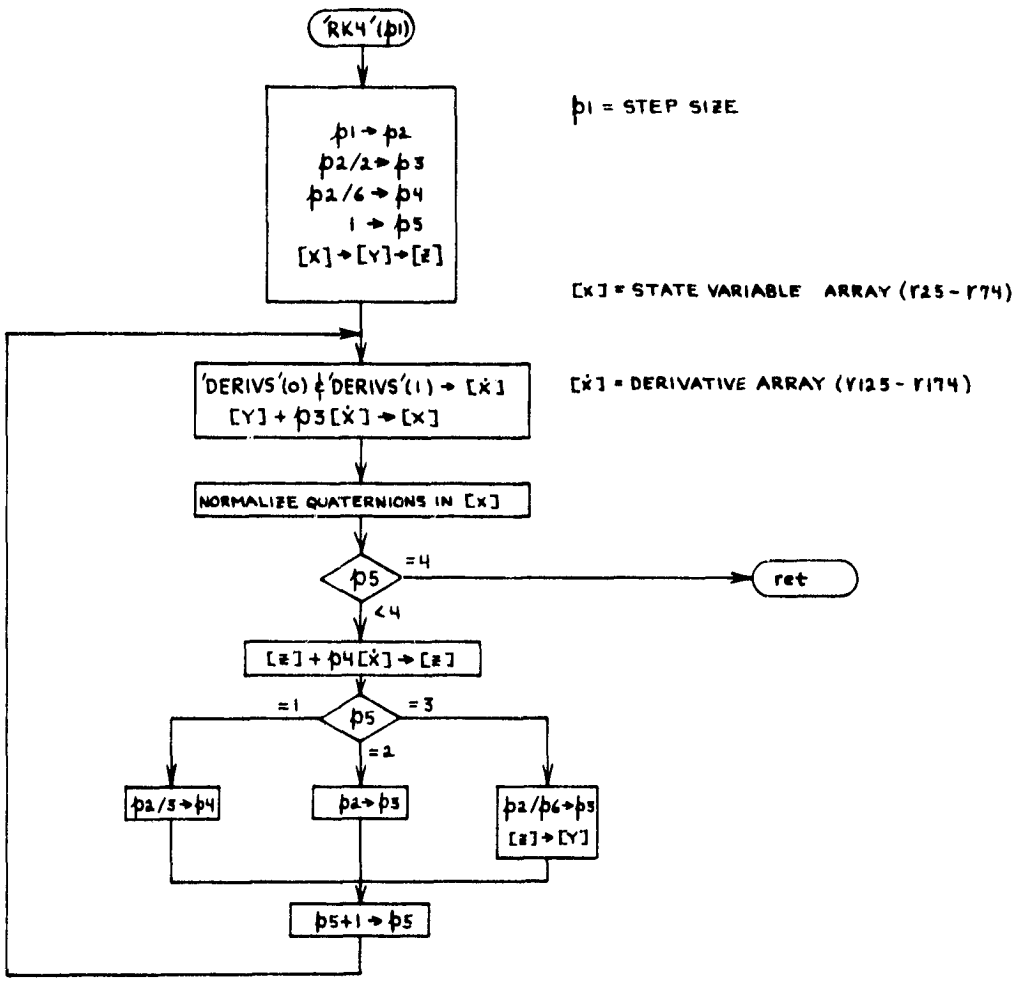


Figure 37. 'RK4' Subroutine Logic Flow

7.3 MAIN LOGIC

The %PROP computations are described by the flow chart contained in Figures 38 a through 38h. The memory allocation table in Appendix D will have to be consulted to correlate the logical symbols appearing in the flow chart with the r-register numbers that appear in the HPL code, which is contained in Appendix C.6.

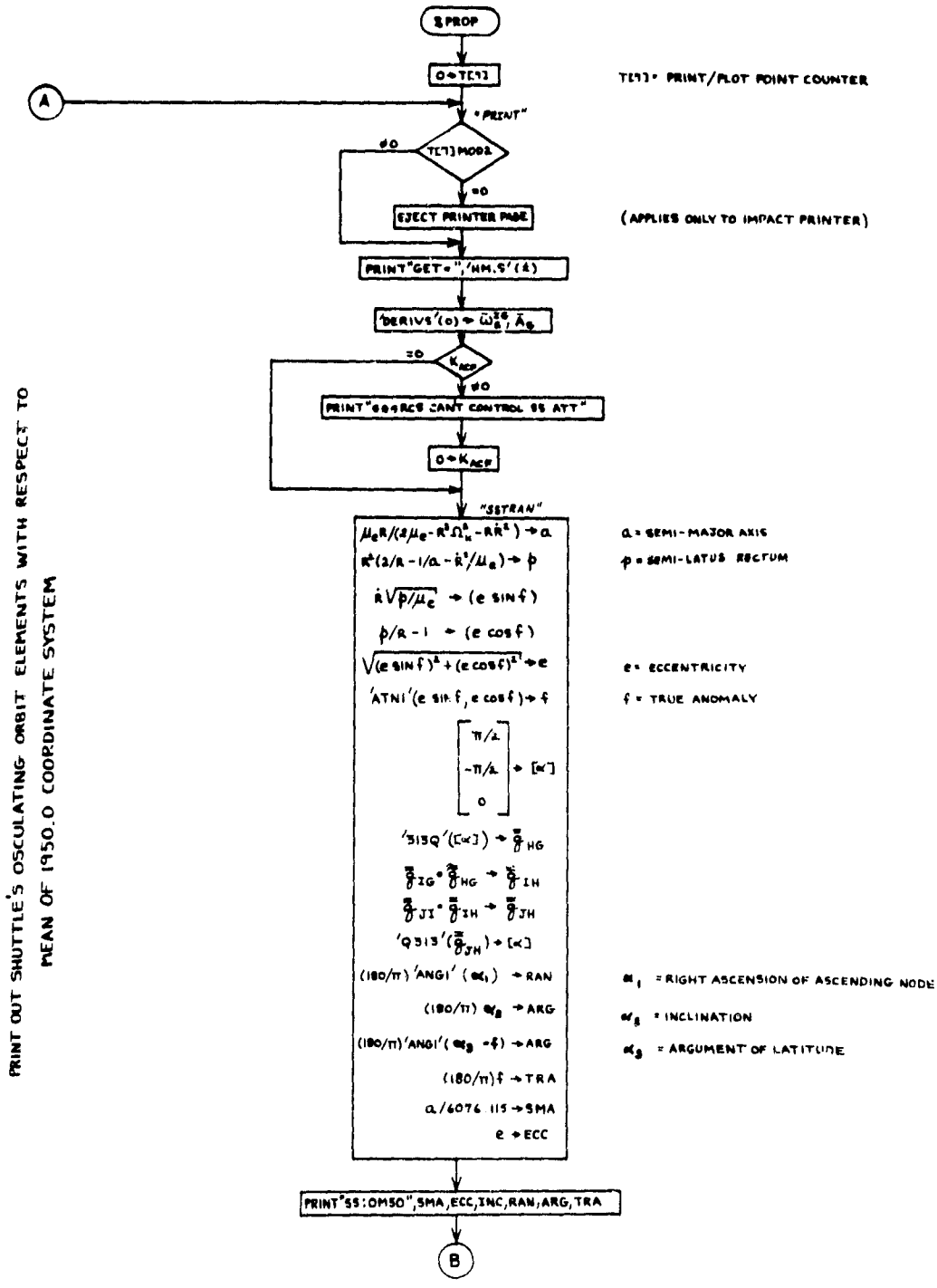


Figure 38a. %PROP Logic Flow

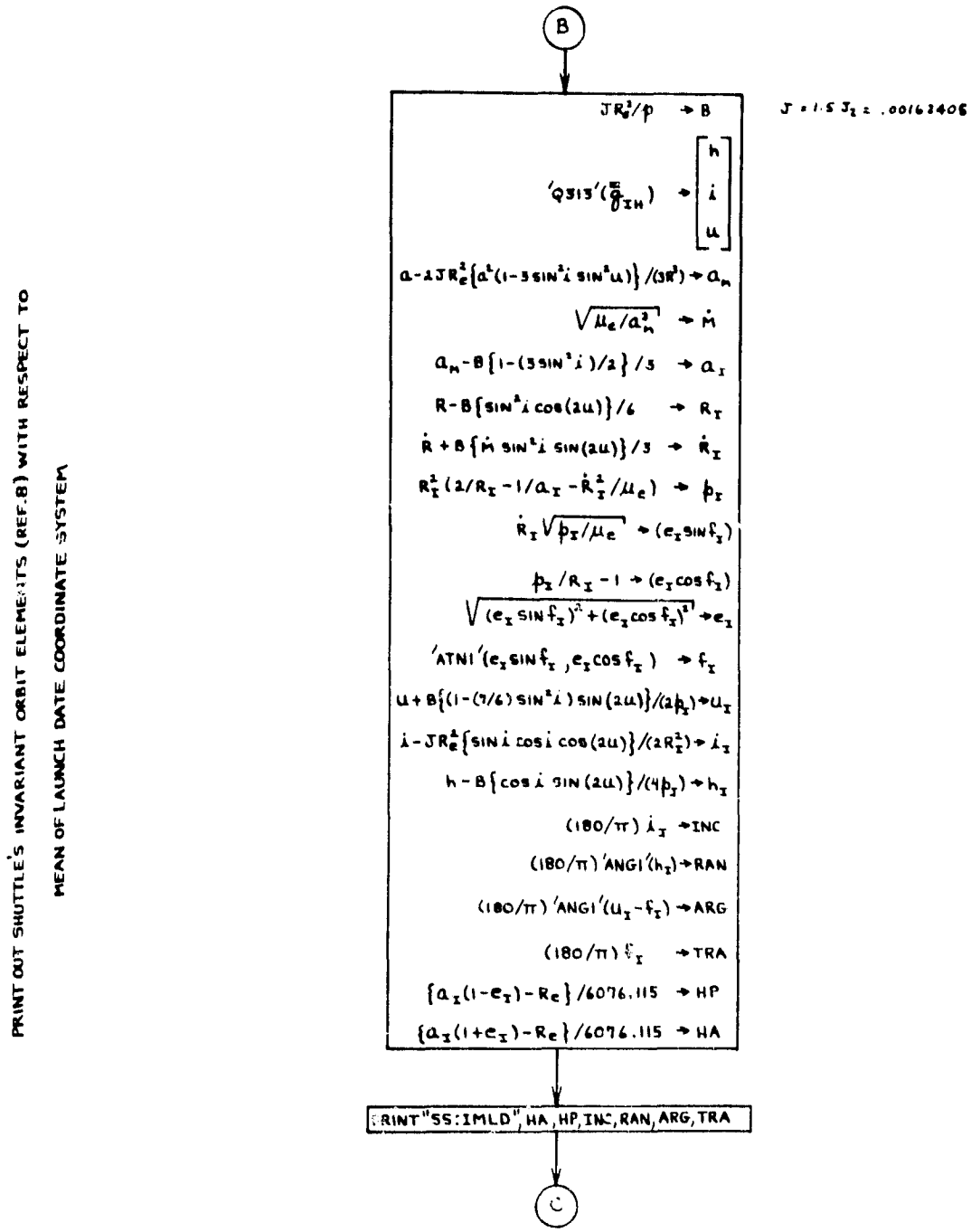


Figure 38b. %PROP Logic Flow (cont.)

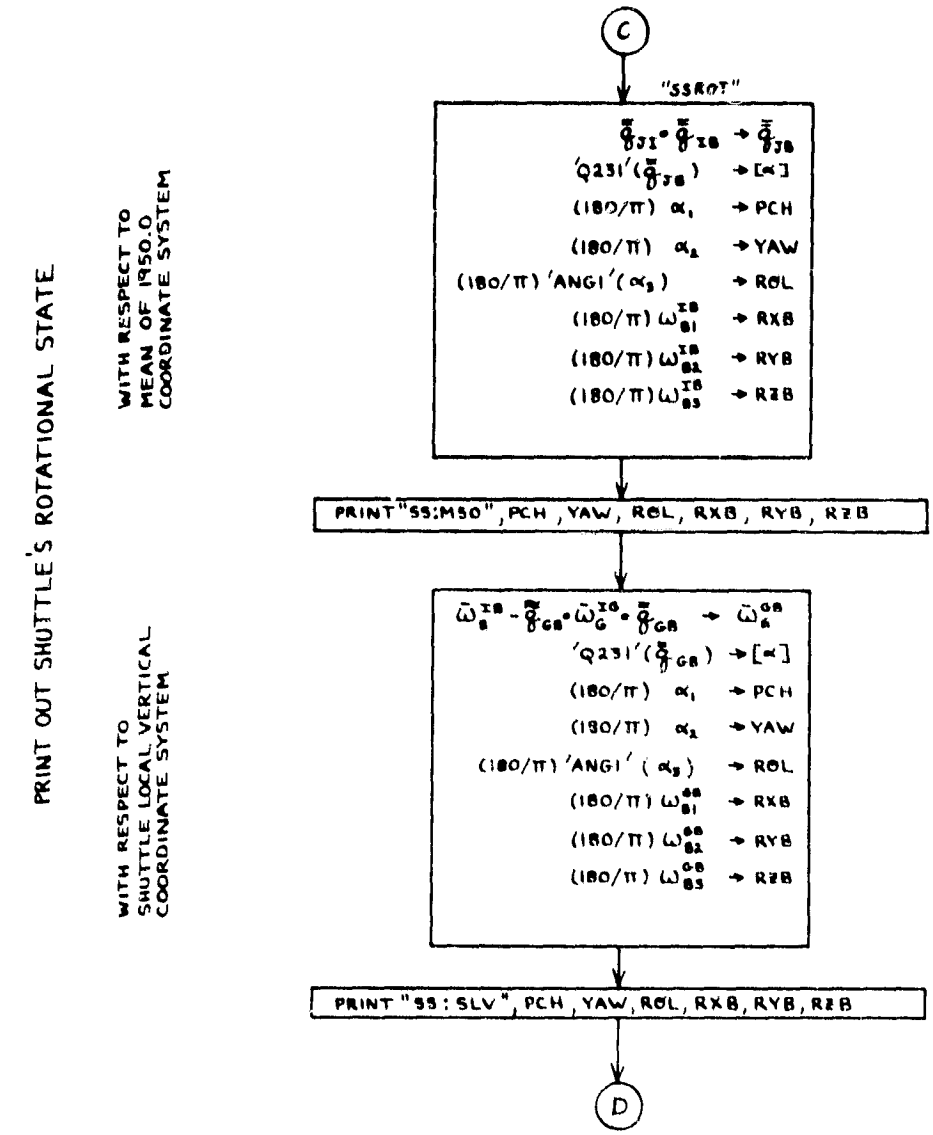


Figure 38c. %PROP Logic Flow (cont.)

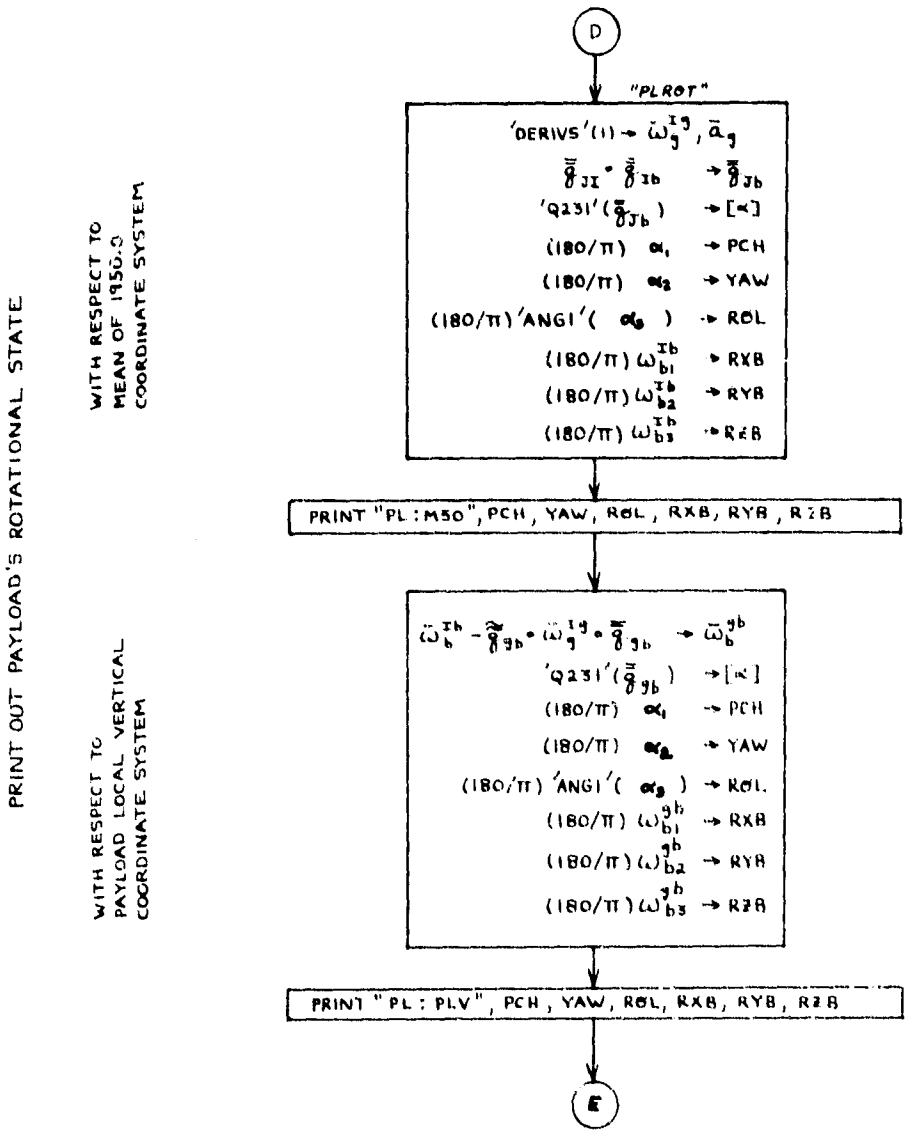


Figure 38d. %PROP Logic Flow (cont.)

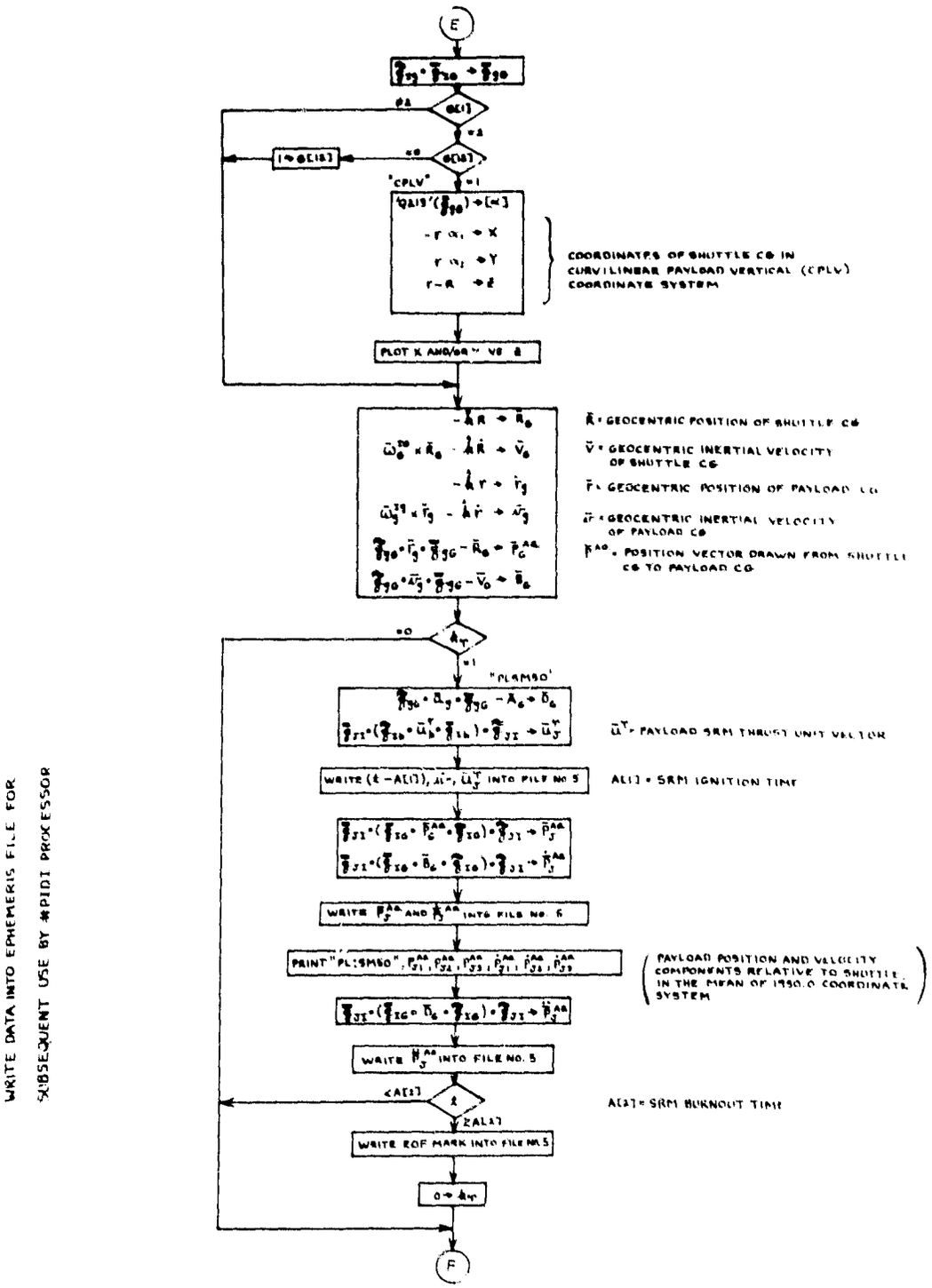


Figure 38e. %PROP Logic Flow (cont.)

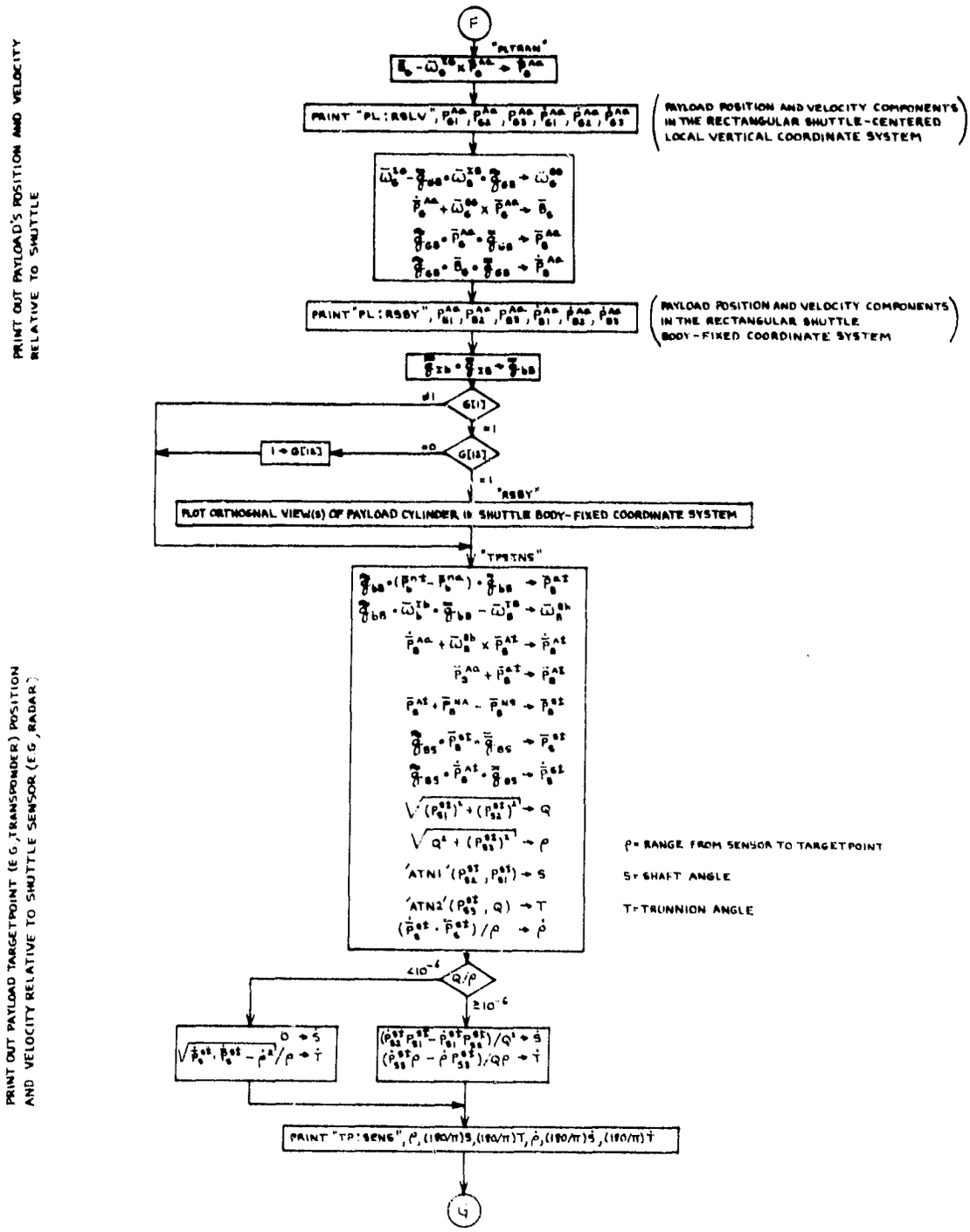


Figure 38f. %PROP Logic Flow (cont.)

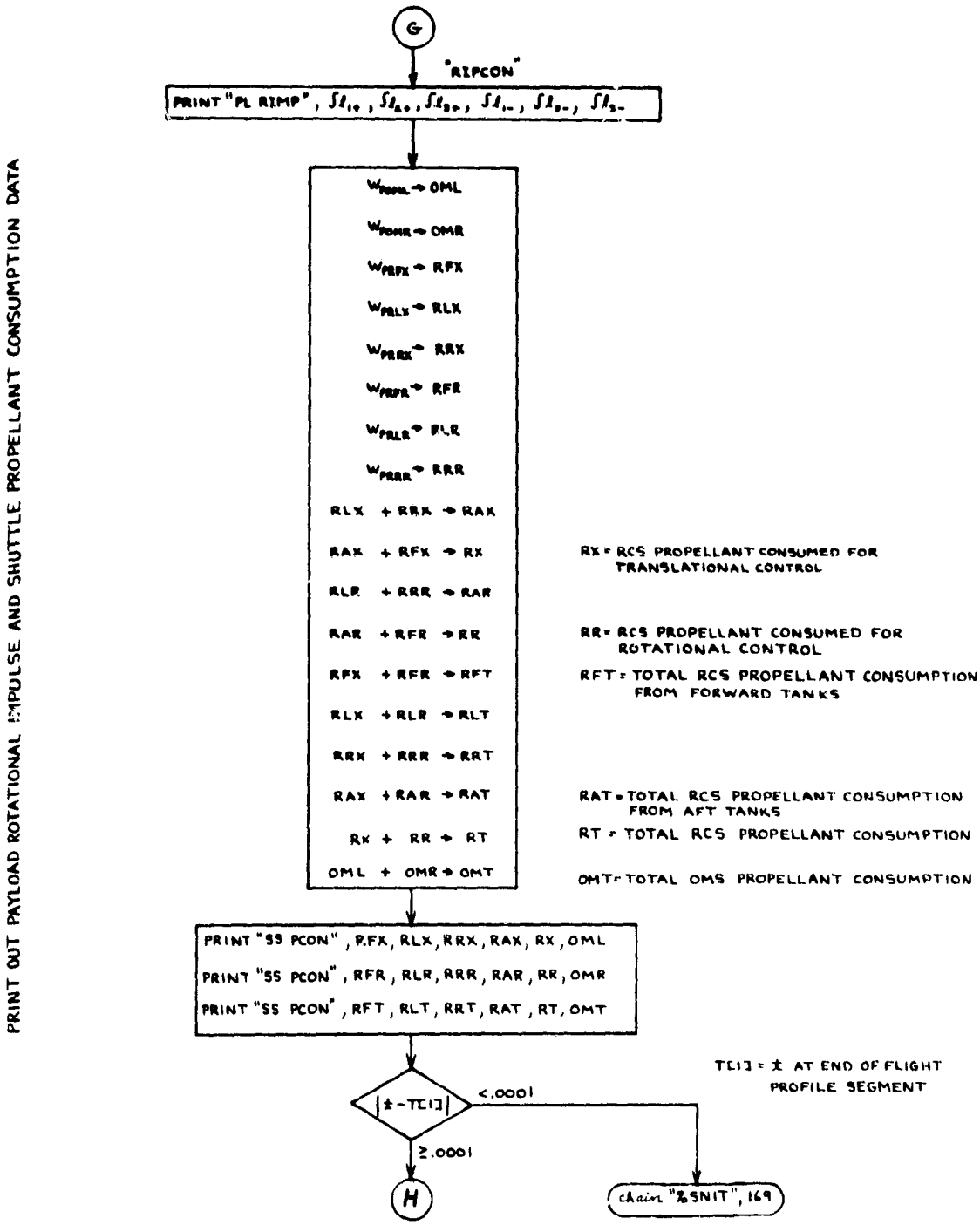


Figure 38g. %PROP Logic Flow (cont.)

ADVANCE STATE OF SHUTTLE/PAYLOAD SYSTEM TO NEXT PRINT/PLOT POINT

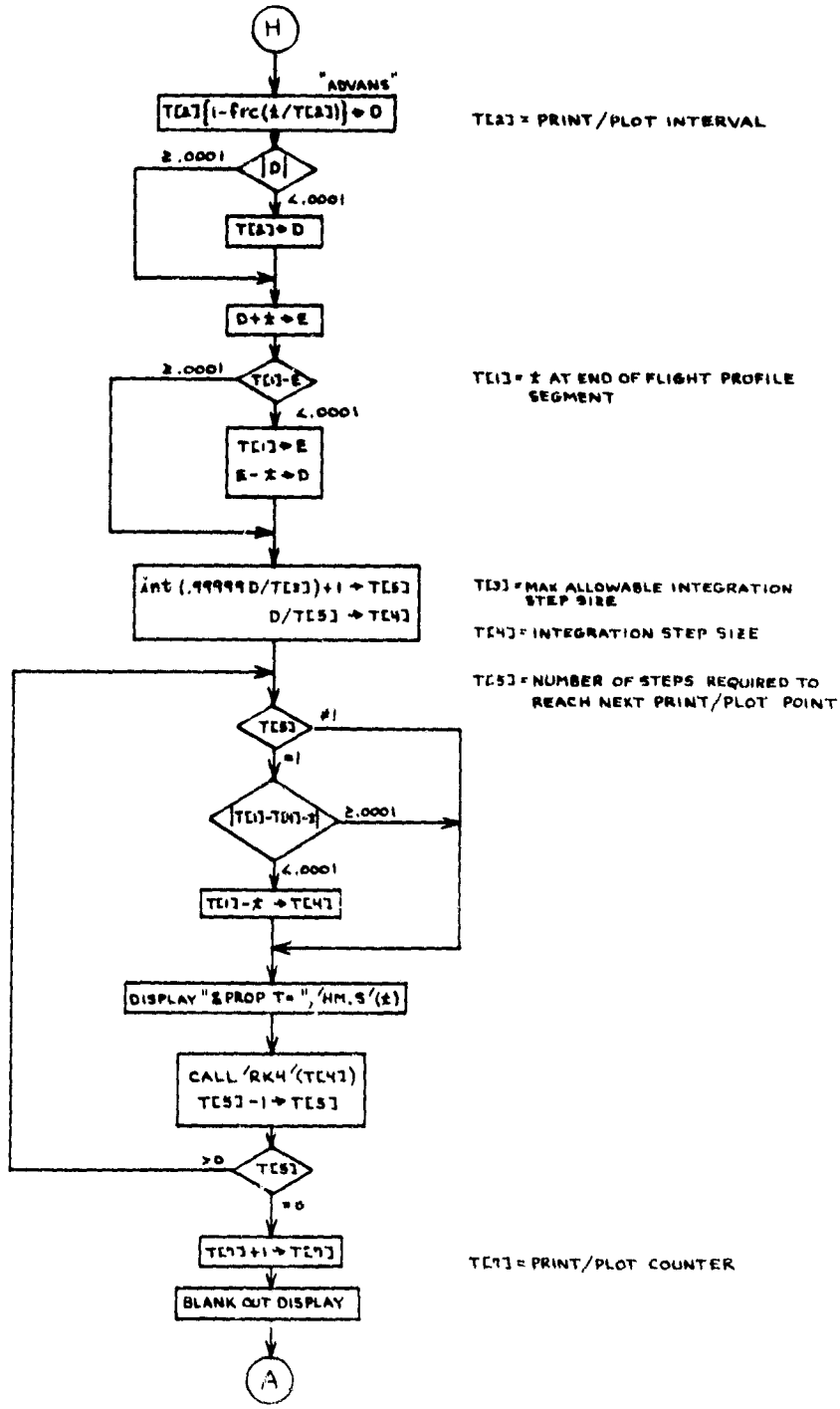


Figure 38h. %PROP Logic Flow (cont.)

REFERENCES

1. S. M. Kindall, "User's Guide for the HP-9825A High Fidelity Relative Motion Program (HFRMP)," TRW IOC No. 78:2511.4-33, 16 October 1978.

2. S. W. Wilson, "Equations for Calculating Orbiter Surface Erosion and Breakage Rates in IUS and SSUS SRM Exhaust Plumes," TRW Report No. 28415-H007-R0-00, 6 June 1978.

3. S. W. Wilson, "A Quaternion Method for the Numerical Integration of Satellite Orbits," TRW Report No. 99815-H001-R0-00, 30 December 1977.

4. United States Committee on Extension to the Standard Atmosphere (COESA), U.S. Standard Atmosphere, 1962, U. S. Government Printing Office, Washington, D. C., December 1962.

5. M. D. Zuteck, "Improved Shuttle On-Orbit Aerodynamic Moment Coefficient Models," TRW IOC No. 77:2511.4-37, 1 August 1977.

6. S. W. Wilson, "Shuttle On-Orbit Aerodynamic Moment Equations," TRW IOC No. 77:2511.1-30, 28 April 1977.

7. M. D. Zuteck, "Shuttle On-Orbit Aerodynamic Drag Model Improvements," TRW IOC No. 76:2511.4-60, 3 September 1976.

8. E. C. Lineberry, "Ivariant Orbital Elements for Use in the Description of Motion About an Oblate Earth," NASA/JSC IN 74-FM-84, 4 December 1974.

9. S. W. Wilson, "Engineering Description of the OMS/RCS/DAP Models Used in the High Fidelity Relative Motion Program (HFRMP)," TRW Report No. 28415-H009-R0-00, 6 October 1978.

10. D. M. Henderson, "Euler Angles, Quaternions, and Transformation Matrices - Working Relationships," NASA/JSC Memorandum 77-FM-37, July 1977.

11. Lyndon B. Johnson Space Center, "Shuttle Operational Data Book, Volume I, Shuttle Systems Performance and Constraints Data," NASA/JSC Document No. JSC-08934 (Vol. I), Revision A, updated 13 June 1978.

12. E. W. Purcell and W. B. Cowan, "Relating Geodetic Latitude and Altitude to Geocentric Latitude and Radius Vector," ARS Journal, July 1961.

13. B. G. Jackson, "Update of On-Orbit Plume Impingement Values," JSC Memorandum No. EX32/7901-4, 17 January 1979.

14. H. F. Fliegel and T. C. Van Flandern, "A Machine Algorithm for Processing Calendar Dates," Communications of the ACM, Vol. 11, No. 10, p. 657, October 1968.

15. A Ralston and H. S. Wilf (editors), Mathematical Models for Digital Computers, Vol. 1, Chapter 7, John Wiley and Sons, New York, 1962.

16. C. A. Denham, "Trajectory Prediction Parameters for AAP, Space Station/ Space Shuttle, and Interplanetary Mission," MSC Internal Note No. 70-FM-33, 5 March 1970.

17. Nautical Almanac Offices of the United Kingdom and the United States of America, Explanatory Supplement to the Ephemeris, p. 30, Her Majesty's Stationery Office, London, 1961.

APPENDIX A:

BASIC QUATERNION OPERATIONS

A.1 QUATERNION ALGEBRA

Reference A-1 defines a quaternion to be a mathematical quantity of the form

$$\overline{Q} = Q_0 + \hat{i}Q_1 + \hat{j}Q_2 + \hat{k}Q_3 \tag{A-1}$$

where Q_0, Q_1, Q_2, and Q_3 are real numbers and where the products of $\hat{i}$, $\hat{j}$, and $\hat{k}$ are governed by the following conventions:

$$\hat{i} \circ \hat{i} = \hat{j} \circ \hat{j} = \hat{k} \circ \hat{k} = -1, \tag{A-2}$$

$$\hat{j} \circ \hat{k} = -\hat{k} \circ \hat{j} = \hat{i}, \tag{A-3}$$

$$\hat{k} \circ \hat{i} = -\hat{i} \circ \hat{k} = \hat{j}, \tag{A-4}$$

and

$$\hat{i} \circ \hat{j} = -\hat{j} \circ \hat{i} = \hat{k}. \tag{A-5}$$

In addition to having the properties of imaginary numbers, the quantities $\hat{i}$, $\hat{j}$, and $\hat{k}$ also have the properties of unit vectors that are aligned with orthongonal coordinate axes in a three dimensional space.

The circular symbol ($\circ$) which is used to denote the quaternion product is adopted from the notation of Reference A-2. This symbol should not be confused with the dot ($\cdot$) that is used to denote the scalar product of vector algebra, which is governed by the following conventions:

$$\hat{i} \cdot \hat{i} = \hat{j} \cdot \hat{j} = \hat{k} \cdot \hat{k} = 1, \tag{A-6}$$

$$\hat{j} \cdot \hat{k} = -\hat{k} \cdot \hat{j} = 0, \tag{A-7}$$

$$\hat{k} \cdot \hat{i} = -\hat{i} \cdot \hat{k} = 0, \tag{A-8}$$

and

$$\hat{i}\cdot\hat{j} = -\hat{j}\cdot\hat{i} = 0. \tag{A-9}$$

It is worthwhile to observe also the similarities and differences between the quaternion product and the vector cross product, which is governed by the conventions

$$\hat{i} \times \hat{i} = \hat{j} \times \hat{j} = \hat{k} \times \hat{k} = 0, \tag{A-10}$$

$$\hat{j} \times \hat{k} = -\hat{k} \times \hat{j} = \hat{i}, \tag{A-11}$$

$$\hat{k} \times \hat{i} = -\hat{i} \times \hat{k} = \hat{j}, \tag{A-12}$$

and

$$\hat{i} \times \hat{j} = -\hat{j} \times \hat{i} = \hat{k}. \tag{A-13}$$

A quaternion can be thought of as having a scalar part Q_0, and a vector part

$$\overline{Q} = \hat{i}Q_1 + \hat{j}Q_2 + \hat{k}Q_3. \tag{A-14}$$

Sometimes, then, it is convenient to express Equation (1) in its equivalent form

$$\overline{\overline{Q}} = Q_0 + \overline{Q}. \tag{A-15}$$

The sum of two quaternions is defined by

$$\overline{\overline{P}}+\overline{\overline{Q}} = (P_0+Q_0) + \hat{i}(P_1+Q_1) + \hat{j}(P_2+Q_2) + \hat{k}(P_3+Q_3), \tag{A-16}$$

and their product by

$$\bar{\bar{P}} \circ \bar{\bar{Q}} = (P_0Q_0 - P_1Q_1 - P_2Q_2 - P_3Q_3)$$

$$+ \hat{i}(P_0Q_1 + P_1Q_0 + P_2Q_3 - P_3Q_2)$$

$$+ \hat{j}(P_0Q_2 + P_2Q_0 + P_3Q_1 - P_1Q_3)$$

$$+ \hat{k}(P_0Q_3 + P_3Q_0 + P_1Q_2 - P_2Q_1). \qquad \text{(A-17)}$$

Equation (A-17) results from applying the distributive law of algebra along with the conventions defined by Equations (A-2) through (A-5). It should be noted that, in general, quaternion multiplication is not commutative (i.e., $\bar{\bar{Q}} \circ \bar{\bar{P}} \neq \bar{\bar{P}} \circ \bar{\bar{Q}}$). Except for the commutative property of multiplication, quaternions satisfy all the requirements for the definition of a field.

The quaternion product of a scalar and a quaternion <u>is</u> commutative, and is given by

$$S \circ \bar{\bar{Q}} = \bar{\bar{Q}} \circ S = SQ_0 + \hat{i}SQ_1 + \hat{j}SQ_2 + \hat{k}SQ_3. \qquad \text{(A-18)}$$

which follows from (A-17) when the scalar is treated as a quaternion whose vector part is zero. In a similar vein, the product of a quaternion and a vector if formed by treating the vector as a quaternion whose scalar part is zero. This results in

$$\bar{\bar{Q}} \circ \bar{V} = (-Q_1V_1 - Q_2V_2 - Q_3V_3)$$

$$+ \hat{i}(Q_0V_1 + Q_2V_3 - Q_3V_2)$$

$$+ \hat{j}(Q_0V_2 + Q_3V_1 - Q_1V_3)$$

$$+ \hat{k}(Q_0V_3 + Q_1V_2 - Q_2V_1) \qquad \text{(A-19)}$$

and

$$\bar{V}\circ\bar{\bar{Q}} = (-V_1Q_1 - V_2Q_2 - V_3Q_3)$$

$$+ \hat{i}(V_1Q_0 + V_2Q_3 - V_3Q_2)$$

$$+ \hat{j}(V_2Q_0 + V_3Q_1 - V_1Q_3)$$

$$+ \hat{k}(V_3Q_0 + V_1Q_2 - V_2Q_1). \qquad \text{(A-20)}$$

By examining Equation (A-17), it is seen that the quaternion product can be expressed in the form

$$\bar{\bar{P}}\circ\bar{\bar{Q}} = P_0Q_0 - \bar{P}\cdot\bar{Q} + P_0\bar{Q} + Q_0\bar{P} + \bar{P} \times \bar{Q}, \qquad \text{(A-21)}$$

which leads to a relationship,

$$\bar{\bar{Q}}\circ\bar{\bar{P}} = \bar{\bar{P}}\circ\bar{\bar{Q}} - 2\,\bar{P} \times \bar{Q}, \qquad \text{(A-22)}$$

that is sometimes useful. Equation (A-22) shows that quaternion multiplication is commutative whenever the vector parts of the two quaternions are parallel to each other.

The conjugate of the quaternion $\bar{\bar{Q}} = Q_0 + \hat{i}Q_1 + \hat{j}Q_2 + \hat{k}Q_3$ is defined by

$$\tilde{\tilde{Q}} = Q_0 - \hat{i}Q_1 - \hat{j}Q_2 - \hat{k}Q_3. \qquad \text{(A-23)}$$

The norm of a quaternion is defined by the product

$$\bar{\bar{Q}}\circ\tilde{\tilde{Q}} = \tilde{\tilde{Q}}\circ\bar{\bar{Q}} = Q_0^2 + Q_1^2 + Q_2^2 + Q_3^2 . \qquad \text{(A24)}$$

A.2 COORDINATE TRANSFORMATION

Quaternions that have a norm of unity exhibit properties that make them very useful for transforming coordinates from one Cartesian system to another having a different orientation. Although the distinction is not made in some of the relevant literature, a quaternion of this special class is more precisely known as a versor. In any event, it is well to remember that when the word "quaternion" is used in relation to coordinate transformation or the orientation of rigid bodies, almost always it refers to a quaternion whose norm is equal to one (1.0). A convention of using lower-case alphabetic symbols to designate versors (unit quaternions) has been adopted in this report. In other words, use of the symbology

$$q = q_0 + \grave{i}q_1 + \grave{j}q_2 + \grave{k}q_3 \tag{A-25}$$

implies that

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{A-26}$$

According to one of Euler's theorems, any two Cartesian coordinate systems F and G that have a common origin can be brought into coincidence by rotating F through some angle α about a single fixed axis. That is to say, the angular displacement of any Cartesian system G with respect to another Cartesian system F can be described in terms of a rotation about a single fixed axis which is usually referred to as the Euler axis. The orientation of the Euler axis can be defined by a unit vector

$$\grave{d} = \grave{d}_F = \grave{d}_G = \grave{i}d_1 + \grave{j}d_2 + \grave{k}d_3 \tag{A-27}$$

with components d_1, d_2, d_3 which not only satisfy the equation

$$d_1^2 + d_2^2 + d_3^2 = 1. \tag{A-28}$$

but which also have identical values in F and G. Assuming that the angle of rotation is restricted to lie in the range $0 \leq \alpha < 2\pi$, the unit vector $\hat{d}$ can be taken to define both the orientation of the Euler axis and the sense (direction) of rotation according to the customary right-hand screw convention.

It can be shown (e.g., see Reference A-2 and A-3) that any vector $\overline{V}$ having the form

$$\overline{V}_F = \hat{i}V_{F1} + \hat{j}V_{F2} + \hat{k}V_{F3} \tag{A-29}$$

in coordinate system F can be transformed to its equivalent form*

$$\overline{V}_G = \hat{i}V_{G1} + \hat{j}V_{G2} + \hat{k}V_{G3} \tag{A-30}$$

in system G by use of the equation

$$\overline{V}_G = \overline{\overline{q}}_{FG} \circ \overline{V}_F \circ \overline{\overline{q}}_{FG}, \tag{A-31}$$

where

$$\overline{\overline{q}}_{FG} = \cos(\tfrac{1}{2}\alpha) + \hat{d}\sin(\tfrac{1}{2}\alpha). \tag{A-32}$$

*It should be noted that in this system of notation the unit vectors $\hat{i}$, $\hat{j}$, and $\hat{k}$ are not associated with a particular set of reference axes. Instead, they are understood to be aligned with the axes of whatever coordinate system is designated by the alphabetic subscript attached to the vector symbol. In this connection, a vector symbol having no subscript represents an intrinsic physical value that exists independently of the system of reference. For instance, the symbol $\overline{V}$ might represent the inertial velocity of one body with respect to another, while $\overline{V}_F$ represents just one of many possible quantifications of $\overline{V}$: namely, that resulting from the projection of $\overline{V}$ onto the axes of coordinate system F.

The double subscript FG in Equation (A-32) identifies the versor as being the orientation versor which defines the angular displacement of coordinate system G with respect to system F[†].

The inverse displacement (of F with respect to G) is defined by

$$\bar{\bar{q}}_{GF} = \tilde{\tilde{q}}_{FG} = \cos(\tfrac{1}{2}\alpha) - \hat{d}\sin(\tfrac{1}{2}\alpha), \tag{A-33}$$

and the inverse transformation of coordinates by

$$\bar{V}_F = \tilde{\tilde{q}}_{GF} \circ \bar{V}_G \circ \bar{\bar{q}}_{GF}. \tag{A-34}$$

It is easily seen that the equations

$$\bar{V}_G = \bar{\bar{q}}_{GF} \circ \bar{V}_F \circ \tilde{\tilde{q}}_{GF} \tag{A-35}$$

and

$$\bar{V}_F = \bar{\bar{q}}_{FG} \circ \bar{V}_G \circ \tilde{\tilde{q}}_{FG} \tag{A-36}$$

are equivalent to (A-31) and (A-34).

We now consider a third coordinate system H whose angular displacement relative to G is defined by the orientation versor

$$\bar{\bar{q}}_{GH} = \cos(\tfrac{1}{2}\beta) + \hat{e}\sin(\tfrac{1}{2}\beta). \tag{A-37}$$

Application of the coordinate transformation law yields

[†]An alternate rotation that produces the same result is defined by $\bar{\bar{q}}'_{FG} = -\cos(\tfrac{1}{2}\alpha) - \hat{d}\sin(\tfrac{1}{2}\alpha)$, which represents a rotation through an angle of $2\pi - \alpha$ in the opposite $(-\hat{d})$ direction. However, there is no need to consider the alternate rotation in the present discussion.

$$\bar{V}_H = \tilde{\bar{q}}_{GH} \circ \bar{V}_G \circ \bar{\bar{q}}_{GH}$$

$$= \tilde{\bar{q}}_{GH} \circ (\tilde{\bar{q}}_{FG} \circ \bar{V}_F \circ \bar{\bar{q}}_{FG}) \circ \bar{\bar{q}}_{GH}$$

$$= (\tilde{\bar{q}}_{GH} \circ \tilde{\bar{q}}_{FG}) \circ \bar{V}_F \circ (\bar{\bar{q}}_{FG} \circ \bar{\bar{q}}_{GH}), \qquad \text{(A-38)}$$

whence it follows that

$$\bar{\bar{q}}_{FH} = \bar{\bar{q}}_{FG} \circ \bar{\bar{q}}_{GH}. \qquad \text{(A-39)}$$

The above result can be extended to any number of successive rotations. For instance, suppose that the body-fixed frame B of a spacecraft is displaced from an inertial reference frame I by rotating the spacecraft first through a pitch angle θ about Y-axis, then through a yaw angle ψ about its body-fixed Z-axis, and finally through a roll angle ϕ about its X-axis. The total displacement is defined by the quaternion product

$$\bar{\bar{q}}_{IB} = [\cos(\tfrac{1}{2}\theta) + \hat{j}\sin(\tfrac{1}{2}\theta)] \circ [\cos(\tfrac{1}{2}\psi) + \hat{k}\sin(\tfrac{1}{2}\psi)] \circ$$

$$[\cos(\tfrac{1}{2}\phi) + \hat{i}\sin(\tfrac{1}{2}\phi)], \qquad \text{(A-40)}$$

which, after carrying out the indicated multiplications, reduces to

$$\bar{\bar{q}}_{IB} = (C_\theta C_\psi C_\phi - S_\theta S_\psi S_\phi)$$

$$+ \hat{i}(C_\theta C_\psi S_\phi + S_\theta S_\psi C_\phi)$$

$$+ \hat{j}(S_\theta C_\psi C_\phi + C_\theta S_\psi S_\phi)$$

$$+ \hat{k}(C_\theta S_\psi C_\phi - S_\theta C_\psi S_\phi) \qquad \text{(A-41)}$$

where $C_\theta = \cos(\frac{1}{2}\theta)$, $S_\theta = \sin(\frac{1}{2}\theta)$, $C_\psi = \cos(\frac{1}{2}\psi)$, etc.

The angles θ, ψ, ϕ represent one of many Euler angle sets that can be used to define the orientation of one Cartesian system relative to another. Any relative orientation can be described by three Euler angles* representing successive rotations about particular coordinates axes, taken in a specified sequence; the only restriction being that the second axis of rotation must not coincide with the first or the third. Reference A-4 contains a useful compendium of the relationships between versors, transformation matrices, and all of the possible Euler angle sets.

*Not to be confused with the single angle of rotation about the Euler axis which produces the same result.

A.3 TIME DERIVATIVE OF AN ORIENTATION VERSOR

If the coordinate system B is rotating with angular velocity $\overline{\Omega}$ relative to system I, the time derivative of $\overline{\overline{q}}_{IB}$ is given by

$$\dot{\overline{\overline{q}}}_{IB} = \tfrac{1}{2}\, \overline{\overline{q}}_{IB} \circ \overline{\Omega}_B \tag{A-42}$$

or, alternatively, by

$$\dot{\overline{\overline{q}}}_{IB} = \tfrac{1}{2}\, \overline{\Omega}_I \circ \overline{\overline{q}}_{IB}. \tag{A-43}$$

The consistency of Equations (A-42) and (A-43) with each other can be readily verified by substituting the relationship

$$\overline{\Omega}_B = \tilde{\overline{\overline{q}}}_{IB} \circ \overline{\Omega}_I \circ \overline{\overline{q}}_{IB} \tag{A-44}$$

into (A-42), which yields

$$\dot{\overline{\overline{q}}}_{IB} = \tfrac{1}{2}\, (\overline{\overline{q}}_{IB} \circ \tilde{\overline{\overline{q}}}_{IB}) \circ \overline{\Omega}_I \circ \overline{\overline{q}}_{IB} \tag{A-45}$$

One of the major advantages of using a versor to define the orientation of a rotating coordinate system lies in the fact that it has a finite derivative at every possible orientation (assuming of course that the angular velocity $\overline{\Omega}$ is finite), thereby facilitating numerical integration of the differential equations that govern the rotational motion of the system. Such is not the case when Euler angles are used. No matter what rotation sequence is chosen, it is possible for the derivatives of two of the angles to approach infinity in the vicinty of certain critical orientations.

The problem of infinite derivatives can be avoided by defining the orientation with direction cosines (i.e., the elements of a coordinate transformation matrix); however, this requires the integration of nine real

variables as compared to four in the case of a versor. The integration of direction cosines is further complicated by the necessity of maintaining the normality and orthogonality of the transformation matrix, as defined by six different equations among the direction cosines. In the case of versor integration, only a single ancillary condition of this nature is of concern: the maintenance or normality as defined by Equation (A-26).

A.4 REFERENCES

A-1 Glenn James and Robert C. James (editors), Mathematics Dictionary, D. Van Nostrand Co., Inc., Princeton, N. J., 1959.

A-2 V. N. Branyetz and I. P. Shmiglevski, Application of Quaternions in the Orientation of Rigid Bodies (in Russian), Science Publishing House, Moscow, 1973.

A-3 Alfred C. Robinson, "On the Use of Quaternions in Simulation of Rigid Body Motion," Wright Air Development Center Technical Report 58-17, December 1958.

A-4 D. M. Henderson, "Euler Angles, Quaternions, and Transformation Matrices-Working Relationships," NASA/JSC Report No. 77-FM-37, July 1977.

APPENDIX B:

COORDINATE SYSTEMS

B.1 SHUTTLE STATION (STRUCTURAL) COORDINATES

Shuttle Station coordinates are referenced to a Cartesian system that is fixed to the Orbiter structure. The origin of the system lies in the plane of symmetry at a point 400 inches below the cargo bay centerline and 576 inches forward of the aft face of the forward bulkhead of the cargo bay. The X-axis is parallel to the cargo bay centerline and positive in the aft direction, the Y-axis is normal to the Orbiter's plane of symmetry and positive to starboard, and the Z-axis completes a right handed orthogonal system. The X coordinate is referred to as the station (STA), the Y coordinate is referred to as the buttock line (BL), and the Z coordinate is referred to as the water line (WL). This system is shown in Figure B1.

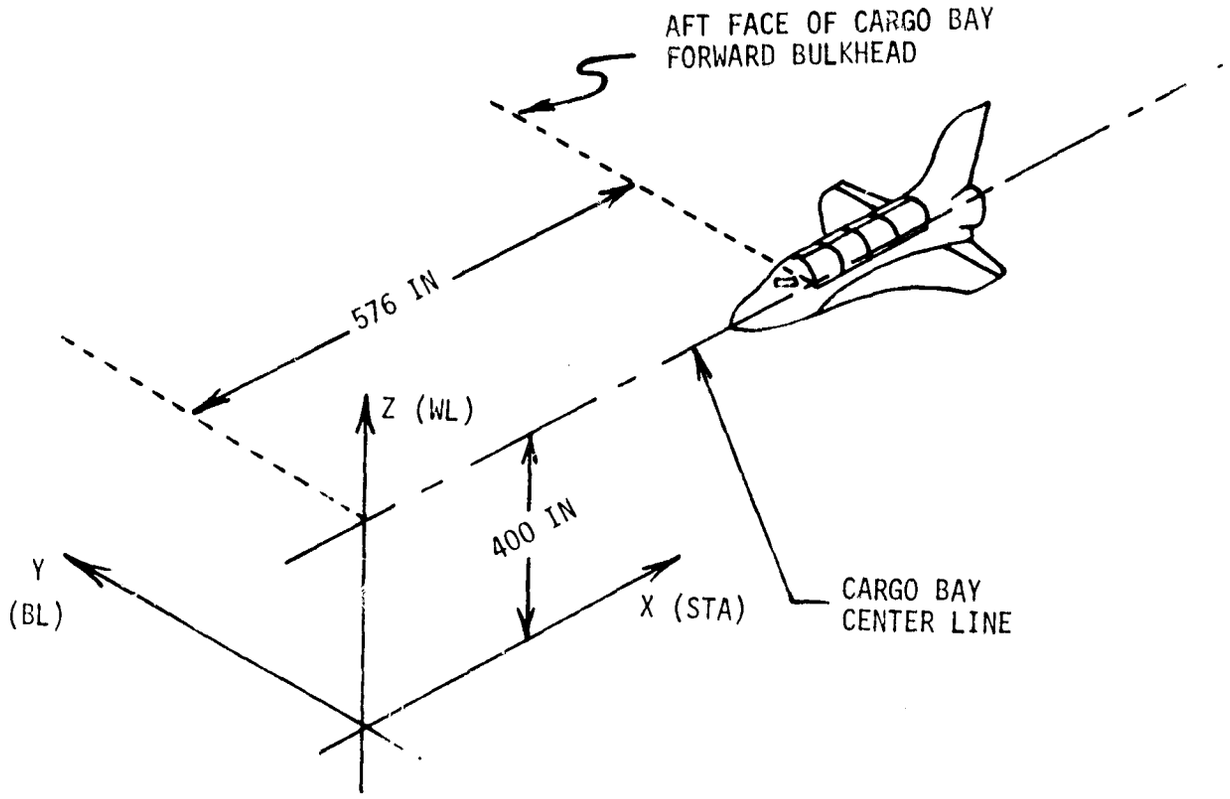


Figure B1. Shuttle Station Coordinates

B.2 PAYLOAD STATION (STRUCTURAL) COORDINATES

The Payload Station reference system is defined such that its coordinate axes are parallel to the Shuttle Station coordinate axes when the payload is stowed in the Orbiter payload bay. The origin of the Payload Station coordinate system is located in the center of the front face of the payload cylinder. This system is shown in Figure B2. Coordinates in this system are always measured in inches.

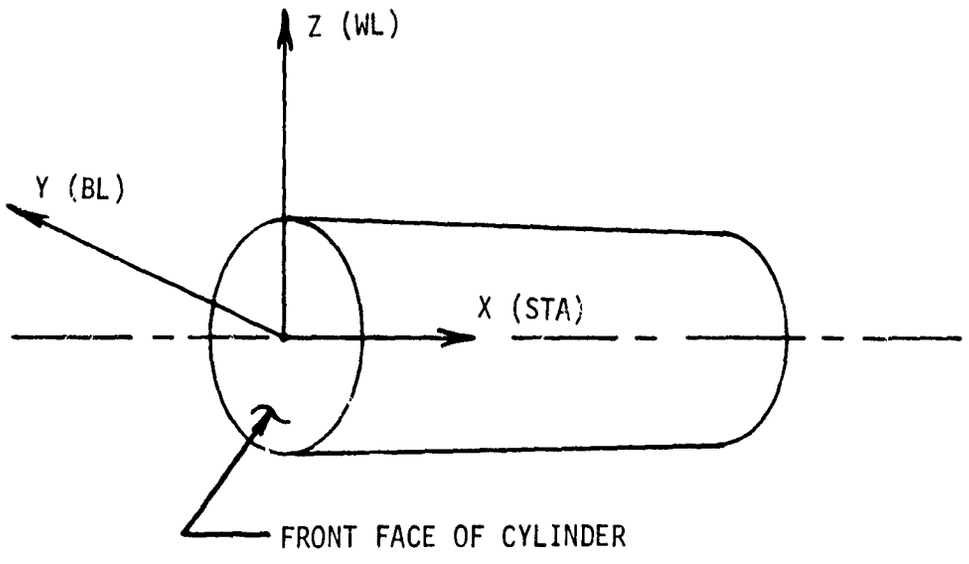


Figure B2. Payload Station Coordinates

B.3 SHUTTLE BODY COORDINATES

Shuttle Body coordinates are referenced to a Cartesian system that is fixed relative to the Orbiter structure, with its origin at the Orbiter CG. The X-axis is parallel to the cargo bay centerline and positive in the forward direction, the Y-axis is normal to the Orbiter's plane of symmetry and positive to starboard, and the Z-axis completes a right handed orthogonal system. These axes are parallel to the Shuttle Station axes; however, the positive directions of the X and Z axes are reversed. This system is shown in Figure B3.

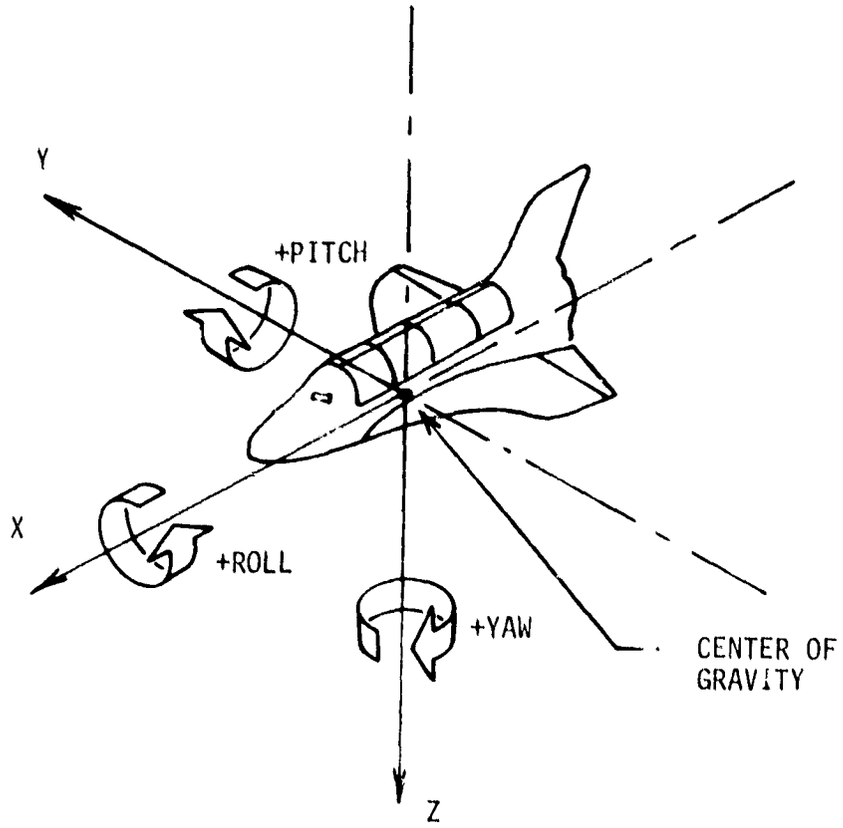


Figure B3. Shuttle Body Coordinates

B.4 PAYLOAD BODY COORDINATES

Payload Body coordinates are referenced to a Cartesian system that is fixed relative to the payload structure, with its origin at the payload CG. The X-axis is parallel to the payload longitudinal axis and positive in the Orbiter's forward dirction when the payload is stowed in the Orbiter's cargo bay. The Y-axis is normal to the Orbiter's plane of symmetry when the Payload is stowed in the cargo bay, and the Z-axis completes a right-handed orthogonal system. This system is shown in Figure B4.

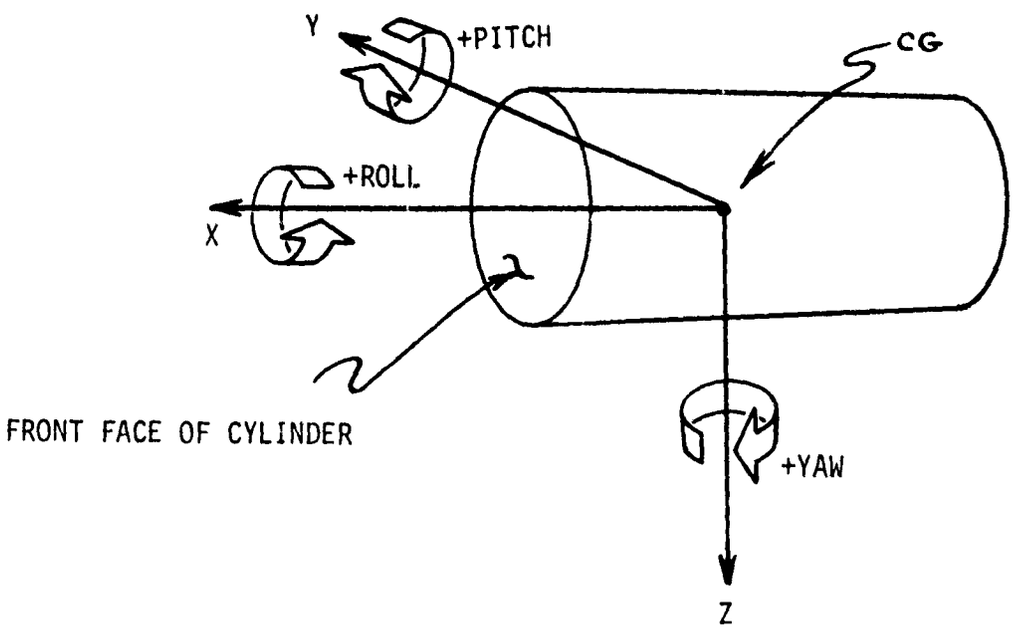


Figure B4. Payload Body Coordinates

B.5 RECTANGULAR LOCAL VERTICAL COORDINATES

Rectangular Local Vertical coordinates are referenced to a rotating Cartesian system centered at the CG of an orbiting vehicle. The direction of the Y-axis is opposite to the orbital angular momentum vector of the vehicle's CG with respect to the center of the earth, the Z-axis points toward the center of the earth, and the X-axis completes a right hand orthogonal system. This system is shown in Figure B5.

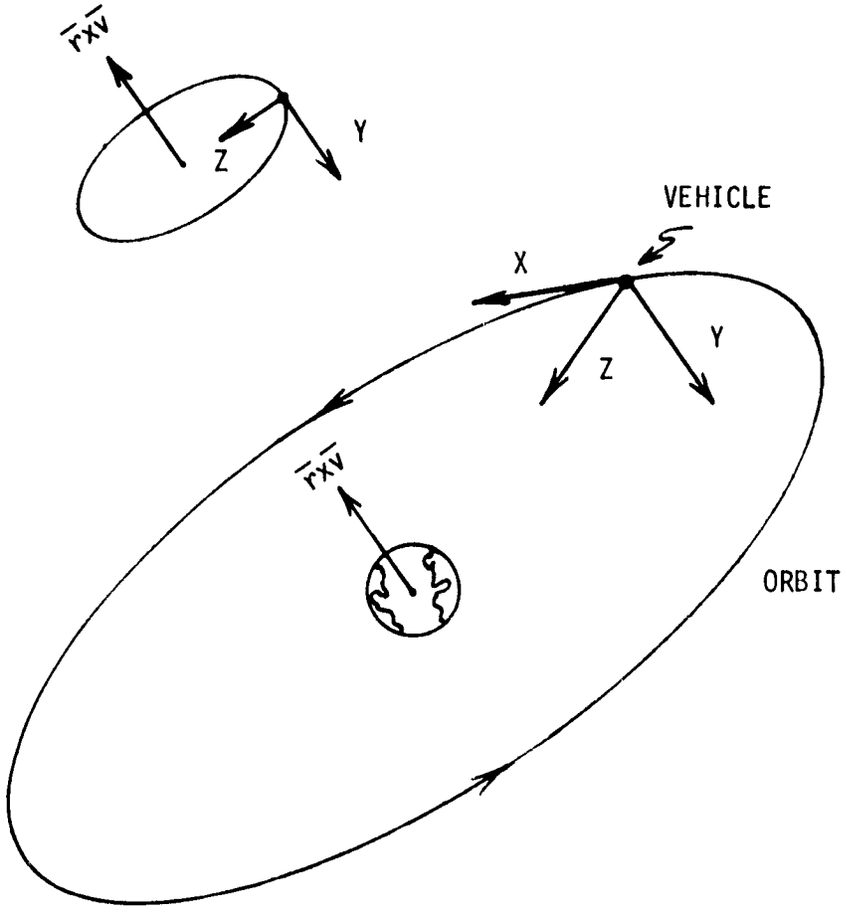
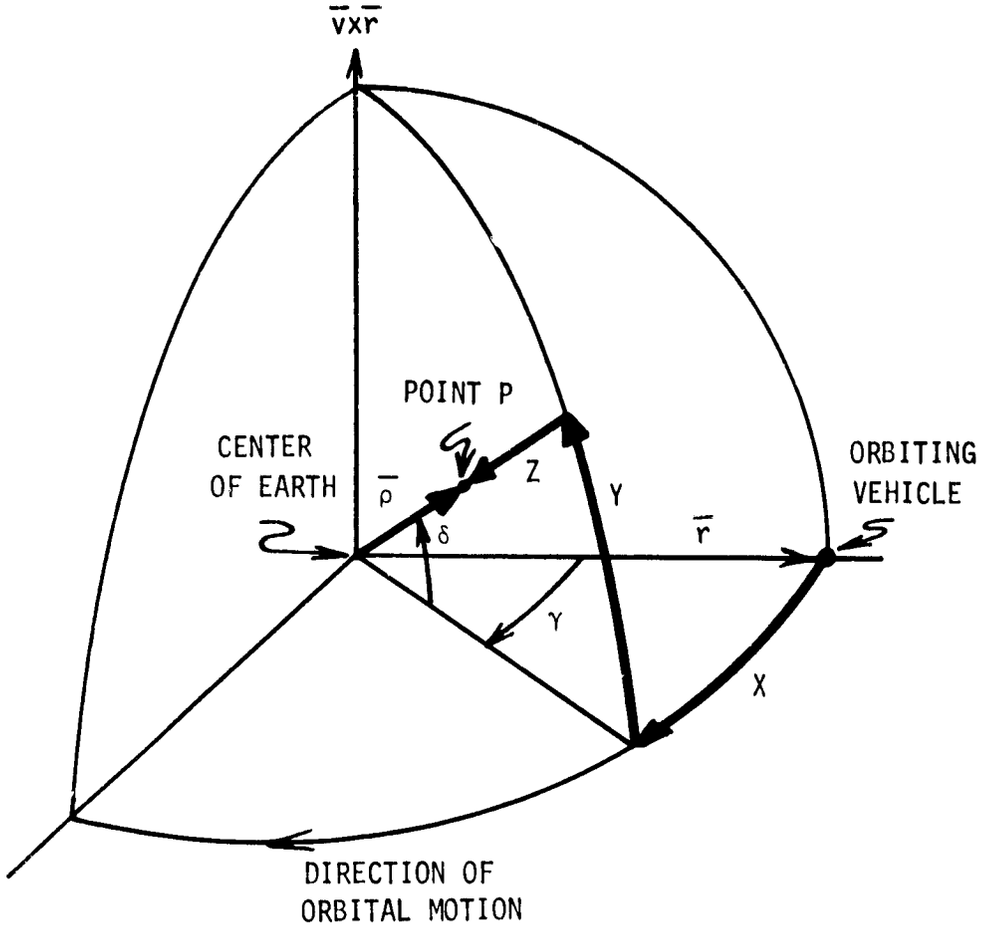


Figure B5. Rectangular Local Vertical Coordinates

B.6 CURVILINEAR LOCAL VERTICAL COORDINATES

Curvilinear Local Vertical coordinates are referenced to the CG of an orbiting vehicle. As shown in Figure B6, the X and Y components of position are measured along the surface of an imaginary earth-centered sphere that passes through the CG of the vehicle. The Z component is measured normal to the sphere, positive in the direction of the earth's center. These coordinates are essentially identical with those defined in Section B.5 when the distance from the vehicle CG is small.



$X = r\gamma$

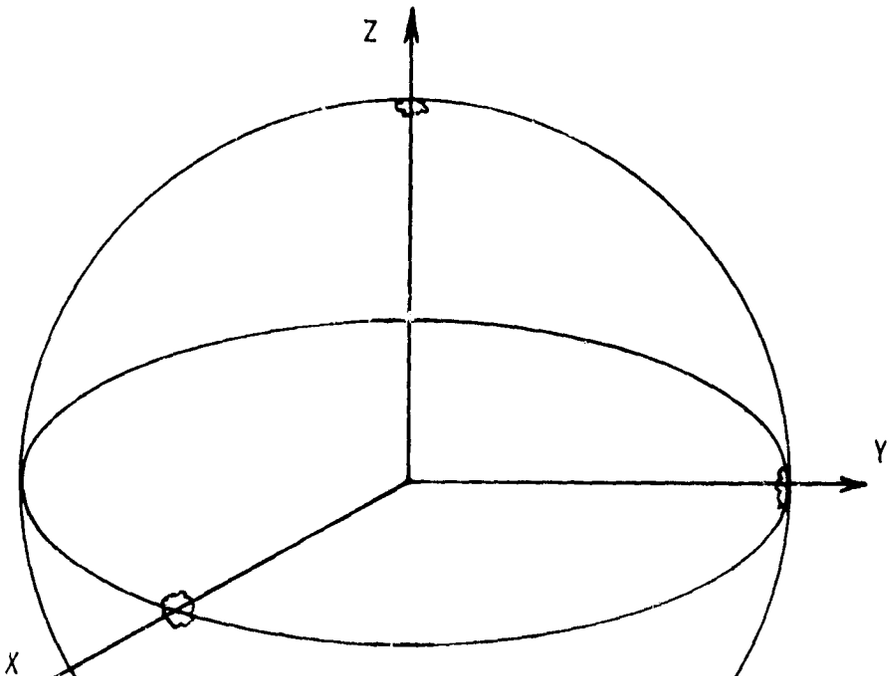
$Y = r\delta$

$Z = r - \rho$

Figure B6. Curvilinear Local Vertical Coordinates

B.7 MEAN OF 1950.0 COORDINATES

Mean of 1950.0 coordinates are referenced to a non-rotating earth-centered Cartesian system that is defined by the orientation of the earth's mean equator at the beginning of the Besselian year 1950.0 (i.e., the Julian Date 2433282.423357). The Z-axis points in the direction of the earth's angular momentum vector, the X-axis is aligned with the intersection of the equatorial and ecliptic planes (positive in the direction of the sun as seen from the earth at the time of the vernal equinox), and the Y-axis completes a right hand orothogonal system. This system is shown in Figure B7.



X-Y plane is Earth's equator of epoch

X is directed toward the mean vernal equinox of epoch

Z is directed along Earth's mean rotational axis of epoch and is positive north

Y completes a right handed system

Epoch is the beginning of Besselian year 1950 for the Mean of 1950.0 system or the date of launch for the Mean of Launch system

Figure B7. Mean of 1950.0 and Mean of Launch Date Coordinates

B.8 MEAN OF LAUNCH DATE COORDINATES

The Mean of Launch Date system of reference is also a non-rotating earth-centered Cartesian system defined in exactly the same manner as the Mean of 1950.0 system, except that the orientation of the equatorial plane is fixed at the date of launch. This system is shown in Figure B7.

APPENDIX C

PROCESSOR CODE

C.1 Base Link (#TRAJ)

```
0: "#TRAJ(1223/10NOV79)":sto "RUN"
1: wtb 6,11;list #6
2: get "%RMAT",0,1
3: "ANG1":2πfrc(p1/2π)→p2
4: if p2<0;2π+p2→p2
5: ret p2
6: "ANG2":2πfrc(p1/2π)→p2
7: if p2>π;p2-2π→p2
8: ret p2
9: "ATN1":2π→p5;sto +2
10: "ATN2":0→p5
11: p1↑2→p3;p2↑2→p4
12: if p3+p4=0;sto +6
13: if p3>p4;sto +3
14: atn(abs(p1/p2))→p3;if p2<0;π-p3→p3
15: sto +2
16: π/2-atn(p2/abs(p1))→p3
17: if p1<0;p5-p3→p3
18: ret p3
19: "HM.S":abs(p1)→p2
20: int(p2/60)→p3;int(p3/60)→p4
21: 100p4+p3-60p4+(p2-60p3)/100→p3
22: ret p3sgn(p1)
23: "SECS":abs(p1)/100→p2
24: int(p2)→p3;100frc(p2)→p4
25: 60(60p3+int(p4))+100frc(p4)→p3
26: ret p3sgn(p1)
27: "DOTP":ret rp1rp2+r(p1+1)r(p2+1)+r(p1+2)r(p2+2)
28: "SXV":p1rp2→rp3;p1r(p2+1)→r(p3+1);p1r(p2+2)→r(p3+2);ret
29: "VADD":1→p4;sto +2
```

#TRAJ Cont.

```
30: "VSUB":-1→p4
31: rp1+p4rp2→rp3
32: r(p1+1)+p4r(p2+1)→r(p3+1)
33: r(p1+2)+p4r(p2+2)→r(p3+2);ret
34: "CRSP":r(p1+1)r(p2+2)-r(p1+2)r(p2+1)→p4
35: r(p1+2)rp2-rp1r(p2+2)→p5
36: rp1r(p2+1)-r(p1+1)rp2→r(p3+2)
37: p5→r(p3+1);p4→rp3;ret
38: "ROT":rp2→H;sto +2
39: "IROT":-rp2→H
40: rp1→X;r(p1+1)→Y;r(p1+2)→Z
41: r(p2+1)→I;r(p2+2)→J;r(p2+3)→K
42: (HH+II-JJ-KK)X+2((IJ+HK)Y+(IK-HJ)Z)→rp3
43: (HH-II+JJ-KK)Y+2((JK+HI)Z+(IJ-HK)X)→r(p3+1)
44: (HH-II-JJ+KK)Z+2((IK+HJ)X+(JK-HI)Y)→r(p3+2);ret
45: "QDOT":1→H;0→W;rp2/2→X;r(p2+1)/2→Y;r(p2+2)/2→Z;sto +5
46: "QXQ":1→H→W;sto +3
47: "QXQC":1→H;-1→W;sto +2
48: "QCXQ":-1→H;1→W
49: Wr(p2+1)→X;Wr(p2+2)→Y;Wr(p2+3)→Z;rp2→W
50: Hr(p1+1)→I;Hr(p1+2)→J;Hr(p1+3)→K;rp1→H
51: HW-IX-JY-KZ→rp3
52: HX+IW+JZ-KY→r(p3+1)
53: HY+JW+KX-IZ→r(p3+2)
54: HZ+KW+IY-JX→r(p3+3);ret
55: "Q231":1→W;r(p1+1)→I;r(p1+3)→K;sto +2
56: "Q213":-1→W;-r(p1+3)→I;r(p1+1)→K
57: rp1→H;r(p1+2)→J
58: HH-II+JJ-KK→X;2(JK-HI)→Y
59: 'ATN2'(2(IJ+HK),r(XX+YY))→r(p2+1)
```

#TRAJ Cont.

```
60: 'ATN2'(J+I,H+K)→X:'ATN2'(J-I,H-K)→Y
61: 'ANG2'(X+Y)→rp2:'ANG2'(W(X-Y))→r(p2+2):ret
62: "Q313":rp1→H:r(p1+1)→I:r(p1+2)→J:r(p1+3)→K
63: 2'ATN2'(r(II+JJ),r(HH+KK))→r(p2+1)
64: 'ATN2'(K,H)→X:'ATN2'(J,I)→Y
65: 'ANG2'(X+Y)→rp2:'ANG2'(X-Y)→r(p2+2):ret
66: "231Q":1→W:ato +2
67: "213Q":-1→W
68: rp1/2→X:cos(X)→I:sin(X)→X
69: r(p1+1)/2→Y:cos(Y)→J:sin(Y)→Y
70: r(p1+2)/2→Z:cos(Z)→K:sin(Z)→Z
71: IJK-WXYZ→rp2
72: IJZ+WXYK→r(p2+2-W)
73: XJK+WIYZ→r(p2+2)
74: IYK-WXJZ→r(p2+2+W):ret
75: "3130":r(p1+1)/2→Y:cos(Y)→J:sin(Y)→Y
76: (rp1+r(p1+2))/2→X:cos(X)→I:sin(X)→X
77: (rp1-r(p1+2))/2→Z:cos(Z)→K:sin(Z)→Z
78: JI→rp2:YK→r(p2+1):YZ→r(p2+2):JX→r(p2+3):ret
79: "IMATQ":r(p1+5)→I:r(p1+6)→J:r(p1+1)→K
80: r(p1+7)→X:r(p1+2)→Y:r(p1+3)→Z
81: 1+rp1+r(p1+4)+r(p1+8)→H:if H<1:ato +2
82: 1/2√H→W:1/4W→rp2:(I-X)W→r(p2+1):(J-Y)W→r(p2+2):(K-Z)W→r(p2+3):ret
83: 2-H→W:W+2rp1→H:if H<1:ato +2
84: 1/2√H→W:(I-X)W→rp2:1/4W→r(p2+1):(K+Z)W→r(p2+2):(J+Y)W→r(p2+3):ret
85: W+2r(p1+4)→H:if H<1:ato +2
86: 1/2√H→W:(J-Y)W→rp2:(K+Z)W→r(p2+1):1/4W→r(p2+2):(I+X)W→r(p2+3):ret
87: W+2r(p1+8)→H
88: 1/2√H→W:(K-Z)W→rp2:(J+Y)W→r(p2+1):(I+X)W→r(p2+2):1/4W→r(p2+3):ret
89: "DERIVS":25p1→Q
```

#TRAJ Cont.

```
90: "GRAV":cll 'QCXQ'(0+25,0+29,0+87);cll 'QXQ'(0+87,0+80,0+129)
91: 0→r1→r2;1→r3;cll 'ROT'(1,0+25,0+133);-1→r3;cll 'ROT'(1,0+129,1)
92: r(0+33)→R;r24/RR→C;r(0+133)→U;r(0+134)→T;r(0+135)→S
93: .0032481(r23/R)↑2→B;C(B(5SS-1)/2-1)→A;CBS→B
94: BU→r7;BT→r8;BS-A→r9
95: r(0+77)→L;r(0+78)→M;r(0+79)→N;3C/R→C
96: Cr2r3(N-M)→r16;Cr1r3(L-N)→r17;Cr1r2(M-L)→r18
97: r124r(0+75)→P;if P=0;for I=10 to 15;0→rI;next I;sto "ANGAC"
98: "ATM":.00669342→E;TT+UU→C;-ESrC→D;EC→C;D/(1-C)→D
99: 20855591(1+(1+3C/4)C/2)→B;BD/R→A;(R-B)(1-DA/2)→H
100: .003377exp(r((H-394134)/12665)+17.449)→Q
101: 7.2921e-5→W;R(r(0+35)+TW)→r1;-RUW→r2;-r(0+34)→r3
102: cll 'ROT'(1,0+87,1);if p1=1;sto "PLAERO"
103: "AERO":r1r1+r3r3→A;r(A+r2r2)→V;r2/V→S;rA/V→C
104: 0→Y;1→X;if C#0;r3/VC→Y;r1/VC→X
105: 'ATN2'(S,C)→B;'ATN2'(Y,X)→A;abs(S)→S;sin(2B)→E
106: (.786+2.413abs(Y)↑1.316)(1-S)+1.928S+.646abs(EYY)→D
107: -1345PDQV→D;cll 'SXV'(D,1,10)
108: .031(sin(3B)-Ecos(2A))-.036B→I
109: (.045cos(3A)-.162Y)CC-.045Xcos(3B)→J
110: .067sin(2.09AB)-.01(Y+Xrobs(E))E→K
111: PQVV→A;533100→B;104986A→A;AI+r13;BJ→r14;AI→r15;sto +4
113: "PLAERO":r122abs(r1)+r123r(r2r2+r3r3)→A
113: -APO→A;cll 'SXV'(A,1,10)
114: -r121/2→r1;0→r2→r3;cll 'CRSP'(1,10,13)
115: "ENDERO":cll 'CRSP'(0+84,10,1)
116: cll 'VSUB'(13,1,13);cll 'IROT'(10,0+87,10)
117: 32.174 r(0+39)→G;cll 'SXV'(G,10,10)
118: cll 'ROT'(13,0+80,1);cll 'VADD'(16,1,16)
119: "ANGAC":cll 'ROT'(0+36,0+80,1);Lr1→r4;Mr2→r5;Nr3→r6
```

```
120: cll 'CRSP'(1,4,4)↓cll 'VSUB'(16,4,1)
121: r1/L→r1↓r2/M→r2↓r3/N→r3↓cll 'IROT'(1,0+80,0+136)
122: if r1=1↓sto "PLTHR"
123: "THR":r94→J↓Jmod2→I↓(J-I)/2→K
124: 19.16K→r141↓19.16I→r140↓0→r139
125: 6000G(I+K)→A↓cll 'SNV'(A,9i,13)
126: r95→J↓if J=0↓for I=142 to 147↓0→r[next I↓sto +11
127: for I=13 to 15↓r[+R[I-12,J]→r[↓next I
128: for I=136 to 138↓r[+R[I-132,J]→r[↓next I
129: for I=142 to 147↓R[I-135,J]→r[↓next I↓sto +8
130: "PLTHR":1→r174↓0→V→r13→r14→r15→r164↓if r119=0↓sto +7
131: r74→T↓if T`A[2]↓sto +6
132: 1→V↓T-A[1]→T↓if T<=A[7]↓sto +3
133: A[7]→X↓A[8]→Y↓read 3,A[7],A[8]↓if T>A[7]↓sto +0
134: (A[8]-Y)-(A[7]-X)→A[6]↓Y-XA[6]→A[5]
135: -(A[5]+TA[6])A[3]→r164→G↓-32.174GA[4]→r64→G
136: for I=13 to 15↓Gr(I+103)→r[↓next I
137: "ENDTHR":cll 'IROT'(13,0+87,13)
138: for I=7 to 9↓r[+r(I+3)+r(I+6)→r(I+0+123)↓next I
139: "AMC":r(0+76)→A↓if A=0↓sto "ENDDER"
140: if A=1↓for I=16 to 18↓r(I+0+120)→r[↓next I↓sto +9
141: 0→r1↓r(0+35)→r2↓-r(0+131)→r3
142: cll 'IROT'(0+36,0+87,7)↓cll 'VSUB'(7,1,4)↓cll 'CRSP'(4,13,4)
143: 2r5→A↓if r1=1↓if V#0↓A-(A[6]→r164+r164→r64→A
144: cll 'CRSP'(0+133,1,4)↓r(0+34)→S↓r(0+35)→W
145: .0032481r24(r23.RR)↑2(r5r(0+135)+r6r(0+134))↑-4Sr8.R+A→A
146: (r(0+130)-2SW).R→W↓0→r4↓-X→r5↓A.RW-(S.P+X.W)→r3→r6
147: cll 'CRSP'(1,7,7)↓cll 'VADD'(4,7,4)↓cll 'PUT'(4,0+87,4)
148: for I=16 to 18↓r(I+0+120)→r(I-12)→r[↓next I
149: if r1=1↓sto "PLAMI"
```

#TRAJ Cont.

```
150: "AMI":for I=7 to 9;0→rI;next I
151: for K=16 to 18;K-12→H;2H→J;J-1→G;if rK<0;G→J;J+1→G
152: abs(rK)→A;abs(R[H,J])→E;if A>E;E→A;1→r285
153: .014→F;r99→C;r(H+92)→D;F+4r(AD)/E→T;Cint(T/C)→T
154: if T>0;T-F→S;0→B→Y→Z;goto +2
155: C→T;T-F→S;EESS/16D-A→B;B/abs(R[H,G])→Z;ZT/S→Y
156: (A+B)/E→X;XT/S→W
157: for I=7 to 9;I-6→H;rI+XR[H,J]+ZR[H,G]→rI;next I
158: for I=136 to 138;I-132→H;rI+XR[H,J]+ZR[H,G]→rI;next I
159: for I=142 to 147;I-135→H;rI+WR[H,J]+YR[H,G]→rI;next I
160: next I;cll 'IROT'(7,87,7);cll 'VADD'(130,7,130);goto +5
161: "PLAMI":for I=161 to 163;rI-r(I-145)→rI;next I
162: cll 'PUT'(16,105,1);-Lr1→r1;-Mr2→r2;-Nr3→r3
163: cll 'IROT'(1,105,1);for I=1 to 3;rI/2→Y
164: abs(Y)→X;X+Y→r(I+164);X-Y→r(I+167);next I
165: "ENDDER":for I=7 to 9;r(I+O+123)→rI;next I
166: r(O+34)→O→r(O+133);r(O+35)→W;RWW-r9→r(O+134)
167: (r7-29W)/R→r(O+135);0→r1;-W→r2;r8/RW→r3
168: cll 'ODOT'(O+25,1,O+125);cll 'QDOT'(O+29,O+36,O+129);ret
169: "RUN":set "LRMAT",169
170: end
-1A018
```

C.2 Response Matrix Computation Link (%RMAT)

```
0: "%RMAT(0403/24SEP79)":goto "RUN"
1: wtb 6,11;list #6
2: get "%INIT",0,1
3: "RUN":lkd ;rod;if fl=11;goto +5
4: dim A[3],B[32],F[12,12],I[3,3],J[8,12]
5: dim L[3],U[12,12],R[12,12],T[7,44]
6: dim B$[15],C$[12,4],D$[25],J$[3,4],K$[5]
7: dim M$[44,3],P$[3],V$[80],W$[80];sfg 11
8: "* WARNING:  SS "→B$
9: " CONTROL IMPOSSIBLE WITH "→D$
10: "   V"→J$[1];"   P"→J$[2];" PZI"→J$[3]
11: " JETS"→K$
12: fmt 1,/,c80,/
13: assn "$CMDID",1;read 1,C$
14: assn "$JFTM",1;for I=1 to 44;read 1,M$[I];next I
15: assn "$JFT",1;read 1,T[*]
16: assn "1*",1;read 1,B[*];if B[32]=9;goto "RMEND"
17: for I=1 to 3;B[I+1]→I[I,I];next I
18: -B[5]→I[2,3]→I[3,2]
19: -B[6]→I[1,3]→I[3,1]
20: -B[7]→I[1,2]→I[2,1]
21: inv I→I
22: for T=1 to 3;dsp "%RMAT";J$[T];imp T
23: files #JSV,*V,*VR;goto +3
24: files #JSP,*P,*PR,*PF;goto +2
25: files #JSPZI,*PZI,*PZIR,*PZIF
26: ino U,P,F;""→V$→W$;read 1,J[*]
27: for C=1 to 12;for N=1 to 8
28: J[N,C]→M;if M=0;goto +13
29: for I=1 to 6;T[I,M]→r[I];next I
```

```
30: for I=4 to 6;(rI-B[I+4])/12→rI;next I
31: -r4→r4;-r6→r6;cll 'CRSP'(4,1,4)
32: for I=4 to 6;rI+T[7,M]r(I-3)→rI;next I
33: for I=1 to 6;U[I,C]+rI→U[I,C];next I
34: M$[M]→P$
35: if P$[1,1]="R";9→I;gto +3
36: if P$[1,1]="L";8→I;gto +2
37: 7→I
38: if C>6;I+3→I
39: 3.1071→W;if M>38;.0923→W
40: U[I,C]+W→U[I,C]
41: next N
42: 32.174/B[1]→A;for I=1 to 3;AU[I,C]→U[I,C];next I
43: for I=4 to 6;U[I,C]→L[I-3];next I
44: mat IL→A
45: for I=4 to 6;A[I-3]→U[I,C];next I
46: if T=1;if C<7;gto +4
47: int((C+1)/2)→P;4P-2C-1→S;if SU[P,C]>0;gto +3
48: B$&C$[C]&D$&J$[T]&K$→V$;wrt 6.1,V$;""→V$
49: if C>6;B$&"ATT"&D$&J$[T]&K$→W$
50: next C;sprt 2,U[*],W$;ora U→R
51: if W$##"";gto "REND"
52: for C=7 to 12;int((C+1)/2)→P
53: for N=1 to 10
54: for H=0 to 1;4+(P+H)mod3→E
55: -R[E,C]→X;2E→K;if X>0;K-1→K
56: X/U[E,K]→Y;for I=1 to 12;R[I,C]+YU[I,K]→R[I,C];next I
57: next H;0→X
58: for H=0 to 1;4+(P+H)mod3→E;X+R[E,C]↑2→X
59: next H;X/R[P,C]↑2→X;if X<1e-8;gto +2
```

```
60: next N;gto +2
61: 4P-2C-1→S;if SR[P,C]>0;gto +2
62: B#&"ATT"&D$&J$[T]&K$→W$;wrt 6.1,W$;ara U→R;gto "REND"
63: for H=0 to 1;4+(P+H)mod3→E;0→R[E,C];next H
64: next C
65: for C=1 to 6
66: for E=4 to 6
67: -R[E,C]→X;2E→K;if X>0;K-1→K
68: % R[E,K]→Y;for I=1 to 12;F[I,C]+YR[I,K]→R[I,C];next I
69: 0→R[E,C];next E
70: if T=1;gto +3
71: int((C+1)/2)→P;4P-2C-1→S;if SR[P,C]>0;gto +2
72: B#&"ROT COMP "&C$[C]&D$&J$[T]&K$→V$;wrt 6.1,V$
73: next C
74: "FEND":sprt 3,R[*],W$;if T=1;gto "NEXTT"
75: ara R→F
76: if W$#"";gto "FEND"
77: if V$#"";gto "ABORTF"
78: for C=1 to 6;int((C+1)/2)→P
79: for H=1 to 10
80: for H=0 to 1;1+(P+H)mod3→E
81: -F[E,C]→X;2E→K;if X>0;K-1→K
82: % R[E,K]→Y;for I=1 to 12;F[I,C]+YR[I,K]→F[I,C];next I
83: next H;0→X
84: for H=0 to 1;1+(P+H)mod3→E;X+F[E,C]↑2→X
85: next H;% F[P,C]↑2→X;if X<1e-8;gto +2
86: next N;gto "ABORTF"
87: 4P-2C-1→S;if SF[P,C]≠0;gto "ABORTF"
88: for H=0 to 1;1+(P+H)mod3→E;0→F[E,C];next H
89: next C
```

```
90: for C=7 to 12
91: for E=1 to 3
92: -F[E,C]→X;2E→K;if X>0;K-1→K
93: X/F[E,K]→Y;for I=1 to 12;F[I,C]+YF[I,K]→F[I,C];next I
94: 0→F[E,C];next E
95: int((C+1)/2)→P;4P-2C-1→S;if SF[P,C]<=0;gto "ABORTF"
96: next C;gto "FEND"
97: "ABORTF":B$&"FULL COMP"&D$&J$[T]&K$→W$
98: wrt 6.1,W$;ara R→F
99: "FEND":sprt 4,F[*],W$
100: "NEXTT":next T
101: 9→B[32]
102: asgn "1*",1;sprt 1,B[*]
103: "RMEND":set "%TNIT",169
104: end
*24706
```

C.3 Trajectory Initialization Link (%TNIT)

```
0: "%TNIT(1238/16OCT79)":gto "RUN"
1: wtb 6,11;list #6
2: get "%SSVU",0,1
3: "JD":int((p2-14)/12)→p4
4: p3-32075+int(1461(p1+4800+p4)/4)→p5
5: p5+int(367(p2-2-12p4)/12)→p5
6: ret p5-int(3int((p1+4900+p4)/100)/4)
7: "PLOTYP":
8: if P$="    ";ret 0
9: if P$="NONE";ret 0
10: if P$="RSBY";ret 1
11: if P$="CPLV";ret 2
12: ret -9
13: "UNIT":
14: if P$="  FT";ret 1
15: if P$=" KFT";ret 1000
16: if P$=" NMI";ret 6076.115
17: if P$="   M";ret 1/.3048
18: if P$="  KM";ret 1000/.3048
19: ret -9
20: "MONTH":
21: if P$=" JAN";ret 1
22: if P$=" FEB";ret 2
23: if P$=" MAR";ret 3
24: if P$=" APR";ret 4
25: if P$=" MAY";ret 5
26: if P$=" JUN";ret 6
27: if P$=" JUL";ret 7
28: if P$=" AUG";ret 8
29: if P$=" SEP";ret 9
```

```
30: if P$=" OCT"⁞ret 10
31: if P$=" NOV"⁞ret 11
32: if P$=" DEC"⁞ret 12
33: ret -9
34: "C"⁞ret p3(p2-1)+p1-1
35: "T":max(p1,p2)→p3⁞ret p3(p3-1)/2+min(p1,p2)-1
36: "DIAG":0→W→X
37: for J=1 to p1⁞for I=1 to J
38: 'T'(I,J)→K⁞p2+K→H⁞p3+K→K⁞rH→rK→Y
39: if I=J⁞W+YY→W⁞1→r(p4+'C'(J,J,p1))⁞gto +2
40: X+2YY→X⁞0→r(p4+'C'(I,J,p1))→r(p4+'C'(J,I,p1))
41: next I⁞next J
42: if p0<5⁞r(W+X)/p1/1e6→p5
43: rX/p1→p6⁞0→p7→p8
44: for J=  to p1⁞for I=1 to J-1
45: p3+'T'(I,J)→Z⁞if abs(rZ)<=p6⁞gto +12
46: 1→p8⁞p3+'T'(I,I)→H⁞p3+'T'(J,J)→W
47: rH-rW→X⁞2rZ→Y⁞Y/r(XX+YY)→K⁞if X<0⁞-K→K
48: K/r(2(1+r(1-KK)))→Y⁞r(1-YY)→X⁞rH→K
49: KXX+2XYrZ+YYrW→rH⁞KYY-2XYrZ+XXrW→rW⁞0→rZ
50: for K=1 to p1⁞if K=I⁞gto +4
51: if K=J⁞gto +3
52: p3+'T'(K,I)→H⁞p3+'T'(K,J)→W
53: rH→Z⁞XZ+YrW→rH⁞XrW-YZ→rW
54: p4+'C'(K,I,p1)→H⁞p4+'C'(K,J,p1)→W
55: rH→Z⁞XZ+YrW→rH⁞XrW-YZ→rW
56: next K
57: next I⁞next J⁞if p8=1⁞0→p8⁞gto +3
58: if p6<p5⁞ret
59: p6/p1→p6
```

```
60: p7+1→p7;if p7<100;goto -16
61: fmt "DIAG ERROR =",e10.2,2/;wrt 6,p1p6;fmt ;ret
62: "RUN":wtb 6,11;if fl≥11;goto +5
63: dim A[8],B[32],G[12],R[12,12],T[7]
64: dim A$[9,2],B$[40],C$[2,4],D$[3,4],E$[5,4],F$[4,4]
65: dim G$[3,4],H$[8,3],I$[6,3],J$[6,1],K$[80],L$[4],M$[6],N$[6]
66: dim O$[6,3],P$[4],Q$[6,3],R$[6,3],S$[6,3],U$[10],W$[80],Z$[80];sfg 11
67: "SS"→A$[1];"PL"→A$[2];"TP"→A$[3];"DX"→A$[4];"DY"→A$[5];"DZ"→A$[6]
68: "DR"→A$[7];"DS"→A$[8];"DT"→A$[9]
69: "OM50"→C$[1];"IMLD"→C$[2]
70: " M50"→D$[1];" SLV"→D$[2];" PLV"→D$[3]
71: "RSBY"→E$[1];"RSLV"→E$[2];"CSLV"→E$[3];"ESLV"→E$[4];"SM50"→E$[5]
72: "RPBY"→F$[1];"RPLV"→F$[2];"CPLV"→F$[3];"EPLV"→F$[4]
73: "SENS"→G$[1];"PCON"→G$[2];"RIMP"→G$[3]
74: "SMA"→H$[1];"ECC"→H$[2];"INC"→H$[3];"RAN"→H$[4]
75: "ARG"→H$[5];"TRA"→H$[6];"HA "→H$[7];"HP "→H$[8]
76: "PCH"→I$[1];"YAW"→I$[2];"ROL"→I$[3];"RWB"→I$[4];"RYB"→I$[5];"RZB"→I$[6]
77: "X"→J$[1];"Y"→J$[2];"Z"→J$[3];"R"→J$[4];"S"→J$[3];"T"→J$[6]
78: "+X "→O$[1];"+Y "→O$[2];"+Z "→O$[3];"-X "→O$[4];"-Y "→O$[5];"-Z "→O$[6]
79: "RFX"→O$[1];"RLX"→O$[2];"RRX"→O$[3];"RAX"→O$[4];"RX "→O$[5];"OML"→O$[6]
80: "RFR"→P$[1];"RLR"→P$[2];"RRR"→R$[3];"RAR"→P$[4];"RR "→P$[5];"OMR"→P$[6]
81: "RFT"→S$[1];"RLT"→S$[2];"RRT"→S$[3];"RAT"→S$[4];"RT "→S$[5];"OMT"→S$[6]
82: asgn "→ID",1;aread 1,Z$;"#TRAJ"→Z$[1,5]
83: beepient "ENTER TRAJECTORY ID TEXT",K$;if fl≥13;goto +0
84: 80-len(K$)→K;if K=0;goto +2
85: for H=1 to K;K$&" "→K$;next H
86: fmt 2/,c80,2/,c80,/;wrt 6,Z$,K$;fmt
87: 1;d ;rad;0→r235;π/180→Q
88: 20925722→r23;1.4076469e16→r24
89: asgn "1*",1;aread 1,B[→];B[1]→r39
```

```
90: B[2]→r1;B[3]→r3;B[4]→r6;-B[5]→r5;-B[6]→r4;-B[7]→r2
91: cll 'DIAG'(3,1,1,7);r1→r77;r3→r78;r6→r79;cll 'IMATQ'(7,80)
92: (1076.7-B[8])/12→r84;B[9]/12→r85;(375-B[10])/12→r86
93: (1076.7-B[11])/12→r275;B[12]/12→r276;(375-B[13])/12→r277
94: for I=1 to 3;0B[I+13]→r[I;next I;cll '2310'(1,278)
95: B[17]/1e3→r99;for I=40 to 47;0→r[I;next I
96: oean "2*",1;sread 1,B[*];B[1]→r64
97: B[2]→r1;B[3]→r3;B[4]→r6;-B[5]→r5;-B[6]→r4;-B[7]→r2
98: cll 'DIAG'(3,1,1,7);r1→r102;r3→r103;r6→r104;cll 'IMATQ'(7,105)
99: -B[8]/12→r109;B[9]/12→r110;-B[10]/12→r111
100: -B[11]/12→r282;B[12]/12→r283;-B[13]/12→r284
101: B[14]/24→R→r120;B[15]/12→L→r121
102: 0RR→r122;2RL→r123
103: for I=65 to 70;0→r[I;next I
104: oean "3*",1;for I=1 to 11;sread 1,G[I];next I
105: fts (G[1])→P#;'PLOTYP'→G[1];if G[1]=0;sto +13
106: fts (G[2])→P#;'UNIT'→G[2]
107: beepient "READY TO PLOT ?",I;if fl≥13=0;sto +0
108: pclr;wrt 705,"ip540,876,14778,10030";pen# 2;fxd 0
109: scl 0,14,0,9;pen;plt 0,-.3;lbl K$
110: G[3]→C;G[9]-CG[6]→Y;if G[5]=0;sto +4
111: scl 0,-14C,0,-9C;ofs -G[8],-G[9];G[8]-CG[5]→X
112: xax 0,G[10],0,G[8],G[11];xax 0,-G[10],0,X+C/10,G[11]
113: xax 0,G[10],0,G[9],G[11];xax 0,-G[10],0,Y,G[11]
114: if G[4]=0;sto +4
115: scl 0,-14C,0,-9C;ofs -CG[5]-G[7],-G[9];G[7]-CG[4]→X
116: xax 0,G[10],0,G[7]-C/10,G[11];xax 0,-G[10],0,X,G[11]
117: xax 0,G[10],0,G[9],G[11];xax 0,-G[10],0,Y,G[11];pen# 1
118: oean "4*",1;sread 1,B[*]
119: B[1]→Y;fts (B[2])→P#;'MONTH'→M;B[3]→D
```

%TNIT Cont.

```
120: ('JD'(Y,M,D)-2433282)/36524.22→T
121: (2304.948+(.302+.018T)T)T→X
122: (2004.256-(.426+.042T)T)T→Y
123: (2304.948+(1.093+.018T)T)T→Z
124: π/648000→C;π/2-CX→r1;CY→r2;-π/2-CZ→r3
125: cll '313Q'(1,19)
126: 'SECS'(B[5])→r74;B[6]→r124
127: osan "7*",1;on end 1,"G02";sread 1,B[*]
128: "G02":B[5]→r75;B[18]→r100;0→r76→r94→r95→r101→r119
129: osan "5*",1;sread 1,B[*];fts (B[1])→P$
130: 6076.115→T;r24→G;if P$="OM50";TB[2]→A;B[3]→E;sto +2
131: TB[2]+r23→A;TB[3]+r23→P;(A-P)/(A+P)→E;(A+P)/2→A
132: A(1-EE)→P;0B[7]→F;P/(1+Ecos(F))→R;r(G/P)Esin(F)→S
133: 0B[5]→r1;0B[4]→r2;0B[6]+F→r3
134: if P$="OM50";cll '313Q'(1,25);cll '0CXQ'(19,25,25);sto +8
135: r1→H;r2→I;r3→U;cos(I)→C;sin(I)→D
136: cos(U)→K;sin(U)→L;KK-LL→M;2KL→N
137: 1.62405e-3→J;Jr23↑2→V;V/P→B
138: R+BDDM/6→R;A+B(1-3DD/2)/3→A;r(G/A)/A→W
139: S-BWDDN/3→S;A+2AAV(1-3DDLL)/3RRR→A
140: U-B(1-7DD/6)N/2P→r3;I+VCDM/2RR→r2;H+BCN/4P→r1
141: cll '3130'(1,25)
142: π/2→r4;-r4→r5;0→r6;cll '3130'(4,0);cll '0X0'(25,0,25)
143: r((2/R-1/A)J-SS)/R→r35;S→r34;R→r33
144: for I=1 to 3;0B[I+8]→rI;next I
145: cll '2310'(1,87);cll '0X0'(25,87,29)
146: for I=36 to 38;0B[I-28]→rI;next I
147: cll 'DERIVS'(0);π/180→Q
148: fts (B[12])→P$;if P$=" SLV";cll 'ROT'(1,87,4);cll 'VADD'(4,36,36)
149: osan "6*",1;sread 1,B[*]
```

%TNIT Cont.

```
150: for I=7 to 12;B[I-5]→rI;next I
151: fts (B[1])→P$;if P$="RSLV";sto +4
152: cll 'ROT'(1,87,4);cll 'VSUB'(36,4,4)
153: cll 'CRSP'(4,7,4);cll 'VADD'(10,4,10)
154: cll 'IROT'(7,87,7);cll 'IROT'(10,87,10)
155: r9-r33→r9;r12-r34→r12
156: cll 'CRSP'(1,7,4);cll 'VADD'(10,4,10)
157: cll 'IROT'(7,25,7);cll 'IROT'(10,25,10)
158: r'DOTP'(7,7)→R→r58;cll 'CRSP'(7,10,4)
159: r'DOTP'(4,4)→H;H/RR→W→r60
160: for I=4 to 6;-rI/H→rI;-r(I+3)/R→r(I+3);next I
161: -'DOTP'(10,7)→r59;cll 'CRSP'(4,7,1)
162: cll 'IMATQ'(1,50)
163: for I=1 to 3;0B[I+8]→rI;next I;cll '2310'(1,112)
164: fts (B[8])→P$;if P$=" PLV";sto +2
165: cll 'OCXQ'(50,29,0);cll 'QXQ'(0,112,112)
166: cll 'QXQ'(50,112,54)
167: for I=61 to 63;0B[I-48]→rI;next I
168: fts (B[12])→P$;if P$=" M50";sto +6
169: if P$=" PLV";sto +3
170: cll 'OCXQ'(29,54,0);cll 'ROT'(36,0,4)
171: cll 'VADD'(4,61,61);sto +3
172: cll 'DERIVS'(*)
173: cll 'ROT'(1,112,4);cll 'VADD'(4,61,61)
174: asam "7*",1
175: if G[1]=1;chain "%SSVU",169
176: chain "%SNIT",169
177: end
*23100
```

C.4 Orbiter Geometry Link (%SSVU)

```
0: "%SSVU(1108/16SEP79)":sto "RUN"
1: wtb 6,11;list #6
2: get "%SNIT",0,1
3: "SYM":abs(int(p6))→H;p5/H→J;cos(J)→I;sin(J)→J
4: cos(p4)→X;sin(p4)→Y;0→K
5: plt p1+Xp3,p2+Yp3;if K=H;ret
6: JX+IY→W;IX-JY→X;W→Y;K+1→K;sto -1
7: "RUN":dea;12G[2]→D;DG[3]→C
8: pen# 2;if G[5]=0;sto "SVU"
9: scl 0,-140,0,90;ofs -DG[8],DG[9]
10: lim DG[8],DG[8]-CG[5],-DG[9],CG[6]-DG[9]
11: ofs -12rp5,12rp6-375
12: -1→S
13: pen;plt 0,261
14: plt S50,262
15: plt S100,263
16: plt S468,300
17: plt S468,312
18: plt S200,327
19: plt S130,336
20: plt S125,403
21: plt S150,389
22: plt S175,389
23: plt S200,403
24: plt S220,412
25: plt S241,424
26: plt S232,424
27: plt S220,422
28: plt S200,413
29: plt S175,409
```

%SSVU Cont.

```
30: plt S150,409
31: plt S125,413
32: plt S120,415
33: plt S130,430
34: plt S139,450
35: plt S143,462
36: plt S145,475
37: plt S144,490
38: plt S140,509
39: plt S130,515
40: plt S115,524
41: plt S100,526
42: plt S80,526
43: plt S60,524
44: plt S40,518
45: plt S30,512
46: plt S20,513
47: plt S4,816
48: if S>0:1+S:goto -35
49: plt -S4,816
50: line 3,1:S15+X:484.8+Y:240+Z
51: pen:plt X+Y:plt X+SZcos52,Y+Zsin52
52: pen:plt X+Y:plt X+SZcos119.5,Y+Zsin119.5:line
53: "SVU":if G[4]-0:goto "END"
54: scl 0,140,0,90:ofs DG[5]+DG[7],DG[9]
55: lim -DG[7],DG[4]-DG[7],-DG[9],DG[6]-DG[9]
56: ofs 12r04-1076.7,12r06-375
57: pen:plt 1613,291
58: plt 1534,298
59: plt 1536,336
```

%SSVU Cont.

```
60: plt 1550,335
61: plt 1575,334
62: plt 1600,334
63: plt 1623,336
64: plt 1609,412
65: plt 1600,418
66: plt 1575,402
67: plt 1550,392
68: plt 1525,382
69: plt 1517,376
70: plt 1502,441
71: plt 1525,441
72: plt 1550,442
73: plt 1575,445
74: plt 1598,453
75: plt 1578,526
76: plt 1550,517
77: plt 1525,506
78: plt 1500,489
79: plt 1492,483
80: plt 1483,519
81: plt 1575,562
82: plt 1700,816
83: plt 1593,816
84: plt 1325,548
85: plt 1317,540
86: plt 1311,533
87: plt 1308,524
88: plt 1307,516
89: plt 1307,420
```

```
90: plt 576,420
91: plt 576,500
92: plt 500,500
93: plt 491,499
94: plt 480,493
95: plt 472,489
96: plt 434,450
97: plt 300,394
98: plt 283,385
99: plt 270,378
100: plt 259,372
101: plt 250,364
102: cll 'SYM'(262.5,338,27.5,135,102,10)
103: plt 263,308
104: plt 286,300
105: plt 312,294
106: plt 337,289
107: plt 362,286
108: plt 387,283
109: plt 414,280
110: plt 450,278
111: plt 492,275
112: plt 535,273
113: plt 575,272
114: plt 625,271
115: plt 675,269
116: plt 725,268
117: plt 775,267
118: plt 900,265
119: plt 1000,264
```

```
120: plt 1100,263.
121: plt 1150,262
122: plt 1250,261
123: plt 1300,261
124: plt 1350,264
125: plt 1400,267
126: plt 1450,271
127: plt 1500,275
128: plt 1550,281
129: plt 1600,288
130: plt 1613,290
131: line 2,1;545.2→X;484.8→Y;24C→Z
132: pen;plt X,Y;plt X+Zcos(100),Y+Zsin(100)
133: pen;plt X,Y;plt X+Zcos(38),Y+Zsin(38)
134: 560→X;480→Y
135: pen;plt X,Y;plt X+Zcos(30),Y+Zsin(30)
136: pen;plt X,Y;plt X+395cos(-32),Y+395sin(-32);line
137: "END":pen;pen# 1
138: chain "%SNIT",169
139: end
*22779
```

C.5 Flight Segment Initialization Link (%SNIT)

```
 0: "%SNIT(1340/27OCT79)":goto "RUN"
 1: wtb 6,11;list #6
 2: get "%PROP",0,1
 3: "RAM":
 4: if P$="   D";ret 0
 5: if P$=" IRH";ret 1
 6: if P$="LVRH";ret 2
 7: ret -9
 8: "JSEL":
 9: if P$="   V";ret 0
10: if P$="   P";ret 1
11: if P$=" PZI";ret 2
12: ret -9
13: "COMP":
14: if P$="    ";ret 0
15: if P$="NONE";ret 0
16: if P$=" ROT";ret 1
17: if P$="FULL";ret 2
18: ret -9
19: "PTOMS":
20: if P$="    ";ret 0
21: if P$="NONE";ret 0
22: if P$="   L";ret 1
23: if P$="   R";ret 2
24: if P$=" L+R";ret 3
25: ret -9
26: "PTRCS":
27: if P$="    ";ret 0
28: if P$="NONE";ret 0
29: if P$="  +X";ret 1
```

```
30: if P$="  -X";ret 2
31: if P$="  +Y";ret 3
32: if P$="  -Y";ret 4
33: if P$="  +Z";ret 5
34: if P$="  -Z";ret 6
35: if P$="+ROL";ret 7
36: if P$="-ROL";ret 8
37: if P$="+PCH";ret 9
38: if P$="-PCH";ret 10
39: if P$="+YAW";ret 11
40: if P$="-YAW";ret 12
41: ret -9
42: "DATYP";jmp p1
43: "FLT SEGMENT TYPE (PROP)................."→B$;"          "→U$;ret 3
44: "FLT SEGMENT LENGTH....................."→B$;"HHMM.SS   "→U$;ret 1
45: "PRINT/PLOT INTERVAL...................."→B$;"HHMM.SS   "→U$;ret 1
46: "MAX INTEG STEPSIZE....................."→B$;"HHMM.SS   "→U$;ret 1
47: "SS AERO FACTOR........................."→B$;"          "→U$;ret 1
48: "SS RATE OPT (INCR,IR,LVR)..............."→B$;"          "→U$;ret 3
49: "SS XB RATE OR INCR....................."→B$;"DEG/SEC   "→U$;ret 1
50: "SS YB RATE OR INCR....................."→B$;"DEG/SEC   "→U$;ret 1
51: "SS ZB RATE OR INCR....................."→B$;"DEG/SEC   "→U$;ret 1
52: "SS ATT MAINT OPT (D,IRH,LVRH)..........."→B$;"          "→U$;ret 3
53: "SS XB DEADBAND........................."→B$;"DEG       "→U$;ret 1
54: "SS YB DEADBAND........................."→B$;"DEG       "→U$;ret 1
55: "SS ZB DEADBAND........................."→B$;"DEG       "→U$;ret 1
56: "SS RCS OPT (V,P,PZI)..................."→B$;"          "→U$;ret 3
57: "SS XC COMPENSATION (NONE,ROT,FULL)......"→B$;"          "→U$;ret 3
58: "SS OMS CMD (NONE,L,R,L+R)..............."→B$;"          "→U$;ret 3
59: "SS RCS CMD (NONE,+X,-X,+Y,...,+YAW,-YAW)"→B$;"          "→U$;ret 3
```

```
60: "PL AERO FACTOR....."→B$;"                    "→U$;ret 1
61: "PL RATE OPT (INCR,IR,LVR)....."→B$;"         "→U$;ret 3
62: "PL XB RATE OR INCR....."→B$;"DEG/SEC         "→U$;ret 1
63: "PL YB RATE OR INCR....."→B$;"DEG/SEC         "→U$;ret 1
64: "PL ZB RATE OR INCR....."→B$;"DEG/SEC         "→U$;ret 1
65: "PL ATT MAINT OPT (D,IRH,LVRH)....."→B$;"     "→U$;ret 3
66: "PL THR TABLE (NONE,IL,SA,SD)....."→B$;"      "→U$;ret 3
67: "SLIST":
68: fmt 1,2/,"FLT PROFILE SEGMENT",f3.0
69: fmt 2,2/,"ITEM",2x,"DESCRIPTION",38x,"VALUE",2/
70: fmt 3,f4.0,c42,f14.4,c11
71: fmt 4,f4.0,c42,f14.0,c11
72: fmt 5,f4.0,c42,c14,c11
73: fmt 9,2/;fmt
74: if p1>1;wtb 6,11
75: wrt 6.1,p1;wrt 6.2
76: for I=1 to p2;jmp 'DATYP'(I)
77: wrt 6.3,I,B$,B[I],U$;sto +3
78: wrt 6.4,I,B$,B[I],U$;sto +2
79: fts (B[I])→P$;wrt 6.5,I,B$,P$,U$
80: next I;wrt 6.9;ret
81: "RUN":radin 180→D
82: T[6]+1→T[6];0→G[12];if T[6]=1;1→G[12];""→L$
83: 24→L;on end 1,"FIN"
84: for I=1 to L;sread 1,B[I];next I;cll 'SLIST'(T[6],L)
85: for I=1 to 3;'SECS'(B[I+1])→T[I];next I;T[1]+r74→T[1]
86: B[5]→r75
87: fts (B[10])→P$;'LAM'→K→r76
88: fts (B[14])→P$;'JSEL'→J
89: if J=0;"→V"→M$;sto +3
```

%SNIT Cont.

```
90: if J=1;"*P"→M$;gto +2
91: "*PZI"→M$
92: fts (B[15])→P$;'COMP'→C
93: if K>0;max(1,C)→C
94: if J=0;min(1,C)→C
95: if C=0;M$&"R"→N$;gto +3
96: if C=1;M$&"R"→M$→N$;gto +2
97: M$&"F"→M$→N$
98: asan M$,2;sread 2,R[*],W$
99: if W$#"";fmt /,c80,/;wrt 6,W$;fmt
100: fts (B[17])→P$;'KTRCS'→r95
101: fts (B[16])→P$;'KTOMS'→r94;if r94=0;gto +6
102: (1518-1076.7)/12→r1;0→r2;(492-375)/12→r3
103: if r94=1;88/12→r2
104: if r94=2;-88/12→r2
105: cll 'VADD'(1,84,91);r'DOTP'(91,91)→X
106: for I=91 to 93;rI/X→rI;next I
107: for I=96 to 98;0B[I-85]→rI;next I
108: cll 'DERIVS'(0);π/180→Q
109: 0→r7→r8;-r33→r9;cll 'CRSP'(1,7,10)
110: r12-r34→r12;cll 'ROT'(10,87,10)
111: fts (B[6])→P$
112: if P$="INCR";for I=4 to 6;0→rI;next I;gto +3
113: if P$="  IR";for I=4 to 6;r(I+32)→rI;next I;gto +2
114: cll 'ROT'(1,87,4);cll 'VSUB'(36,4,4)
115: for I=4 to 6;0B[I+3]-rI→rI;next I
116: asan N$,2;sread 2,R[*],W$
117: if W$#"";fmt /,c80,/;wrt 6,W$;fmt
118: for K=4 to 6;if rK=0;gto +6
119: 2K→C;if rK<0;C-1→C
```

```
120: max(0,rK/R[K,C])→T
121: for I=10 to 12;rI+TR[I-9,C]→rI;next I
122: for I=36 to 38;rI+TR[I-32,C]→rI;next I
123: for I=42 to 47;rI+TR[I-35,C]→rI;next I
124: next K
125: cll 'IROT'(7,25,7);cll 'IROT'(10,29,10)
126: cll 'CRSP'(7,10,4);r33→R
127: r'DOTP'(4,4)→H;H/RR→W→r35
128: for I=4 to 6;-rI/H→rI;-r(I+3)/R→r(I+3);next I
129: -'DOTP'(10,7)→r34;cll 'CRSP'(4,7,1)
130: cll 'IMATQ'(1,25)
131: oesn M$,2;sread 2,R[*]
132: B[18]→r100;fts (B[23])→P$;'KAM'→r101
133: fts (B[24])→P$
134: if P$="NONE";sto +4
135: if P$="    ";sto +3
136: if L$="";sto +4
137: fmt "SRM ",c4," IGNITION COMMAND IGNORED",/;wrt 6,P$;fmt
138: if r119#0;fmt "SRM ",c4," BURN CONTINUES",/;wrt 6,L$;fmt
139: sto +8
140: P$→L$
141: for I=1 to 4;if P$[I,I]=" ";next I
142: "00"&P$[I,4]→N$;N$[3,6]→P$
143: oesn N$,3;sread 3,A[2],A[3],A[4],A[8]
144: r74→A[1];r74+A[2]→A[2];A[8]→A[5];0→A[6]→A[7]
145: 1→r119→r116;0→r117→r118
146: oesn "8*",5;sprt 5,P$,K$
147: cll 'DERIVS'(1);π/180→W
148: 0→r7→r8;-r58→r9;cll 'CRSP'(1,7,10)
149: r12-r59→r12;cll 'ROT'(10,112,10)
```

```
150: fts (B[19])→P$
151: if P$="INCR";for I=4 to 6;0→rI;next I;gto +3
152: if P$="  IR";for I=4 to 6;r(I+57)→rI;next I;gto +2
153: cll 'ROT'(1,112,4);cll 'VSUB'(61,4,4)
154: for I=4 to 6;0B[I+16]-rI→rI;next I
155: for I=61 to 63;rI+r(I-57)→rI;next I
156: cll 'ROT'(4,105,1)
157: for I=1 to 3;rIr(I+101)→rI;next I
158: cll 'IROT'(1,105,1)
159: for I=1 to 3;rI/2→Y;abs(Y)→X
160: r(I+64)+X+Y→r(I+64);r(I+67)+X-Y→r(I+67);next I
161: chain "%PROP";169
162: "FIN":if r119=0;gto +3
163: fmt "SRM ",c4," STILL BURNING AT TRAJECTORY TERMINATION",2/;wrt 6,L$
164: sprt 5,"end"
165: fmt 2/,c60,2/,c80,2/;wrt 6,Z$,K$
166: for I=1 to 3;beep;wait 500;next I
167: if L$#"";set "#PIDI"
168: ent "START NEW RUN ?",I;if fla13=0;gto +2
169: set "#DBED"
170: stp
171: end
+29819
```

C.6 State Propagation Link (%PROP)

```
 0: "%PROP(1103/16OCT79)":gto "RUN"
 1: wtb 6,11;list #6
 2: get "#PIDI",0,1
 3: "RELIP":abs(int(p5))→H;p4/H→J;cos(J)→I;sin(J)→J
 4: cos(p3)→X;sin(p3)→Y;0→K
 5: plt Xp1p6-Yp2p7,Xp1p7+Yp2p6;if K=H;ret
 6: JX+IY→W;IX-JY→X;W→Y;K+1→K;gto -1
 7: "RK4":p1→p2;p2/2→p3;p2/6→p4;1→p5
 8: for I=25 to 74;rI→r(I+150)→r(I+200);next I
 9: cll 'DERIVS'(0);cll 'DERIVS'(1)
10: for I=25 to 74;r(I+150)+p3r(I+100)→rI;next I
11: for K=0 to 25 by 25;for J=25 to 29 by 4;J+K→H;0→W
12: for I=0 to 3;W+r(I+H)↑2→W;next I;rW→W
13: for I=0 to 3;r(I+H)/W→r(I+H);next I;next J;next K
14: if p5=4;ret
15: for I=225 to 274;rI+p4r(I-100)→rI;next I
16: if p5=1;p2/3→p4;gto +3
17: if p5=2;p2→p3;gto +2
18: p2/6→p3;for I=175 to 224;r(I+50)→rI;next I
19: p5+1→p5;gto -10
20: "RUN":rad;0→T[7]
21: fmt 1,2/,"GET =",f10.3," HHMM.SS",/
22: fmt 2,c2,":",c4,f9.1,c4,f8.1,c4,f8.2,c4,f8.2,c4,f8.2,c4,f8.2,c4
23: fmt 3,c2,":",c4,f9.1,c4,f8.5,c4,f8.2,c4,f8.2,c4,f8.3,c4,f8.2,c4
24: fmt 4,c2,":",c4,f9.2,c4,f8.2,c4,f8.2,c4,f8.3,c4,f8.3,c4,f8.3,c4
25: fmt 5,c2,":",c4,f10.2,c2,f9.2,c2,f9.2,c2,f10.2,c3,f10.2,c3,f10.2,c3
26: fmt 6,c2,":",c4,f10.0,c2,f9.0,c2,f9.0,c2,f10.2,c3,f10.2,c3,f10.2,c3
27: fmt 7,c2,":",c4,f10.0,c2,f9.2,c2,f9.2,c2,f10.2,c3,f10.3,c3,f10.3,c3
28: fmt 8,c2,c5,f9.0,c4,f8.0,c4,f8.0,c4,f8.0,c4,f8.0,c4,f8.0,c4
29: fmt
```

```
30: "PRINT":if T[7]mod2=0;wtb 6,11
31: wrt 6.1,'HM.S'(r74)
32: cll 'DERIVS'(0);180/π→Q
33: if r285#0;fmt "*** RCS CAN'T CONTROL SS ATT",/;wrt 6;fmt ;0→r285
34: for I=1 to 3;rI→r(I+174);next I
35: for I=7 to 9;rI→r(I+178);next I
36: "SSTRAN":r24→G;r33→R;r34→S;r35→W
37: GR/(2G-RRRWW-RSS)→A;RR(2/R-1/A-SS/G)→P
38: Sr(P/G)→Y;P/R-1→X;r(XX+YY)→E;'ATN1'(Y,X)→F
39: π/2→r4;-r4→r5;0→r6;cll '313Q'(4,7);cll 'QXQC'(25,7,11)
40: cll 'QXQ'(19,11,7);cll 'Q313'(7,4)
41: Q'ANG1'(r4)→N;Qr5→I;Q'ANG1'(r6-F)→J
42: 6076.115→T;QF→F
43: wrt 6.3,A#[1],C#[1],A/T,H#[1],E,H#[2],I,H#[3],N,H#[4],J,H#[5],F,H#[6]
44: 1.62405e-3→J;Jr2312→V;V/P→B
45: cll '0313'(11,4);cos(r5)→C;sin(r5)→D
46: cos(r6)→K;sin(r6)→L;KK-LL→M;2KL→N
47: A-2AAV(1-3DDLL)/3RRR→A;r(G/A)/A→W
48: A-B(1-3DD/2)/3→A;R-BDDM/6→R;S+BWDDN/3→S
49: RR(2/R-1/A-SS/G)→P;Sr(P/G)→Y;P/R-1→X
50: r(XX+YY)→E;'ATN1'(Y,X)→F
51: r6+B(1-7DD/6)N/2P→U;r5-VDDM/2RR→I;r4-BDN/4P→H
52: 0I→I;Q'ANG1'(H)→N;0'ANG1'(U-F)→J;QF→F
53: (A(1-E)-r23)/T→P;(A(1+E)-r23)/T→A
54: wrt 6.2,A#[1],C#[2],A,H#[7],P,H#[8],I,H#[3],N,H#[4],J,H#[5],F,H#[6]
55: wrt 6
56: "SSROT":cll '0X0'(19,29,7);cll '0231'(7,4)
57: Qr4→I;Qr5→J;0'ANG1'(r6)→K;0r36→X;0r37→Y;0r38→Z
58: wrt 6.4,A#[1],D#[1],I,I#[1],J,I#[2],K,I#[3],X,I#[4],Y,I#[5],Z,I#[6]
59: cll 'ROT'(1,87,1);cll 'VSUB'(36,1,7);cll '0231'(87,4)
```

%PROP Cont.

```
60: Qr4→I;Qr5→J;Q'ANG1'(r6)→K;Qr7→X;Qr8→Y;Qr9→Z
61: wrt 6.4,A$[1],D$[2],I,I$[1],J,I$[2],K,I$[3],X,I$[4],Y,I$[5],Z,I$[6]
62: wrt 6
63: "PLROT":cll 'DERIVS'(1);180/π→Q
64: for I=1 to 3;rI→r(I+177);next I
65: for I=7 to 9;rI→r(I+181);next I
66: cll 'QXQ'(19,54,7);cll 'Q231'(7,4)
67: Qr4→I;Qr5→J;Q'ANG1'(r6)→K;Qr61→X;Qr62→Y;Qr63→Z
68: wrt 6.4,A$[2],D$[1],I,I$[1],J,I$[2],K,I$[3],X,I$[4],Y,I$[5],Z,I$[6]
69: cll 'ROT'(1,112,1);cll 'VSUB'(61,1,7);cll 'Q231'(112,4)
70: Qr4→I;Qr5→J;Q'ANG1'(r6)→K;Qr7→X;Qr8→Y;Qr9→Z
71: wrt 6.4,A$[2],D$[3],I,I$[1],J,I$[2],K,I$[3],X,I$[4],Y,I$[5],Z,I$[6]
72: wrt 6
73: cll 'QCXQ'(50,25,0);if G[1]#2;gto +9
74: if G[12]=0;1→G[12];gto +8
75: "CPLV":G[2]G[3]→S;G[2]G[9]→T
76: cll 'Q213'(0,4);-r4r58→r7;r5r58→r8;r58-r33→r9
77: for L=0 to 1;if G[5-L]=0;gto +4
78: scl 0,-14S,0,-9S;G[2]G[8-L]→X→W;if L=1;W+SG[5]→W
79: ofs -W,-T;lim X,X-SG[5-L],T,T-SG[6]
80: ofs r(8-L),r9;plt 0,0;pen;cll 'RELIP'(S/20,S/20,0,2π,30,1,0)
81: pen;next L
82: 0→r7→r8;-r33→r9;cll 'CRSP'(175,7,10);r12-r34→r12
83: 0→r13→r14;-r58→r15;cll 'CRSP'(178,13,16);r18-r59→r18
84: cll 'ROT'(13,0,13);cll 'ROT'(16,0,16)
85: cll 'VSUB'(13,7,13);cll 'VSUB'(16,10,16);if r119=0;gto +12
86: "PLSM50":cll 'ROT'(188,0,188);cll 'VSUB'(188,185,188)
87: cll 'IROT'(116,54,1);cll 'IROT'(1,19,1)
88: sprt 5,r74-A[1],r164,r1,r2,r3
89: cll 'IPOT'(13,25,1);cll 'IROT'(1,19,1)
```

```
90: cll 'IROT'(16,25,4)⁞cll 'IROT'(4,19,4)
91: sprt 5,r1,r2,r3,r4,r5,r6
92: wrt 6.6,A$[2],E$[5],r1,J$[1],r2,J$[2],r3,J$[3],r4,A$[4],r5,A$[5],r6,A$[6]
93: cll 'IROT'(188,25,1)⁞cll 'IROT'(1,19,1)
94: sprt 5,r1,r2,r3
95: if r74<A[2]⁞sto +2
96: sprt 5,"end"⁞0→r119
97: "PLTRAN":for I=1 to 3⁞r(I+12)→r[I]⁞next I
98: cll 'CRSP'(175,1,4)⁞cll 'VSUB'(16,4,4)
99: wrt 6.6,A$[2],E$[2],r1,J$[1],r2,J$[2],r3,J$[3],r4,A$[4],r5,A$[5],r6,A$[6]
100: cll 'IROT'(36,87,7)⁞cll 'VSUB'(175,7,7)
101: cll 'CRSP'(7,1,7)⁞cll 'VADD'(4,7,4)
102: cll 'ROT'(1,87,1)⁞cll 'ROT'(4,87,4)
103: wrt 6.6,A$[2],E$[1],r1,J$[1],r2,J$[2],r3,J$[3],r4,A$[4],r5,A$[5],r6,A$[6]
104: wrt 6
105: cll '0CW0'(54,29,181)⁞if G[1]#1⁞sto +15
106: if G[12]=0⁞1→G[12]⁞sto +14
107: "RSBY":G[2]G[3]→S⁞G[2]G[9]→T⁞r120→A⁞-r121→r10⁞0→r11→r12
108: cll 'ROT'(109,181,7)⁞cll 'ROT'(10,181,10)
109: for L=0 to 1⁞if G[5-L]=0⁞sto +10
110: scl 0,-148,0,-98⁞G[2]G[8-L]→X→W⁞if L=1⁞W+SG[5]→W
111: ofs -W,-T⁞lim X,X-SG[5-L],T,T-SG[6]
112: ofs r(2-L),r3⁞plt 0,0⁞pen⁞cll 'RELIP'(S/20,S/20,π/2,2π,3,1,0)⁞pen
113: r(11-L)→U⁞r12→V⁞-r(8-L)→X⁞-r9→Y
114: if (1-2L)r(10+L)>0⁞X+U→X⁞Y+V→Y⁞-U→U⁞-V→V
115: r(UU+VV)→E⁞Ar max(0,1-(E/r121)↑2)→B⁞if E=0⁞1→C⁞0→D⁞sto +2
116: V/E→C⁞-U/E→D
117: ofs X,Y⁞cll 'RELIP'(A,B,0,2π,30,C,D)⁞if E=0⁞sto +2
118: ofs U,V⁞cll 'RELIP'(A,B,0,π,15,C,D)⁞plt -U,-V
119: pen⁞next L
```

```
120: "TPSENS":cll 'VSUB'(282,109,7)!cll 'ROT'(7,181,7)
121: cll 'ROT'(61,181,13)!cll 'VSUB'(13,36,13)!cll 'CRSP'(13,7,10)
122: cll 'VADD'(4,10,4)!cll 'VADD'(1,7,1)
123: cll 'VADD'(1,84,1)!cll 'VSUB'(1,275,1)
124: cll 'ROT'(1,278,1)!cll 'ROT'(4,278,4)
125: r1→X!r2→Y!r3→Z!r4→A!r5→B!r6→C
126: XX+YY→Q!r(Q+ZZ)→R!rQ→Q
127: 'ATN1'(Y,X)→S!'ATN2'(Z,Q)→T
128: (AX+BY+CZ)/R→U
129: if Q/R<1e-6!0→V!r(AA+BB+CC-UU)/R→W!go to +2
130: (BX-AY)/QQ→V!(CR-UZ)/QR→W
131: 180/π→Q!QS→S!QT→T!QV→V!QW→W
132: wrt 6,7,A$[3],G$[1],R,J$[4],S,J$[5],T,J$[6],U,A$[7],V,A$[8],W,A$[9]
133: wrt 6
134: "RIPCON":for I=1 to 6!r(I+64)→r1!next I
135: wrt 6,8,A$[2],G$[3],r1,O$[1],r2,O$[2],r3,O$[3],r4,O$[4],r5,O$[5],r6,O$[6]
136: wrt 6
137: for I=0 to 7!r(I+40)→r1!next I
138: r3+r4→A!A+r2→B
139: r6+r7→C!C+r5→D
140: r2+r5→I!r3+r6→J!r4+r7→K
141: A+C→M!B+D→Y!r0+r1→Z
142: wrt 6,8,A$[1],G$[2],r2,O$[1],r3,O$[2],r4,O$[3],A,O$[4],B,O$[5],r0,O$[6]
143: wrt 6,8,A$[1],G$[2],r5,P$[1],r6,P$[2],r7,P$[3],C,P$[4],D,P$[5],r1,K$[6]
144: wrt 6,8,A$[1],G$[2],I,S$[1],J,S$[2],K,S$[3],M,S$[4],Y,S$[5],Z,S$[6]
145: wrt 6
146: if abs(r74-T[1])>.0001!chain "%SHIT",169
147: "ADVANS":T[2]+1→r(r74/T[2])→D!if abs(D)<.0001!T[2]→D
148: D+r74→E!if T[1]-E<.0001!T[1]→E!E-r74→D
149: int(.99999D T[3])+1→T[5]!D.T[5]→T[4]
```

```
150: if T[5]=1;if abs(T[1]-T[4]-r74)<.0001;T[1]-r74→T[4]
151: fxd 2;dsp "%PROP  T = ";'HM.S'(r74);cll 'RK4'(T[4])
152: T[5]-1→T[5];if T[5]=0;T[7]+1→T[7];dsp "";sto "PRINT"
153: sto -3
154: end
*2907
```

APPENDIX D:

MEMORY ALLOCATION

MISCELLANEOUS DATA

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
0			volatile
1			volatile; ω_1 contains $\bar{\omega}_G^{IG}$ or $\bar{\omega}_g^{Ig}$ upon return from DERIVS
2			volatile; ω_2 contains $\bar{\omega}_G^{IG}$ or $\bar{\omega}_g^{Ig}$ upon return from DERIVS
3			volatile; ω_3 contains $\bar{\omega}_G^{IG}$ or $\bar{\omega}_g^{Ig}$ upon return from DERIVS
4			volatile
5			volatile
6			volatile
7			volatile; A_1 Contains $\bar{A}_G$ or $\bar{a}_g$ upon return from DERIVS
8			volatile; A_2 Contains $\bar{A}_G$ or $\bar{a}_g$ upon return from DERIVS
9			volatile; A_3 Contains $\bar{A}_G$ or $\bar{a}_g$ upon return from DERIVS
10			volatile
11			volatile
12			temporary storage for
13			working arrays
14			
15			
16			
17			
18			
19	q_0		$\bar{\bar{q}}_{JI}$ (orientation versor, MLD ECI with respect to
20	q_1		M50 ECI).
21	q_2		
22	q_3		
23	R_e	ft	20925722 = equatorial radius
24	μ_e	$ft^3\ sec^{-2}$	14076469×10^9 = gravitational parameter

SS STATE VARIABLES

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
25	q_0		$\bar{\bar{q}}_{IG}$ — SS local-vertical system orientation versor, with respect to MLD ECI system.
26	q_1		
27	q_2		
28	q_3		
29	q_0		$\bar{\bar{q}}_{IB}$ — SS body orientation versor, with respect to MLD ECI system.
30	q_1		
31	q_2		
32	q_3		
33	R	ft	SS CG distance from center of earth.
34	$\dot{R}$	ft sec^{-1}	dR/dt
35	Ω_K	sec^{-1}	SS Keplerian angular velocity magnitude.
36	ω_1	sec^{-1}	$\bar{\omega}_B^{IB}$ — SS body angular velocity with respect to inertial space, resolved onto SS body axes.
37	ω_2	sec^{-1}	
38	ω_3	sec^{-1}	
39	W	lb	SS gross weight (constant)
40	W_{POML}	lb	left — OMS propellant consumption
41	W_{POMR}	lb	right
42	W_{PRFX}	lb	forward — translation control — RCS propellant consumption
43	W_{PRLX}	lb	aft left — translation control — RCS propellant consumption
44	W_{PRRX}	lb	aft right — translation control — RCS propellant consumption
45	W_{PRFR}	lb	forward — rotational control — RCS propellant consumption
46	W_{PRLR}	lb	aft left — rotational control — RCS propellant consumption
47	W_{PRRR}	lb	aft right — rotational control — RCS propellant consumption
48			(RESERVED)
49			(RESERVED)

PL STATE VARIABLES

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
50	q_0		$\bar{\bar{q}}_{Ig}$: PL local vertical system orientation versor, with respect to MLD ECI system
51	q_1		
52	q_2		
53	q_3		
54		q_0	$\bar{\bar{q}}_{Ib}$: PL body orientation versor, with respect to MLD ECI system.
55		q_1	
56		q_2	
57		q_3	
58	r	ft	PL CG distance from center of earth.
59	$\dot{r}$	ft sec^{-1}	dr/dt
60	ω_K	sec^{-1}	PL Keplerian angular veolcity magnitude.
61	ω_1	sec^{-1}	$\bar{\omega}_b^{Ib}$: PL body angular velocity with respect to inertial space, resolved onto PL body axes.
62	ω_2	sec^{-1}	
63	ω_3	sec^{-1}	
64	w	lb	PL gross weight
65	$\int \ell_{1+}$	lb ft sec	Integrals of control torque applied in the positive direction about each body axis.
67	$\int \ell_{2+}$	lb ft sec	
68	$\int \ell_{3+}$	lb ft sec	
69	$\int \ell_{1-}$	lb ft sec	Integrals of the control torgue magnitude applied in the negative direction about each body axis.
70	$\int \ell_{2-}$	lb ft sec	
71	$\int \ell_{3-}$	lb ft sec	
71			(RESERVED)
72			(RESERVED)
73			(RESERVED)
74	t	sec	State time

SS CONSTANTS

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
75	$K\alpha$		SS aero factor.
76	K_{AM}		SS att maint flag (0:D, 1:IRH, 2:LVRH)
77	I'_{xx}	slug ft^2	SS principal moments of inertia
78	I'_{yy}	slug ft^2	
79	I'_{zz}	slug ft^2	
80	q_0		$\bar{\bar{q}}_{BP}$ (SS principal axes of inertia orientation versor, with respect to SS body axes.)
81	q_1		
82	q_2		
83	q_3		
84	P_1	ft	$\bar{P}_B^{NA}$ (Position of SS actual CG with respect to nominal CG, in SS body axes system.)
85	P_2	ft	
86	P_3	ft	
87	q_0		$\bar{\bar{q}}_{GB} = q_{IG} \circ \bar{\bar{q}}_{IB}$ (not constant; value computed in 'DERIVS')
88	q_1		
89	q_2		
90	q_3		
91	U_1		$\bar{U}_B^{\tau}$ (OMS thrust unit vector, in SS body axes system.)
92	U_2		
93	U_3		
94	$K_{\tau OMS}$		OMS thrust flag (0:none, 1:L, 2:R, 3:L+R)
95	$K_{\tau RCS}$		RCS thrust command flag (0,1,2,$\cdots$,12)
96	D_1		SS attitude deadbands
97	D_2		
98	D_3		
99	C	sec	SS DAP cycle time

PL CONSTANTS

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
100	k_α		PL aero factor
101	k_{AM}		PL att maint flag (0:D, 1:IRH, 2:LVRH)
102	i'_{xx}	slug ft^2	PL principal moments of inertia.
103	i'_{yy}	slug ft^2	
104	i'_{zz}	slug ft^2	
105	q_0		$\bar{\bar{q}}_{bp}$ (PL principal axes of inertia orientation versor, with respect to PL body axes.)
106	q_1		
107	q_2		
108	q_3		
109	P_1	ft	$\bar{P}_b^{na}$ (Position of PL actual CG with respect to "nominal CG" (front face of cylinder), in PL body axes system.)
110	P_2	ft	
111	P_3	ft	
112	q_0		$\bar{\bar{q}}_{gb} = q_{Ig} \circ \bar{\bar{q}}_{Ib}$ (not constant; value computed in 'DERIVS')
113	q_1		
114	q_2		
115	q_3		
116	u_1		$\bar{u}_b^\tau$ (PL thrust unit vector, in PL body axes system.)
117	u_2		
118	u_3		
119	k_τ		PL thrust flag (o:no, 1:yes)
120	d/2	ft	PL radius
121	ℓ	ft	PL length
122	A_E	ft^2	$\pi\, d^2/4$
123	A_S	ft^2	$d\ell$
124	K_ρ		Atmospheric density factor (applies to SS & PL)

SS STATE VARIABLE DERIVATIVES

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
125	$\dot{q}_0$	sec^{-1}	
126	$\dot{q}_1$	sec^{-1}	$\dot{\bar{q}}_{IG}$
127	$\dot{q}_2$	sec^{-1}	
128	$\dot{q}_3$	sec^{-1}	
129	$\dot{q}_0$	sec^{-1}	
130	$\dot{q}_1$	sec^{-1}	$\dot{\bar{q}}_{IB}$
131	$\dot{q}_2$	sec^{-1}	
132	$\dot{q}_3$	sec^{-1}	
133	$\dot{R}$	$ft\ sec^{-1}$	
134	$\ddot{R}$	$ft\ sec^{-2}$	
135	$\dot{\Omega}_K$	sec^{-2}	
136	$\dot{\omega}_1$	sec^{-2}	$\dot{\bar{\omega}}_B^{IB}$
137	$\dot{\omega}_2$	sec^{-2}	
138	$\dot{\omega}_3$	sec^{-2}	
139	$\dot{W}$	$lb\ sec^{-1}$	$\equiv 0.0$
140	$\dot{W}_{POML}$	$lb\ sec^{-1}$	
141	$\dot{W}_{POMR}$	$lb\ sec^{-1}$	
142	$\dot{W}_{PRFX}$	$lb\ sec^{-1}$	
143	$\dot{W}_{PRLX}$	$lb\ sec^{-1}$	
144	$\dot{W}_{PRRX}$	$lb\ sec^{-1}$	
145	$\dot{W}_{PRFR}$	$lb\ sec^{-1}$	
146	$\dot{W}_{PRLR}$	$lb\ sec^{-1}$	
147	$\dot{W}_{PRRR}$	$lb\ sec^{-1}$	
148			(RESERVED)
149			(RESERVED)

(q_0, q_1, q_2, q_3) $\bar{\bar{q}}_{GP}$ (A_1, A_2, A_3) $\bar{A}_G$

(U_1, U_2, U_3) $\bar{U}_G^{IZ}$

(intermediate data used in 'DERIVS' computations)

PL STATE VARIABLE DERIVATIVES

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
150	$\dot{q}_0$	sec^{-1}	$\dot{\bar{\bar{q}}}_{Ig}$
151	$\dot{q}_1$	sec^{-1}	
152	$\dot{q}_2$	sec^{-1}	
153	$\dot{q}_3$	sec^{-1}	
154	$\dot{q}_0$	sec^{-1}	$\dot{\bar{\bar{q}}}_{Ib}$
155	$\dot{q}_1$	sec^{-1}	
156	$\dot{q}_2$	sec^{-1}	
157	$\dot{q}_3$	sec^{-1}	
158	$\dot{r}$	$ft\ sec^{-1}$	
159	$\ddot{r}$	$ft\ sec^{-2}$	
160	$\dot{\omega}_K$	sec^{-2}	
161	$\dot{\omega}_1$	sec^{-2}	$\dot{\bar{\omega}}_{Ib}$
162	$\dot{\omega}_2$	sec^{-2}	
163	$\dot{\omega}_3$	sec^{-2}	
164	$\dot{w}$	$lb\ sec^{-1}$	
165	ℓ_{1+}	lb ft	
166	ℓ_{2+}	lb ft	
167	ℓ_{3+}	lb ft	
168	ℓ_{1-}	lb ft	
169	ℓ_{2-}	lb ft	
170	ℓ_{3-}	lb ft	
171			(RESERVED)
172			(RESERVED)
173			(RESERVED)
174	$\dot{t}$		$\equiv 1.0$

(q_0, q_1, q_2, q_3) $\bar{\bar{q}}_{gp}$ (a_1, a_2, a_3) $\bar{a}_g$

(U_1, U_2, U_3) $\bar{U}_g^{IZ}$

(intermediate data used in 'DERIVS' computations)

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
175			ω_1
176			ω_2 $\bar{\omega}_G^{IG}$
177			ω_3
178			ω_1
179			ω_2 $\bar{\omega}_g^{Ig}$
180			ω_3
181			q_0
182			q_1 $\bar{\bar{q}}_{bB}$
183			q_2
184			q_3
185			A_1
186			A_2 $\bar{A}_G$
187			A_3
188			a_1
189			a_2 $\bar{a}_g$
190			a_3
191			
192			
193			
194			
195			Used by RKUT to store data
196			which must be preserved
197			between calls to 'DERIVS'.
198			
199			

(Used for temporary storage by %PROP between calls to 'RKUT')

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
200			
201			
202			
203			
204			
205			
206			
207			
208			
209			
210			
211			
212			Used by 'RKUT' to store data
213			which must be preserved between
214			calls to 'DERIVS'.
215			
216			
217			
218			
219			
220			
221			
222			
223			
224			

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
225			
226			
227			
228			
229			
230			
231			
232			
233			
234			
235			
236			
237			Used by 'RKUT' to store data
238			which must be preserved between
239			calls to 'DERIVS'.
240			
241			
242			
243			
244			
245			
246			
247			
248			
249			

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
250			
251			
252			
253			
254			
255			
256			
257			
258			
259			
260			
261			Used by 'RKUT' to store data
262			which must be preserved bewteen
263			calls to 'DERIVS'.
264			
265			
266			
267			
268			
269			
270			
271			
272			
273			
274			

MISCELLANEOUS DATA

REG.	SYMBOL	UNITS	DESCRIPTION/NOTES
275	P_1	ft	$\bar{P}_B^{NS}$ — SS sensor position with respect to nominal CG, in SS body axes system.
276	P_2	ft	
277	P_3	ft	
278	q_0		$\bar{q}_{BS}$ — SS sensor orientation versor, with respect to SS body axes.
279	q_1		
280	q_2		
281	q_3		
282	P_1	ft	$\bar{P}_b^{nt}$ — PL target point position with respect to "nominal CG", front fact of PL cylinder), in PL body axes system.
283	P_2	ft	
284	P_3	ft	
285	K_{ACF}		Shuttle attitude control failure flag

APPENDIX E

USER-SUPPLIED INPUT DATA BASE

E.1 SHUTTLE DATA FILE

SHUTTLE DATA FILE $

ITEM	DESCRIPTION	VALUE	
1	WEIGHT	200017.0000	LB
2	IXX	887302.0000	SLUG-FT↑2
3	IYY	6386877.0000	SLUG-FT↑2
4	IZZ	6694367.0000	SLUG-FT↑2
5	IYZ	-971.0000	SLUG-FT↑2
6	IZX	247376.0000	SLUG-FT↑2
7	IXY	5622.0000	SLUG-FT↑2
8	CG STA	1095.3000	IN
9	CG BL	0.3000	IN
10	CG WL	377.4000	IN
11	SENSOR STA	1095.3000	IN
12	SENSOR BL	0.3000	IN
13	SENSOR WL	377.4000	IN
14	SENSOR AXES PITCH	0.0000	DEG
15	SENSOR AXES YAW	0.0000	DEG
16	SENSOR AXES ROLL	0.0000	DEG
17	DAP CYCLE	80.0000	MILLISEC

E.2 PAYLOAD DATA FILE

PAYLOAD DATA FILE $

ITEM	DESCRIPTION	VALUE	
1	WEIGHT	37253.0000	LB
2	IXX	7439.0000	SLUG-FT↑2
3	IYY	45468.0000	SLUG-FT↑2
4	IZZ	45433.0000	SLUG-FT↑2
5	IYZ	0.0000	SLUG-FT↑2
6	IZX	0.0000	SLUG-FT↑2
7	IXY	0.0000	SLUG-FT↑2
8	CG STA	300.6300	IN
9	CG BL	0.0000	IN
10	CG WL	0.2000	IN
11	TARGETPOINT STA	300.6300	IN
12	TARGETPOINT BL	0.0000	IN
13	TARGETPOINT WL	0.2000	IN
14	DIAMETER	112.0000	IN
15	LENGTH	428.0000	IN

E.3 GRAPHICS DATA FILE

GRAPHICS DATA FILE $

ITEM	DESCRIPTION	VALUE	
1	PLOT TYPE (NONE,RSBY,CPLV).....	RSBY	
2	PLOT UNIT, PU (FT,KFT,NMI,M,KM).....	FT	
3	PLOT SCALE.....	120	PU/IN
4	X PANEL WIDTH.....	14.0000	IN
5	Y PANEL WIDTH.....	0.0000	IN
6	PANEL HEIGHT.....	9.0000	IN
7	X MAX.....	1500.0000	PU
8	Y MAX.....	0.0000	PU
9	Z MAX.....	80.0000	PU
10	TIC INTERVAL.....	20	PU
11	LABEL INTERVAL.....	5	TICS

E.4 TIME/ATM DATA FILE

TIME/ATM DATA FILE $

ITEM	DESCRIPTION	VALUE	
1	LAUNCH YEAR.....	1981	
2	LAUNCH MONTH (JAN,FEB,MAR,.....,NOV,DEC)	FEB	
3	LAUNCH DAY.....	27	
4	LAUNCH GMT.....	1935.0000	HHMM.SS
5	INITIAL GET.....	1005.0000	HHMM.SS
6	ATMOSPHERIC DENSITY FACTOR.....	1.0000	

E.5 SHUTTLE ISTATE DATA FILE

```
SHUTTLE ISTATE      $

ITEM  DESCRIPTION                                          VALUE

   1  STATE TYPE (OM50,IMLD).....................          OM50
   2  SMA........................................    3598.1726 NMI
   3  ECC........................................       0.0012
   4  INC........................................      28.3014 DEG
   5  RAN........................................     -80.5443 DEG
   6  ARG........................................     -29.4152 DEG
   7  TRA........................................       3.3863 DEG
   8  ATT REF (SLV)..............................          SLV
   9  PITCH......................................    -100.9000 DEG
  10  YAW........................................     -33.9000 DEG
  11  ROLL.......................................    -160.0000 DEG
  12  RATE REF (M50,SLV).........................          M50
  13  XB RATE....................................       0.0000 DEG/SEC
  14  YB RATE....................................       0.0000 DEG/SEC
  15  ZB RATE....................................       0.0000 DEG/SEC
```

E.6 PAYLOAD ISTATE DATA FILE

```
PAYLOAD ISTATE      $

ITEM  DESCRIPTION                                        VALUE

   1  STATE TYPE (RSLV,RSBY).....................        RSBY
   2  X..........................................    -6.3775 FT
   3  Y..........................................    -0.0250 FT
   4  Z..........................................    -8.1292 FT
   5  XDOT.......................................     0.2120 FT/SEC
   6  YDOT.......................................     0.0000 FT/SEC
   7  ZDOT.......................................    -0.3392 FT/SEC
   8  ATT REF (PLV,SBY)..........................         SBY
   9  PITCH......................................    58.0000 DEG
  10  YAW........................................     0.0000 DEG
  11  ROLL.......................................     0.0000 DEG
  12  RATE REF (M50,PLV,SBY).....................         M50
  13  XB RATE.....  .............................     0.0000 DEG/SEC
  14  YB RATE....................................     0.0000 DEG/SEC
  15  ZB RATE....................................     0.0000 DEG/SEC
```

APPENDIX F

USER-SUPPLIED FLIGHT PROFILE DEFINITION

F.1 FLIGHT PROFILE SEGMENT

FLT PROFILE SEGMENT 20

ITEM	DESCRIPTION	VALUE	
1	FLT SEGMENT TYPE (PROP).....	PROP	
2	FLT SEGMENT LENGTH.....	2.4000	HHMM.SS
3	PRINT/PLOT INTERVAL.....	0.0500	HHMM.SS
4	MAX INTEG STEPSIZE.....	10.0000	HHMM.SS
5	SS AERO FACTOR.....	1.0000	
6	SS RATE OPT (INCR,IR,LVR).....	IR	
7	SS XB RATE OR INCR.....	0.0000	DEG/SEC
8	SS YB RATE OR INCR.....	0.0000	DEG/SEC
9	SS ZB RATE OR INCR.....	0.0000	DEG/SEC
10	SS ATT MAINT OPT (D,IRH,LVRH).....	IRH	
11	SS XB DEADBAND.....	2.0000	DEG
12	SS YB DEADBAND.....	2.0000	DEG
13	SS ZB DEADBAND.....	2.0000	DEG
14	SS RCS OPT (V,P,PZI).....	P	
15	SS XC COMPENSATION (NONE,ROT,FULL).....	ROT	
16	SS OMS CMD (NONE,L,R,L+R).....		
17	SS RCS CMD (NONE,+X,-X,+Y,...,+YAW,-YAW)		
18	PL AERO FACTOR.....	1.0000	
19	PL RATE OPT (INCR,IR,LVR).....	IR	
20	PL XB RATE OR INCR.....	0.0000	DEG/SEC
21	PL YB RATE OR INCR.....	0.0000	DEG/SEC
22	PL ZB RATE OR INCR.....	0.0000	DEG/SEC
23	PL ATT MAINT OPT (D,IRH,LVRH).....	IRH	
24	PL THR TABLE (NONE,IL,SA,SD).....	IL	

APPENDIX G

DIGITAL OUTPUT DATA

G.1 DIGITAL OUTPUT DATA

GET = 1056.350 HHMM.SS

SS:OM50	3617.3	SMA	0.00407	ECC	28.31	INC	279.22	RAN	26.00	ARG	153.42	TRA
SS:IMLD	187.5	HA	149.8	HP	28.46	INC	279.57	RAN	19.90	ARG	159.58	TRA
SS: M50	-21.13	PCH	-57.48	YAW	98.13	ROL	-0.000	RXB	0.000	RYB	0.000	RZB
SS: SLV	-114.70	PCH	0.00	YAW	180.00	ROL	0.000	RXB	-0.065	RYB	0.000	RZB
PL: M50	161.78	PCH	-12.72	YAW	87.86	ROL	0.000	RXB	0.000	RYB	0.000	RZB
PL: PLV	-3.51	PCH	-10.14	YAW	0.03	ROL	-0.011	RXB	0.068	RYB	0.013	RZB
PL:SM50	-325994	X	-277410	Y	-196164	Z	-868.89	DX	-716.98	DY	-347.56	DZ
PL:RSLV	413200	X	-992	Y	225774	Z	1273.34	DX	-155.27	DY	106.44	DZ
PL:RSBY	32454	X	1011	Y	469739	Z	101.56	DX	155.31	DY	1164.21	DZ
TP:SENS	470860	R	1.78	S	86.05	T	1168.77	DR	0.268	DS	-0.003	DT
PL RIMP	173	+X	205	+Y	272	+Z	173	-X	331	-Y	261	-Z
SS PCON	62	RFX	0	RLX	0	RRX	0	RAX	62	RX	671	OML
SS PCON	46	RFR	30	RLR	31	RRR	61	RAR	107	RR	671	OMR
SS PCON	108	RFT	30	RLT	31	RRT	61	RAT	169	RT	1341	OMT

GET = 1056.400 HHMM.SS

SS:OM50	3617.3	SMA	0.00407	ECC	28.31	INC	279.22	RAN	25.90	ARG	153.85	TRA
SS:IMLD	187.5	HA	149.8	HP	28.46	INC	279.57	RAN	19.90	ARG	159.91	TRA
SS: M50	-21.13	PCH	-57.48	YAW	98.13	ROL	-0.000	RXB	0.000	RYB	0.000	RZB
SS: SLV	-114.37	PCH	0.00	YAW	180.00	ROL	0.000	RXB	-0.065	RYB	0.000	RZB
PL: M50	161.78	PCH	-12.72	YAW	87.86	ROL	0.000	RXB	0.000	RYB	0.000	RZB
PL: PLV	-3.16	PCH	-10.07	YAW	0.03	ROL	-0.011	RXB	0.068	RYB	0.014	RZB
PL:SM50	-330744	X	-281099	Y	-198034	Z	-1033.87	DX	-759.41	DY	-401.51	DZ
PL:RSLV	420006	X	-1849	Y	226314	Z	1452.11	DX	-188.22	DY	109.08	DZ
PL:RSBY	32821	X	1868	Y	475968	Z	44.17	DX	188.27	DY	1330.19	DZ
TP:SENS	477102	R	3.26	S	86.05	T	1330.80	DR	0.323	DS	0.004	DT
PL RIMP	173	+X	205	+Y	272	+Z	173	-X	331	-Y	261	-Z
SS PCON	62	RFX	0	RLX	0	RRX	0	RAX	62	RX	671	OML
SS PCON	46	RFR	30	RLR	31	RRR	61	RAR	107	RR	671	OMR
SS PCON	108	RFT	30	RLT	31	RRT	61	RAT	169	RT	1341	OMT

APPENDIX H

GRAPHICAL OUTPUT DATA

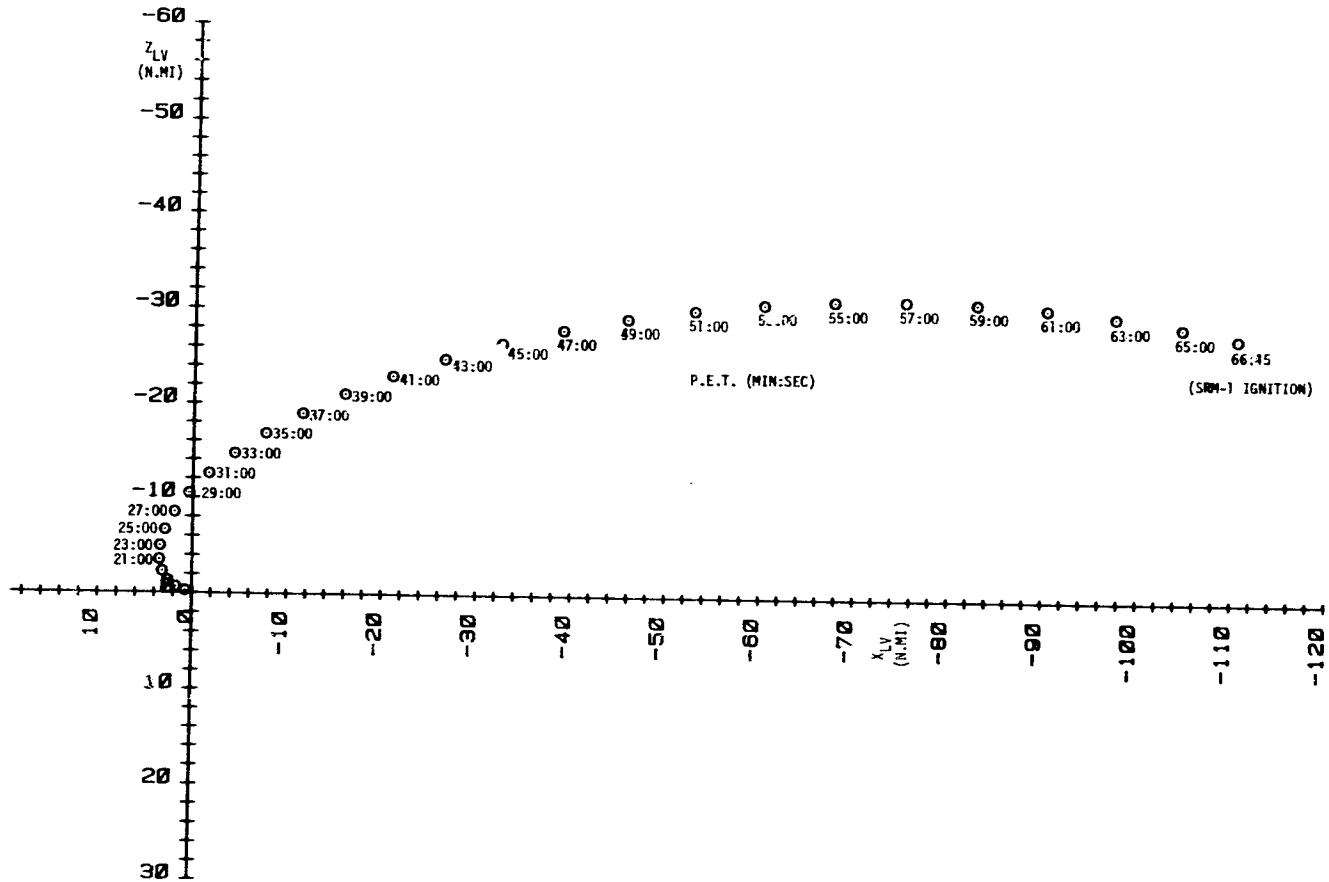


Figure H1. Illustration of Graphical Plot for CPLV Plot Option

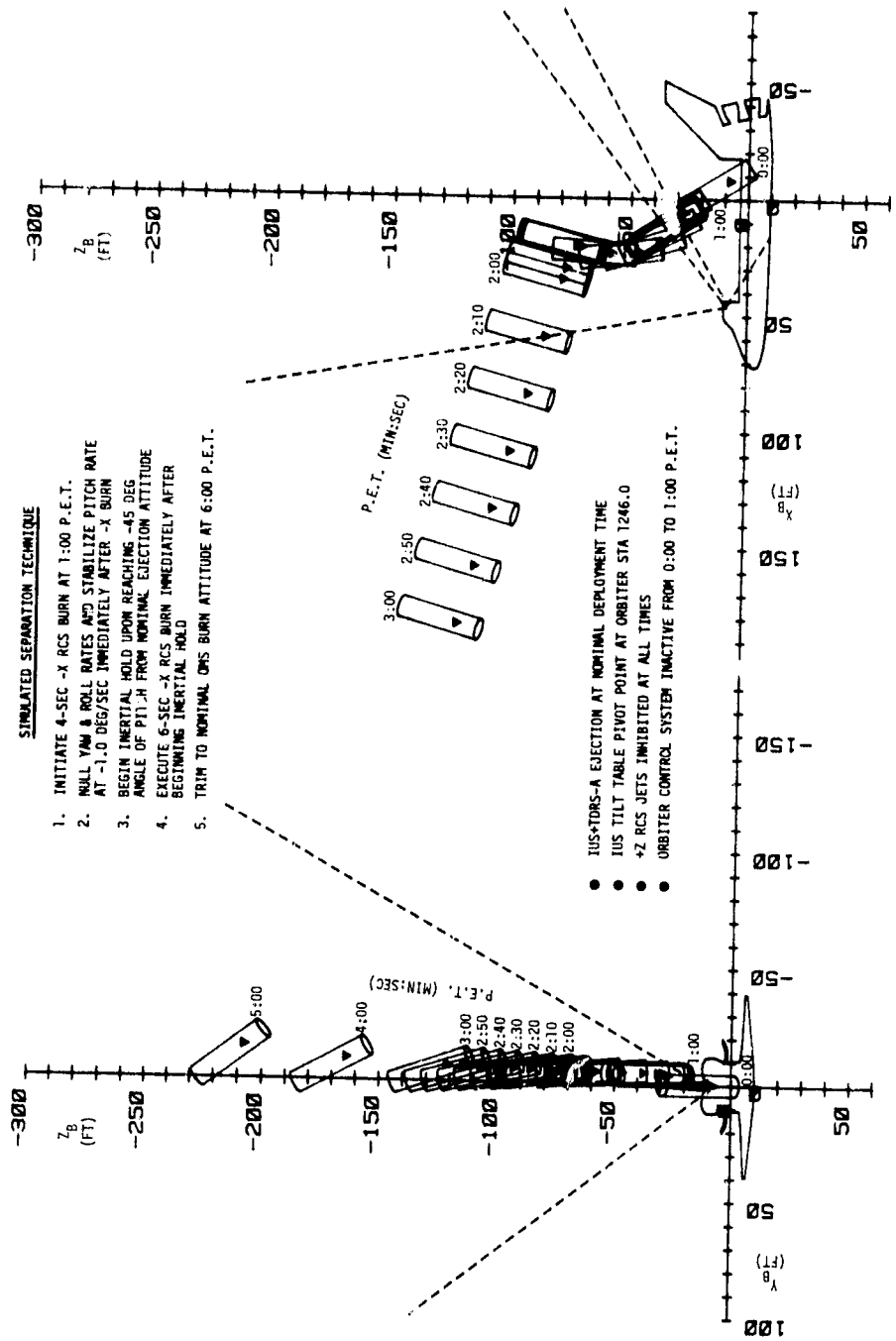


Figure H2. Illustration of Graphical Plot for RSBY Plot Option