

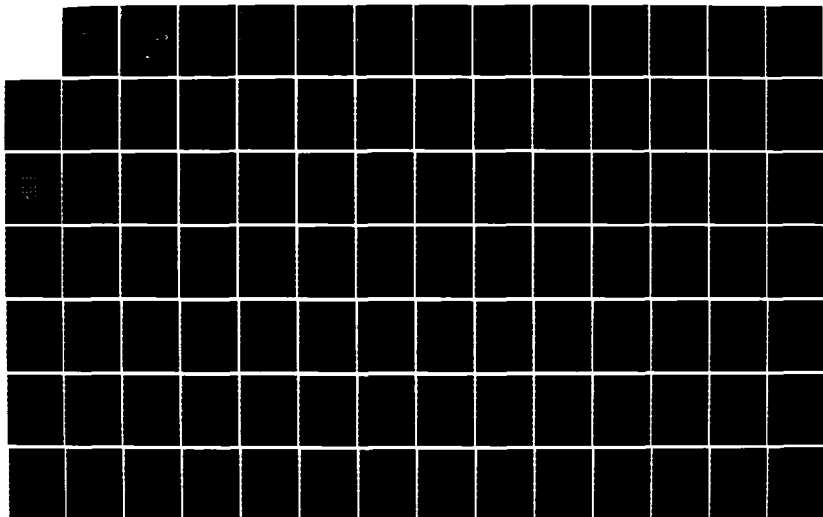
AD-A162 461

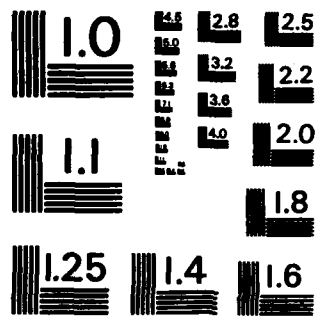
STS (SPACE TRANSPORTATION SYSTEM) TASK SIMULATOR(U) AIR 1/2
FORCE ACADEMY CO 5 ALFANO 15 AUG 85 USAFA-TR-85-12

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

USAFA TR 85-12

STS TASK SIMULATOR

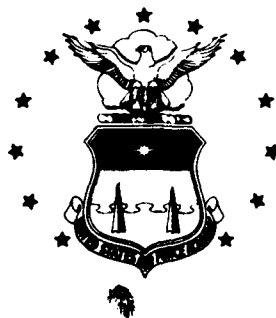
AD-A162 461

SALVATORE ALFANO, CAPTAIN, USAF

DTIC
EXTRACTED
DEC 18 1985
D

AUGUST 1985
FINAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED



DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO SPRINGS, CO 80840

DTIC FILE COPY

85 12 17 112

This research report is presented as a competent treatment of the subject, worthy of publication. The United States Air Force Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the author.

This report has been cleared for open publication and/or public release by the appropriate Office of Information in accordance with AFM 190-1, AFR 12-30, and AFR 80-3. There is no objection to unlimited distribution of this report to the public at large, or by DTIC to the National Technical Information Service.

This research report has been reviewed and is approved for publication.

Thomas E. McCann

THOMAS E. McCANN, Lt Col, USAF
Director of Research and
Computer Based Education

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A162461

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1d. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release. Unlimited distribution.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) USAFA TR 85-12			7a. NAME OF MONITORING ORGANIZATION		
6a. NAME OF PERFORMING ORGANIZATION Department of Astronautics		6b. OFFICE SYMBOL (If applicable) DFAS	7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION			9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8b. OFFICE SYMBOL (If applicable)			10. SOURCE OF FUNDING NOS.		
8c. ADDRESS (City, State and ZIP Code) U.S. Air Force Academy Colorado Springs, CO 80840-5981			11. TITLE (Include Security Classification) STS Task Simulator		
12. PERSONAL AUTHOR(S) SALVATORE ALFANO, Capt, USAF			13a. TYPE OF REPORT Final report		
13b. TIME COVERED FROM Aug 84 to Aug 85			14. DATE OF REPORT (Yr., Mo., Day) 15 August 1985		
15. PAGE COUNT 133			16. SUPPLEMENTARY NOTATION		
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB. GR.	Simulation, STS, Shuttle, Orbiter, Space Transportation System, Task, Rendezvous, Proximity Ops.		
22	01				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
This paper describes simplified mathematical models of the Space Transportation System (STS), Reaction Control Systems (RCS), and Digital Autopilot (DAP), used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. Included in the modeling are propellant expenditures for translational maneuvers, rotational maneuvers, and attitude maintenance and on-orbit trajectory deviations induced by RCS cross coupling. This simulator serves as a learning aid for cadets studying orbital dynamics and STS mission planning and as a research platform for the Department of Astronautics.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL SALVATORE ALFANO, Capt, USAF			22b. TELEPHONE NUMBER (Include Area Code) (303) 472-4110		22c. OFFICE SYMBOL DFAS

STS TASK SIMULATOR

by

Capt Salvatore Alfano

**United States Air Force Academy
Department of Astronautics**

August 15, 1985

ABSTRACT

This paper describes simplified mathematical models of the Space Transportation System (STS) Reaction Control System (RCS) and Digital Autopilot (DAP) used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. Included in the modeling are propellant expenditures for translational maneuvers, rotational maneuvers, and attitude maintenance and on-orbit trajectory deviations induced by RCS cross coupling. This simulator serves as a learning aid for cadets studying orbital dynamics and STS mission planning and as a research platform for the Department of Astronautics.

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Available and/or Special
A-1	



TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	ii
LIST OF TABLES	iv
LIST OF FIGURES	v
1. INTRODUCTION	1
2. RCS ACCELERATION MODELING	2
3. RCS THRUSTER FIRING SELECTION	8
4. DETERMINING POSITION AND ATTITUDE	14
5. GENERATION OF VISUAL DISPLAY	15
REFERENCES	18
APPENDIX A - DAP PANEL FUNCTIONS	A-1
APPENDIX B - TERMINAL RENDEZVOUS/DOCKING	B-1
APPENDIX C - ATTITUDE DETERMINATION USING QUATERNIONS	C-1
APPENDIX D - THRUSTERS PROGRAM LISTING	D-1
APPENDIX E - MAIN PROGRAM LISTING	E-1

LIST OF TABLES

<u>Table</u>		<u>Page</u>
1	RCS Thruster Data	3
2	Typical Orbiter Mass Properties	4
3	Response Matrix for RCS Thrusters	5
4	Vernier Jet Select Table	10
5	Primary Jet Select Table	11
6	Primary with +Z Thrusters Inhibited Jet Select Table	12
7	Primary with Additional +Z Thrusters Jet Select Table	13

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	Visual Display Layout	17
2	DAP Panel	A-1
3	Clohessey-Wiltshire Coordinate System	B-1

1. INTRODUCTION

The Space Transportation System (STS) Proximity Operations Simulator is a nine degrees-of-freedom trajectory integrator (six degrees of freedom for the STS and three degrees of freedom for the target) which generates digital and graphical data to describe and record relative motion of the STS Orbiter and a free-flying payload. This motion is obtained by applying the Clohessy-Wiltshire equations for terminal rendezvous/docking with the earth modeled as a uniform sphere (Appendix B) and aerodynamic forces ignored. STS position relative to target is computed by a first-order Euler integrator which uses quaternions to define the rotational state (Appendix C).

The payload is modeled as a spinning Communications satellite. The Orbiter is treated as a rigid body whose mass properties (gross weight, moments and products of inertia, and center of gravity location) are set by Program THRUSTERS and automatically read in at the beginning of the simulation. These properties remain constant for the entire simulation.

The initial state of the simulation is defined by the user. The program requires altitude and inclination of target orbit to determine proper viewing perspective and orbital dynamics. The user must also input Orbiter position relative to payload. The program sets relative velocity and rotation rates to zero and defines initial Orbiter attitude such that the payload bay faces the target with the Orbiter nose pointed away from the earth. The user also has the option of multiplying shuttle responsiveness to facilitate proximity operations training. Digital Autopilot (DAP) parameters are predefined and cannot be changed by the user.

After program initialization, user inputs are made through the hand controllers and DAP panel located in the Shuttle Aft Flight Deck Mockup. The left controller is the Translational Hand Controller (THC) and is used for positioning. The right hand controller is the Rotational Hand Controller (RHC) and controls Orbiter attitude. Cross-coupling accelerations are fully modeled in the simulation. The program reads inputs from the mockup and updates position and attitude every 40 milliseconds and monitors fuel expended from each tank. Every ten seconds Orbiter position is written to data file STS.TRK should the user wish to view his flightpath at the completion of the simulation using Program STSPATH.

2. RCS ACCELERATION MODELING

(This chapter is taken largely from chapters 3 & 4 of Reference 1)

Table 1 contains the data used by program THRUSTERS to compute forces and torques produced by individual RCS jets (with plume impingement). The left column contains array index numbers used for thruster identification. The second column contains jet identification mnemonics (not used in actual program). The next three columns contain thrust components in Orbiter body axes. These are followed by three columns containing station coordinates of thrust application point. These coordinates are used in conjunction with the last column to calculate torque about the Orbiter CG.

For a CG location defined by arbitrary station coordinates (STA_{cg} , BL_{cg} , WL_{cg}), the torque produced by a particular jet is computed from the equation

$$L_j = R_j \times F_j + C_j F_j \quad (1)$$

where

$$\bar{F}_j = \begin{bmatrix} F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} \quad (2)$$

$$\bar{R}_j = \begin{bmatrix} -(STA_j - STA_{cg})/12 \\ (BL_j - BL_{cg})/12 \\ -(WL_j - WL_{cg})/12 \end{bmatrix} \quad (3)$$

and where the values of F_{xj} , F_{yj} , F_{zj} , STA_j , BL_j , WL_j , and C_j are obtained from columns 3-9 of Table 1.

Table 1. RCS Thruster Data (with plume impingement)

THRUSTER NO.	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
	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	413.46	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	0.0	870.5	-8.4	1516.06	-149.83	455.21	-0.5887
20	L2L	0.0	870.5	-8.4	1529.07	-149.83	455.21	-0.6061
21	L3L	0.0	870.5	-8.4	1542.07	-149.83	455.21	-0.6235
22	L1L	0.0	870.5	-8.4	1555.07	-149.83	455.21	-0.6410
23	R4R	0.0	-870.5	-8.4	1516.06	149.83	455.21	0.5887
24	R2R	0.0	-870.5	-8.4	1529.07	149.83	455.21	0.6061
25	R3R	0.0	-870.5	-8.4	1542.07	149.83	455.21	0.6235
26	R1R	0.0	-870.5	-8.4	1555.07	149.83	455.21	0.6410
27	L4U	29.0	72.0	870.0	1520.04	-116.51	481.65	-0.4615
28	L2U	29.0	72.0	870.0	1532.96	-116.54	481.65	-0.3725
29	L1U	29.0	72.0	870.0	1545.87	-116.58	481.65	-0.2836
30	R4U	29.0	-72.0	870.0	1520.04	116.51	481.65	0.4615
31	R2U	29.0	-72.0	870.0	1532.96	116.54	481.65	0.3725
32	R1U	29.0	-72.0	870.0	1545.87	116.58	481.65	0.2836
33	L4D	312.4	346.8	-545.7	1498.11	-101.47	420.49	1.7413
34	L2D	312.4	346.8	-545.7	1513.68	-100.61	424.63	1.4807
35	L3D	312.4	346.8	-545.7	1529.23	-99.79	428.76	1.2208
36	R4D	312.4	-346.8	-545.7	1498.11	101.47	420.49	-1.7413
37	R2D	312.4	-346.8	-545.7	1513.68	100.61	424.63	-1.4807
38	R3D	312.4	-346.8	-545.7	1529.23	99.79	428.76	-1.2208
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

(Reproduced from Reference 1)

Table 2. Typical Orbiter Mass Properties

DESCRIPTION	VALUE
WEIGHT.....	200017.0 LB
Ixx.....	887302.0 SLUG-FT ²
Iyy.....	6386877.0 SLUG-FT ²
Izz.....	6694367.0 SLUG-FT ²
Iyz.....	-971.0 SLUG-FT ²
Izx.....	247376.0 SLUG-FT ²
Ixy.....	5622.0 SLUG-FT ²
CG STA.....	1095.3 IN
CG BL.....	0.3 IN
CG WL.....	377.4 IN

TABLE 3. RESPONSE MATRIX FOR RCS THRUSTERS

THRUSTER NUMBER	LINEAR ACCELERATION			ANGULAR ACCELERATION			PROPELLANT CONSUMPTION RATES		
	AX	AY	AZ	TX	QY	QZ	PAV TANK	APT LEFT TANK	APT RIGHT TANK
1	-0.14157	-0.00422	0.01930	0.00009	-0.00106	-0.00010	3.10710	0.00000	0.00000
2	-0.14159	0.00000	0.01275	-0.00001	-0.00137	0.00000	3.10710	0.00000	0.00000
3	-0.14157	0.00422	0.01930	0.00001	-0.00136	0.00009	3.10710	0.00000	0.00000
4	-0.14066	0.00293	0.00293	0.00011	-0.00107	0.00001	3.10710	0.00000	0.00000
5	-0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
6	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
7	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
8	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
9	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
10	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
11	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
12	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
13	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
14	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
15	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
16	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
17	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
18	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
19	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
20	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
21	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
22	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
23	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
24	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
25	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
26	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
27	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
28	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
29	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
30	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
31	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
32	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
33	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
34	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
35	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
36	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
37	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
38	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
39	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
40	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
41	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
42	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
43	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000
44	-0.14066	0.00000	0.00000	0.00000	0.00000	0.00000	3.10710	0.00000	0.00000

Program THRUSTERS computes steady-state accelerations and propellant consumption rates for each of the 44 RCS thrusters and writes this information to data file RESPONSES. The user may specify a set of Orbiter mass properties or use the preprogrammed properties from Table 2.

Table 3 shows the RESPONSES data file for the properties listed in Table 2. Each row corresponds to a particular RCS thruster. The first three columns contain the body axis components of the steady-state linear acceleration vector, and columns 4-6 contain the corresponding components of angular acceleration. Columns 7-9 contain rates of propellant flow from the forward, aft left, and aft right tanks respectively.

Linear accelerations for each thruster are calculated by the equation

$$\bar{a}_j = (32.174/W) \begin{bmatrix} F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix} \quad (4)$$

where W is the Orbiter gross weight and F_{xj} , F_{yj} , and F_{zj} represent the thrust components (from Table 1). The corresponding angular accelerations are given by

$$\bar{\alpha} = [I]^{-1} L \quad (5)$$

where

$$\bar{\alpha} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix} \quad (6)$$

$$\mathbf{L} = \begin{bmatrix} L_x \\ L_y \\ L_z \end{bmatrix} \quad (7)$$

$$[\mathbf{I}] = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{zx} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{yz} & I_{zz} \end{bmatrix} \quad (8)$$

and where L_j represents the individual torque vectors produced by the designated thruster as calculated from Equations (1) through (3). The $[\mathbf{I}]$ matrix represents the Orbiter's moments of inertia.

Propellant flow rates (columns 7-9 of Table 3) are assumed to be 3.1071 lb/sec for each active primary jet and 0.0923 lb/sec for each active vernier jet. These rates are based on the nominal vacuum thrust magnitudes (870 lb and 24 lb) and specific impulses (290 sec and 260 sec) that are given in Reference 2. Each thruster is always assumed to be fed from its nominal source (tank) with no provisions for simulating propellant crossfeed.

3. RCS THRUSTER FIRING SELECTION

(This chapter is taken largely from chapter 4 of Reference 1)

Each iteration the program must determine which thrusters are to be fired for attitude and/or translational control. Rotation Hand Controller (RHC), Translation Hand Controller (THC), and Digital Auto Pilot (DAP) inputs are read from the shuttle mockup every 40 milliseconds and applied to one of four available firing options as commanded by the DAP. The available options are designated V (vernier jets), P (primary jets), PZI (primary jets with +Z thrusters inhibited), and HIGH Z (primary jets with additional +Z thrusters). Corresponding to each of these options is a jet-select table (Tables 4-7) which identifies the particular jet or combination of jets that is to be fired in response to each of the six possible translation acceleration commands (+X, -X, +Y, -Y, +Z, -Z) and the six possible rotational commands (+ROL, -ROL, +PCH, -PCH, +YAW, -YAW). These jet-select tables are read from the RESPONSES data file and stored in array JET. Jets are identified by the mnemonics listed in the second column of Table 1.

As indicated in Table 4 by the absence of any jet designations for the execution of translational acceleration commands, the V option can be used only for attitude control. The other three options 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 jet 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.

The digital auto pilot controls the attitude and translation of the Orbiter for both automatic and manned maneuvers by specifying the appropriate commands to the primary and vernier reaction control system (PRCS/VRCS) jets. The crew uses the DAP panel (Figure 1) to exercise vehicle control. The DAP panel consists of hardware pushbuttons

that allow the crew to select the translation DAP auto/manual control operations sequence; to determine whether A or B DAP configuration values will be used with auto/manual control; to select translational control; and to determine if primary or vernier jets will be commanded to fire. Appendix A contains a detailed description of the functions of the DAP control panel.

The precalculated acceleration and propellant consumption information for each thruster in the RESPONSES data file is read at the beginning of the program and stored in array JETSEL. Once the program has determined which thrusters are to be fired, the elements of the rows in JETSEL (Table 3) corresponding to those thrusters are summed. This yields net vectors for translational and rotational acceleration as well as propellant loss rate from each tank. The elements corresponding to acceleration in array JETSEL are not recomputed to account for propellant loss during the simulation.

Table 4. Vernier (V) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+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

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+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

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+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							

Table 7. Primary with Additional +Z Thrusters
(HIGH Z) Jet Select Table

CMD	THRUSTERS TO BE FIRED								
	1	2	3	4	5	6	7	8	9
+X	R1A	L1A							
-X	F2F	F1F							
+Y	F1L	L4L							
-Y	F2R	R4R							
+Z	F2U	F3U	F1U	L4U	L2U	L1U	R4U	R2U	R1U
-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							

4. DETERMINING POSITION AND ATTITUDE

The Orbiter translational accelerations, once computed, are added to the orbital drift accelerations. These accelerations are then integrated twice to yield velocity and position. Attitude is determined by applying rotational accelerations to the present quaternions.

The Clohessy-Wiltshire equations for terminal rendezvous/docking are used to model orbital drift. These are linearized equations of motion for an interceptor vehicle relative to a target vehicle in a circular orbit with Keplerian motion.

$$\ddot{x} = f_x' - 2\omega\dot{y} \quad (9)$$

$$\ddot{y} = f_y' + 3\omega^2 y + 2\omega\dot{x} \quad (10)$$

$$\ddot{z} = f_z' - \omega^2 z \quad (11)$$

where f_x' , f_y' , and f_z' are the Orbiter translational acceleration components (due to thrust), and ω is the rotation rate of the target about the planet.

The target frame is a right handed orthogonal system where x is the direction of target velocity, y is the zenith direction (along target radius vector), and z is out of orbital plane (opposite the angular momentum vector). For further explanation and derivation of Equations 9-11 refer to Appendix B.

The translational accelerations due to drift and thrust are summed in Subroutine THRUST and then integrated in Subroutine LINTEG. LINTEG is a first order Euler integrator which was chosen for fast computational speed (needed in real time simulation) in light of the fact that accelerations, and hence error, will be small.

The rotational accelerations are transformed from the body frame to the reference (target) frame by Subroutine BTOR. These accelerations are then used by Subroutine ROTATE to determine a new attitude quaternion and transformation matrix. The equations used in Subroutine ROTATE are included in Appendix C.

5. GENERATION OF VISUAL DISPLAY

The simulator uses an Evans and Sutherland PS300 computer for visual display. Three dimensional object data are loaded and stored in the mass memory of the PS300 at the beginning of the program. These data are then rotated, translated, and displayed repeatedly as commanded by the main program from the VAX 11/780.

Subroutine PS300 is responsible for loading the object data, known as vector lists, and providing a hierarchy of rotation, translation and viewing commands for later input by Subroutine LOOK. The vector lists consist of a spherical outline of the earth's continents, a star sphere, a circular horizon, target satellite, and heads-up display imagery. The reference coordinate system is the Clohessy-Wiltshire system, centered at the target and described in detail in Appendix B.

The earth's vector list is scaled at one distance unit (DU) and rotated about its axis at the rate of 15 degrees per hour. The earth is then inclined and counter-rotated at a rate equivalent to the angular rate of the target orbit. Finally, it is translated in the -Y direction an amount equal to its radius plus target altitude. To complete the earth picture a horizon circle is added and the earth vector list is clipped to prevent viewing beyond the horizon.

The star vector list is triplicated and rotated 90 degrees about each axis to create a unit sphere of stars. This representation does not reflect true star positions. The sphere is scaled up by a factor of one earth radius plus twice target altitude and set counter-rotating (as was the earth). The star sphere is then translated in the -Y direction an amount equal to one DU plus target altitude and clipped so no stars appear below the horizon.

The target vector list is scaled, placed at the center of the reference coordinate system, and rotated at fifty degrees per minute to give the effect of a spinning satellite. It is not clipped and remains at the origin throughout the simulation.

Heads-up display information includes on screen printout of range, range rate, fuel used, and elapsed time in the upper left hand corner. The upper right hand corner gives an X-Y plane view of the shuttle and predicted future position in five minute increments. The increment markers

appear as small diamonds originating from the shuttle silhouette. Affixed to the target are radius, velocity, and out-of-plane arrows to facilitate quadrant determination. A fixed reticle is also displayed to help the operator judge target drift.

Subroutine LOOK uses shuttle position and attitude data relative to the target to continuously update rotation, translation, and viewing of all the predefined vector lists. Current position and attitude are used to generate three viewing vectors in the reference (target) frame: AT, FROM, and UP. AT is the line of sight vector, FROM is the position vector and UP is the overhead vector (perpendicular to AT). The Evans and Sutherland PS300 uses these viewing vectors to scale, translate, and re-orient the stored images for perspective viewing. Figure 1 shows the relationship of the stored images to each other and how they are viewed given AT, FROM and UP vectors.

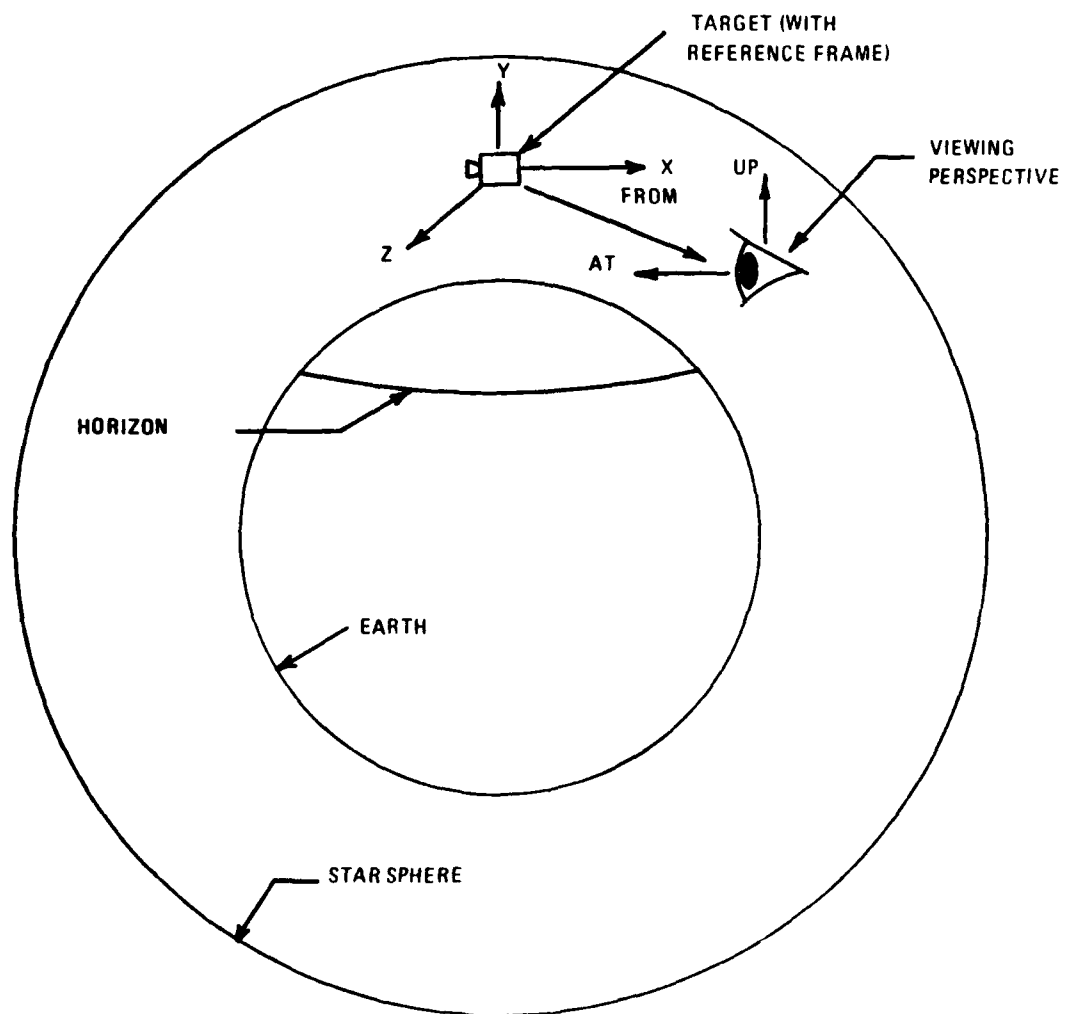


Figure 1. Visual Display Layout

REFERENCES

1. S. W. Wilson, "Engineering Description of the OMS/RCS/DAP Models used in the HP-9825A High Fidelity Relative Motion Program (HFRMP)," TRW Report No. 28415-H009-R0-00, 6 October 1978
2. Lyndon B. Johnson Space Center, "Shuttle Operational Data Book, Volume 1, Shuttle Systems Performance and Constraints Data," NASA/JSC Document No. JSC-08934 (Vol. 1), Revision A, updated 13 June 1978.
3. Lyndon B. Johnson Space Center, "Attitude and Pointing Flight Procedures Handbook", NASA/JSC Document No. JSC-10511, January 1982.
4. Astro 451 Course Handout, Astro 451, Course Readings #2, Fall 1984, Chapter F, USAF Academy, Colorado, 1984
5. Hamilton Standard Paper, Strapdown Attitude Determination Using Quaternions, Hamilton Standard Division of United Technologies, Undated

APPENDIX A

(This appendix was taken largely from Reference 3)

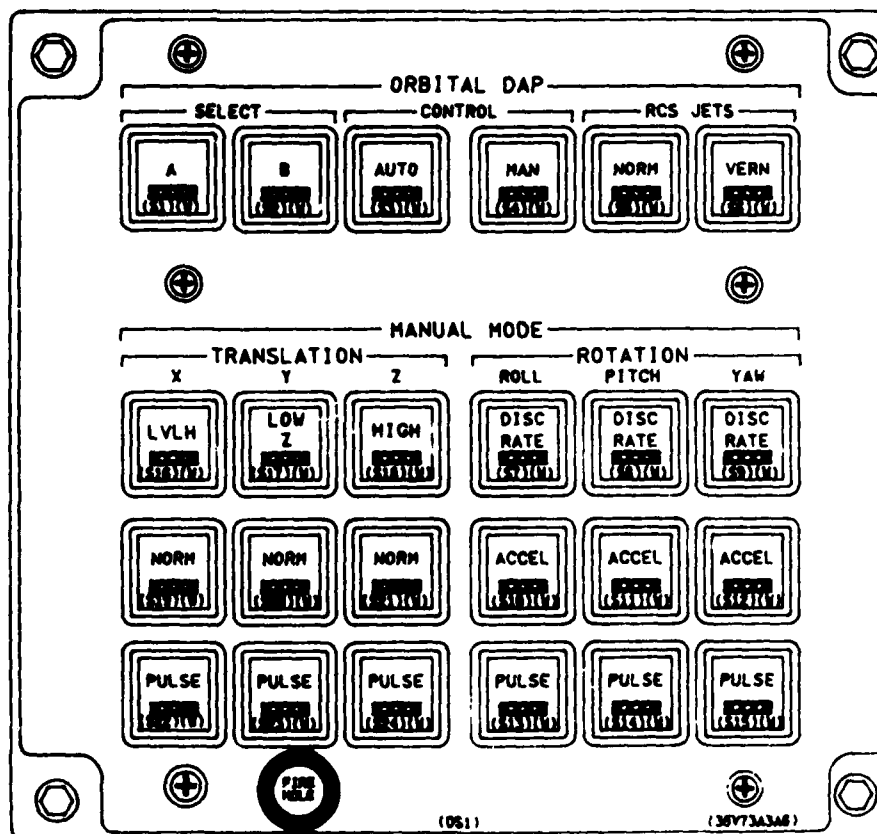


Figure 2. Panel C3: Orbital DAP Panel

AUTO/MAN Selection

- AUTO A - Shuttle automatically flies directly to target.
- AUTO B - Shuttle flies Clohessy-Wiltshire approach to 40 feet in front/behind and then stationkeeps.
- MAN - Allows user to fly shuttle manually.

LVLH - With DISC RATE selected in all three axes and with the LVLH mode selected, the attitude that existed at the time of selection of LVLH will be held fixed within the LVLH frame until changed by use of the RHC. The RHC can be used with LVLH selected to change LVLH attitude. When the RHC returns to detent, the LVLH attitude begins to be held fixed again. This mode would be used manually for precise PROX OPS attitude control in the near vicinity of a payload.

NORM VERN JET SELECT - Primary RCS (PRCS or NORM) and vernier RCS (VRCS or VERN) jets can be used for attitude hold and maneuvers. NORM jets must be selected for translation maneuvers. With primaries selected, there is a choice of jets as shown in Tables 5-7.

Translation Submodes (NORM Jets Required)

PULSE X, Y, or Z - With this submode selected, each deflection of the THC will result in a change in velocity of the magnitude prespecified in the program. (DAP A yields .05 feet per second, DAP B yields .01 feet per second).

NORM X, Y, Z - With one of these submodes selected, continuous acceleration will occur at whatever level is available with the commanded jets for the axis and direction (+ or -) of the THC deflection.

HIGH Z - An ACCEL submode that is available for +Z translation only. Selection of this submode fires more jets than are fired for the NORM +Z command (Table 7).

LOW Z - LOW Z is a submode that takes advantage of some +Z thrust components that exist for both the +X and -X translation jets (Table 6). To get the +Z translation, the +X and -X translational jets are fired simultaneously so that the jets nearly cancel the X components of each other and combine the small Z components. The result is a small +Z velocity change. This mode was added because it provides translation away from a deployed payload without firing thrusters in the direction of the payload (thus minimizing plume impingement) as would the other +Z translation submodes. LOW Z translations also affect pitch and roll, since only downfiring jets are used. This results in significant -Z translational cross-coupling, but reduces the duty cycle for attitude maintenance.

Manual Rotational Submodes

DISC RATE - ROLL, PITCH, or YAW - When the DAP mode is manual, the submode is DISC RATE and the RHC is deflected out of the detent in an axis, jets are fired until an angular rate is achieved in that axis equal to the preprogrammed rate (DAP A yields .000873 radians per second, DAP B yields .000175 radians per second). When the rate is achieved the jets are turned off until the RHC is returned to detent in that axis unless the jets are momentarily required to maintain attitude or rates within deadbands. When the RHC is returned to detent in an axis, the attitude in that axis is snapshotted and jets are fired to stop the rate and to begin holding the snapshotted attitude.

ACCEL - ROLL, PITCH, YAW - When the DAP mode is MAN/ACCEL and the RHC is deflected out of detent, a fixed number of jets are fired to provide constant angular acceleration. When the RHC is returned to detent, the jets are turned off but the angular rates continue. The result is a free mode since the attitude is uncontrolled when the RHC is in detent ('free drift at the existing angular rates).

PULSE - ROLL, PITCH, or YAW - When the DAP mode is in MAN/PULSE and the RHC is deflected out of detent; jets are fired just long enough to achieve the preprogrammed angular rate (DAP A yields .3 radians per second squared, DAP B yields .06 radians per second squared). PULSE is also a free mode.

RHC and THC - The handcontrollers are used to input commands to the DAP, which then (depending on DAP configuration) outputs appropriate RCS jet fire commands.

APPENDIX B

(This appendix was taken wholly from Reference 4)

TERMINAL RENDEZVOUS/DOCKING

This appendix develops the equations of motion for an interceptor vehicle relative to a target vehicle in orbit about a central body when the range between vehicles is less than 8 km (5 miles).

The following approach will be taken to solve this engineering problem.

1. Establish a coordinate system:

- a. Origin at target vehicle
- b. Orthogonal right-handed system
- c. x -- In the local horizontal, in direction of target vehicle velocity vector
- d. y -- In zenith direction (along target vehicle position vector R)

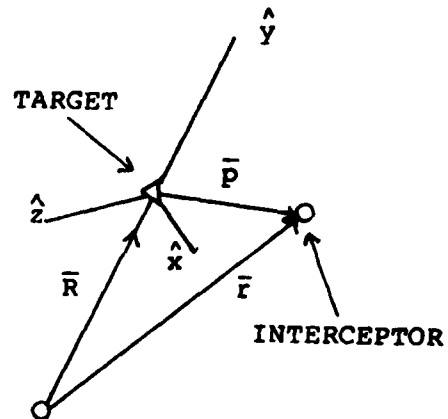


Figure 3

- e. R -- Vector to target in fixed frame

- f. r -- Vector to interceptor in fixed frame

- g. $\bar{p} = \bar{r} - \bar{R}$ -- Position of the interceptor relative to the target.

2. Apply $\Sigma \bar{F}_{ext} = \frac{d}{dt}(\bar{m}\bar{v})$ in the rotating coordinate system:

- a. This is a double application of the law of coriolis for derivatives in a rotating frame. (If you can't derive this, see BMW pp. 92-93.)

$$\frac{d^2(\quad)}{dt^2} \bar{F} = \frac{d^2(\quad)}{dt^2} \bar{R} + \dot{\bar{\omega}} \times (\quad) + \bar{\omega} \times \bar{\omega} \times (\quad) + 2 \bar{\omega} \times \frac{d(\quad)}{dt} \bar{R}$$

b. When () is the position of the interceptor vehicle, ($\bar{r}$), then

$$\frac{d^2 \bar{r}}{dt^2} = \frac{\Sigma \bar{F}_{ext}}{m} = \bar{F} = \ddot{\bar{r}}_R + \dot{\bar{u}} \times \bar{r} + \bar{u} \times \dot{\bar{u}} \times \bar{r} + 2 \bar{u} \times \dot{\bar{r}}_R$$

c. Noting the following relationships

$$(1) \bar{u} = -u \hat{z}$$

$$(7) \dot{\bar{r}}_R = \dot{R} \hat{y}$$

$$(2) \dot{\bar{u}} = -\dot{u} \hat{z}$$

$$(8) \ddot{\bar{r}}_R = \ddot{R} \hat{y}$$

$$(3) \bar{p} = x \hat{x} + y \hat{y} + z \hat{z}$$

$$(9) \bar{r} = x \hat{x} + (R + y) \hat{y} + z \hat{z}$$

$$(4) \dot{\bar{p}}_R = \dot{x} \hat{x} + \dot{y} \hat{y} + \dot{z} \hat{z}$$

$$(10) \dot{\bar{r}}_R = \dot{x} \hat{x} + (\dot{R} + \dot{y}) \hat{y} + \dot{z} \hat{z}$$

$$(5) \ddot{\bar{p}}_R = \ddot{x} \hat{x} + \ddot{y} \hat{y} + \ddot{z} \hat{z}$$

$$(11) \ddot{\bar{r}}_R = \ddot{x} \hat{x} + (\ddot{R} + \ddot{y}) \hat{y} + \ddot{z} \hat{z}$$

$$(6) \bar{R} = R \hat{y}$$

d. Developing the required cross products

$$\dot{\bar{u}} \times \bar{r} = \dot{u}(R + y) \hat{x} - \dot{u}x \hat{y}$$

$$\bar{u} \times \dot{\bar{r}} = u(\dot{R} + \dot{y}) \hat{x} - ux \dot{\hat{y}}$$

$$\bar{u} \times \dot{\bar{u}} \times \bar{r} = -u^2 x \hat{x} - u^2 (R + y) \hat{y}$$

e. Now substituting from c and d into the component form of the vector equation developed in 2.b, we obtain

$$f_x = \ddot{x} + \dot{u}(R + y) - u^2 x + 2u(\dot{R} + \dot{y})$$

$$f_y = \ddot{R} + \ddot{y} - \dot{u}x - u^2 (R + y) - 2u\dot{x}$$

$$f_z = \ddot{z}$$

f. Now, let f' be the specific force other than the gravitational attraction of the central body (i.e., drag, thrust, solar pressure, etc.)

$$\vec{F} = \vec{F}' + \vec{F}_g = \vec{F}' - \frac{\mu \vec{r}}{r^3}$$

OR

$$f_x = f'_x - \frac{\mu x}{r^3}$$

$$f_y = f'_y - \frac{\mu(R + y)}{r^3}$$

$$f_z = f'_z - \frac{\mu z}{r^3}$$

g. Then the relative motion is described by

$$\ddot{x} = f'_x - \dot{u}(R + y) + u^2 x - 2u\dot{y} - \frac{\mu x}{r^3}$$

$$\ddot{y} = f'_y - \ddot{R} + \dot{u}x + u^2(R + y) + 2u\dot{x} - \frac{\mu(R + y)}{r^3}$$

$$\ddot{z} = f'_z - \frac{\mu z}{r^3} \quad \text{where} \quad r^3 = [x^2 + (R + y)^2 + z^2]^{3/2}$$

THESE ARE VERY NON-LINEAR, BUT EXACT EQUATIONS OF MOTION.

3. Linearize the equations to find an analytic solution:

$$r^3 = R^3 \left[\left(\frac{x}{R}\right)^2 + \left(1 + \frac{y}{R}\right)^2 + \left(\frac{z}{R}\right)^2 \right]^{3/2}$$

$$= R^3 \left[1 + \frac{2y}{R} + \left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2 + \left(\frac{z}{R}\right)^2 \right]^{3/2}$$

a. Since x , y and z are small compared with R ,

$$r^3 \approx R^3 \left(1 + \frac{2y}{R} \right)^{3/2}$$

b. Then the last terms in the equations of 2.g. take on the form

$$-\frac{\mu}{r^3} \approx -\frac{\mu}{R^3(1 + \frac{2y}{R})^{3/2}}$$

c. Since the term $(\frac{y}{R})$ is small (less than 0.00125), use the binominal expansion to begin to simplify these equations

$$(1 + \epsilon)^{-3/2} = 1 - \frac{3}{2}\epsilon + \frac{15}{8}\epsilon^2 - \dots$$

d. Then, substituting into 2g and neglecting the small high order terms,

$$-\frac{\mu x}{R^3(1 + \frac{2y}{R})^{-3/2}} \approx -\frac{\mu x}{R^3} + \frac{3\mu xy}{R^4}, \text{ etc.}$$

So that

$$\ddot{x} = f_x' - \dot{u}(R + y) + u^2 x - 2u\dot{x}(R + \dot{y}) - \frac{\mu x}{R^3} + \frac{3\mu xy}{R^4}$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{u}x + u^2(R + y) + 2u\dot{x} - \frac{\mu}{R^2} + \frac{2\mu y}{R^3} + \frac{3\mu y^2}{R^4}$$

$$\ddot{z} = f_z' - \frac{\mu z}{R^3} + \frac{3\mu zy}{R^4}$$

e. The only nonlinear terms are the last ones in the equations. These terms are smaller than the proceeding terms by $(\frac{y}{R})$. Neglecting these terms results in

$$\ddot{x} = f_x' - \dot{u}(R + y) + (u^2 - \frac{\mu}{R^3})x - 2u\dot{x}(R + \dot{y})$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{u}x + (u^2 - \frac{\mu}{R^3})R + (u^2 + \frac{2\mu}{R^3})y + 2u\dot{x}$$

$$\ddot{z} = f_z' - \frac{\mu}{R^3}z$$

4. Now ASSUME the target vehicle is in a CIRCULAR orbit.

a. Then

$$u = \frac{v_{CS}}{R} = \sqrt{\frac{\mu}{R^3}}, \quad u^2 = \frac{\mu}{R^3}, \quad \dot{u} = 0, \quad \dot{R} \text{ and } \dot{R} = 0$$

b. The equations of motion become

$$\ddot{x} = f_x' - 2uy\dot{y}$$

$$\ddot{y} = f_y' + 3u^2y + 2ux\dot{x}$$

$$\ddot{z} = f_z' - u^2z$$

THESE ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPTOR VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT WITH KEPLERIAN MOTION.

APPENDIX C

(This appendix was taken wholly from Reference 5)

ATTITUDE DETERMINATION USING QUATERNIONS

The quaternion q describing the orientation of a body with respect to a reference coordinate frame may be found by integrating the quaternion differential equation:

$$\dot{q} = q \frac{\omega}{2} \quad (1)$$

where

$$q = 1q_1 + j q_2 + k q_3 + q_4$$

$$\omega = 1\omega_x + j\omega_y + k\omega_z$$

ω = rate of rotation of body with respect to the reference frame (in body coordinate frame).

Expansion of (1) yields the following form:

$$\frac{d}{dt} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad (2)$$

The quaternion components in the body (X, Y, Z) coordinate frame are

$$\begin{aligned} q_1 &= \frac{u_X}{w} \sin \frac{ut}{2} \\ q_2 &= \frac{u_Y}{w} \sin \frac{ut}{2} \\ q_3 &= \frac{u_Z}{w} \sin \frac{ut}{2} \\ q_4 &= \cos \frac{ut}{2} \end{aligned} \tag{3}$$

where

$$w = \sqrt{u_X^2 + u_Y^2 + u_Z^2}$$

A vector is transformed from body to reference coordinates by

$$\bar{x}^R = q \bar{x}^B q^* \tag{4}$$

where

$$\begin{aligned} q^* &= -iq_1 - jq_2 - kq_3 + q_4 \\ &= \text{conjugate of } q \\ qq^* &= q^* q = 1 \end{aligned}$$

The equivalent equation in vector-matrix notation is

$$\bar{x}^R = \begin{bmatrix} T_B^R \end{bmatrix} \bar{x}^B \tag{5}$$

If (4) and (5) are expanded and compared, it is seen that

$$\begin{bmatrix} T_B^R \end{bmatrix} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 - q_3 q_4) & 2(q_1 q_3 + q_2 q_4) \\ 2(q_1 q_2 + q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 - q_1 q_4) \\ 2(q_1 q_3 - q_2 q_4) & 2(q_2 q_3 + q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \quad (6)$$

From (6), the quaternion components, expressed as functions of the matrix elements, are

$$\begin{aligned} q_1 &= \frac{1}{4q_4} (T_{32} - T_{23}) \\ q_2 &= \frac{1}{4q_4} (T_{13} - T_{31}) \\ q_3 &= \frac{1}{4q_4} (T_{21} - T_{12}) \\ q_4 &= \frac{1}{2} \sqrt{1 + T_{11} + T_{22} + T_{33}} \end{aligned} \quad (7)$$

APPENDIX D

THRUSTERS PROGRAM LISTING

C THIS PROGRAM GENERATES ROTATIONAL AND TRANSLATIONAL ACCELERATIONS
 C AND ASSOCIATED FUEL COSTS FOR EACH THRUSTER AND STORES THEM IN
 C ARRAY "THRUST". TO DO THIS IT APPLIES EACH THRUSTER'S FORCE
 C AND POSITION ON SHUTTLE BODY (STORED IN ARRAY 'JET') TO SHUTT
 C MASS, CG, AND MOMENTS OF INERTIA.
 C FOR A DETAILED EXPLANATION OF EQUATIONS/PARAMETERS REFER TO
 C TRW PAPER "ENGINEERING DESCRIPTION OF THE CMS/RCS/DAP MODELS
 C USED IN THE HP-9825A HIGH FIDELITY RELATIVE MOTION PROGRAM (HFRMP)
 C 6 OCT 78. OR SEE CAPT. ALFANO, DFAS
 C

REAL MASS,STACG,BLCG,WLCG,RX,RY,RZ
 REAL MOI(3,3),IMOI(3,3),JET(44,12),THRUST(44,9)
 REAL VUN(12),PUN(12),PZIUN(12),PZHI(12)
 INTEGER I,J,TYPE,TABLE(4,12,9),IOSTAT

C
 C
 C
 C

SELECTION TABLES ARE AS FOLLOWS:

DATA (TABLE(1,1,I),I=1,9)/9*0/
 DATA (TABLE(1,2,I),I=1,9)/9*0/
 DATA (TABLE(1,3,I),I=1,9)/9*0/
 DATA (TABLE(1,4,I),I=1,9)/9*0/
 DATA (TABLE(1,5,I),I=1,9)/9*0/
 DATA (TABLE(1,6,I),I=1,9)/9*0/
 DATA (TABLE(1,7,I),I=1,9)/44,8*0/
 DATA (TABLE(1,8,I),I=1,9)/43,8*0/
 DATA (TABLE(1,9,I),I=1,9)/39,40,7*0/
 DATA (TABLE(1,10,I),I=1,9)/44,43,7*0/
 DATA (TABLE(1,11,I),I=1,9)/41,8*0/
 DATA (TABLE(1,12,I),I=1,9)/42,8*0/

C

DATA (TABLE(2,1,I),I=1,9)/16,18,7*0/
 DATA (TABLE(2,2,I),I=1,9)/1,3,7*0/
 DATA (TABLE(2,3,I),I=1,9)/4,19,7*0/
 DATA (TABLE(2,4,I),I=1,9)/6,23,7*0/
 DATA (TABLE(2,5,I),I=1,9)/9,27,30,6*0/
 DATA (TABLE(2,6,I),I=1,9)/12,11,33,34,36,37,3*0/
 DATA (TABLE(2,7,I),I=1,9)/33,30,7*0/
 DATA (TABLE(2,8,I),I=1,9)/27,36,7*0/
 DATA (TABLE(2,9,I),I=1,9)/12,11,27,30,5*0/
 DATA (TABLE(2,10,I),I=1,9)/9,33,36,6*0/
 DATA (TABLE(2,11,I),I=1,9)/4,23,7*0/
 DATA (TABLE(2,12,I),I=1,9)/6,19,7*0/

C

DATA (TABLE(3,1,I),I=1,9)/16,18,7*0/
 DATA (TABLE(3,2,I),I=1,9)/1,3,7*0/
 DATA (TABLE(3,3,I),I=1,9)/4,19,7*0/
 DATA (TABLE(3,4,I),I=1,9)/6,23,7*0/
 DATA (TABLE(3,5,I),I=1,9)/1,3,16,18,5*0/
 DATA (TABLE(3,6,I),I=1,9)/12,11,33,34,36,37,3*0/
 DATA (TABLE(3,7,I),I=1,9)/33,8*0/
 DATA (TABLE(3,8,I),I=1,9)/36,8*0/
 DATA (TABLE(3,9,I),I=1,9)/12,11,7*0/
 DATA (TABLE(3,10,I),I=1,9)/33,36,7*0/
 DATA (TABLE(3,11,I),I=1,9)/4,23,7*0/
 DATA (TABLE(3,12,I),I=1,9)/6,19,7*0/

C

DATA (TABLE(4,1,I),I=1,9)/16,18,7*0/
 DATA (TABLE(4,2,I),I=1,9)/1,3,7*0/
 DATA (TABLE(4,3,I),I=1,9)/4,19,7*0/

```

DATA (TABLE(4,4,I),I=1,9)/6,23,7*0/
DATA (TABLE(4,5,I),I=1,9)/8,9,10,27,28,29,30,31,32/
DATA (TABLE(4,6,I),I=1,9)/12,11,33,34,36,37,3*0/
DATA (TABLE(4,7,I),I=1,9)/33,30,7*0/
DATA (TABLE(4,8,I),I=1,9)/27,36,7*0/
DATA (TABLE(4,9,I),I=1,9)/12,11,27,30,5*0/
DATA (TABLE(4,10,I),I=1,9)/9,33,36,6*0/
DATA (TABLE(4,11,I),I=1,9)/4,23,7*0/
DATA (TABLE(4,12,I),I=1,9)/6,19,7*0/

```

C
C
C

ASSIGN VALUES TO JET ARRAY

```

DATA (JET(1,I),I=1,9)/-879.4,-26.2,119.9,306.72,14.65,392.96,
*0.3,1071,1/
DATA (JET(2,I),I=1,9)/-879.5,0.0,122.7,306.72,0.0,394.45,0.0,
*3.1071,1/
DATA (JET(3,I),I=1,9)/-879.4,26.2,119.9,306.72,-14.65,392.96,
*0.0,3.1071,1/
DATA (JET(4,I),I=1,9)/-26.3,873.6,18.2,362.67,-69.5,373.73,
*0.0,3.1071,1/
DATA (JET(5,I),I=1,9)/-21.0,870.3,.5,364.71,-71.65,359.25,0.0,
*3.1071,1/
DATA (JET(6,I),I=1,9)/-26.3,-873.6,18.2,362.67,69.5,373.73,
*0.0,3.1071,1/
DATA (JET(7,I),I=1,9)/-21.0,-870.3,0.5,364.71,71.65,359.25,
*0.0,3.1071,1/
DATA (JET(8,I),I=1,9)/-32.3,-11.7,874.4,350.93,14.39,413.46,
*0.0,3.1071,1/
DATA (JET(9,I),I=1,9)/-31.9,0.0,873.5,350.92,0.0,414.53,
*0.0,3.1071,1/
DATA (JET(10,I),I=1,9)/-32.3,11.7,874.4,350.93,-14.39,413.46,
*0.0,3.1071,1/
DATA (JET(11,I),I=1,9)/-28.0,-616.4,-639.5,333.84,61.42,356.95,
*0.0,3.1071,1/
DATA (JET(12,I),I=1,9)/-28.0,616.4,-639.5,333.84,-61.42,356.95,
*0.0,3.1071,1/
DATA (JET(13,I),I=1,9)/-24.8,-612.6,-639.4,348.44,66.23,358.44,
*0.0,3.1071,1/
DATA (JET(14,I),I=1,9)/-24.8,612.6,-639.4,348.44,-66.23,358.44,
*0.0,3.1071,1/
DATA (JET(15,I),I=1,9)/856.8,0.0,151.1,1555.29,137.0,473.06,
*0.0,3.1071,2/
DATA (JET(16,I),I=1,9)/856.8,0.0,151.1,1555.29,124.0,473.06,
*0.0,3.1071,2/
DATA (JET(17,I),I=1,9)/856.8,0.0,151.1,1555.29,-137.0,473.06,
*0.0,3.1071,3/
DATA (JET(18,I),I=1,9)/856.8,0.0,151.1,1555.29,-124.0,473.06,
*0.0,3.1071,3/
DATA (JET(19,I),I=1,9)/0.0,870.5,-8.4,1516.06,-149.83,455.21,
*-0.5887,3.1071,3/
DATA (JET(20,I),I=1,9)/0.0,870.5,-8.4,1529.07,-149.83,455.21,
*-0.6061,3.1071,3/
DATA (JET(21,I),I=1,9)/0.0,870.5,-8.4,1542.07,-149.83,455.21,
*-0.6235,3.1071,3/
DATA (JET(22,I),I=1,9)/0.0,870.5,-8.4,1555.07,-149.83,455.21,
*-0.6410,3.1071,3/
DATA (JET(23,I),I=1,9)/0.0,-870.5,-8.4,1516.06,149.83,455.21,
*0.5887,3.1071,2/
DATA (JET(24,I),I=1,9)/0.0,-870.5,-8.4,1529.07,149.83,455.21,
*0.6061,3.1071,2/

```

```

DATA (JET(25,I),I=1,9)/0.0,-870.5,-8.4,1542.07,149.83,455.21,
*0.6235,3.1071,2/
DATA (JET(26,I),I=1,9)/0.0,-870.5,-8.4,1555.07,149.83,455.21,
*0.641,3.1071,2/
DATA (JET(27,I),I=1,9)/29.0,72.0,870.0,1520.04,-116.51,481.65,
*-0.4615,3.1071,3/
DATA (JET(28,I),I=1,9)/29.0,72.0,870.0,1532.96,-116.54,481.65,
*-0.3725,3.1071,3/
DATA (JET(29,I),I=1,9)/29.0,72.0,870.0,1545.87,-116.58,481.65,
*-0.2836,3.1071,3/
DATA (JET(30,I),I=1,9)/29.0,-72.0,870.0,1520.04,116.51,481.65,
*0.4615,3.1071,2/
DATA (JET(31,I),I=1,9)/29.0,-72.0,870.0,1532.96,116.54,481.65,
*0.3725,3.1071,2/
DATA (JET(32,I),I=1,9)/29.0,-72.0,870.0,1545.87,116.58,481.65,
*0.2836,3.1071,2/
DATA (JET(33,I),I=1,9)/312.4,346.8,-545.7,1498.11,-101.47,420.49,
*1.7413,3.1071,3/
DATA (JET(34,I),I=1,9)/312.4,346.8,-545.7,1513.68,-100.61,424.63,
*1.4807,3.1071,3/
DATA (JET(35,I),I=1,9)/312.4,346.8,-545.7,1529.23,-99.79,428.76,
*1.2208,3.1071,3/
DATA (JET(36,I),I=1,9)/312.4,-346.8,-545.7,1498.11,101.47,420.49,
*-1.7413,3.1071,2/
DATA (JET(37,I),I=1,9)/312.4,-346.8,-545.7,1513.68,100.61,424.63,
*-1.4807,3.1071,2/
DATA (JET(38,I),I=1,9)/312.4,-346.8,-545.7,1529.23,99.79,428.76,
*-1.2208,3.1071,2/
DATA (JET(39,I),I=1,9)/-0.8,-17.0,-17.6,324.35,59.7,350.12,
*0.0,.0923,1/
DATA (JET(40,I),I=1,9)/-0.8,17.0,-17.6,324.35,-59.7,350.12,
*0.0,.0923,1/
DATA (JET(41,I),I=1,9)/0.0,-24.0,-0.6,1565.0,149.87,459.0,
*0.0,.0923,2/
DATA (JET(42,I),I=1,9)/0.0,24.0,-0.6,1565.0,-149.87,459.0,
*0.0,.0923,3/
DATA (JET(43,I),I=1,9)/0.0,0.0,-24.0,1565.0,118.0,455.44,
*0.0,.0923,2/
DATA (JET(44,I),I=1,9)/0.0,0.0,-24.0,1565.0,-118.0,455.44,
*0.0,.0923,3/

```

```

C
C      REASSIGN VALUES TO JET
C

```

```

DO 900 I=1,44
  JET(I,11)=JET(I,8)
  JET(I,12)=JET(I,9)
900 CONTINUE
C

```

```

MASS=200017.0/32.2
MOI(1,1)=887302.00
MOI(2,2)=6386877.0
MOI(3,3)=6694367.0
MOI(1,2)=-5622.0
MOI(2,1)=-5622.0
MOI(2,3)=971.0
MOI(3,2)=971.0
MOI(1,3)=-247376.0
MOI(3,1)=-247376.0
STACG=1095.3
BLCG=0.3

```



```
C C PZIUN( )=PRIMARY WITH +Z INHIBITED/UNCOMPENSATED RESPONSES  
C C PZHI( )=PRIMARY WITH HIGH Z/UNCOMPENSATED RESPONSES
```

```
CALL INVERT(MOI,IMOI)  
CALL JETSEL(IMOI,JET,MASS,THRUST,VUN,PUN,PZIUN,PZHI,TABLE)
```

```
C C OPEN FILES HERE
```

```
*OPEN(UNIT=2,NAME='[ALFANO.DATA]RESPONSES.DAT',TYPE='NEW',  
8 *FORM='UNFORMATTED',INITIALSIZE=40)  
C *OPEN(UNIT=3,FILE='RSP.TBL',STATUS='NEW',  
C *IOSTAT=ISTAT)
```

```
C WRITE TO THE FILE
```

```
105 FORMAT(7X,I2,9(3X,F9.5))  
DO 825 I=1,44  
    WRITE(2)(THRUST(I,J),J=1,9)  
    WRITE(3,105)I,(THRUST(I,J),J=1,9)  
CONTINUE  
825 DO 850 I=1,12  
    WRITE(2)(TABLE(1,I,J),J=1,9)  
    WRITE(2)(TABLE(2,I,J),J=1,9)  
    WRITE(2)(TABLE(3,I,J),J=1,9)  
    WRITE(2)(TABLE(4,I,J),J=1,9)  
    WRITE(2)VUN(I)  
    WRITE(2)PUN(I)  
    WRITE(2)PZIUN(I)  
    WRITE(2)PZHI(I)  
850 CONTINUE
```

```
C CLOSE FILES
```

```
C CLOSE(UNIT=2)  
CLOSE(UNIT=3)
```

```
C END
```

```
SUBROUTINE ROWOP(MAT,RC)  
    REAL MAT(3,6)  
    INTEGER RC,I,J  
    REAL FACTOR
```

```
C THIS ROUTINE UNITIZES ONE ROW OF A 3 BY 6 MATRIX (MAT)  
ABOUT THE PIVOT (RC) AND THEN PERFORMS ELEMENTARY ROW  
OPERATIONS ON THE REMAINING TWO ROWS.  
(TO BE USED BY SUBROUTINE INVERT)
```

```
FACTOR=MAT(RC,RC)  
DO 100 I=1,6  
    MAT(RC,I)=MAT(RC,I)/FACTOR  
100 CONTINUE
```

```

DO 200 J=1,3
  IF (J .EQ. RC) GOTO 200
  IF (ABS(MAT(J,RC)) .LE. .0000000001) GOTO 200
  FACTOR=-MAT(J,RC)/MAT(RC,RC)
  DO 400 I=1,6
    MAT(J,I)=MAT(J,I)+MAT(RC,I)*FACTOR
400  CONTINUE
200  CONTINUE
END

C
C
C
C

SUBROUTINE INVERT(MOI,IMOI)
C
C  THIS ROUTINE SETS UP A 3 BY 6 MATRIX (DUMMY). LEFT 3 BY 3
C  MATRIX IS THE MOMENT OF INERTIA MATRIX (MOI). THE RIGHT
C  3 BY 3 IS THE IDENTITY MATRIX. CALLS TO SUBROUTINE ROWOP
C  PERFORM ELEMENTARY ROW OPERATIONS MAKING THE LEFT 3 BY 3
C  OF DUMMY THE IDENTITY MATRIX AND THE RIGHT 3 BY 3 THE INVERSE
C  MOMENT OF INERTIA MATRIX (IMOI).
C
C  REAL MOI(3,3),IMOI(3,3),DUMMY(3,6)
C  INTEGER K,L
C
DO 100 K=1,3
  DO 300 L=1,3
    DUMMY(K,L)=MOI(K,L)
    DUMMY(K,L+3)=0.0
300  CONTINUE
    DUMMY(K,K+3)=1.0
100  CONTINUE
    CALL ROWOP(DUMMY,1)
    CALL ROWOP(DUMMY,2)
    CALL ROWOP(DUMMY,3)
    DO 150 K=1,3
      DO 350 L=1,3
        IMOI(K,L)=DUMMY(K,L+3)
350  CONTINUE
150  CONTINUE
END

C
C
C
C

SUBROUTINE JETSEL(IMOI,JET,MASS,THRUST,VUN,PUN,PZIUN,
* PZHI,TABLE)
C
C  SUMS FORCES AND TORQUES OF THRUSTERS, CONVERTS TO ACCELERATIONS
C  ASSIGNS PROPELLANT LOSSES FOR THRUSTER.
C
C  INTEGER I,J,TABLE(4,12,9),ROW,COL
C  REAL M(12),ALPHA(3),IMOI(3,3),JET(44,12),MASS,THRUST(44,9)
C  REAL VUN(12),PUN(12),PZIUN(12),PZHI(12)
C
DO 100 I=1,44
  M(1)=JET(I,1)
  M(2)=JET(I,2)
  M(3)=JET(I,3)
  M(4)=JET(I,8)

```

```

M(5)=JET(I,9)
M(6)=JET(I,10)

C
C   ASSIGNING OF PROPELLANT
C
M(7)=0.0
M(9)=0.0
M(8)=JET(I,11)
IF (JET(I,12) .LE. 1.5) THEN
  M(7)=JET(I,11)
  M(8)=0.0
ENDIF
IF (JET(I,12) .GE. 2.5) THEN
  M(9)=JET(I,11)
  M(8)=0.0
ENDIF

C
C   CONVERT FORCES AND TORQUES TO ACCELERATIONS
C
M(1)=M(1)/MASS
M(2)=M(2)/MASS
M(3)=M(3)/MASS
DO 400 J=1,3
  ALPHA(J)=IMO(I,J,1)*M(4)+IMO(I,J,2)*M(5)+IMO(I,J,3)*M(6)
400 CONTINUE
M(4)=ALPHA(1)
M(5)=ALPHA(2)
M(6)=ALPHA(3)

C
C   ASSIGN THRUST ARRAY AS FOLLOWS
C   COL 1-3  TRANSLATIONAL ACCELERATION (FPS2)
C   COL 4-6  ROTATIONAL ACCELERATION (RAD/S2)
C   COL 7-9  FUEL USED (LBS/S) (7-FWD 8-RT 9-LT)
C
DO 200 J=1,9
  THRUST(I,J)=M(J)
200 CONTINUE
100 CONTINUE

C
C   ASSIGN RATES CORRESPONDING TO COMMANDS FOR GIVEN
C   THRUST PROFILE
C
DO 500 I=1,12
  VUN(I)=0.0
  DO 600 J=1,9
    IF (TABLE(1,I,J) .EQ. 0) GOTO 500
    ROW=TABLE(1,I,J)
    COL=INT((I+1)/2)
    VUN(I)=VUN(I)+THRUST(ROW,COL)
600 CONTINUE
500 CONTINUE

C
DO 525 I=1,12
  PUN(I)=0.0
  DO 625 J=1,9
    IF (TABLE(2,I,J) .EQ. 0) GOTO 525
    ROW=TABLE(2,I,J)
    COL=INT((I+1)/2)
    PUN(I)=PUN(I)+THRUST(ROW,COL)
625 CONTINUE

```

525 CONTINUE

C

DO 550 I=1,12

PZIUN(I)=0.0

DO 650 J=1,9

IF (TABLE(3,I,J) .EQ. 0) GOTO 550

ROW=TABLE(3,I,J)

COL=INT((I+1)/2)

PZIUN(I)=PZIUN(I)+THRUST(ROW,COL)

650 CONTINUE

550 CONTINUE

C

DO 575 I=1,12

PZHI(I)=0.0

DO 675 J=1,9

IF (TABLE(4,I,J) .EQ. 0) GOTO 575

ROW=TABLE(4,I,J)

COL=INT((I+1)/2)

PZHI(I)=PZHI(I)+THRUST(ROW,COL)

675 CONTINUE

575 CONTINUE

C

END

APPENDIX E

MAIN PROGRAM LISTING

```

C *****
C   PRODUCED BY THE UNITED STATES AIR FORCE ACADEMY
C   DEPARTMENT OF ASTRONAUTICS AND IS NOT PROTECTED
C   BY COPYRIGHT.          (17 U.S.C. SECTION 105)
C   PROGRAM IS BASED ON WORK BY USAFA/DFAS.
C   AUTOVON 259-4110        COMMERCIAL 303-472-4110
C *****
C
C   EXECUTES THRUST COMMANDS FOR PROXIMITY OPERATIONS
C   DAP PANEL MODE DEFINITIONS CAN BE FOUND IN:
C   FLIGHT PROCEDURES HANDBOOK/PROXIMITY OPERATIONS
C   PRELIMINARY NOV 11, 1982 NASA
C   PAGES 3-16 TO 3-20
C   QUESTIONS ?????? CONTACT CAPT ALFANO, DFAS
C
C   INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
C   REAL X(6),T(3,3),Q(4),JET(4,12),JETSEL(44,9),RT,RVX
C   REAL W9(3),MAXWX,MAXWY,MAXWZ,ALT,AB(3),AR(3),ANG(3)
C   REAL FUEL(3),DELTAT,DVX,DVY,DVZ,DWX,DWY,DWZ,GAS(3),FX(6)
C   REAL DBX,DBY,DBZ,SNAP(3),REF(4),DELT1,DELT2,INCLIN,TIME
C   REAL TW(3,3),TARG(3),AP(4),RANGE,OMEGA,APSTOP,OLDX(3)
C   REAL UP(3),HAFWAY,RRATE,TFUEL,FUDGE,STOPIT
C
C   INTEGER*4 SYSS$SETEF,SYSS$WAITFR,CHAN,COUNT
C   INTEGER*2 I$BUF(337),I$SB(4)
C   INTEGER S(24),C(12),P(12),I,J,PICK,L(24),CO(12),COMM,LITE
C   INTEGER DC(12),NLOOPS,TABLE(4,12,9),PCOUNT,APSW,RESET,FLAG
C   INTEGER PIC,DS(24),DL(24),OVHD(12),IOSTAT,TRKCN,RESTOP
C   CHARACTER*16 TIMBUF
C   INTEGER*4 TIMADR
C
C   AB(3) - ACCELERATION (BODY FRAME)
C   ALT - ALTITUDE OF TARGET (KM)
C   ANG(3) - ANGULAR ACCELERATION (BODY FRAME)
C   APSTOP - RANGE FROM TARGET YOU WISH AUTOPILOT TO STOP
C   APSW - AUTOPILOT SWITCH (S(3)) FROM LAST ITERATION
C   AP(4) - UNIT VECTOR FROM TARGET TO SHUTTLE FOR AUTOPILOT
C   AR(3) - ACCELERATION (REF FRAME)
C   CHAN - CHANNEL FOR PHYSICAL I/O WITH PS300
C   COMM - 1=RHC IN NEUTRAL, 0=RHC NOT IN NEUTRAL
C   COUNT - COUNTER FOR FAST AUTOPILOT GRAPHICS
C   DBX - X DEADBAND FOR ATTITUDE CONTROL
C   DBY - Y DEADBAND FOR ATTITUDE CONTROL
C   DBZ - Z DEADBAND FOR ATTITUDE CONTROL
C   DELT1,2 - TIME COUNTERS TO SEE IF LOOP EXCEEDS .04 SEC
C   DELTAT - TIME STEP
C   DVX - X VELOCITY PULSE
C   DVY - Y VELOCITY PULSE
C   DVZ - Z VELOCITY PULSE
C   DWX - X ROTATIONAL PULSE
C   DWY - Y ROTATIONAL PULSE
C   DWZ - Z ROTATIONAL PULSE
C   FLAG - A PARAMETER TO SLOW DOWN TRANSMISSION RATE TO PS300
C   FUDGE - FUDGE FACTOR TO CHANGE STS RESPONSIVENESS
C   FUEL(3) - TOTAL FUEL EXPENDED FROM EACH TANK
C   FX(6) - PREDICTED FUTURE POS/VEL OF SHUTTLE
C   GAS(3) - FUEL CONSUMED FOR PRESENT ITERATION (FROM EACH TANK)
C   HAFWAY - HALF THE DISTANCE SINCE LAST AUTOPILOT UPDATE
C             (USED FOR BRAKING PURPOSES)

```

C INCLIN - INCLINATION OF TARGET ORBIT (DEG)
 C IBUF - BUFFER FOR ALL MOVEMENT (I/O WITH PS300)
 C IOSB - UNKNOWN PARAMETER (?) FOR I/O WITH PS300
 C IOSTAT - FLAG FOR CHECKING I/O ON UNIT 20
 C JET(4,12) - RESPONSE TO COMMANDS
 C (ROW: 1-VERNIER 2-PRIMARY 3-LOW Z 4-HIGH Z)
 C JETSEL(44,9) - PRECOMPUTED ACCEL AND FUEL FOR EACH THRUSTER
 C LITE - 1=LVLH+3 DISC MODES SELECTED, 0=NOT SELECTED
 C MAXWX - DISC RATE FOR ROLL
 C MAXWY - DISC RATE FOR PITCH
 C MAXWZ - DISC RATE FOR YAW
 C NLOOPS - LOOP COUNTER
 C OLDX(3) - ORIGINAL SHUTTLE POSITION FROM TARGET IN C-W FRAME
 C OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH (RAD/SEC)
 C P(12)- PULSE COMMAND COUNTER
 C PCOUNT - A COUNTER FOR PRINT STATEMENTS
 C PIC - FLAG FOR PS300 GRAPHICS USAGE
 C PICK - NUMBER OF SELECTED RESPONSE MATRIX (JET)
 C Q(4) - QUATERNION
 C RANGE - RANGE FROM TARGET
 C REF(4) - QUATERNION (LVLH TO C-W FRAME)
 C RESET - COMMAND FLAG TO START OVER FROM ORIGINAL POSITION
 C RESTOP - COUNTER TO ALLOW STOPPING OF PROGRAM W/ RESET BUTTON
 C RRATE - RANGE RATE FROM TARGET
 C RT - RENDEZVOUS TIME FOR BANANA AUTOPILOT
 C RVX - FINAL RENDEZVOUS OFFSET DISTANCE (35 FT)
 C SNAP(3) - SNAPSHOT ANGLE (NEEDED FOR DISC RATE MODE)
 C STOPIT - MAXIMUM NUMBER OF MINUTES YOU ANTICIPATE RUNNING
 C T(3,3) - TRANSITION MATRIX (BODY TO REFERENCE FRAME)
 C TABLE(4,12,9) - LIST OF THRUSTERS FIRED VS COMMAND
 C TARG(3) - TARGET POSITION WRT AFT BAY WINDOW
 C TIMBUF,TIMADR - USED TO SET ITERATIVE LOOP TO DELTAT
 C TIME - TIME (SEC)
 C TFUEL - TOTAL FUEL USED
 C TW(3,3) - TRANSITION MATRIX (REF TO WINDOW FRAME)
 C TRKCNT - COUNTER FOR WRITING TO UNIT 20
 C UP(3) - UP VECTOR FOR PS300 AND ALIGNMENT SUBROUTINE
 C WB(3) - ROTATION RATE (BODY FRAME)
 C X(6) - SHUTTLE POSITION/VELOCITY FROM TARGET IN C-W FRAME
 C
 C ASSIGNMENT OF DAP PANEL SWITCHES, LIGHTS, AND THRUST COMMANDS
 C
 C S() - SWITCH ARRAY (READ FROM DAP PANEL)
 C L() - LIGHT ARRAY (WRITTEN TO DAP PANEL)
 C C() - THRUST COMMAND ARRAY
 C OVHD() - COMMAND ARRAY FOR OVERHEAD LOS (READ FROM THC/RHC)
 C CO() - COMMAND ARRAY FROM LAST ITERATION
 C DC() - DUMMY COMMAND ARRAY
 C
 C S(1) - SELECT A / PARAMETERS FOR FAR AWAY MANEUVERING (MAN MODE)
 C / DIRECT AUTOPILOT (AUTO MODE)
 C S(2) - SELECT B / PARAMETERS FOR CLOSE IN MANEUVERING (MAN MODE)
 C / C-W (BANANA) AUTOPILOT (AUTO MODE)
 C
 C S(3) - AUTOPILOT
 C S(4) - MAN / MANUAL
 C S(5) - NORM RCS JETS
 C S(6) - VERNIER RCS JETS
 C S(7) - DISCRETE RATE ROLL
 C S(8) - DISCRETE RATE PITCH
 C S(9) - DISCRETE RATE YAW

```

C      S(10) - ACCEL ROLL
C      S(11) - ACCEL PITCH
C      S(12) - ACCEL YAW
C      S(13) - PULSE ROLL
C      S(14) - PULSE PITCH
C      S(15) - PULSE YAW
C      S(16) - LVLH
C      S(17) - LOW Z Y
C      S(18) - HIGH Z
C      S(19) - NORM X
C      S(20) - NORM Y
C      S(21) - NORM Z
C      S(22) - PULSE X
C      S(23) - PULSE Y
C      S(24) - PULSE Z
C
C      L( ) - SAME AS S ARRAY
C
C      C(1) - +X
C      C(2) - -X
C      C(3) - +Y
C      C(4) - -Y
C      C(5) - +Z
C      C(6) - -Z
C      C(7) - +ROL
C      C(8) - -ROL
C      C(9) - +PCH
C      C(10) - -PCH
C      C(11) - +YAW
C      C(12) - -YAW
C
C      CO( ) - SAME AS C ARRAY
C      DC( ) - SAME AS C ARRAY
C
C      1=ON 0=OFF
C
C      INITIALIZATIONS
C
C      DATA TIMBUF/'0000 00:00:00.12'/
C      DATA (S(I),I=1,24)/24*0/
C      DATA (L(I),I=1,12)/1,0,0,1,1,0,3*0,3*1/
C      DATA (L(I),I=13,24)/6*0,3*1,3*0/
C      DATA (C(I),I=1,12)/12*0/
C      DATA (CO(I),I=1,12)/12*0/
C      DATA (P(I),I=1,12)/12*0/
C      DATA (X(I),I=1,6)/6*0/
C      DELTAT=.04
C      TIME=0.
C      FLAG=0
C
C      SET TIMER
C
C      CALL SYSSBINTIM(TIMBUF,TIMADR)
C
C      READ IN JETSEL, TABLE AND JET MATRICES
C
C      OPEN(UNIT=2,NAME='[ALFANO.DATA]RESPONSES.DAT',
+      TYPE='OLD',FORM='UNFORMATTED',READONLY)
C
C      DO 115 I=1,44

```

```

115 READ(2)(JETSEL(I,J),J=1,9)
C CONTINUE
DO 120 I=1,12
  READ(2)(TABLE(1,I,J),J=1,9)
  READ(2)(TABLE(2,I,J),J=1,9)
  READ(2)(TABLE(3,I,J),J=1,9)
  READ(2)(TABLE(4,I,J),J=1,9)
  READ(2)JET(1,I)
  READ(2)JET(2,I)
  READ(2)JET(3,I)
  READ(2)JET(4,I)
120 CONTINUE
C
C CLOSE(2)
C
C   ECOCHECK DATA THAT WAS READ IN
C
C   CALL ECOCHK(TABLE,JET)
C
C   DETERMINE USER REQUIREMENTS
C
C PRINT *, 'DO YOU WISH : '
C PRINT *, ' 1 - REAL TIME GRAPHIC SIMULATION'
C PRINT *, ' 2 - FAST GRAPHIC AUTOPILOT DEMO'
C PRINT *, ' 3 - FAST AUTOPILOT W/O GRAPHICS'
C READ *, PIC
C
C   SET UP AUTOPILOT, IF CHOSEN
C
C IF ((PIC .EQ. 2).OR.(PIC .EQ. 3)) THEN
  L(1)=0
  L(2)=1
  L(3)=1
  L(4)=0
  APSW=1
  APSTOP=35.
ENDIF
C
C   READ IN FUDGE FACTOR
C
C PRINT *, 'BY WHAT FACTOR DO YOU WISH TO MULTIPLY'
C PRINT *, 'STS RESPONSIVENESS ?'
C READ *, FUDGE
C
C   APPLY FUDGE FACTOR TO THRUSTER DATA
C
C DO 37 I=1,44
  DO 38 J=1,9
    JETSEL(I,J)=JETSEL(I,J)*FUDGE
38 CONTINUE
37 CONTINUE
C
C   READ IN MAX RUN TIME
C
C STOPIT=1000.
C IF (PIC .EQ. 1) THEN
  PRINT *, 'HOW LONG DO YOU ANTICIPATE RUNNING ?'
  PRINT *, '(IF OVER 10 MINUTES PLEASE COORDINATE',
  *      ' WITH SEILER)'

```

```

      READ *,STOPIT
    ENDIF

    C
    C      READ IN TARGET ALTITUDE
    C
    PRINT *, ' ENTER ALTITUDE OF TARGET (KM)'
    READ *,ALT

    C
    C      COMPUTE ROTATION RATE OF TARGET ABOUT EARTH
    C
    OMEGA=SQRT(398600.8/(ALT+6378.135)**3)

    C
    C      READ IN INCLINATION OF TARGET ORBIT
    C
    PRINT *, ' ENTER INCLINATION OF TARGET ORBIT (DEG)'
    READ *,INCLIN

    C
    C      READ IN INITIAL SHUTTLE POSITION
    C
    PRINT *, ' ENTER COORDINATES OF SHUTTLE FROM TARGET'
    PRINT *, ' X,Y,Z CLOHESSY-WILTSHIRE FRAME (FEET)'
    READ *,OLDX(1),CLDX(2),OLDX(3)

    C
    C      ZERO VELOCITY AND ANGULAR MOTION ARE ASSUMED FOR SHUTTLE.
    C      SHUTTLE ALIGNMENT IS INITIALIZED SUCH THAT TARGET
    C      STARTS IN CENTER OF SCREEN (-Z IN BODY FRAME)
    C
    C      INITIALIZE PS300 DISPLAY
    C      INITIALIZE BUFFER FOR PS300 MEMORY ADDRESS
    C
    IF ((PIC .EQ. 1).OR.(PIC .EQ. 2)) THEN
      CALL PS300(ALT,INCLIN)
      CALL INITBUF(IBUF,CHAN,IOSB)
    ENDIF

    C
    C      READ AND WRITE TO SHUTTLE MOCKUP
    C
    IF (PIC .EQ. 1) CALL IOBUFF(1,S,L,OVHD,RESET)

    C
    C      OPEN UNIT 20 FOR WRITING AND SET FORMAT.
    C      IF UNABLE, PRINT MESSAGE AND END PROGRAM
    C
    OPEN(UNIT=20,FILE='STS.TRK',STATUS='NEW',IOSTAT=ISTAT)
    IF (IOSTAT .NE. 0) THEN
      PRINT *, 'CAN NOT OPEN OUTPUT FILE'
      PRINT *, 'FILE = STS.TRK'
      PRINT *, 'ISTAT =', IOSTAT
      GOTO 999
    ENDIF
444  FORMAT('P',4(3X,F8.1),4(3X,F8.6))
445  FORMAT('L',4(3X,F8.1),4(3X,F8.6))

    C
    C      SET EVENT FLAG FOR WAIT COMMAND
    C
    CALL SYS$SETEF(7)

    C
    C      RESET BEGINS HERE
    C
135  CONTINUE
    C

```

```

C      INITIALIZE PARAMETERS
C
      SNAP(1)=0.0
      SNAP(2)=0.0
      SNAP(3)=0.0
      FUEL(1)=0.0
      FUEL(2)=0.0
      FUEL(3)=0.0
      WB(1)=0.0
      WB(2)=0.0
      WB(3)=0.0
      FLAG=0
C
      UP(1)=0.0001
      UP(2)=1.0
      UP(3)=0.0001
C
      X(1)=OLDX(1)
      X(2)=OLDX(2)
      X(3)=OLDX(3)
      X(4)=0.0
      X(5)=0.0
      X(6)=0.0
      RESET=0
C
      DO 175 I=1,12
        C(I)=0
        DC(I)=0
175    CONTINUE
C
      HAFWAY=10000000.
C
      WRITE INITIAL POSITION, TIME, AND QUATERNION
      TO UNIT 20
C
      WRITE(20,444),X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)
C
      ALIGN SHUTTLE SO TARGET IS CENTERED IN WINDOW
C
      CALL ALIGN(X(1),X(2),X(3),T,Q,UP)
C
      INITIALIZE LVLH REF TO ALIGNMENT QUATERNION
C
      REF(1)=Q(1)
      REF(2)=Q(2)
      REF(3)=Q(3)
      REF(4)=Q(4)
C
      INITIALIZE COUNTERS
C
      NLOOPS=0
      PCOUNT=999
      DELT1=SECNDS(0.)
      TRKCNT=C
      COUNT=0
C
      ITERATIVE LOOP BEGINS HERE
C
450    CONTINUE
C

```

```

C      UPDATE COUNTERS
C
NLOOPS=NLOOPS+1
TIME=NLOOPS*DELTAT
PCOUNT=PCOUNT+1
TRKCNT=TRKCNT+1
COUNT=COUNT+1
FLAG=FLAG+1

C
C      PICK APPROPRIATE THRUST RESPONSES.
C      VALUES OF PICK ARE AS FOLLOWS:
C      1 - VERNIER
C      2 - PRIMARY
C      3 - PRIMARY WITH +Z INHIBITED
C      4 - PRIMARY WITH HIGH Z
C
C      CHOOSE THRUST RESPONSES FROM DAP PANEL LIGHTS
C
IF (L(6) .EQ. 1) PICK=1
IF (L(5) .EQ. 1) PICK=2
IF (L(17) .EQ. 1) PICK=3
IF (L(18) .EQ. 1) PICK=4

C
C      SELECT RENDEZVOUS CONSTRAINTS (FROM DAP PANEL LIGHTS)
C
IF ((L(2)+L(3)) .GT. 0) THEN
  DVX=.01
  DUY=.01
  DVZ=.01
  DWX=.000175
  DWY=.000175
  DWZ=.000175
  DBX=.02
  DBY=.02
  DBZ=.02
  MAXWX=.06
  MAXWY=.06
  MAXWZ=.06
ELSE
  DVX=.05
  DUY=.05
  DVZ=.05
  DWX=.000873
  DWY=.000873
  DWZ=.000873
  DBX=.1
  DBY=.1
  DBZ=.1
  MAXWX=.3
  MAXWY=.3
  MAXWZ=.3
ENDIF

C
C      CHECK PULSE MODE LIGHTS
C
IF (L(22) .EQ. 1) CALL PULSE(P(1),P(2),C(1),C(2),DVX,
* DELTAT*JET(PICK,1),DELTAT*JET(PICK,2))
IF (L(23) .EQ. 1) CALL PULSE(P(3),P(4),C(3),C(4),DUY,
* DELTAT*JET(PICK,3),DELTAT*JET(PICK,4))
IF (L(24) .EQ. 1) CALL PULSE(P(5),P(6),C(5),C(6),DVZ,

```

```

* DELTAT*JET(PICK,5),DELTAT*JET(PICK,6))
  IF (L(13) .EQ. 1) CALL PULSE(P(7),P(8),C(7),C(8),DWX,
* DELTAT*JET(PICK,7),DELTAT*JET(PICK,8))
  IF (L(14) .EQ. 1) CALL PULSE(P(9),P(10),C(9),C(10),DWY,
* DELTAT*JET(PICK,9),DELTAT*JET(PICK,10))
  IF (L(15) .EQ. 1) CALL PULSE(P(11),P(12),C(11),C(12),DWZ,
* DELTAT*JET(PICK,11),DELTAT*JET(PICK,12))

```

C
C
C

CHECK LVLH MODE USING COMM AND LITE

```

COMM=0
LITE=0
IF ((C(7)+C(8)+C(9)+C(10)+C(11)+C(12)) .EQ. 0) COMM=1
IF ((L(16)+L(7)+L(8)+L(9)) .EQ. 4) LITE=1
IF ((LITE+COMM) .EQ. 2) THEN
  CALL LVLH(C,CO,REF,DBX,DBY,DBZ,
* WS,JET,DELTAT,MAXWX,MAXWY,MAXWZ,PICK,T)
ENDIF

```

C
C
C

CHECK DISC MODE LIGHTS

```

IF (L(16) .EQ. 0) THEN
  IF (L(7) .EQ. 1) CALL DISC(JET(PICK,7),JET(PICK,8),WB(1),
* DELTAT,MAXWX,C(7),C(8),CO(7),CO(8),DBX,SNAP(1))
  IF (L(8) .EQ. 1) CALL DISC(JET(PICK,9),JET(PICK,10),WB(2),
* DELTAT,MAXWY,C(9),C(10),CO(9),CO(10),DBY,SNAP(2))
  IF (L(9) .EQ. 1) CALL DISC(JET(PICK,11),JET(PICK,12),WB(3),
* DELTAT,MAXWZ,C(11),C(12),CO(11),CO(12),DBZ,SNAP(3))
ENDIF

```

C
C
C
C
C

CHECK IF THE AUTOPILOT (A OR B) IS SELECTED

A - DIRECT APPROACH
B - CLOSED LOOP CLOWESSY-WILTSHIRE

IF (L(3) .EQ. 1) THEN

C
C
C

ZERO OUT THE COMMANDS

```

C(1)=0
C(2)=0
C(3)=0
C(4)=0
C(5)=0
C(6)=0

```

C
C
C

PREDICT POS/VEL PRODUCED BY RHC COMMANDS AND DRIFT

```

FX(1)=X(1)
FX(2)=X(2)
FX(3)=X(3)
FX(4)=X(4)
FX(5)=X(5)
FX(6)=X(6)

```

C
C
C

```

CALL THRUST(JETSEL,AB,ANG,GAS,C,PICK,TABLE)
CALL BTOR(T,AB,AR)
CALL LINTEG(FX,OMEGA,AR,DELTAT)

```

C
C
C

CHOOSE PROPER AUTOPILOT

```

      IF (L(1) .EQ. 1) THEN
        CALL APILOT(L,C,REF,X,FX,T,APSW,
*         S(3),AP,DELTAT,Q,APSTOP,JET,PICK)
      ELSE
        CALL BANANA(FX,OMEGA,C,T,REF,L,
*         S(3),APSW,Q,HAFWAY,RT,DELTAT,UP,JET,PICK,RVX)
      ENDIF
    ENDIF

C
C     IF NO OTHER MODE IS CHOSEN, NORM/ACCEL MODE IS ASSUMED.
C     COMMANDS REMAIN THE SAME, SO NO SUBROUTINE IS NEEDED.
C
C     APPLY COMMANDS (C) TO RESPONSE MATRIX (JETSEL)
C
C     CALL THRUST(JETSEL,AB,ANG,GAS,C,PICK,TABLE)
C
C     TRANSLATE BODY ACCELERATIONS (AB) TO REF FRAME (AR)
C
C     CALL BTOR(T,AB,AR)
C
C     APPLY ACCELERATIONS TO DETERMINE POSITION (X(1-3))
C     AND VELOCITY (X(4-6)) OF SHUTTLE RELATIVE TO TARGET
C     USING CLOHESSY-WILTSHIRE EQUATIONS.
C
C     CALL LINTEG(X,OMEGA,AR,DELTAT)
C
C     APPLY ANGULAR ACCELERATIONS TO FIGURE QUATERNIONS AND NEW T MAT
C
C     CALL ROTATE(ANG,DELTAT,WB,Q,T)
C
C     UPDATE FUEL USED
C
C     DO 900 I=1,3
C       FUEL(I)=FUEL(I)+GAS(I)*DELTAT
900  CONTINUE
C     TFUEL=FUEL(1)+FUEL(2)+FUEL(3)
C
C     DETERMINE TARGET POSITION AND ATTITUDE
C     AS SEEN THRU AFT BAY WINDOW.
C     FOR REAL SIMULATION, DISPLAY ON THE PS300.
C
C     IF (PIC .EQ. 1)
C       + CALL LOOK(T,X,OMEGA,TIME,TFUEL,IBUF,CHAN,IOSB)
C
C     IF ((PIC .EQ. 2).AND.(COUNT .EQ. 13)) THEN
C       CALL LOOK(T,X,OMEGA,TIME,TFUEL,IBUF,CHAN,IOSB)
C       COUNT=0
C     ENDIF
C
C     WRITE POSITION, TIME, AND ATTITUDE DATA
C     TO UNIT 20 EVERY 10 SECONDS.
C
C     IF (TRKCNT .GE. 250) THEN
C       WRITE(20,445),X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)
C       TRKCNT=0
C     ENDIF
C
C     UPDATE OLD COMMAND ARRAY (CO)
C
C     DO 950 I=1,12

```

```

          CO(I)=DC(I)
950      CONTINUE
C
C          UPDATE AUTOPILOT SWITCH
C
          APSW=S(3)
C
C          PRINT TO SCREEN EVERY 1000 ITERATIONS IF PIC=3
C
          IF ((PCOUNT .GE. 1000) .AND. (PIC .EQ. 3)) THEN
              PCOUNT=0
              PRINT *, ' RANGE (FT)    RANGE RATE (FPS)',
                *      ' FUEL USED (LBS)  TIME (SEC)'
              RANGE=SQRT(X(1)**2+X(2)**2+X(3)**2)
              RRATE=(X(1)*X(4)+X(2)*X(5)+X(3)*X(6))/RANGE
              PRINT *, RANGE,RRATE,(FUEL(1)+FUEL(2)+FUEL(3)),TIME
          ENDIF
C
C          ZERO OUT SWITCHES AND RESTORE THRUST COMMANDS,
C          SET MANUAL LIGHT FLASHING IF AUTOPILOT ON
C
          DO 645 I=1,12
              S(I)=0
              S(I+12)=0
              C(I)=DC(I)
645      CONTINUE
C
          IF ((L(3).EQ.1).AND.((TIME-INT(TIME)).GT..5)) L(4)=1
C
C          READ SWITCHES AND THRUST COMMANDS FROM SHUTTLE MOCKUP.
C          WRITE TO LIGHTS ON DAP PANEL OF SHUTTLE MOCKUP.
C          IF RESET BUTTON HELD FOR ONE SECOND, STOP PROGRAM
C          IF RESET COMMANDED, GO TO 135
C
          IF ((PIC .EQ. 1).AND.(FLAG .GE. 3)) THEN
              CALL IOBUFF(2,S,L,OVHD,RESET)
              CALL SYSSWAITFR(7)
              CALL SYSSSETIMR(7,TIMADR,,)
              IF (RESET .EQ. 1) THEN
                  RESTOP=RESTOP+1
                  IF (RESTOP .EQ. 15) GOTO 999
                  GOTO 135
              ELSE
                  RESTOP=0
              ENDIF
              FLAG=0
C
C          CHANGE INPUT TO REFLECT OVERHEAD LINE OF SIGHT
C
          DC(1)=OVHD(6)
          DC(2)=OVHD(5)
          DC(3)=OVHD(3)
          DC(4)=OVHD(4)
          DC(5)=OVHD(1)
          DC(6)=OVHD(2)
          DC(7)=OVHD(12)
          DC(8)=OVHD(11)
          DC(9)=OVHD(9)
          DC(10)=OVHD(10)
          DC(11)=OVHD(7)

```

```

      DC(12)=OVMD(8)
C
C      ASSIGN REAL COMMAND ARRAY (C) WITH DUMMY COMMANDED VALUES
C      THIS IS TO PRESERVE DATA READ FROM THC/RHC WHILE PROGRAM
C      CHANGES C ARRAY DUE TO VARIOUS SELECTED MODES
C
      DO 475 I=1,12
        C(I)=DC(I)
475    CONTINUE
C
      ENDIF
C
      IF AUTOPILOT ON, SET L(4) (MANUAL LIGHT) TO ZERO
C
      IF (L(3) .EQ. 1) L(4)=0
C
      APPLY SWITCH COMMANDS TO DAP PANEL LIGHTS
C
      CALL SWITCH(S,L)
C
C
      RETURN TO 450 FOR NEXT ITERATION
C
      IF (NLOOPS .LT. (1500*STOPIT)) GOTO 450
C
      COMPUTE AVERAGE LOOP TIME
C
      DELT2=SECNDS(DELT1)
      PRINT *, ' AVERAGE LOOP TIME = ',DELT2/NLOOPS,' SECONDS'
C
      CLOSE UNIT 20
C
      CLOSE(20)
C
999    CONTINUE
C
      END

```

SUBROUTINE ALIGN(X,Y,Z,T,Q,UP)

COMPUTES BODY AXIS COMPONENTS IN REFERENCE FRAME.
ZBOD IS THE POSITION VECTOR. XBOD AND YBOD
COMPLETE THE RIGHT HANDED SYSTEM.
DETERMINES QUATERNION (Q) AND TRANSFORMATION
MATRIX (T).

REAL X,Y,Z,T(3,3),Q(4),UP(3),CONST
REAL XBOD(3),YBOD(3),ZBOD(3)

CAMERA VECTOR BODY COMPONENTS (0,0,-1)

ZBOD(1)=X
ZBOD(2)=Y
ZBOD(3)=Z

DO NOT ALLOW ZBOD TO LIE DIRECTLY ON AN AXIS.
THIS IS DONE TO PREVENT QUATERNION AMBIGUITIES.

IF (ABS(ZBOD(1)) .LT. .001) ZBOD(1)=.001
IF (ABS(ZBOD(2)) .LT. .001) ZBOD(2)=.001
IF (ABS(ZBOD(3)) .LT. .001) ZBOD(3)=.001

COMPUTE TRANSFORMATION MATRIX

CALL UNITIZE(ZBOD)
CALL CROSS(YBOD,ZBOD,UP)
CALL UNITIZE(YBOD)
CALL CROSS(XBOD,YBOD,ZBOD)
CALL UNITIZE(XBOD)

UP(1)=XBOD(1)
UP(2)=XBOD(2)
UP(3)=XBOD(3)

T(1,1)=XBOD(1)
T(1,2)=YBOD(1)
T(1,3)=ZBOD(1)
T(2,1)=XBOD(2)
T(2,2)=YBOD(2)
T(2,3)=ZBOD(2)
T(3,1)=XBOD(3)
T(3,2)=YBOD(3)
T(3,3)=ZBOD(3)

COMPUTE QUATERNIONS

Q(4)=1.0+T(1,1)+T(2,2)+T(3,3)
IF (Q(4) .LT. .1E-30) Q(4)=.1E-30
Q(4)=SQRT(Q(4))/2.0
Q(1)=(T(3,2)-T(2,3))/(4.0*Q(4))
Q(2)=(T(1,3)-T(3,1))/(4.0*Q(4))
Q(3)=(T(2,1)-T(1,2))/(4.0*Q(4))

NORMALIZE QUATERNIONS

CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
Q(1)=Q(1)/CONST
Q(2)=Q(2)/CONST

Q(3)=Q(3)/CONST
Q(4)=Q(4)/CONST

C

END

```

SUBROUTINE APILOT(L,C,REF,X,FX,T,OLDSW,NEWSW,
* AP,DELTAT,Q,APSTOP,JET,PICK)

```

```

THIS AUTOPILOT FLIES THE SHUTTLE DIRECTLY TO
TARGET (DIRECT APPROACH) USING LVLH FOR ATTITUDE
CONTROL. IT 'THINKS' ONE ITERATION IN ADVANCE
USING PREDICTED POS/VEL DUE TO RHC THRUSTING
AND ORBITAL DRIFT.

```

```

REAL REF(4),X(6),MAXD,DELTAV(3),VDB,DELTAT
REAL AP(4),DP(3),Q(4),APSTOP,FX(6),JET(4,12)
REAL DELVB(3),T(3,3),RANGE,REQV(3)
INTEGER L(24),C(12),I,OLDSW,NEWSW,PICK

```

```

REF - REFERENCE QUATERNION FOR ATTITUDE CONTROL
X - PRESENT POS AND VEL OF SHUTTLE FROM TARGET
FX - FUTURE POS AND VEL OF SHUTTLE FROM TARGET
MAXD - MAX DISTANCE CHANGE FOR ONE ITERATION
DELTAV - VELOCITY CHANGE (TARGET FRAME)
DELTAT - TIME INCREMENT
VDB - VELOCITY DEADBAND
AP - REFERENCE UNIT VECTOR FROM TARGET TO SHUTTLE
DP - DESIRED POSITION
DELVB - VELOCITY CHANGE (BODY FRAME)
T - TRANSFORMATION MATRIX
L - DAP PANEL LIGHTS
C - TWC/RHC COMMANDS
OLDSW - LAST POSITION OF AUTOPILOT SELECT SWITCH
NEWSW - PRESENT POSITION OF AUTOPILOT SELECT SWITCH
APSTOP - RANGE YOU WISH TO MAINTAIN FROM TARGET
REQV - REQUIRED VELOCITY
Q - PRESENT QUATERNION

```

```

COMPUTE RANGE

```

```

RANGE=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))

```

```

INITIALIZE LIGHTS AND COMMANDS

```

```

DO 100 I=1,12

```

```

L(I)=0

```

```

L(I+12)=0

```

```

100 CONTINUE

```

```

L(1)=1

```

```

L(3)=1

```

```

L(5)=1

```

```

L(7)=1

```

```

L(8)=1

```

```

L(9)=1

```

```

L(16)=1

```

```

L(20)=1

```

```

COMPUTE MAX DISTANCE CHANGE

```

```

MAXD=.005*(RANGE)*DELTAT

```

```

IF (RANGE .LT. 1.5*APSTOP) THEN

```

```

MAXD=MAXD*(RANGE-APSTOP)/(.5*APSTOP)

```

```

ENDIF

```

```

SWITCH TO LOW Z INSIDE 200 FEET

```

```

C      IF (RANGE .GE. 200.0) THEN
C          L(21)=1
C      ELSE
C          L(17)=1
C      ENDIF

C      CHECK TO SEE IF AUTOPILOT JUST SELECTED
C
C      IF (OLDSW .NE. NEWSW) THEN
C          SET REF TO PRESENT QUATERNION
C          (TO BE USED BY LVLH)
C
C          REF(1)=Q(1)
C          REF(2)=Q(2)
C          REF(3)=Q(3)
C          REF(4)=Q(4)
C
C          SET REFERENCE VECTOR
C
C          AP(1)=X(1)/RANGE
C          AP(2)=X(2)/RANGE
C          AP(3)=X(3)/RANGE
C          AP(4)=RANGE+1.0
C
C      ENDIF
C
C      FIGURE DESIRED FUTURE POSITION (DP) (+2 ITERATIONS)
C
C      IF (ABS(RANGE-APSTOP) .LT. ABS(AP(4)-APSTOP)) THEN
C          AP(4)=RANGE
C      ELSE
C          MAXD=0.0
C      ENDIF
C
C      DP(1)=(AP(4)-2.0*MAXD)*AP(1)
C      DP(2)=(AP(4)-2.0*MAXD)*AP(2)
C      DP(3)=(AP(4)-2.0*MAXD)*AP(3)
C
C      COMPUTE REQUIRED VELOCITY (NEXT ITERATION)
C
C      REQV(1)=(DP(1)-FX(1))/DELTAT
C      REQV(2)=(DP(2)-FX(2))/DELTAT
C      REQV(3)=(DP(3)-FX(3))/DELTAT
C
C      FIGURE VELOCITY TO BE GAINED (DELTAV, TARGET FRAME)
C
C      DELTAV(1)=REQV(1)-FX(4)
C      DELTAV(2)=REQV(2)-FX(5)
C      DELTAV(3)=REQV(3)-FX(6)
C
C      TRANSFORM TO BODY FRAME
C
C      CALL RTOB(T,DELTAV,DELVB)
C
C      DETERMINE IF THRUST IS REQUIRED
C      (VDB IS VELOCITY DEADBAND)
C
C      VDB=1.1*DELTAT

```

C

```
IF (DELVB(1) .GT. VDB*JET(PICK,1)) C(1)=1
IF (DELVB(1) .LT. VDB*JET(PICK,2)) C(2)=1
IF (DELVB(2) .GT. VDB*JET(PICK,3)) C(3)=1
IF (DELVB(2) .LT. VDB*JET(PICK,4)) C(4)=1
IF (DELVB(3) .GT. VDB*JET(PICK,5)) C(5)=1
IF (DELVB(3) .LT. VDB*JET(PICK,6)) C(6)=1
```

C

END

SUBROUTINE ARM

THIS ROUTINE BUILDS THE PAYLOAD ARM ON THE PS300
ALL DIMENSIONS ARE IN FEET.

INCLUDE 'PROCONST.FOR/MOLIST'

REAL*4 V(3)

COMMANDS FOR WRIST ROLL

SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION

CALL PFN('WRROLL','YROTATE',ERR)

CALL PSNST('WR ROLL',1,'DLABEL1',ERR)
CALL PCONN('DIALS',1,1,'WRROLL',ERR)

CONNECT INPUTS TO ACCUMULATOR

CALL PFN('ACC1','ACCUMULATE',ERR)

CALL PCONN('WRROLL',1,1,'ACC1',ERR)
CALL PSNREA(0.0,2,'ACC1',ERR)
CALL PSNREA(0.1,3,'ACC1',ERR)
CALL PSNREA(1.0,4,'ACC1',ERR)
CALL PSNREA(447.0,5,'ACC1',ERR)
CALL PSNREA(-447.0,6,'ACC1',ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM

CALL PCONN('ACC1',1,1,'WRIST.ROLL',ERR)

COMMANDS FOR WRIST YAW

SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION

CALL PFN('WRYAW','ZROTATE',ERR)

CALL PSNST('WR YAW',1,'DLABEL2',ERR)
CALL PCONN('DIALS',2,1,'WRYAW',ERR)

CONNECT INPUTS TO ACCUMULATOR

CALL PFN('ACC2','ACCUMULATE',ERR)

CALL PCONN('WRYAW',1,1,'ACC2',ERR)
CALL PSNREA(0.0,2,'ACC2',ERR)
CALL PSNREA(0.1,3,'ACC2',ERR)
CALL PSNREA(1.0,4,'ACC2',ERR)
CALL PSNREA(120.0,5,'ACC2',ERR)
CALL PSNREA(-120.0,6,'ACC2',ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM

CALL PCONN('ACC2',1,1,'WRIST.YAW',ERR)

COMMANDS FOR WRIST PITCH

```

C
C      SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
C      CALL PFN('WRPITCH','XROTATE',ERR)
C
C      CALL PSNST('WR PITCH',1,'DLABEL3',ERR)
C      CALL PCONN('DIALS',3,1,'WRPITCH',ERR)
C
C      CONNECT INPUTS TO ACCUMULATOR
C
C      CALL PFN('ACC3','ACCUMULATE',ERR)
C
C      CALL PCONN('WRPITCH',1,1,'ACC3',ERR)
C      CALL PSNREA(0.0,2,'ACC3',ERR)
C      CALL PSNREA(0.1,3,'ACC3',ERR)
C      CALL PSNREA(1.0,4,'ACC3',ERR)
C      CALL PSNREA(120.0,5,'ACC3',ERR)
C      CALL PSNREA(-120.0,6,'ACC3',ERR)
C
C      CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
C      CALL PCONN('ACC3',1,1,'WRIST.PITCH',ERR)
C
C  COMMANDS FOR ELBOW PITCH
C
C      SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
C      CALL PFN('ELPITCH','XROTATE',ERR)
C
C      CALL PSNST('EL PITCH',1,'DLABEL4',ERR)
C      CALL PCONN('DIALS',4,1,'ELPITCH',ERR)
C
C      CONNECT INPUTS TO ACCUMULATOR
C
C      CALL PFN('ACC4','ACCUMULATE',ERR)
C
C      CALL PCONN('ELPITCH',1,1,'ACC4',ERR)
C      CALL PSNREA(0.0,2,'ACC4',ERR)
C      CALL PSNREA(0.1,3,'ACC4',ERR)
C      CALL PSNREA(1.0,4,'ACC4',ERR)
C      CALL PSNREA(160.0,5,'ACC4',ERR)
C      CALL PSNREA(-2.0,6,'ACC4',ERR)
C
C      CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
C      CALL PCONN('ACC4',1,1,'ELBOW.PITCH',ERR)
C
C  COMMANDS FOR SHOULDER YAW
C
C      SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
C      CALL PFN('SHYAW','ZROTATE',ERR)
C
C      CALL PSNST('SH YAW',1,'DLABEL5',ERR)
C      CALL PCONN('DIALS',5,1,'SHYAW',ERR)
C
C      CONNECT INPUTS TO ACCUMULATOR
C
C      CALL PFN('ACC5','ACCUMULATE',ERR)

```

```

C      CALL PCONN('SHYAW',1,1,'ACC5',ERR)
C      CALL PSNREA(0.0,2,'ACC5',ERR)
C      CALL PSNREA(0.1,3,'ACC5',ERR)
C      CALL PSNREA(1.0,4,'ACC5',ERR)
C      CALL PSNREA(180.0,5,'ACC5',ERR)
C      CALL PSNREA(-180.0,6,'ACC5',ERR)
C
C      CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
C      CALL PCONN('ACC5',1,1,'SHOULDER.YAW',ERR)
C
C      COMMANDS FOR SHOULDER PITCH
C
C      SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
C
C      CALL PFN('SHPITCH','XROTATE',ERR)
C
C      CALL PSNST('SH PITCH',1,'DLABEL6',ERR)
C      CALL PCONN('DIALS',6,1,'SHPITCH',ERR)
C
C      CONNECT INPUTS TO ACCUMULATOR
C
C      CALL PFN('ACC6','ACCUMULATE',ERR)
C
C      CALL PCONN('SHPITCH',1,1,'ACC6',ERR)
C      CALL PSNREA(0.0,2,'ACC6',ERR)
C      CALL PSNREA(0.1,3,'ACC6',ERR)
C      CALL PSNREA(1.0,4,'ACC6',ERR)
C      CALL PSNREA(2.0,5,'ACC6',ERR)
C      CALL PSNREA(-145.0,6,'ACC6',ERR)
C
C      CONNECT ACCUMULATOR OUTPUT TO PROGRAM
C
C      CALL PCONN('ACC6',1,1,'SHOULDER.PITCH',ERR)
C
C      CREATE A 3D VECTOR (IN FEET) TO TRANSLATE ARM
C
C      CALL PFN('XYVEC','VEC',ERR)
C      CALL PFN('XYZVEC','VEC',ERR)
C      CALL PCONN('XYVEC',1,1,'XYZVEC',ERR)
C      CALL PCONN('XYZVEC',1,1,'SHOULDER.TRAN',ERR)
C
C      BUILD A WRIST
C
C      CALL PBEGS('WRIST',ERR)
C      CALL PROTX('PITCH',0.0,'',ERR)
C      V(1)=.5415
C      V(2)=1.23
C      V(3)=.5415
C      CALL PSCALE('','V','CYLINDER',ERR)
C      V(1)=0.0
C      V(2)=1.23
C      V(3)=0.0
C      CALL PTRANS('','V','',ERR)
C      CALL PROTZ('YAW',0.0,'',ERR)
C      CALL PROTY('RCLL',0.0,'',ERR)
C      V(1)=.5415
C      V(2)=4.93

```

```

      V(3)=.5415
      CALL PSCALE('','V','CYLINDER',ERR)
      CALL PENDS(ERR)
C
C   BUILD A FOREARM (W/ ELBOW)
C
      CALL PBEGS('ELBOW',ERR)
      V(1)=0.0
      V(2)=0.0
      V(3)=.5415
      CALL PTRANS('','V','',ERR)
      CALL PROTX('PITCH',0.0,'',ERR)
      V(1)=0.0
      V(2)=0.0
      V(3)=-.5415
      CALL PTRANS('','V','',ERR)
      V(1)=.5415
      V(2)=23.1625
      V(3)=.5415
      CALL PSCALE('','V','CYLINDER',ERR)
      V(1)=0.0
      V(2)=23.1625
      V(3)=0.0
      CALL PTRANS('','V','WRIST',ERR)
      CALL PENDS(ERR)
C
C   BUILD THE SHOULDER
C
      CALL PBEGS('SHOULDER',ERR)
      V(1)=0.0
      V(2)=0.0
      V(3)=0.0
      CALL PTRANS('TRAN',V,'',ERR)
      CALL PROTZ('','90.0','CYL',ERR)
      V(1)=.58
      V(2)=1.083
      V(3)=.58
      CALL PSCALE('CYL',V,'CYLINDER',ERR)
      CALL PROTX('','90.0','',ERR)
      CALL PROTX('PITCH',0.0,'',ERR)
      CALL PROTZ('YAW',0.0,'',ERR)
      V(1)=.5415
      V(2)=20.921
      V(3)=.5415
      CALL PSCALE('','V','CYLINDER',ERR)
      V(1)=0.0
      V(2)=20.921
      V(3)=0.0
      CALL PTRANS('','V','ELBOW',ERR)
      CALL PENDS(ERR)
C
C   RESCALE SHOULDER FROM FEET TO KM AND NAME IT 'ARM'
C
      V(1)=.0003048
      V(2)=.0003048
      V(3)=.0003048
      CALL PSCALE('ARM',V,'SHOULDER',ERR)
C
      END

```

```

SUBROUTINE BANANA(X,OMEGA,C,T,REF,L,
*NEWSW,OLDSW,Q,HAFWAY,RT,DELTAT,UP,JET,PICK,RVX)

```

```

C      THIS IS AN EXPLICIT GUIDANCE SCHEME.
C      IT SOLVES THE CLOHESSEY-WILTSHIRE EQUATIONS FOR
C      VELOCITY REQUIRED (VX,VY,VZ, TARGET FRAME) GIVEN
C      A SPECIFIED RENDEZVOUS TIME (RT). VELOCITY TO
C      BE GAINED (DELTAV) IS THEN COMPUTED AND TRANSFORMED
C      TO BODY FRAME (DELV8). IF OUTSIDE THE VELOCITY
C      DEADBAND (VDB), THRUST IS COMMANDED.
C      LVLH IS USED TO MAINTAIN ATTITUDE WITH REPEATED
C      CALLS TO 'ALIGN'.

```

```

C      REAL X(6),OMEGA,REF(4),DUMT(3,3),VX,VY
C      REAL T(3,3),DELTAV(3),DELV8(3),A,D,E,F
C      REAL RANGE,B,VZ,Q(4),HAFWAY,RT,DELTAT,UP(3)
C      REAL POS(3),BPCS(3),JET(4,12),RVX

```

```

C      INTEGER C(12),I,L(24),NEWSW,OLDSW,PICK

```

```

C      L - DAP PANEL LIGHTS
C      T - TRANSFORMATION MATRIX
C      DUMT - DUMMY TRANSFORMATION MATRIX
C      REF - REFERENCE QUATERNION FOR LVLH MODE
C      OMEGA - ROTATION RATE OF TARGET ABOUT EARTH
C      C - COMMAND SWITCHES
C      DELTAV - VEL CHANGE (TARGET FRAME)
C      DELV8 - VEL CHANGE (BOD FRAME)
C      RANGE - RANGE FROM TARGET
C      RT - RENDEZVOUS TIME (SECONDS)
C      A,B,D,E,F - INTERMEDIATE VARIABLES
C      VX,VY,VZ - VELOCITY NEEDED
C      NEWSW - PRESENT CONDITION OF S(3)
C      OLDSW - LAST CONDITION OF S(3)
C      HAFWAY - HALF DISTANCE TO TARGET
C      X(6) - FUTURE PREDICTED POS/VEL
C      DELTAT - TIME INCREMENT
C      UP - UP VECTOR (FOR ALIGNMENT)
C      JET - ACCELERATION AVAILABLE (F/S**2)
C      PICK - THRUST PROFILE SELECTED
C      RVX - FINAL RENDEZVOUS OFFSET DISTANCE

```

```

C      INITIALIZATIONS

```

```

C      RANGE=SQRT(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))

```

```

C      DO 100 I=1,12
C          L(I)=0
C          L(I+12)=0

```

```

100    CONTINUE

```

```

C      SET DAP PANEL LIGHTS

```

```

C      L(2)=1
C      L(3)=1
C      L(5)=1
C      L(7)=1
C      L(8)=1
C      L(9)=1
C      L(16)=1

```

L(20)=1

C
C
C

USE LOW Z FROM 60-200 FEET

IF ((RANGE.LE.200.0).AND.(RANGE.GE.60.)) THEN

L(17)=1

ELSE

L(21)=1

ENDIF

C
C
C

UPDATE TIME TO RENDEZVOUS

RT=RT-DELTAT

C
C
C

CHECK TO SEE IF AUTOPILOT JUST SELECTED

IF (OLDSW .NE. NEWSW) THEN

HAFWAY=1000000.

RVX=0.

ENDIF

C
C
C
C

CHECK TO SEE IF YOU ARE 'HALFWAY' THERE
(ACTUALLY 4/5 OF THE WAY THERE)

IF (RANGE .LT. HAFWAY) THEN

RT=4.*RANGE

IF (RT .GT. 1200.) RT=1200.

IF (RANGE .GE. 250.) THEN

HAFWAY=250.

ELSE

HAFWAY=(RANGE-35.)/5.+35.

ENDIF

IF (X(1) .LT. 0.0) THEN

RVX=+35.

ELSE

RVX=-35.

ENDIF

ENDIF

C

IF (RT .LT. 0.0) RT=100.

C
C
C

COMPUTE NEW REFERENCE QUATERNION BASED ON POSITION

CALL ALIGN(X(1),X(2),X(3),DUMT,REF,UP)

C
C
C
C

CHANGE X(1) TO MISS HITTING TARGET BY
RVX DISANCE FEET

X(1)=X(1)+RVX

C
C
C

COMPUTE DELTAV (TARGET FRAME)

SW=SIN(OMEGA*RT)

CW=COS(OMEGA*RT)

C

A=-3.0*RT+4.0*SW/OMEGA

B=(2.0/OMEGA)*(CW-1.0)

F=X(1)-6.0*OMEGA*X(2)*RT+6.0*X(2)*SW

D=SW/OMEGA

E=4.0*X(2)-3.0*X(2)*CW

```

C      VY=(-E-F*B/A)/(B*B/A+D)
      VX=(-B*VY-F)/A
      VZ=-X(3)*OMEGA*CW/SW
C
      DELTAV(1)=VX-X(4)
      DELTAV(2)=VY-X(5)
      DELTAV(3)=VZ-X(6)
C
      TRANSFORM TO BODY FRAME
C
      CALL RTOB(T,DELTAV,DELV8)
C
      DETERMINE IF THRUST IS REQUIRED
      VDB IS VELOCITY DEADBAND
C
      VDB=1.1*DELTAT
C
      IF (DELV8(1) .GT. VDB*JET(PICK,1)) C(1)=1
      IF (DELV8(1) .LT. VDB*JET(PICK,2)) C(2)=1
      IF (DELV8(2) .GT. VDB*JET(PICK,3)) C(3)=1
      IF (DELV8(2) .LT. VDB*JET(PICK,4)) C(4)=1
      IF (DELV8(3) .GT. VDB*JET(PICK,5)) C(5)=1
      IF (DELV8(3) .LT. VDB*JET(PICK,6)) C(6)=1
C
      END

```

SUBROUTINE BTOR(T,BOD,REF)

TRANSFORMS FROM BOD TO REF FRAME GIVEN TRANSFORMATION MATRIX T

REAL T(3,3),BOD(3),REF(3)

REF(1)=BOD(1)*T(1,1)+BOD(2)*T(1,2)+BOD(3)*T(1,3)

REF(2)=BOD(1)*T(2,1)+BOD(2)*T(2,2)+BOD(3)*T(2,3)

REF(3)=BOD(1)*T(3,1)+BOD(2)*T(3,2)+BOD(3)*T(3,3)

END

SUBROUTINE CHECK(ANG,DB,WB,C1,C2,JET1,JET2,MAX,DELTAT)

THIS SUBROUTINE EXAMINES ANGLE (ANG) AND BODY RATE (WB)
TO DETERMINE THRUST REQUIRED TO RESTORE ANGLE WITHIN
DEADBAND (DB) WHILE NOT EXCEEDING MAX BODY RATE (MAX)
A WINDOW IS SET UP (SA) TO STOP ROTATION. IF INSIDE WINDOW
AND APPROACHING ZERO, THRUST TO STOP.

REAL ANG,DB,WB,JET1,JET2,MAX,DELTAT,SA
INTEGER C1,C2,N

DELTAT - TIME INCREMENT
SA - STOP ANGLE (ANGLE REQUIRED TO MAKE WB=0)
N - COUNTER FOR STOP ANGLE SOLUTION
JET - ANGULAR ACCELERATION AVAILABLE

INITIALIZATIONS

C1=0
C2=0

CHECK IF OUTSIDE DEADBAND

IF (ABS(ANG) .GT. DB) THEN

CHECK IF WB IS GOING AWAY FROM SNAPSHOT AND CHOOSE THRUST

IF (ANG*WB .GE. 0.0) THEN
IF (ANG .GT. 0.0) C2=1
IF (ANG .LT. 0.0) C1=1
ELSE

ELSE WB IS MOVING TOWARDS SNAPSHOT. DETERMINE STOP
ANGLE (SA) AND CHOOSE THRUST.

IF (ANG .GT. 0.0) THEN
SA=ABS(.5*WB*WB/JET1)+DB*.9
N=INT(-WB/(JET2*DELTAT))
SA=ABS(N*WB+JET2*DELTAT*N*(N-1)/2.0)*DELTAT+.9*DB
IF (ANG .LE. SA) C1=1
SA=ABS(.5*(WB+JET2*DELTAT)**2/JET1)
SA=SA+.9*DB+WB*DELTAT
IF (ANG .GT. SA) C2=1
ELSE

SA=ABS(.5*WB*WB/JET2)+DB*.9
N=INT(-WB/(JET1*DELTAT))
SA=ABS(N*WB+JET1*DELTAT*N*(N-1)/2.0)*DELTAT+.9*DB
IF (ABS(ANG) .LE. SA) C2=1
SA=ABS(.5*(WB+JET1*DELTAT)**2/JET2)
SA=SA+.9*DB+WB*DELTAT
IF (ABS(ANG) .GT. SA) C1=1
ENDIF
ENDIF
ENDIF

COMPARE RATES TO MAX RATE AND SET THRUST COMMANDS

IF (WB .GT. 0.0) THEN
IF ((WB+DELTAT*JET1) .GT. MAX) C1=0
IF (WB .GT. MAX) C2=1

```
ELSE  
  IF ((WB+DELTAT*JET2) .LT. -MAX) C2=0  
  IF (WB .LT. -MAX) C1=1  
ENDIF
```

C

END

```
C      SUBROUTINE CROSS(A,B,C)
C      COMPUTES  A = B X C
C      REAL A(3),B(3),C(3)
C      A(1)=B(2)*C(3)-B(3)*C(2)
      A(2)=B(3)*C(1)-B(1)*C(3)
      A(3)=B(1)*C(2)-B(2)*C(1)
C
      END
```

SUBROUTINE DISC(JET1,JET2,WB,DELTAT,MAX,C1,C2,CO1,CO2,DB,ANG)

ATTITUDE CONTROL SCHEME FOR ONE BODY AXIS. ALSO
APPLIES DISCRETE RATE (MAX) TO PRESENT BODY RATE (WB).
IF DISC RATE WILL BE (OR IS BEING) EXCEEDED, COMMAND (C1,C2)
IS SET TO BRING/MAINTAIN RATE WITHIN LIMITS.

REAL JET1,JET2,DELTAT,MAX,WB,T,DB,SA,ANG
INTEGER C1,C2,CO1,CO2

ANG - ANGLE FROM WHEN SNAPSHOT TAKEN
SNAP - TRIGGER TO TAKE NEW SNAPSHOT
CO1,2 - OLD COMMAND FROM RHC
C1,2 - PRESENT COMMAND FROM RHC
DB - DEADBAND WIDTH (+/-)
JET1,2 - ANGULAR ACCELERATION AVAILABLE
WB - BODY RATE (FOR SELECTED AXIS)

COMPUTE PRESENT ANGLE FROM SNAPSHOT ANGLE

ANG=ANG+WB*DELTAT

TAKE SNAPSHOT (SNAP=1) IF CONTROLLER HAS JUST BEEN MOVED TO NEUTRAL

SNAP=0

IF ((C1+C2) .EQ. 0) THEN
IF ((CO1-C1) .GT. 0) SNAP=1
IF ((CO2-C2) .GT. 0) SNAP=1
IF (SNAP .EQ. 1) ANG=0.0

DETERMINE THRUST PROFILE

CALL CHECK(ANG,DB,WB,C1,C2,JET1,JET2,
MAX,DELTAT)

ELSE

COMPARE RATES TO MAX RATE AND SET THRUST COMMANDS

IF (WB .GT. 0.0) THEN
IF ((WB+DELTAT*JET1) .GT. MAX) C1=0
IF (WB .GT. MAX) C2=1
ELSE
IF ((WB+DELTAT*JET2) .LT. -MAX) C2=0
IF (WB .LT. -MAX) C1=1

ENDIF

ENDIF

END

SUBROUTINE ECOCHK(TABLE,JET)

ECOCHK OF TABLE AND JET

REAL JET(4,12)

INTEGER I,J, TABLE(4,12,9)

175 FORMAT(1X,A5,3X,9(1X,I2),5X,F8.5)

```

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT *,' ECOCHK OF VERNIER THRUSTER TABLE AND ACCEL'
PRINT *,' '
PRINT 175,' + X',(TABLE(1,1,J),J=1,9),JET(1,1)
PRINT 175,' - X',(TABLE(1,2,J),J=1,9),JET(1,2)
PRINT 175,' + Y',(TABLE(1,3,J),J=1,9),JET(1,3)
PRINT *,' '
PRINT 175,' - Y',(TABLE(1,4,J),J=1,9),JET(1,4)
PRINT 175,' + Z',(TABLE(1,5,J),J=1,9),JET(1,5)
PRINT 175,' - Z',(TABLE(1,6,J),J=1,9),JET(1,6)
PRINT *,' '
PRINT 175,' + ROL',(TABLE(1,7,J),J=1,9),JET(1,7)
PRINT 175,' - RCL',(TABLE(1,8,J),J=1,9),JET(1,8)
PRINT 175,' + PCH',(TABLE(1,9,J),J=1,9),JET(1,9)
PRINT *,' '
PRINT 175,' - PCH',(TABLE(1,10,J),J=1,9),JET(1,10)
PRINT 175,' + YAW',(TABLE(1,11,J),J=1,9),JET(1,11)
PRINT 175,' - YAW',(TABLE(1,12,J),J=1,9),JET(1,12)
PRINT *,' '
PRINT *,' HIT RETURN TO CONTINUE'
READ *
```

```

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT *,' ECOCHK OF PRIMARY THRUSTER TABLE AND ACCEL'
PRINT *,' '
PRINT 175,' + X',(TABLE(2,1,J),J=1,9),JET(2,1)
PRINT 175,' - X',(TABLE(2,2,J),J=1,9),JET(2,2)
PRINT 175,' + Y',(TABLE(2,3,J),J=1,9),JET(2,3)
PRINT *,' '
PRINT 175,' - Y',(TABLE(2,4,J),J=1,9),JET(2,4)
PRINT 175,' + Z',(TABLE(2,5,J),J=1,9),JET(2,5)
PRINT 175,' - Z',(TABLE(2,6,J),J=1,9),JET(2,6)
PRINT *,' '
PRINT 175,' + ROL',(TABLE(2,7,J),J=1,9),JET(2,7)
PRINT 175,' - RCL',(TABLE(2,8,J),J=1,9),JET(2,8)
PRINT 175,' + PCH',(TABLE(2,9,J),J=1,9),JET(2,9)
PRINT *,' '
PRINT 175,' - PCH',(TABLE(2,10,J),J=1,9),JET(2,10)
PRINT 175,' + YAW',(TABLE(2,11,J),J=1,9),JET(2,11)
PRINT 175,' - YAW',(TABLE(2,12,J),J=1,9),JET(2,12)
PRINT *,' '
PRINT *,' HIT RETURN TO CONTINUE'
READ *
```

```

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT *,' ECOCHK OF PZI THRUSTER TABLE AND ACCEL'
PRINT *,' '
PRINT 175,' + X',(TABLE(3,1,J),J=1,9),JET(3,1)
PRINT 175,' - X',(TABLE(3,2,J),J=1,9),JET(3,2)
PRINT 175,' + Y',(TABLE(3,3,J),J=1,9),JET(3,3)
PRINT *,' '
```

```

PRINT 175,' - Y',(TABLE(3,4,J),J=1,9),JET(3,4)
PRINT 175,' + Z',(TABLE(3,5,J),J=1,9),JET(3,5)
PRINT 175,' - Z',(TABLE(3,6,J),J=1,9),JET(3,6)
PRINT *,' '
PRINT 175,'+ ROL',(TABLE(3,7,J),J=1,9),JET(3,7)
PRINT 175,'- RCL',(TABLE(3,8,J),J=1,9),JET(3,8)
PRINT 175,'+ PCH',(TABLE(3,9,J),J=1,9),JET(3,9)
PRINT *,' '
PRINT 175,'- PCH',(TABLE(3,10,J),J=1,9),JET(3,10)
PRINT 175,'+ YAW',(TABLE(3,11,J),J=1,9),JET(3,11)
PRINT 175,'- YAW',(TABLE(3,12,J),J=1,9),JET(3,12)
PRINT *,' '
PRINT *,' HIT RETURN TO CONTINUE'
READ *
```

C

```

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT *,' ECOCHECK OF PZHI THRUSTER TABLE AND ACCEL'
PRINT *,' '
PRINT 175,' + X',(TABLE(4,1,J),J=1,9),JET(4,1)
PRINT 175,' - X',(TABLE(4,2,J),J=1,9),JET(4,2)
PRINT 175,' + Y',(TABLE(4,3,J),J=1,9),JET(4,3)
PRINT *,' '
PRINT 175,' - Y',(TABLE(4,4,J),J=1,9),JET(4,4)
PRINT 175,' + Z',(TABLE(4,5,J),J=1,9),JET(4,5)
PRINT 175,' - Z',(TABLE(4,6,J),J=1,9),JET(4,6)
PRINT *,' '
PRINT 175,'+ ROL',(TABLE(4,7,J),J=1,9),JET(4,7)
PRINT 175,'- RCL',(TABLE(4,8,J),J=1,9),JET(4,8)
PRINT 175,'+ PCH',(TABLE(4,9,J),J=1,9),JET(4,9)
PRINT *,' '
PRINT 175,'- PCH',(TABLE(4,10,J),J=1,9),JET(4,10)
PRINT 175,'+ YAW',(TABLE(4,11,J),J=1,9),JET(4,11)
PRINT 175,'- YAW',(TABLE(4,12,J),J=1,9),JET(4,12)
PRINT *,' '
PRINT *,' HIT RETURN TO CONTINUE'
READ *
```

C

C

```

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))

END
```

THE FOLLOWING ERROR HANDLER DEMONSTRATES THE GENERAL
OVERALL RECOMMENDED FORM THAT THE USER'S OWN ERROR
HANDLER SHOULD FOLLOW.

THIS ERROR HANDLER UPON BEING INVOKED WRITES ALL
MESSAGES TO THE DATA FILE: 'PROERROR.LOG'. ERROR
AND WARNING EXPLANATION MESSAGES ARE WRITTEN TO
A DATA FILE FOR 2 REASONS:

1. THE ERROR HANDLER SHOULD NOT IMMEDIATELY
WRITE INFORMATION OUT ON THE PS 300 SCREEN
SINCE THE EXPLANATORY TEXT DEFINING THE ERROR
OR WARNING CONDITION MAY BE TAKEN AS DATA BY
THE PS 300 AND THEREFORE WIND UP NOT BEING
DISPLAYED ON THE PS 300 SCREEN (AS IN THE
CASE OF A CATASTROPHIC DATA TRANSMISSION
ERROR).
2. THE LOGGING OF ERRORS AND WARNINGS TO A
LOGFILE ALLOWS ANY ERRORS AND/OR WARNINGS
TO BE REVIEWED AT A LATER TIME.

SUBROUTINE ERR (ERRCOD)

PROCEDURAL INTERFACE ERROR HANDLER:

```
INCLUDE 'PROCONST.FOR/NOLIST'
INTEGER*4  ERRCOD
INTEGER*4  PSVMSERR
LOGICAL    FILOPN
DATA       FILOPN /.FALSE./
EXTERNAL   PSVMSERR, DETERM, PIDCOD
```

IF (FILOPN) GOTO 1

OPEN ERROR FILE FOR LOGGING OF ERRORS:

```
OPEN (UNIT=10, FILE='PROERROR.LOG', STATUS='NEW',
&      DISP='KEEP', ORGANIZATION='SEQUENTIAL',
&      ACCESS='SEQUENTIAL', CARRIAGECONTROL='LIST')
FILOPN = .TRUE.
END IF
1 CALL PIDCOD (ERRCOD)
IF (ERRCOD .LT. 512) GOTO 3
WRITE (10, *) 'PS-I-ATDCOMLNK: ATTEMPTING TO '
&           // 'DETACH PS 300/HOST COMMUNICATIONS '
&           // 'LINK.'
```

WHEN WE ATTEMPT TO PERFORM THE DETACH, USE A

```

C      DIFFERENT ERROR HANDLER SO AS NOT TO GET CAUGHT
C      IN A RECURSIVE LOOP IF WE CONSISTENTLY GET AN
C      ERROR WHEN ATTEMPTING TO DETACH.

```

```

C      CALL PDTACH (DETERM)
C      CLOSE (UNIT=10)
C      IF ((ERRCOD .LT. PSFPAF) .OR.
&        (ERRCOD .GT. PSFPPF)) GOTO 2
C
C      IDENTIFY VMS ERROR IF THERE WAS ONE
C
C      CALL LIB$STOP (XVAL (PSVMSERR ()))
C      GOTO 3
C      ELSE
C      2  STOP
C      END IF
C      END IF
C      3  RETURN
C      END

```

```

SUBROUTINE DETERM (ERRCOD)

```

```

C
C      MAIN ERROR HANDLER DETACH ERROR HANDLER:
C

```

```

      INTEGER*4  ERRCOD
      EXTERNAL  PIDCOD

      WRITE (10, *) 'PS-I-ERRWARDET:  ERROR/WARNING '
&      // 'TRYING TO DETACH '
&      // 'THE COMMUNICATIONS'
      WRITE (10, *) 'LINK BETWEEN THE PS 300 AND THE HOST.'
      CALL PIDCOD (ERRCOD)
      RETURN
      END

```

```

SUBROUTINE PIDCOD (ERRCOD)

```

```

C
C      PIDCOD:  IDENTIFY PROCEDURAL INTERFACE COMPLETION
C      CODE.
C

```

```

      INCLUDE 'PROCONST.FOR/NOLIST'
      INTEGER*4  ERRCOD
      CHARACTER  VMSDEF*133, PIDEF*133
      INTEGER*4  PSVMSERR
      CHARACTER  MSSG1*55, MSSG2*67
      PARAMETER  (MSSG1 = 'PS-W-UNRCOMCOD:  PROCEDURAL '
&      // 'INTERFACE '
&      // '(GSR) COMPLETION ')
      EXTERNAL  PSVMSERR

      WRITE (10, *) 'PS-I-PROERRWAR:  PROCEDURAL '
&      // 'INTERFACE WARNING/'

```

```

      &          // 'ERROR COMPLETION CODE WAS '
      WRITE (10, *) 'RECEIVED.'
      IF (ERRCOD .NE. PSWBNC) GOTO 1
      WRITE (10, *) 'PS-W-BADNAMCHR:  BAD CHARACTER '
      &          // 'IN NAME WAS '
      &          // 'TRANSLATED TO:  "-".'
      GOTO 1000
C     ELSE
1  IF (ERRCOD .NE. PSWNTL) GOTO 2
      WRITE (10, *) 'PS-W-NAMTOOLON:  NAME TOO '
      &          // 'LONG.  NAME WAS '
      &          // 'TRUNCATED TO '
      WRITE (10, *) '256 CHARACTERS.'
      GOTO 1000
C     ELSE
2  IF (ERRCOD .NE. PSWSTL) GOTO 7
      WRITE (10, *) 'PS-W-STRTOOLON:  STRING TOO '
      &          // 'LONG.  STRING '
      &          // 'WAS TRUNCATED '
      WRITE (10, *) 'TO 240 CHARACTERS.'
      GOTO 1000
C     ELSE
7  IF (ERRCOD .NE. PSWAAD) GOTO 8
      WRITE (10, *) 'PS-W-ATTALRDON:  ATTACH '
      &          // 'ALREADY DONE. '
      &          // 'MULTIPLE CALL TO PATTCH WITHOUT'
      WRITE (10, *) 'INTERVENING PDTACH CALL IGNORED.'
      GOTO 1000
C     ELSE
8  IF (ERRCOD .NE. PSWAKS) GOTO 9
      WRITE (10, *) 'PS-W-ATNKEYSEE:  ATTENTION KEY '
      &          // 'SEEN (DEPRESSED).'
      CALL PIBMSP
      GOTO 1000
C     ELSE
9  IF (ERRCOD .NE. PSWBGCC) GOTO 10
      WRITE (10, *) 'PS-W-BADGENCHR:  BAD GENERIC '
      &          // 'CHANNEL CHARACTER.  BAD '
      WRITE (10, *) 'CHARACTER IN STRING SENT VIA: '
      &          // 'PPUTGX WAS TRANSLATED TO '
      WRITE (10, *) 'A BLANK.'
      CALL PIBMSP
      GOTO 1000
C     ELSE
10 IF (ERRCOD .NE. PSWBSC) GOTO 11
      WRITE (10, *) 'PS-W-BADSTRCHR:  BAD '
      &          // 'CHARACTER IN STRING WAS '
      &          // 'TRANSLATED TO A BLANK.'
      CALL PIBMSP
      GOTO 1000
C     ELSE
11 IF (ERRCOD .NE. PSWBPC) GOTO 12
      WRITE (10, *) 'PS-W-BADPARCHR:  BAD PARSER '
      &          // 'CHANNEL CHARACTER.  BAD '
      &          // 'CHARACTER IN STRING SENT TO'
      WRITE (10, *) 'PS 300 PARSER VIA:  PPUTP '
      &          // 'WAS TRANSLATED TO A BLANK.'
      CALL PIBMSP
      GOTO 1000
C     ELSE

```

```

12 IF (ERRCOD .NE. PSEIMC) GOTO 13
    WRITE (10, *) 'PS-E-INVMUXCHA: INVALID '
    &           // 'MULTIPLEXING CHANNEL '
    &           // 'SPECIFIED IN CALL TO:'
    WRITE (10, *) 'PMUXCI, PMUXP, OR PMUXG.'
    GOTO 1000
C   ELSE
13 IF (ERRCOD .NE. PSEIVC) GOTO 14
    WRITE (10, *) 'PS-E-INVVECCLA: INVALID '
    &           // 'VECTOR LIST CLASS '
    &           // 'SPECIFIED'
    WRITE (10, *) 'IN CALL TO: PVCBEG.'
    GOTO 1000
C   ELSE
14 IF (ERRCOD .NE. PSEIVD) GOTO 15
    WRITE (10, *) 'PS-E-INVVECDIM: INVALID '
    &           // 'VECTOR LIST DIMENSION '
    &           // 'SPECIFIED IN CALL TO'
    WRITE (10, *) 'PVCBEG.'
    GOTO 1000
C   ELSE
15 IF (ERRCOD .NE. PSEPOE) GOTO 16
    WRITE (10, *) 'PS-E-PROPEEXP: PREFIX '
    &           // 'OPERATOR CALL WAS '
    &           // 'EXPECTED.'
    GOTO 1000
C   ELSE
16 IF (ERRCOD .NE. PSEFOE) GOTO 17
    WRITE (10, *) 'PS-E-FOLOPEEXP: FOLLOW '
    &           // 'OPERATOR CALL WAS '
    &           // 'EXPECTED.'
    GOTO 1000
C   ELSE
17 IF (ERRCOD .NE. PSELBE) GOTO 18
    WRITE (10, *) 'PS-E-LABBLKEXP: CALL TO '
    &           // 'PLAADD OR PLAEND WAS '
    &           // 'EXPECTED.'
    GOTO 1000
C   ELSE
18 IF (ERRCOD .NE. PSEVLE) GOTO 19
    WRITE (10, *) 'PS-E-VECLISEXP: CALL TO '
    &           // 'PVCLIS OR PVCEND '
    &           // 'WAS EXPECTED.'
    GOTO 1000
C   ELSE
19 IF (ERRCOD .NE. PSEAMV) GOTO 20
    WRITE (10, *) 'PS-E-ATTMULVEC: ATTEMPTED '
    &           // 'MULTIPLE CALL '
    &           // 'SEQUENCE TO PVCLIS IS NOT'
    WRITE (10, *) 'PERMITTED FOR BLOCK '
    &           // 'NORMALIZED VECTORS.'
    GOTO 1000
C   ELSE
20 IF (ERRCOD .NE. PSEMLB) GOTO 21
    WRITE (10, *) 'PS-E-MISLABBEG: MISSING '
    &           // 'LABEL BLOCK BEGIN CALL. '
    &           // 'CALL TO PLAADD OR PLAEND'
    WRITE (10, *) 'WITHOUT CALL TO: PLABEG.'
    GOTO 1000
C   ELSE

```

```

21 IF (ERRCOD .NE. PSEMV) GOTO 22
    WRITE (10, *) 'PS-E-MISVECBEG: MISSING '
    &           // 'VECTOR LIST BEGIN '
    &           // 'CALL. CALL TO PVCLIS'
    WRITE (10, *) 'OR PVCEND WITHOUT CALL '
    &           // 'TO: PVCBEG.'
    GOTO 1000
C   ELSE
22 IF (ERRCOD .NE. PSENU) GOTO 23
    WRITE (10, *) 'PS-E-NULNAM: NULL NAME '
    &           // 'PARAMETER IS NOT ALLOWED.'
    GOTO 1000
C   ELSE
23 IF (ERRCOD .NE. PSEBCT) GOTO 24
    WRITE (10, *) 'PS-E-BADCOMTYP: BAD '
    &           // 'COMPARISON TYPE OPERATOR '
    &           // 'SPECIFIED IN '
    WRITE (10, *) 'CALL TO: PIFLEV.'
    GOTO 1000
C   ELSE
24 IF (ERRCOD .NE. PSEIFN) GOTO 25
    WRITE (10, *) 'PS-E-INVFUNNAM: INVALID '
    &           // 'FUNCTION NAME. '
    &           // 'ATTEMPTED PS 300'
    WRITE (10, *) 'FUNCTION INSTANCE FAILED '
    &           // 'BECAUSE THE NAMED '
    &           // 'FUNCTION CANNOT POSSIBLY'
    WRITE (10, *) 'EXIST. THE FUNCTION NAME '
    &           // 'IDENTIFYING THE '
    &           // 'FUNCTION TYPE TO INSTANCE'
    WRITE (10, *) 'WAS LONGER THAN 256 CHARACTERS.'
    GOTO 1000
C   ELSE
25 IF (ERRCOD .NE. PSENNR) GOTO 26
    WRITE (10, *) 'PS-E-NULNAMREQ: NULL NAME '
    &           // 'PARAMETER IS '
    &           // 'REQUIRED IN OPERATE MODE'
    WRITE (10, *) 'CALL FOLLOWING A PPREF OR '
    &           // 'PFOLL PROCEDURE CALL.'
    GOTO 1000
C   ELSE
26 IF (ERRCOD .NE. PSETME) GOTO 27
    WRITE (10, *) 'PS-E-TOOMANEND: TOO '
    &           // 'MANY END_STRUCTURE CALLS '
    &           // 'INVOKED.'
    GOTO 1000
C   ELSE
27 IF (ERRCOD .NE. PSENOA) GOTO 28
    WRITE (10, *) 'PS-E-NOTATT: THE PS 300 '
    &           // 'COMMUNICATIONS LINK '
    &           // 'HAS NOT '
    WRITE (10, *) 'YET BEEN ESTABLISHED. '
    &           // 'PATCH HAS NOT BEEN '
    &           // 'CALLED OR FAILED.'
    GOTO 1000
C   ELSE
28 IF (ERRCOD .NE. PSEODR) GOTO 29
    WRITE (10, *) 'PS-E-OVEDURREA: AN '
    &           // 'OVERRUN OCCURRED DURING '
    &           // 'A READ OPERATION.'

```

```

        WRITE (10, *) 'THE SPECIFIED INPUT BUFFER '
&        // 'IN CALL TO: PGET '
&        // 'OR: PGETW'
        WRITE (10, *) 'WAS TOO SMALL AND '
&        // 'TRUNCATION HAS OCCURRED.'
        GOTO 1000
C      ELSE
29 IF (ERRCOD .NE. PREICP) GOTO 38
38 IF (ERRCOD .NE. PSEPDT) GOTO 39
        WRITE (10, *) 'PS-E-PHYDEVTP: MISSING '
&        // 'OR INVALID PHYSICAL '
&        // 'DEVICE TYPE'
        WRITE (10, *) 'SPECIFIER IN CALL TO PATTC.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
39 IF (ERRCOD .NE. PSELDN) GOTO 40
        WRITE (10, *) 'PS-E-LOGDEVNAM: MISSING '
&        // 'OR INVALID LOGICAL '
&        // 'DEVICE NAME'
        WRITE (10, *) 'SPECIFIER IN CALL TO PATTC.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
40 IF (ERRCOD .NE. PSEADE) GOTO 41
        WRITE (10, *) 'PS-E-ATTDELEXP: ATTACH '
&        // 'PARAMETER STRING '
&        // 'DELIMITER'
        WRITE (10, *) '"/" WAS EXPECTED.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
41 IF (ERRCOD .NE. PSFPAF) GOTO 42
        WRITE (10, *) 'PS-F-PHYATTFAI: '
&        // 'PHYSICAL ATTACH OPERATION '
&        // 'FAILED.'
        GOTO 1000
C      ELSE
42 IF (ERRCOD .NE. PSFPDF) GOTO 43
        WRITE (10, *) 'PS-F-PHYDETFAI: PHYSICAL '
&        // 'DETACH OPERATION '
&        // 'FAILED.'
        GOTO 1000
C      ELSE
43 IF (ERRCOD .NE. PSFPGF) GOTO 44
        WRITE (10, *) 'PS-F-PHYGETFAI: PHYSICAL '
&        // 'GET OPERATION FAILED.'
        GOTO 1000
C      ELSE
44 IF (ERRCOD .NE. PSFPPF) GOTO 45
        WRITE (10, *) 'PS-F-PHYPUTFAI: PHYSICAL '
&        // 'PUT OPERATION FAILED.'
        GOTO 1000
C      ELSE
45 IF (ERRCOD .NE. PSFBTL) GOTO 46
        WRITE (10, *) 'PS-F-SUFTOCLAR: BUFFER '
&        // 'TOO LARGE ERROR IN '
&        // 'CALL TO: PSPUT.'
        WRITE (10, *) 'THIS ERROR SHOULD NEVER '
&        // 'OCCUR AND INDICATES A '

```

```

      &          // 'PROCEDURAL INTERFACE (GSR)'
      WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
      CALL PVAXSP
      GOTO 1000
C   ELSE
46 IF (ERRCOD .NE. PSFWNA) GOTO 47
      &          // 'PS-F-WRONUMARG:  WRONG '
      WRITE (10, *) 'PS-F-WRONUMARG:  WRONG '
      &          // 'NUMBER OF ARGUMENTS '
      &          // 'IN CALL TO PROCEDURAL'
      WRITE (10, *) 'INTERFACE (GSR) LOW-LEVEL '
      &          // 'I/O PROCEDURE '
      &          // '(SOURCE FILE:  PROIOLIB.MAR).'
      WRITE (10, *) 'THIS ERROR SHOULD NEVER '
      &          // 'OCCUR AND INDICATES A '
      &          // 'PROCEDURAL INTERFACE (GSR)'
      WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
      CALL PVAXSP
      GOTO 1000
C   ELSE
47 IF (ERRCOD .NE. PSFPTL) GOTO 48
      &          // 'PS-F-PROTOOLAR:  PROMPT '
      WRITE (10, *) 'PS-F-PROTOOLAR:  PROMPT '
      &          // 'BUFFER TOO LARGE '
      &          // 'ERROR IN CALL TO:  PSPRCV.'
      WRITE (10, *) 'THIS ERROR SHOULD NEVER '
      &          // 'OCCUR AND INDICATES A '
      &          // 'PROCEDURAL INTERFACE (GSR)'
      WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
      CALL PVAXSP
      GOTO 1000
C   ELSE
C   C
C   C   UNKNOWN ERROR MESSAGE ERROR MESSAGE.
C   C
48 IF (ERRCOD .GE. 512) GOTO 49
      MSSG2 = MSSG1 // 'WARNING'
      GOTO 51
C   ELSE
49 IF (ERRCOD .GE. 1024) GOTO 50
      MSSG2 = MSSG1 // 'ERROR '
      GOTO 51
C   ELSE
50 MSSG2 = MSSG1 // 'FATAL ERROR '
C   END IF
C   END IF
51 WRITE (10, *) MSSG2
   WRITE (10, *) 'CODE IS UNRECOGNIZED.'
   WRITE (10, *) 'PROBABLE PROCEDURAL '
      &          // 'INTERFACE (GSR) INTERNAL '
      &          // 'VALIDITY CHECK ERROR.'
C   END IF
1000 IF ((ERRCOD .LT. PSFPAF) .OR.
      & (ERRCOD .GT. PSFPPF)) GOTO 2000
      CALL PSFVMSERR ( VMSDEF, PIDEF )
      WRITE (10, *) 'DEC VAX/VMS ERROR '
      &          // 'DEFINITION IS:'
      WRITE (10, *) VMSDEF
      WRITE (10, *) 'PROCEDURAL INTERFACE '
      &          // '(GSR) INTERPRETATION OF '
      &          // 'DEC VAX/VMS COMPLETION CODE:'
      WRITE (10, *) PIDEF

```

```

      WRITE (10, *) 'DEC VAX/VMS ERROR CODE '
      & // 'VALUE WAS: ', PSVMSERR ()
C   END IF
2000 WRITE (10, *)
      RETURN
      END

```

SUBROUTINE PIBMSP

```

C   PIBMSP:  WRITE THE "IBM VERSION SPECIFIC"
C             MESSAGE TO THE ERROR HANDLER FILE.
C

```

```

      WRITE (10, *) 'THIS ERROR/WARNING IS '
      & // 'APPLICABLE ONLY TO THE IBM '
      & // 'VERSION OF THE'
      WRITE (10, *) 'PROCEDURAL INTERFACE (GSR).'
      RETURN
      END

```

SUBROUTINE PVAXSP

```

C   PVAXSP:  WRITE THE "DEC VAX/VMS VERSION
C             SPECIFIC" MESSAGE TO THE ERROR
C             HANDLER FILE.
C

```

```

      WRITE (10, *) 'THIS ERROR/WARNING IS '
      & // 'APPLICABLE ONLY TO THE DEC '
      & // 'VAX/VMS VERSION OF'
      WRITE (10, *) 'THE PROCEDURAL INTERFACE (GSR).'
      RETURN
      END

```

SUBROUTINE INITBUF(IBUF,CHAN,IOSB)

THIS SUBROUTINE FILLS IBUF WITH DATA THAT DOES NOT CHANGE DURING PROGRAM. THIS DATA IS USED TO ADDRESS PS300 MEMORY LOCATIONS.

INTEGER*4 SYSSQIO,SYSSSETEF,SYSSWAITFR
 INTEGER*4 CHAN,STATUS,SYSSASSIGN,SYSSQIOW,NAMADX(20)
 INTEGER*2 IBUF(3),NAMES(5,20),NAMADR(2,20),IOSB(4),BUFNUM
 CHARACTER*4 UNIT
 EQUIVALENCE (NAMADR,NAMADX)
 DATA NAMES/

1	'TA'	'RG'	'T.'	'LO'	'OK'	'XY'	'ZV'	'C.'	'LO'	'OK'
1	'GL'	'OB'	'E.'	'LO'	'OK'	'ST'	'AR'	'S.'	'LO'	'OK'
1	'GL'	'OB'	'E.'	'TR'	'OT'	'ST'	'AR'	'S.'	'TR'	'OT'
1	'GL'	'OB'	'E.'	'ER'	'OT'	'SA'	'TE'	'L.'	'RO'	'TX'
1	'XY'	'PT'	'H.'	'TR'	'N1'	'XY'	'PT'	'H.'	'TR'	'N2'
1	'XY'	'PT'	'H.'	'TR'	'N3'	'XY'	'PT'	'H.'	'TR'	'N4'
1	'XY'	'PT'	'H.'	'TR'	'N5'					
1	'IN'	'FO'	'R.'	'AN'	'GE'	'IN'	'FO'	'R.'	'RA'	'TE'
1	'IN'	'FO'	'R.'	'FU'	'EL'	'IN'	'FO'	'R.'	'TI'	'ME'
1	'XY'	'PT'	'H.'	'SS'	'TR'	'XY'	'PT'	'H.'	'RO'	'TZ'
1	'XY'	'PT'	'H.'	'RO'	'TX'					

DATA UNIT/'PIAO'/

WAIT

DO 5 J=1,100000
 CONTINUE

SET UP HOUSEKEEPING

GET A CHANNEL NUMBER

STATUS=SYSSASSIGN(UNIT,CHAN,,)
 IF(STATUS.NE.1) THEN
 TYPE *, 'BAD ASSIGN! <STATUS> = ',STATUS
 STOP
 ENDIF

DETACH FOR SAFETY: 34 --> DETACH FUNCTION CODE

STATUS=SYSSQIOW(,XVAL(CHAN),XVAL(34),IOSB,,,,,,,,)

ATTACH: 33 --> ATTACH FUNCTION CODE

STATUS=SYSSQIOW(,XVAL(CHAN),XVAL(33),IOSB,,,,,,,,)
 IF(STATUS.NE.1) THEN
 TYPE *, 'BAD ATTACH! <STATUS> = ',STATUS
 STOP
 ENDIF

GET THE ADDRESSES OF THE ENTITIES TO UPDATE
 43 --> LOOKUP NAMED ENTITIES FUNCTION CODE

DO 25 I=1,20
 STATUS=SYSSQIOW(,XVAL(CHAN),XVAL(43),IOSB,,NAMES(1,I),
 XVAL(10),XVAL(1),,,)

IF(STATUS.EQ.1.AND.IOSB(1).EQ.1.AND.

```

1      (IOSB(3).OR.IOSB(4)).NE.0) GOTO 21
      TYPE *, 'BAD ENTITY FETCH! <STAT,IOSB> -- ',STATUS,IOSB
      STOP

C
C      GET THE ADDRESS FROM OUT OF THE IO STATUS BLOCK (IOSB)
C
21      DO 24 J=1,2
          NAMADR(J,I)=IOSB(J+2)
24      CONTINUE
25      CONTINUE

C
C      OFFSET THE ADDRESSES TO GET PAST THE FIRST THREE FIELDS
C      OF THE NODE WHICH WE DO NOT WANT TO CHANGE.
C
      DO 30 I=1,20
          NAMADX(I)=NAMADX(I)+8
30      CONTINUE

C
C      OFFSET TEXT BY AN ADDITIONAL 16
C
      DO 31 I=14,17
          NAMADX(I)=NAMADX(I)+16
31      CONTINUE

C
C      BUFFER 1  SETUP
C      TRANSLATION NEEDS  7 ELEMENTS
C      ROTATION NEEDS 19 ELEMENTS
C      LOOKAT NEEDS 28 ELEMENTS
C

IBUF(1)      = 20      ! TWENTY BLOCKS
IBUF(2)      = NAMADR(1,1) ! BLOCK ONE ADDRESS - TARGET.LOOK
IBUF(3)      = NAMADR(2,1) ! BLOCK ONE ADDRESS
IBUF(4)      = 27      ! WORD COUNT FOR BLOCK 1
IBUF(24)     = 1       ! TRAN FLAG
IBUF(32)     = NAMADR(1,2) ! BLOCK TWO ADDRESS - XYZVC.LOOK
IBUF(33)     = NAMADR(2,2) ! BLOCK TWO ADDRESS
IBUF(34)     = 27      ! WORD COUNT FOR BLOCK 2
IBUF(54)     = 1       ! TRAN FLAG
IBUF(62)     = NAMADR(1,3) ! BLOCK 3 ADDRESS - GLOBE.LOOK
IBUF(63)     = NAMADR(2,3) ! BLOCK 3 ADDRESS
IBUF(64)     = 27      ! WORD COUNT FOR BLOCK 3
IBUF(84)     = 1       ! TRAN FLAG
IBUF(92)     = NAMADR(1,4) ! BLOCK 4 ADDRESS - STARS.LOOK
IBUF(93)     = NAMADR(2,4) ! BLOCK 4 ADDRESS
IBUF(94)     = 27      ! WORD COUNT FOR BLOCK 4
IBUF(114)    = 1       ! TRAN FLAG

IBUF(122)    = NAMADR(1,5) ! BLOCK 5 ADDRESS - GLOBE.TROT
IBUF(123)    = NAMADR(2,5) ! BLOCK 5 ADDRESS
IBUF(124)    = 19      ! WORD COUNT FOR BLOCK 5
IBUF(144)    = NAMADR(1,6) ! BLOCK 6 ADDRESS - STARS.TROT
IBUF(145)    = NAMADR(2,6) ! BLOCK 6 ADDRESS
IBUF(146)    = 19      ! WORD COUNT FOR BLOCK 6

IBUF(166)    = NAMADR(1,7) ! BLOCK 7 ADDRESS - GLOBE.ENOT

```

IBUF(167)	= NAMADR(2,7)	! BLOCK 7 ADDRESS
IBUF(168)	= 19	! WORD COUNT FOR BLOCK 7
IBUF(188)	= NAMADR(1,8)	! BLOCK 8 ADDRESS - SATEL.P
IBUF(189)	= NAMADR(2,8)	! BLOCK 8 ADDRESS
IBUF(190)	= 19	! WORD COUNT FOR BLOCK 8
IBUF(210)	= NAMADR(1,9)	! BLOCK 9 ADDRESS - XYPTH.TRN1
IBUF(211)	= NAMADR(2,9)	! BLOCK 9 ADDRESS
IBUF(212)	= 7	! WORD COUNT FOR BLOCK 9
IBUF(220)	= NAMADR(1,10)	! BLOCK 10 ADDRESS - XYPTH.TRN2
IBUF(221)	= NAMADR(2,10)	! BLOCK 10 ADDRESS
IBUF(222)	= 7	! WORD COUNT FOR BLOCK 10
IBUF(230)	= NAMADR(1,11)	! BLOCK 11 ADDRESS - XYPTH.TRN3
IBUF(231)	= NAMADR(2,11)	! BLOCK 11 ADDRESS
IBUF(232)	= 7	! WORD COUNT FOR BLOCK 11
IBUF(240)	= NAMADR(1,12)	! BLOCK 12 ADDRESS - XYPTH.TRN4
IBUF(241)	= NAMADR(2,12)	! BLOCK 12 ADDRESS
IBUF(242)	= 7	! WORD COUNT FOR BLOCK 12
IBUF(250)	= NAMADR(1,13)	! BLOCK 13 ADDRESS - XYPTH.TRN5
IBUF(251)	= NAMADR(2,13)	! BLOCK 13 ADDRESS
IBUF(252)	= 7	! WORD COUNT FOR BLOCK 13
IBUF(260)	= NAMADR(1,14)	! BLOCK 14 ADDRESS - INFO.RANGE
IBUF(261)	= NAMADR(2,14)	! BLOCK 14 ADDRESS
IBUF(262)	= 3	! WORD COUNT FOR BLOCK 14
IBUF(266)	= NAMADR(1,15)	! BLOCK 15 ADDRESS - INFO.RRATE
IBUF(267)	= NAMADR(2,15)	! BLOCK 15 ADDRESS
IBUF(268)	= 3	! WORD COUNT FOR BLOCK 15
IBUF(272)	= NAMADR(1,16)	! BLOCK 16 ADDRESS - INFO.RFUEL
IBUF(273)	= NAMADR(2,16)	! BLOCK 16 ADDRESS
IBUF(274)	= 3	! WORD COUNT FOR BLOCK 16
IBUF(278)	= NAMADR(1,17)	! BLOCK 17 ADDRESS - INFO.RTIME
IBUF(279)	= NAMADR(2,17)	! BLOCK 17 ADDRESS
IBUF(280)	= 3	! WORD COUNT FOR BLOCK 17
IBUF(284)	= NAMADR(1,18)	! BLOCK 18 ADDRESS - XYPTH.SSTR
IBUF(285)	= NAMADR(2,18)	! BLOCK 18 ADDRESS
IBUF(286)	= 7	! WORD COUNT FOR BLOCK 18
IBUF(294)	= NAMADR(1,19)	! BLOCK 19 ADDRESS - XYPTH.ROTZ
IBUF(295)	= NAMADR(2,19)	! BLOCK 19 ADDRESS
IBUF(296)	= 19	! WORD COUNT FOR BLOCK 19
IBUF(316)	= NAMADR(1,20)	! BLOCK 20 ADDRESS - XYPTH.ROTX
IBUF(317)	= NAMADR(2,20)	! BLOCK 20 ADDRESS
IBUF(318)	= 19	! WORD COUNT FOR BLOCK 20

C

END

SUBROUTINE IOBUFF (IFUNC,S,L,C,RESET)

```

INTEGER S(24),L(24),C(12)
INTEGER IFUNC,RESET
INTEGER*2 DATAOUT,DATAIN
INTEGER*2 DIBUF(10),DOBUF(10)
INTEGER*4 ISTAT,ICALL
INTEGER*4 NCHAN,DOFLAG,DIFLAG

```

```

REAL ARATE,RATE

```

```

DATA DOFLAG,DIFLAG/3,5/
DATA NFRAME/10/
DATA MODEOUT,MODEIN/8,7/
DATA IUNIT/1/
DATA ICHAN/1/
DATA NCHAN/1/
DATA ISMODE/0/
DATA INIT/1/
DATA RATE/80000.0/

```

```

GO TO (100,200) IFUNC

```

```

100  CALL LPAIO (INIT,IUNIT,,RATE,,,,,ARATE,ISTAT,ICALL,,)
     IF (.NOT. ISTAT) GO TO 950
     RETURN

200  CONTINUE
     DATAOUT='0000'X
     DO 210 I=1,6
210  DATAOUT=DATAOUT+L(I)*2** (I-1)
     DO 220 I=1,NFRAME
220  DOBUF(I)=DATAOUT .XOR. 'FFFF'X
     CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
1  ISTAT,ICALL,,ISMODE)
     IF (.NOT. ISTAT) GO TO 950
     CALL SYSSWAITFR (XVAL(DOFLAG))
230  CALL LPAIO (18,,,,,,ISTAT,ICALL,LSTAT,)
     IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 230
     CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
1  ISTAT,ICALL,,ISMODE)
     IF (.NOT. ISTAT) GO TO 950
     CALL SYSSWAITFR (XVAL(DIFLAG))
240  CALL LPAIO (17,,,,,,ISTAT,ICALL,LSTAT,)
     IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 240
     DATAIN=(DIBUF(1).XOR.'FFFF'X)
     S(1)=DATAIN .AND. 1
     S(2)=(DATAIN .AND. 2)/2
     S(3)=(DATAIN .AND. 4)/4
     S(4)=(DATAIN .AND. 8)/8
     S(5)=(DATAIN .AND. 16)/16
     S(6)=(DATAIN .AND. 32)/32
     RESET=((64.XOR.DATAIN) .AND. 64)/64
     DATAOUT='4000'X
     DO 250 I=7,15
250  DATAOUT=DATAOUT+L(I)*2** (I-7)
     DO 260 I=1,NFRAME
260  DOBUF(I)=DATAOUT .XOR. 'FFFF'X
     CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
1  ISTAT,ICALL,,ISMODE)

```



```

360      CALL LPAIO (17,,,,,,ISTAT,ICALL,LSTAT)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 360
        DATAIN=(DIBUF(1).XOR.'FFFF'X)
        C(1)=(1.XOR.DATAIN) .AND. 1
        C(2)=((2.XOR.DATAIN) .AND. 2)/2
        C(3)=(DATAIN .AND. 4)/4
        C(4)=(DATAIN .AND. 8)/8
        C(5)=(DATAIN .AND. 16)/16
        C(6)=(DATAIN .AND. 32)/32
        C(7)=(DATAIN .AND. 64)/64
        C(8)=(DATAIN .AND. 128)/128
        C(9)=(DATAIN .AND. 256)/256
        C(10)=(DATAIN .AND. 512)/512
        C(11)=(DATAIN .AND. 1024)/1024
        C(12)=(DATAIN .AND. 2048)/2048
        RETURN

950      CONTINUE
        WRITE (5,1950) ISTAT,ICALL
1950     FORMAT (' ERROR IN CALL: STATUS = ',I6,' FROM CALL #',I6)
        CALL EXIT
        END

```

SUBROUTINE LINTEG(X,OMEGA,A,DELTAT)

THIS IS A FIRST ORDER INTEGRATION SCHEME FOR TRANSLATIONAL
ACCELERATIONS IN THE REF (C-W) FRAME USING C-W EQUATIONS

REAL X(6),XDOT(6),A(3),OMEGA,DELTAT
INTEGER J

A(1-3) - X,Y,Z ACCELERATION
X(1-3) - X,Y,Z POSITION
X(4-6) - X,Y,Z VELOCITY
DELTAT - TIME STEP
OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH

XDOT(1)=X(4)
XDOT(2)=X(5)
XDOT(3)=X(6)

BELOW ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPT
VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT
WITH KEPLARIAN MOTION

XDOT(4)=A(1)-2.0*OMEGA*X(5)
XDOT(5)=A(2)+3.0*OMEGA*OMEGA*X(2)+2.0*OMEGA*X(4)
XDOT(6)=A(3)-X(3)*OMEGA*OMEGA

DO 100 J=1,6
X(J)=X(J)+DELTAT*XDOT(J)

CONTINUE

END

SUBROUTINE LOOK(T,X,OMEGA,TIME,FUEL,IBUF,CHAN,IOSB)

DETERMINES WHERE CAMERA IS POINTING (AT)
FROM SHUTTLE POSITION (X1,X2,X3) AND TRANSFORMS
FROM RH TO LH CARTESIAN COORDINATES (FM)
FOR VIEWING ON EVANS & SUTHERLAND PS300.
DETERMINES UP VECTOR FOR PS300 AND COMPUTES
EARTH AND STAR ROTATIONS AND POSITION OF HORIZON.
ALSO COMPUTES HEADS UP DISPLAY DATA.
INFORMATION IS THEN SENT TO THE PS300 FOR DISPLAY.

INCLUDE 'CALFANO]PROCONST.FOR/NOLIST'

REAL CAM(3),T(3,3),X(6),MAT(4,4)
REAL OMEGA,TIME,UP(3),RRATE,FUEL,RANGE,DT
REAL*4 TROT,EROT,AT(3),FM(3),V(3),UPPS(3),RSAT
REAL*4 XYPTH(3),NEWX(3),BMAT(3,3),BVEC(3),AMAT(3,3)

INTEGER*4 CHAN,SYSSQIO,SYSSQIOW
INTEGER*2 IOSB(4),IBUF(337)

COMPUTE SHUTTLE POSITION, RANGE AND RANGE RATE
WRT C-W FRAME (IN LH SYSTEM).

FM(1)=X(1)
FM(2)=X(2)
FM(3)=-X(3)

RANGE=SQRT(X(1)**2+X(2)**2+X(3)**2)
RRATE=(X(1)*X(4)+X(2)*X(5)+X(3)*X(6))/RANGE
IF (ABS(RRATE) .LT. .1) RRATE=0.0

COMPUTE SHUTTLE POSITION AND SCALE DOWN FOR
HUD XYPTH DISPLAY.

XYPTH(1)=X(1)/100.
XYPTH(2)=X(2)/100.
XYPTH(3)=-X(3)/100.

ASSIGN VALUES TO CAMERA LOOK VECTOR IN BODY FRAME
(I. E. - LOOK UP OUT OF PAYLOAD BAY)

CAM(1)=0.
CAM(2)=0.
CAM(3)=-1000.

TRANSFORM TO REF FRAME

CALL BTOR(T,CAM,AT)

ADD SHUTTLE POSITION TO CAMERA VECTOR (REF FRAME)
AND CONVERT TO LEFT HANDED GRAPHICS COORDINATE SYSTEM.

AT(1)=AT(1)+FM(1)
AT(2)=AT(2)+FM(2)
AT(3)=-AT(3)+FM(3)

ASSIGN VALUES TO UP VECTOR IN BODY FRAME.
(I. E. - OUT NOSE OF SHUTTLE)

UP(1)=1000.
UP(2)=0.
UP(3)=0.

C
C TRANSFORM TO REF FRAME, TRANSLATE TO 'AT' IN LH SYSTEM
C

CALL BTOR(T,UP,UPPS)

C
UPPS(1)=AT(1)+UPPS(1)
UPPS(2)=AT(2)+UPPS(2)
UPPS(3)=AT(3)-UPPS(3)

C
C CONVERT AT,FM AND UPPS FROM FEET TO METERS
C

AT(1)=AT(1)*.3048
AT(2)=AT(2)*.3048
AT(3)=AT(3)*.3048
FM(1)=FM(1)*.3048
FM(2)=FM(2)*.3048
FM(3)=FM(3)*.3048
UPPS(1)=UPPS(1)*.3048
UPPS(2)=UPPS(2)*.3048
UPPS(3)=UPPS(3)*.3048

C
C DETERMINE EARTH ROTATION (DEG)
C

EROT=TIME*.004178075

C
C DETERMINE TARGET ROTATION (DEG)
C

TROT= OMEGA*57.29577951*TIME

C
C DETERMINE SATELLITE SPIN (DEG)
C

RSAT=-TIME*30.0

C
C SEND INFORMATION TO PS300
C

CALL P919CV(XYPH,3,IBUF(287))
CALL ROT(90.,1,AMAT)
CALL P919CV(AMAT,9,IBUF(319))
CALL TLHT(T,AMAT)
CALL P919CV(AMAT,9,IBUF(297))

C
CALL ROT(RSAT,1,AMAT)
CALL P919CV(AMAT,9,IBUF(191))
CALL ROT(-EROT,2,AMAT)
CALL P919CV(AMAT,9,IBUF(169))
CALL ROT(TROT,3,AMAT)
CALL P919CV(AMAT,9,IBUF(125))
CALL P919CV(AMAT,9,IBUF(147))

C
CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
CALL P919CV(BMAT,9,IBUF(5))
CALL P919CV(BVEC,3,IBUF(25))
CALL P919CV(BMAT,9,IBUF(35))
CALL P919CV(BVEC,3,IBUF(55))

C CALL PSNREA(RANGE*.01,1,'XYZVECSC',ERR)

```

C
C      RESCALE TO DUS
C
C      DO 57 I=1,3
C          AT(I)=AT(I)/6378135.
C          FM(I)=FM(I)/6378135.
C          UPPS(I)=UPPS(I)/6378135.
57      CONTINUE
C
C      CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
C      CALL P919CV(BMAT,9,IBUF(65))
C      CALL P919CV(BVEC,3,IBUF(85))
C      CALL P919CV(BMAT,9,IBUF(95))
C      CALL P919CV(BVEC,3,IBUF(115))
C
C      CALL NUMBER(RANGE,IBUF(263))
C      CALL NUMBER(RRATE,IBUF(269))
C      CALL NUMBER(FUEL,IBUF(275))
C      CALL NUMBER(TIME,IBUF(281))
C
C      COMPUTE FUTURE POSITIONS FOR HEADS UP DISPLAY
C      AND SEND TO PS300 (DT IS TIME INCREMENT)
C
C      DT=300.
C      CALL PREDPATH(OMEGA,DT,X,NEWX)
C      CALL P919CV(NEWX,3,IBUF(213))
C      CALL PREDPATH(OMEGA,2.*DT,X,NEWX)
C      CALL P919CV(NEWX,3,IBUF(223))
C      CALL PREDPATH(OMEGA,3.*DT,X,NEWX)
C      CALL P919CV(NEWX,3,IBUF(233))
C      CALL PREDPATH(OMEGA,4.*DT,X,NEWX)
C      CALL P919CV(NEWX,3,IBUF(243))
C      CALL PREDPATH(OMEGA,5.*DT,X,NEWX)
C      CALL P919CV(NEWX,3,IBUF(253))
C
C      DO A WRITE SYNC
C      42      WRITE SYNC FUNCTION CODE
C      IOSB    IO STATUS BLOCK
C      IBUF    DATA BUFFER (ACTUALLY ADDRESS OF BUFFER, BY REFERENCE)
C      674    DATA BYTE COUNT (337 WORDS)
C      0      NOT CHARACTER DATA (1 = CHARACTER DATA)
C
C      SEND ALL DATA
C
C      STATUS=SYSSQIO(XVAL(1),XVAL(CHAN),XVAL(42),IOSB,,,
+      IBUF(1),XVAL(674),XVAL(0),,,)
C
C      IF (STATUS .NE. 1) THEN
C          TYPE *, 'BAD WRITE <STATUS,IOSB>', STATUS, IOSB
C          STOP
C      ENDIF
C
C      END
C
C
C      SUBROUTINE ROT(ANGLE,IAXIS,AMAT)
C
C      ROUTINE TO GENERATE ROTATION REQUESTS TO PS300

```

```

C
C CALLING SEQUENCE:
C
C      CALL ROT(ANGLE,IAXIS,AMAT)
C
C WHERE:
C
C ANGLE IS THE REAL*4 ANGLE FOR ROTATION, IN DEGREES.  NEED NOT BE LIMITED
C      TO A SINGLE CIRCLE.
C IAXIS IS THE INTEGER*2 AXIS OF ROTATION (1=X, 2=Y, 3=Z).
C AMAT IS THE REAL 3X3 MATRIX CALCULATED
C
C
C      INTEGER*2 IAXIS,I,J
C      REAL ANGLE,AMAT(3,3),PI180
C      DATA PI180/0.017453/
C
C      IDX(I)=MOD(I+2,3)+1      ! STATEMENT FUNCTION (NOT AN ARRAY)
C
C      IF(IAXIS.LT.1.OR.IAXIS.GT.3) STOP 'PSLIB--AXIS OUT OF BOUNDS'
C      DO 10 I=1,3
C      DO 10 J=1,3
10    AMAT(I,J)=0.E0
C      RADIAN = ANGLE * PI180
C      C=COS(RADIAN)
C      S=SIN(RADIAN)
C      AMAT(IAXIS,IAXIS)=1.E0
C      I=IDX(IAXIS-1)
C      J=IDX(IAXIS+1)
C      AMAT(I,I)=C
C      AMAT(J,J)=C
C      AMAT(I,J)=S
C      AMAT(J,I)=-S
C      RETURN
C      END
C
C
C
C
C      SUBROUTINE TLHT(T,AMAT)
C
C      ROUTINE TO CONVERT A RIGHT HANDED ROTATION MATRIX (T)
C      TO A LEFT HANDED ROTATION MATRIX (AMAT)
C
C CALLING SEQUENCE:
C
C      CALL TLHT(T,AMAT)
C
C      REAL AMAT(3,3),T(3,3)
C
C      AMAT(1,1)=T(1,1)
C      AMAT(2,1)=T(2,1)
C      AMAT(3,1)=-T(3,1)
C      AMAT(1,2)=T(1,2)
C      AMAT(2,2)=T(2,2)
C      AMAT(3,2)=-T(3,2)
C      AMAT(1,3)=-T(1,3)
C      AMAT(2,3)=-T(2,3)
C      AMAT(3,3)=T(3,3)

```

AD-A162 461

STS (SPACE TRANSPORTATION SYSTEM) TASK SIMULATOR(U) AIR 2/2
FORCE ACADEMY CO S ALFANO 15 AUG 85 USAFA-TR-85-12

UNCLASSIFIED

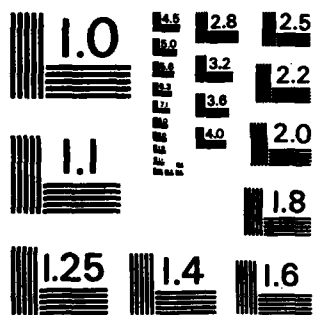
F/G 9/2

NL

END

FORMED

END



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RETURN
END

SUBROUTINE P919CV(MATRIX,N,BUFFER)

ROUTINE TO CONVERT A VAX REAL ARRAY TO ACP FLOATING-POINT FORMAT

THIS ROUTINE CONVERTS AN ARRAY OF VAX SINGLE-PRECISION REAL NUMBERS INTO A NORMALIZED ARRAY OF 32-BIT ACP MANTISSAS, WITH THE ARRAY PRECEDED BY A 16-BIT EXPONENT. THE MOST SIGNIFICANT ELEMENT IN THE ARRAY IS NORMALIZED.

FORTRAN CALLING SEQUENCE:

CALL P919CV(MATRIX,N,BUFFER)

WHERE:

MATRIX IS AN N-ELEMENT REAL*4 ARRAY OF VAX FLOATING-POINT NUMBERS.

N IS THE INTEGER*2 SIZE OF ARRAY MATRIX.

BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH 2N+1, INTO WHICH THE RESULT IS PLACED, WITH THE EXPONENT WORD FIRST, FOLLOWED BY THE ARRAY OF 32-BIT MANTISSAS.

INTEGER*2 N,BUFFER(2*N+1),FWORD(2)

REAL MATRIX(N),DMAX

INTEGER*4 PSMEXP,PSMFRA,PSMNOR,DWORD,EDWORD

EQUIVALENCE (DWORD,FWORD(1))

FIND LARGEST REAL NUMBER TO OBTAIN EXPONENT

DMAX=0.

DO 10 I=1,N

DMAX=AMAX1(DMAX,ABS(MATRIX(I)))

USE EXPONENT OF LARGEST NUMBER FOR NORMALIZATION

DWORD=PSMEXP(DMAX)

BUFFER(1)=FWORD(1)

EDWORD=DWORD

OBTAIN NORMALIZED FRACTIONS AND LOAD INTO BUFFER

DO 20 I=1,N

DWORD=PSMNOR(MATRIX(I),EDWORD)

BUFFER(2+I)=FWORD(2)

BUFFER(2+I+1)=FWORD(1)

CONTINUE

RETURN

END

SUBROUTINE LOOKAT(AT,FM,UP,MAT,VEC)

ROUTINE TO GENERATE LOOK AT, LOOK FROM,
LOOK UP REQUEST TO THE PS300

THIS ROUTINE UPDATES A PS300 DISPLAY LOOK NODE
WITH THE NECESSARY LOOK MATRIX

CALLING SEQUENCE:

```
CALL LOOKAT(AT,FM,UP,MAT,VEC)
```

WHERE:

INDEX IS THE INTEGER*2 NODE SUFFIX ($1 \leq \text{INDEX} \leq 256$) WHICH CORRESPONDS TO ONE OF THE DISPLAY STRUCTURE NODES N001 THROUGH N256.

AT, FM, UP : ARE THE VIEWING VECTORS

MAT : IS THE RESULTING 3X3 VIEWING MATRIX

BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH $2N+1$,

INTO WHICH THE RESULT IS PLACED, WITH THE EXPONENT

WORD FIRST, FOLLOWED BY THE ARRAY OF 32-BIT MANTISSAS.

VEC IS A 3 ELEMENT ARRAY CONTAINING ROW 4

COMPUTES 4X4 MATRIX FOR LOOK FUNCTION

```
REAL AT(3),FM(3),UP(3),MAT(3,3),T(3),VEC(3)
REAL D(3),E(3),M,F(3),G(3),H(3),MAG
```

```
D(1)=AT(1)-FM(1)
```

```
D(2)=AT(2)-FM(2)
```

```
D(3)=AT(3)-FM(3)
```

```
MAG=D(1)**2+D(2)**2+D(3)**2
```

```
IF (MAG .GT. .1E-30) THEN
```

```
  MAG=SQRT(MAG)
```

```
  ELSE
```

```
    MAG=.1E-30
```

```
ENDIF
```

```
D(1)=D(1)/MAG
```

```
D(2)=D(2)/MAG
```

```
D(3)=D(3)/MAG
```

```
E(1)=UP(1)-AT(1)
```

```
E(2)=UP(2)-AT(2)
```

```
E(3)=UP(3)-AT(3)
```

```
M=D(1)*E(1)+D(2)*E(2)+D(3)*E(3)
```

```
F(1)=E(1)-M*D(1)
```

```
F(2)=E(2)-M*D(2)
```

```
F(3)=E(3)-M*D(3)
```

```
IF ((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
```

```
  E(1)=0.0
```

```
  E(2)=1.0
```

```
  E(3)=0.0
```

```
  F(1)=E(1)-M*D(1)
```

```
  F(2)=E(2)-M*D(2)
```

```
  F(3)=E(3)-M*D(3)
```

```
  IF ((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
```

```
    E(1)=0.0
```

```
    E(2)=0.0
```

```
    E(3)=1.0
```

```
    F(1)=E(1)-M*D(1)
```

```
    F(2)=E(2)-M*D(2)
```

```
    F(3)=E(3)-M*D(3)
```

```
  ENDIF
```

```

      ENDIF
C
      MAG=SQRT(F(1)**2+F(2)**2+F(3)**2)
C
      G(1)=F(1)/MAG
      G(2)=F(2)/MAG
      G(3)=F(3)/MAG
C
      H(1)=G(2)*D(3)-G(3)*D(2)
      H(2)=G(3)*D(1)-G(1)*D(3)
      H(3)=G(1)*D(2)-G(2)*D(1)
C
      T(1)=-FM(1)*H(1)-FM(2)*H(2)-FM(3)*H(3)
      T(2)=-FM(1)*G(1)-FM(2)*G(2)-FM(3)*G(3)
      T(3)=-FM(1)*D(1)-FM(2)*D(2)-FM(3)*D(3)
C
      MAT(1,1)=H(1)
      MAT(1,2)=H(2)
      MAT(1,3)=H(3)
      VEC(1)=T(1)
C
      MAT(2,1)=G(1)
      MAT(2,2)=G(2)
      MAT(2,3)=G(3)
      VEC(2)=T(2)
C
      MAT(3,1)=D(1)
      MAT(3,2)=D(2)
      MAT(3,3)=D(3)
      VEC(3)=T(3)
C
C
      RETURN
      END
C
C
C
C
      SUBROUTINE P920CV(MAT,VEC,BUFFER)
C
C      THIS SUBROUTINE FILLS IN A 4 X 4 ARRAY INTO THE BUFFER.
C      YOU LOAD A 3 X 4 MAT, AND AN ARRAY VEC(3) IS THE FOURTH ROW.
C      IT FILLS 34 ELEMENTS IN THE BUFFER.
C
      INTEGER*2 ITEMP1(9), ITEMP2(25), BUFFER(34)
      REAL MAT(3,4), VEC(4), MAT1(4,3)
      DO 100 I=1,3
      DO 99 J=1,4
        MAT1(J,I)=MAT(I,J)
99    CONTINUE
100   CONTINUE
      CALL P919CV(VEC,4,ITEMP1)
      CALL P919CV(MAT1,12,ITEMP2)
      BUFFER(1) = ITEMP1(1)
      DO 5 I = 1,25
        BUFFER(I+1) = ITEMP2(I)
5     CONTINUE
      DO 10 J = 1,8
        BUFFER(J+26) = ITEMP1(J+1)
10    CONTINUE

```

RETURN
END

SUBROUTINE NUMBER(X,BUFFER)

C
C
C
C

THIS SUBROUTINE WILL TAKE ANY NUMBER FROM 9999.9 TO -999.9,
ENCODE IT, AND PUT IT IN THE PROPER PLACE IN THE OUTPUT
BUFFER, IBUF

INTEGER*2 BUFFER(3), TBUF(3)
CHARACTER*6 CHAR
LOGICAL *1 TEMPO,TEMP1(6)
EQUIVALENCE (CHAR,TEMP1,TBUF)

IF(X.GT.9999.9) X=9999.9
IF(X.LT.-999.9) X=-999.9
ENCODE(6,201,CHAR)X
FORMAT(F6.1)

201

TEMPO=TEMP1(1)
TEMP1(1)=TEMP1(2)
TEMP1(2)=TEMPO
TEMPO=TEMP1(3)
TEMP1(3)=TEMP1(4)
TEMP1(4)=TEMPO
TEMPO=TEMP1(5)
TEMP1(5)=TEMP1(6)
TEMP1(6)=TEMPO
BUFFER(1)=TBUF(1)
BUFFER(2)=TBUF(2)
BUFFER(3)=TBUF(3)
RETURN
END

C

C *****
 C <<< LPAIO >>>> LPA11-K I/O ROUTINES; J.M. LIND; 03 MAY 83; REV C
 C *****
 C

 SUBROUTINE LPAIO (MODE,IUNIT,IFLAG,DRATE,ICHAN,NCHAN,NFRAME,IOSUF,
 1 ARATE,ISTAT,ICALL,LSTAT,ISMODE)

C
 C REV A DESIGNED TO USE AST CALLS AT COMPLETION OF SWEEP (FILE NAME LPAIOA.
 C REV B MODIFIED TO USE EVENT FLAGS INSTEAD OF AST CALLS 03 FEB 83 (JML)
 C REV C MODIFIED TO ADD I/O MODE CALLING PARAMETER "ISMODE" AND TO ALLOW
 C FOR 2 CHANNEL DIGITAL I/O WITH THE ADDITION OF MODES 7,8,17, & 18.
 C

C CALLING SEPCIFICATIONS:

C MODE MODE OF CALL WITH:

C MODE = 1 INITIALIZE LPA11-K UNIT IUNIT
 = 3 ANALOG INPUT
 = 4 ANALOG OUTPUT
 = 5 DIGITAL INPUT A
 = 6 DIGITAL OUTPUT A
 = 7 DIGITAL INPUT B
 = 8 DIGITAL OUTPUT B
 =13 ANALOG INPUT STATUS
 =14 ANALOG OUTPUT STATUS
 =15 DIGITAL INPUT STATUS A
 =16 DIGITAL OUTPUT STATUS A
 =17 DIGITAL INPUT STATUS B
 =18 DIGITAL OUTPUT STATUS B

C IUNIT UNIT NUMBER OF THE DESIRED LPA11 SUBSYSTEM:

C IUNIT = 0 USES LAEO:
 = 1 USES LABO:

C IFLAG NUMBER OF THE EVENT FLAG WHICH IS TO BE SET A COMPL
 DRATE DESIRED SAMPLE RATE (DO NOT EXCEED 80 KHZ)
 ICHAN START CHANNEL NUMBER
 NCHAN NUMBER OF CHANNELS (MUST BE 1 FOR DIGITAL I/O)
 NFRAME NUMBER OF FRAMES (NCHAN PER FRAME)
 IOSUF BUFFER FOR DATA (NFRAME * NCHAN 2 BYTE WORDS LONG)

C RETURNED INFORMATION:

C ARATE ACTUAL SAMPLE RATE USED (0 FOR ERROR)
 ISTAT THREE WORD ARRAY WITH:

C ISTAT = 0 ERROR IN CALL
 = 1 SUCCESFUL
 = X VMS ERROR CODE

LSTAT INTEGER*1 (BYTE) VARIABLE USED WITH ISTAT TO DEFINE

ISTAT	LSTAT	MEANING
0	0	NORMAL - BUFFER 0 DONE
-1	1	SWEEP TERMINATED OK
-1	X	X = LPA11 ERROR CODE (USER'S GD P

ICALL CALL NUMBER OF THIS PROGRAM (RELATES TO LPA11 I/O FUNCTION WHICH WAS LAST USED BEFORE RETURN TO THE CALLING PROGRAM)
IF ICALL = 0, THEN MODE IS UNDEFINED!

ISMODE SPECIFY MODE OF LPA11 SWEEP

NOTES:

FOR MODES 5 - 8 (DIGITAL I/O), NCHAN MUST BE 1.

CHANNEL NUMBERING ALWAYS STARTS WITH 0.

IN MODE 1, THE SAMPLE RATE OF THE LPA11 CLOCK

IS SET, AND THE SAME RATE IS USED ON ALL LPA11 FUNCTIONS.

DRATE MUST NOT EXCEED 80 KHZ. HOWEVER, THE LPA11 USER'S

MANUAL SPECIFIES MAXIMUM AGGREGATE THROUGHPUT FOR MULTIR

ACTIVITIES AT 15 KHZ FOR ALL OPERATIONS COMBINED. (PARA 2

WARNINGS:

WE SPECIFYING ISMODE = 512 IN THE DIGITAL INPUT MODE, ONLY ONE CHANNEL OF DIGITAL I/O CAN BE USED AT A TIME.

OTHERWISE, THE PROGRAM WILL HANG WAITING FOR THE INPUT

FLAG TO BE SET BY THE LPA11. WHEN USING ONLY ONE

CHANNEL OF DIGITAL I/O, THE ISMODE = 512 WILL WORK

PROPERLY. IF ISMODE = 0 IS SPECIFIED FOR BOTH DIGIT

INPUT CHANNELS, THEN TWO CHANNELS MAY BE USED AT THE SAME TIME.

WHEN USING A/D OR D/A MODE, YOU MUST SPECIFY AN ISMODE OF 64 IN ORDER TO USE THE MULTIREQUEST MICROCODE WHICH IS LOADED BY THIS ROUTINE.

1. IN THE CASE OF DIGITAL OUTPUT, THE MODE OF THE LPA11 IN RUNNING THE DR11-K IS TO START OUTPUT IMMEDIATELY (THE MODE SPECIFIED IN THE CALL SHOULD BE ISMODE = 0.)

2. IN THE CASE OF DIGITAL INPUT, THE MODE OF THE LPA11 IN RUNNING THE DR11-K IS TO START INPUT ON EXTERNAL TRIGGER (THE MODE SPECIFIED IN THE CALL SHOULD BE ISMODE = 512). THE "EXTERNAL" TRIGGER IS ACTUALLY THE DR11-K "EXTERNAL DATA READY" LINE FOR THE EXTERNAL DEVICE. (SEE DR11-K TIMING DIAGRAM ON PAGE 4-7 OF THE DR11-K INTERFACE USER'S GUIDE AND MAINTENANCE MANUAL.) IN THIS MODE, INTERRUPT WILL OCCUR ONLY AFTER THE EXTERNAL DEVICE CYCLES THE "EXTERNAL DATA READY" LINE.

3. NOTE THAT THE CONFIGURATION OF THE DR11-K JUMPERS IS VERY IMPORTANT TO PROPER OPERATION OF THE DR11-K. IN PARTICULAR, ALL S1 AND S2 SWITCHES SHOULD BE OFF TO DISABLE INTERRUPT BY TRANSITION OF THE DATA BITS (SEE TABLE 5-3). IN ADDITION, JUMPERS W5 - W20 MUST BE IN POSITION "B" IN ORDER TO READ DATA DIRECT FROM THE DATA INPUT LINES (AS OPPOSED TO THE BUFFER REGISTER INPUT). THIS IS DUE TO THE FACT THAT IN "BUFFER REGISTER" MODE, THE INDIVIDUAL DATA BITS IN THE BUFFER ARE SET ONLY ON TRANSITION OF THE DATA LINE (SEE PARAGRAPH 4-6 OF DR11-K INTERFACE USER'S GUIDE). SINCE

C ALL SWITCHES ON S1 AND S2 ARE OFF, THE STATE OF JUMPERS W1-W4 IS A
C DON'T CARE. JUMPERS W21-W23 SHOULD BE SET FOR APPROPRIATE POLARITY
C OF THE INTERNAL DATA ACCEPT AND INTERNAL DATA READY LINES.
C

C 4. REMBER THAT ON DIGITAL OUTPUT, THIS PROGRAM SPECIFIES AT
C LEAST A 150 MICROSECOND DELAY BEFORE OUTPUT OF THE FIRST DIGITAL WORD
C (SEE PAGE 2-14 OF THE LPA11 USER'S GUIDE). THIS IS NECESSARY
C IN ORDER TO ALLOW TIME FOR THE LPA11 TO RETRIEVE DATA FROM
C MEMORY.

C
C PARAMETER USAGE

C	MODE	IUNIT	IFLAG	DRATE	ICHAN	NCHAN	NFRAME	IOBUF	ARA	IS
C	1	X		X					X	X
C	3	X	X		X	X	X	X		X
C	4	X	X		X	X	X	X		X
C	5	X	X		X	X	X	X		X
C	6	X	X		X	X	X	X		X
C	7	X	X		X	X	X	X		X
C	8	X	X		X	X	X	X		X
C	13									X
C	14									X
C	15									X
C	16									X
C	17									X
C	18									X

C ***** PROGRAM DECLARATIONS *****

C VARIABLE DEFINITION SECTION

C IOBUF DATA BUFFER AREA (INTEGER*2)
 C XYBUF LPA11 CONTROL BLOCK (50 LONGWORDS)
 C XSTAT LPA11 COMPLETION STATUS (FORM LPASIGTBUF CALL)
 C IFLAG LPA11 FLAG TO BE SET AT COMPLETION OF SWEEP
 C XYIOSB I/O STATUS BLOCK FOR LPA11 (4 WORDS)
 C DXIOSZ I/O STATUS BLOCK FOR DIGITAL I/O TO BUFFER Z (CH A OR B)
 C XYMSKB LPA11 SUBSYSTEM MASKS AND NUM BUFFER
 C ISTAT LPA11 STATUS LONGWORD
 C LSTAT LPA11 I/O COMPLETION STATUS BYTE
 C NBUF NUMBER OF BUFFERS TO BE FILLED (LONGWORD)

C WHERE "XY" IS AD FOR ANALOG-TO-DIGITAL
 C DA FOR DIGITAL-TO-ANALOG
 C DI FOR DIGITAL INPUT
 C DO FOR DIGITAL OUTPUT

C VARIABLE TYPE SPECIFICATIONS

REAL	LPASXRATE
INTEGER*4	SYSSCLREF
INTEGER*2	IOBUF(1),ADIOSB(4),DAIOSB(4)
INTEGER*2	DIIOSA(4),DOIOSA(4),DIIOSB(4),DOIOSB(4)
INTEGER*4	ADMSKB(2),DAMSKB(2)
INTEGER*4	DIMSKA(2),DOMSKA(2),DIMSKB(2),DOMSKB(2)
INTEGER*4	ISTAT,BUFNUM,NBUF,IFLAG
BYTE	IDSC,IEMC,LSTAT
INTEGER*2	IDSW,IEMW
SET AREA FOR CONTRCL	
INTEGER*4	ADBUF(50),DABUF(50)
INTEGER*4	DIBUFA(50),DOBUFA(50),DIBUFB(50),DOBUFB(50)
EQUIVALENCE	(ADIOSB(1),ADBUF(1)),(DAIOSB(1),DABUF(1))
EQUIVALENCE	(DIIOSA(1),DIBUFA(1)),(DOIOSA(1),DOBUFA(1))
EQUIVALENCE	(DIIOSB(1),DIBUFB(1)),(DOIOSB(1),DOBUFB(1))

C ***** START OF PROGRAM *****

C DETERMINE MODE

C

C LSTAT = 0
 GO TO (100,50,300,400,500,600,700,800,50,50,50,50,
 1 1300,1400,1500,1600,1700,1800),MODE

C EXECUTION STARTS HERE IF MODE IS UNDEFINED

50 ICALL = 0
 ISTAT = 0
 GO TO 1950

C

C

C ***** MODE = 1 *****

C LOAD LPA11 SPECIFIED BY IUNIT WITH MICROCODE FOR MULTIREQUEST MODE

C

100 CONTINUE
 ICALL = 101
 CALL LPASLOADMC (1,IUNIT,ISTAT,IERROR)
 IF (.NOT. ISTAT) GO TO 1950

C

C USE XRATE ROUTINE TO CALCULATE RATE AND PRESET VALUES FOR CLOCK A

C

C RATES ARE SUPPLIED/RETURNED BUT LPASXRATE REQUIRES INTERVALS

 AINTRVL = 1./DRATE
 ICALL = 102
 ACTUAL = LPASXRATE (AINTRVL,IRATE,IPRSET,0)
 ARATE = 1./ACTUAL

C

C SET CLOCK RATE TO SPECIFIED SAMPLE RATE (DO NOT EXCEED ABOUT 80 KHZ)

C

 ICALL = 103
 CALL LPASCLOCKA (IRATE,IPRSET,ISTAT,IUNIT)
 GO TO 1950

```

C ***** MODE = 3 *****
C START ANALOG-TO-DIGITAL INPUT SWEEP
C
300    CONTINUE
C
C CLEAR A/D EVENT FLAG
      ICALL = 300
      ISTAT = SYSSCLREF (XVAL(IFLAG))
      IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE ADBUF ARRAY FOR SWEEP
C
      ICALL = 301
      CALL LPASSETIBF (ADBUF,ISTAT,ADMSKB,I0BUF)
      IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
      ICALL = 302
      CALL LPASLANSKS (ADMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
      ICALL = 303
      CALL LPASRLSBUF (ADBUF,ISTAT,0)
      IF (.NOT. ISTAT) GO TO 1950
C
C START A/D SWEEP BY SPECIFYING ONLY ONE BUFFER
C
      NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
      NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
      ICALL = 304
      CALL LPASADSWP (ADBUF,NPOINT,NBUF,ISMODE,,XVAL(IFLAG),,ICHAN,
1 NCHAN,ISTAT)
      IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF A/D SWEEP CALL
      GO TO 1950
C

```

```

C ***** MODE = 4 *****
C START DIGITAL-TO-ANALOG OUTPUT SWEEP
C
  400    CONTINUE
C
C CLEAR D/A EVENT FLAG
      ICALL = 400
      ISTAT = SYSSCLREF (XVAL(IFLAG))
      IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE DABUF ARRAY FOR SWEEP
C
      ICALL = 401
      CALL LPASSETIBF (DABUF,ISTAT,DAMSKB,I0BUF)
      IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
      ICALL = 402
      CALL LPASLAMS KS (DAMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
      ICALL = 403
      CALL LPASRLSBUF (DABUF,ISTAT,0)
      IF (.NOT. ISTAT) GO TO 1950
C
C CALCULATE NUMBER OF DATA POINTS
      NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
      NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C IN D/A MODE, A DELAY OF AT LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE. SINCE THE LPASXRATE CALL RETURNS THE
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET. (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
      LDELAY = 1
      IF (IRATE.EQ.1) LDELAY = 150
      IF (IRATE.EQ.2) LDELAY = 15
      IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
      IDWELL = 1
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
      ICALL = 404
      CALL LPASDASWP (DABUF,NPOINT,NBUF,ISMODE,IDWELL,XVAL(IFLAG),
1 LDELAY,ICHAN,NCHAN,ISTAT)
      IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF D/A SWEEP CALL
      GO TO 1950

```

```

C ***** MODE = 5 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
500    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
      ICALL = 500
      ISTAT = SYSSCLREF (XVAL(IFLAG))
      IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
      ISTAT = 0
      ICALL = 501
      IF (NCHAN.NE.1) GO TO 1950
      ISTAT = 1
C
C INITIALIZE DIBUFA ARRAY FOR SWEEP
C
      ICALL = 502
      CALL LPA$SETIBF (DIBUFA,ISTAT,DIMSKA,IOBUF)
      IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
      IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
      IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
      IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
      IEMW = -1
      ICALL = 503
      CALL LPA$LAMSKS (DIMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
      ICALL = 504
      CALL LPA$RLSBUF (DIBUFA,ISTAT,0)
      IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
      NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
      NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
      ICALL = 505
      CALL LPA$DISWP (DIBUFA,NPOINT,NBUF,ISMODE,,XVAL(IFLAG),,
1 ICHAN,NCHAN,ISTAT)
      IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT S... P C
      GO TO 1950

```

```

C ***** MODE = 6 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
C 600 CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
C ICALL = 600
C ISTAT = SYSSCLREF (XVAL(IFLAG))
C IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
C ISTAT = 0
C ICALL = 601
C IF (NCHAN.NE.1) GO TO 1950
C ISTAT = 1
C
C INITIALIZE DOBUFA ARRAY FOR SWEEP
C
C ICALL = 602
C CALL LPASSETIBF (DOBUFA,ISTAT,DOMSKA,IOBUF)
C IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
C IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
C IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
C IDSW = -1
C SPECIFY EVENT MARK WORD MASK OF ALL BITS
C IEMW = -1
C ICALL = 603
C CALL LPASLAMSKS (DOMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
C ICALL = 604
C CALL LPASRLSBUF (DOBUFA,ISTAT,0)
C IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
C NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
C NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE. SINCE THE LPASXRATE CALL RETURNS THE
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET. (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
C LDELAY = 1
C IF (IRATE.EQ.1) LDELAY = 150
C IF (IRATE.EQ.2) LDELAY = 15
C IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
C IDWELL = 1
C PROCEED WITH SWEEP START CALL

```

C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION

ICALL = 605

CALL LPASDOSWP (DOBUFA,NPOINT,NBUF,ISMODE,IDWELL,XVAL(FLAG

1 LDELAY,ICHAN,NCHAN,ISTAT)

IF (.NOT. ISTAT) GO TO 1950

C

C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP
GO TO 1950

```

C ***** MODE = 7 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
700    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
      ICALL = 700
      ISTAT = SYSSCLREF (XVAL(IFLAG))
      IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
      ISTAT = 0
      ICALL = 701
      IF (NCHAN.NE.1) GO TO 1950
      ISTAT = 1
C
C INITIALIZE DIBUFB ARRAY FOR SWEEP
C
      ICALL = 702
      CALL LPASSETIBF (DIBUFB,ISTAT,DIMSKB,IOBUF)
      IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
      IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
      IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
      IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
      IEMW = -1
      ICALL = 703
      CALL LPASLAMS (DIMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
      ICALL = 704
      CALL LPASRLSBUF (DIBUFB,ISTAT,0)
      IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
      NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
      NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
      ICALL = 705
      CALL LPASDISWP (DIBUFB,NPOINT,NBUF,ISMODE,,XVAL(IFLAG),,
1 ICHAN,NCHAN,ISTAT)
      IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT SWEEP CALI
GO TO 1950

```

```

C ***** MODE = 8 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
C 800    CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
C       ICALL = 800
C       ISTAT = SYSSCLREF (XVAL(IFLAG))
C       IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
C       ISTAT = 0
C       ICALL = 801
C       IF (NCHAN.NE.1) GO TO 1950
C       ISTAT = 1
C
C INITIALIZE DOBUF8 ARRAY FOR SWEEP
C
C       ICALL = 802
C       CALL LPASSETIBF (DOBUF8,ISTAT,DOMSKB,IOBUF)
C       IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
C       IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
C       IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
C       IDSW = -1
C SPECIFY EVENT MARK WORD MASK OF ALL BITS
C       IEMW = -1
C       ICALL = 803
C       CALL LPASLMSKS (DOMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
C       ICALL = 804
C       CALL LPASRLSBUF (DOBUF8,ISTAT,0)
C       IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
C       NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
C       NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFO
C THE FIRST CONVERSION TAKES PLACE. SINCE THE LPASXRATE CALL RETURNS TH
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET. (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
C       LDELAY = 1
C       IF (IRATE.EQ.1) LDELAY = 150
C       IF (IRATE.EQ.2) LDELAY = 15
C       IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
C       IDWELL = 1

```

C PROCEED WITH SWEEP START CALL

C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION

ICALL = 805

CALL LPASDOSWP (DOBUF, NPOINT, NBUF, ISMODE, IDWELL, XVAL(IFLAG),

1 LDELAY, ICHAN, NCHAN, ISTAT)

IF (.NOT. ISTAT) GO TO 1950

C

C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP CAL
GO TO 1950

```
C ***** MODE = 13 *****
C GET STATUS OF A/D SWEEP
C
1300 CONTINUE
C
      ICALL = 1301
      ISTAT = LPASIWTBUF(ADBUF)
      LSTAT = IAND(ADIOSB(3),'FF00'X)/256
      GO TO 1950
C
C
C ***** MODE = 14 *****
C GET STATUS OF D/A SWEEP
C
1400 CONTINUE
C
      ICALL = 1401
      ISTAT = LPASIWTBUF(DABUF)
      LSTAT = IAND(DAIOSB(3),'FF00'X)/256
      GO TO 1950
C
C
C ***** MODE = 15 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
1500 CONTINUE
C
      ICALL = 1501
      ISTAT = LPASIWTBUF(DIBUFA)
      LSTAT = IAND(DIIOSA(3),'FF00'X)/256
      GO TO 1950
C
C
C ***** MODE = 16 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
1600 CONTINUE
C
      ICALL = 1601
      ISTAT = LPASIWTBUF(DOBUFA)
      LSTAT = IAND(DOIOSA(3),'FF00'X)/256
      GO TO 1950
C
C
C ***** MODE = 17 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
1700 CONTINUE
C
      ICALL = 1701
      ISTAT = LPASIWTBUF(DIBUFB)
      LSTAT = IAND(DIIOSB(3),'FF00'X)/256
      GO TO 1950
C
C
C ***** MODE = 18 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
1800 CONTINUE
C
```

```
ICALL = 1801
ISTAT = LPASIWTFB(DOBUEB)
LSTAT = IAND(DOIOSB(3),'FF00'X)/256
GO TO 1950
```

C
C
C
C
C

***** ERROR SERVICE ROUTINE *****

C

1950 CONTINUE

C

TRANSFER STATUS INFORMATION ON CALLING PARAMETER

C

RETURN

C

END

```

SUBROUTINE LVLH(C,CO,Q,REF,DBX,DBY,DBZ,WB,JET,DELTAT,MAXWX,
* MAXWY,MAXWZ,PICK,T)

```

```

C
C      ATTEMPTS TO HOLD BODY AXES IN ALIGNMENT WITH LVLH AXES
C
      INTEGER C(12),CO(12),SNAP,I,J,PICK
      REAL Q(4),REF(4),YAW,PITCH,ROLL,DBX,DBY,DBZ
      REAL WB(3),JET(4,12),DELTAT,MAXWX,MAXWY,MAXWZ
      REAL ANG(3),T(3,3),PI,SWT,WT
      REAL FQ(4),FWB(3),FT(3,3),QT(3,3),REFT(3,3)
      REAL CONST,QQ(4)

C
      PI=2.0*ACOS(0.0)

C
C      WAS RMC JUST MOVED TO NEUTRAL ? IF SO, TAKE SNAPSHOT.
C      REF(4) - QUATERNION AT TIME OF SNAPSHOT.
C      SNAP - FLAG TO TAKE SNAPSHOT OF QUATERNION.
C
      SNAP=0
      DO 150 I=7,12
        IF ((CO(I)-C(I)) .GT. 0) SNAP=1
150    CONTINUE
C
      IF (SNAP .EQ. 1) THEN
        REF(1)=Q(1)
        REF(2)=Q(2)
        REF(3)=Q(3)
        REF(4)=Q(4)
      ENDIF

C
C      COMPUTE FUTURE QUATERNION (NEXT ITERATION)
C      FWB - FUTURE BODY RATE
C      FQ - FUTURE QUATERNION
C
      ANG(1)=0.0
      ANG(2)=0.0
      ANG(3)=0.0
C
      FQ(1)=Q(1)
      FQ(2)=Q(2)
      FQ(3)=Q(3)
      FQ(4)=Q(4)
C
      FWB(1)=WB(1)
      FWB(2)=WB(2)
      FWB(3)=WB(3)
C
      CALL ROTATE(ANG,DELTAT,FWB,FQ,FT)

C
C      COMPUTE TRANSFORMATION FROM REF FRAME TO C-W FRAME
C
      CALL TRNSFM(REFT,REF)

C
C      COMPUTE TRANSFORMATION FROM FQ FRAME TO REF FRAME
C
      DO 81 I=1,3
        DO 82 J=1,3
          QT(I,J)=REFT(1,I)*FT(1,J)+REFT(2,I)*FT(2,J)+
*          REFT(3,I)*FT(3,J)
82    CONTINUE

```

```

81  CONTINUE
C
C      COMPUTE QUATERNIONS (FROM FQ TO REF)
C
QQ(4)=1.0+QT(1,1)+QT(2,2)+QT(3,3)
IF (QQ(4) .LT. .1E-30) QQ(4)=.1E-30
QQ(4)=SQRT(QQ(4))/2.0
QQ(1)=(QT(3,2)-QT(2,3))/(4.0*QQ(4))
QQ(2)=(QT(1,3)-QT(3,1))/(4.0*QQ(4))
QQ(3)=(QT(2,1)-QT(1,2))/(4.0*QQ(4))
C
C      NORMALIZE QUATERNIONS
C
CONST=SQRT(QQ(1)*QQ(1)+QQ(2)*QQ(2)+
*      QQ(3)*QQ(3)+QQ(4)*QQ(4))
QQ(1)=QQ(1)/CONST
QQ(2)=QQ(2)/CONST
QQ(3)=QQ(3)/CONST
QQ(4)=QQ(4)/CONST
C
C      COMPUTE ANGLE OF ROTATION BETWEEN FRAMES
C
WT=2.0*ACOS(QQ(4))
SWT=SIN(WT/2.0)
C
C      COMPUTE ANGULAR DIFFERENCE (OF FQ W.R.T. REF)
C
IF (WT .LT. .0000001) THEN
    ROLL=0.0
    PITCH=0.0
    YAW=0.0
ELSE
    ROLL=QQ(1)*WT/SWT
    PITCH=QQ(2)*WT/SWT
    YAW=QQ(3)*WT/SWT
ENDIF
C
C      DETERMINE THRUST PROFILE BY CHECKING ROLL, PITCH
C      AND YAW ANGLES/RATES AGAINST ATTITUDE HOLD CRITERIA.
C
CALL CHECK(ROLL,DBX,WB(1),C(7),C(8),JET(PICK,7),
*  JET(PICK,8),MAXWX,DELTAT)
CALL CHECK(PITCH,DBY,WB(2),C(9),C(10),JET(PICK,9),
*  JET(PICK,10),MAXWY,DELTAT)
CALL CHECK(YAW,DBZ,WB(3),C(11),C(12),JET(PICK,11),
*  JET(PICK,12),MAXWZ,DELTAT)
C
END

```

SUBROUTINE MODE(SA,SB,SC,LA,LB,LC)

TAKES ONE COLUMN OF DAP PANEL MANUAL MODE AND
FIGURES OUT WHAT LIGHTS TO TURN ON/OFF (LA,LB,LC)
USING SWITCH COMMANDS (SA,SB,SC)

INTEGER SA,SB,SC,LA,LB,LC

IF (SA .EQ. 1) THEN

LA=1

LB=0

LC=0

ENDIF

IF (SB .EQ. 1) THEN

LA=0

LB=1

LC=0

ENDIF

IF (SC .EQ. 1) THEN

LA=0

LB=0

LC=1

ENDIF

END

SUBROUTINE PREDPATH(OMEGA,TIME,OLDX,X)

THIS SUBROUTINES PREDICTS POSITION (X) RELATIVE TO
TARGET GIVEN PRESENT POSITION AND SPEED (OLDX) AND
FUTURE TIME (TIME)

REAL OMEGA,X(3),OLDX(6),TIME,CW,SW

CW=COS(OMEGA*TIME)
SW=SIN(OMEGA*TIME)

X(1)=OLDX(1)-2.0*OLDX(5)/OMEGA
X(1)=X(1)-3.0*(OLDX(4)+2.0*OMEGA*OLDX(2))*TIME
X(1)=X(1)+2.0*(3.0*OLDX(2)+2.0*OLDX(4)/OMEGA)*SW
X(1)=X(1)+2.0*OLDX(5)*CW/OMEGA

X(2)=4.0*OLDX(2)+2.0*OLDX(4)/OMEGA
X(2)=X(2)-CW*(3.0*OLDX(2)+2.0*OLDX(4)/OMEGA)
X(2)=X(2)+OLDX(5)*SW/OMEGA

X(3)=OLDX(6)*SW/OMEGA+OLDX(3)*CW

SCALE DOWN FOR HUD DISPLAY
CONVERT TO LH SYSTEM

X(1)=X(1)/100.
X(2)=X(2)/100.
X(3)=-X(3)/100.

END

SUBROUTINE PS300(ALTD,INCLIN)

THIS SUBROUTINE IS THE GRAPHICS PROGRAM FOR THE SHUTTLE SIMULATOR. IT GENERATES A TARGET, ROTATING EARTH, HORIZON AND STAR FIELD FOR BACKGROUND. SUBROUTINE LOOK SENDS NEW DATA TO THIS PROGRAM IN THE PS300 TO UPDATE TARGET RANGE AND ATTITUDE AND EARTH/STAR ROTATIONS.

INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

LOGICAL*1 POSLIN(73)
 REAL ALT,INCLIN,AT(3),FM(3),UP(3),DSTAR
 REAL HERR,HORT,HORD,LONG,ANG,ALTD
 REAL*4 V(3)
 INTEGER I,NVEC
 DIMENSION VEC(4,73)

DATA POSLIN/.FALSE.,72*.TRUE./

ATTACH GRAPHICS DEVICE AND INITIALIZE GRAPHICS

CALL PATTCH('LOGDEVNAM=PIAO:/PHYDEVTYP=PARALLEL',ERR)
 CALL PINIT(ERR)

V(3) - VECTOR ARRAY FOR PS300
 HERR - RADIUS OF EARTH HORIZON AS SEEN FROM TARGET (DU)
 HORD - DISTANCE OF HORIZON FROM EARTH CENTER (DU)
 HORT - DISTANCE OF HORIZON FROM TARGET (DU)
 DSTAR - DISTANCE OF STAR CLIPPING PLANE FROM TARGET (DU)

COMPUTE EARTH HORIZON RADIUS (HERR),
 DISTANCE FROM EARTH CENTER (HORD),
 AND DISTANCE FROM TARGET TO HORIZON (HORT)

ALT=ABS(ALTD/6378.135)
 HORD=1.0/(1.0+ALT)
 HERR=SQRT(1.0-HORD*HORD)
 HORT=HERR/HORD

UNITS VARY FROM METERS TO DUS DEPENDING ON WHAT IS BEING DISPLAYED. THIS UNIT JUGGLING IS DONE TO MINIMIZE SCALING ERRORS INHERENT IN THE PS300.

INITIALIZE VECTORS FOR PS300 VIEWING

V - DUMMY VECTOR
 AT,FM,UP - VIEWING VECTORS

V(1)=0.0
 V(2)=0.0
 V(3)=0.0
 AT(1)=0.0
 AT(2)=0.0
 AT(3)=0.0
 FM(1)=0.0
 FM(2)=0.0
 FM(3)=-20.0
 UP(1)=0.0
 UP(2)=10.0


```

CALL PCHS('RANGE',12.,79.,1.,1.,0.,'000000',ERR)
CALL PCHS('RRATE',12.,77.,1.,1.,0.,'000000',ERR)
CALL PCHS('RFUEL',12.,75.,1.,1.,0.,'000000',ERR)
CALL PCHS('RTIME',12.,73.,1.,1.,0.,'000000',ERR)
CALL PENDS(ERR)

```

COMPUTE DISPLAY DATA FOR SPINNING SATELLITE

```

CALL PBEGS('TARGET',ERR)
UNITS ARE IN METERS
CALL PSEDCL('CLIP',.FALSE.,'','',ERR)
CALL PVIEWP('','',-1.0,1.0,-1.0,1.0,1.0,1.0,'','',ERR)
CALL PFOV('FOV',35.0,1.0,3000.,'','',ERR)
CALL PLOOKA('LOOK',AT,FM,UP,'','',ERR)
CALL PINST('','',SATEL',ERR)
CALL PENDS(ERR)

```

CREATE A TDRS SATELLITE

```

CALL PBEGS('SATEL',ERR)
V(1)=-1.8
V(2)=0.0
V(3)=0.0
CALL PTRANS('','',V,'','',ERR)
CALL PROTX('ROTX',0.0,'','',ERR)
V(1)=.2
V(2)=.2
V(3)=.2
CALL PSCALE('','',V,'','',ERR)
CALL PSECOL('','',300.0,1.0,'','',ERR)
DO 150 I=10,360,10
  LONG=I
  CALL PROTX('','',LONG,'RTDRS',ERR)
150 CONTINUE
CALL PENDS(ERR)

```

CREATE A SPINNING/INCLINED GLOBE OF THE EARTH,
ADD A STAR FIELD AND SET THE ENTIRE PICTURE
COUNTER-ROTATING W.R.T. TARGET ORBIT.
ADD A STATIONARY HORIZON

```

CALL PBEGS('GLOBE',ERR)
UNITS ARE IN DU
CALL PSEDCL('CLIP',.TRUE.,'','',ERR)
CALL PVIEWP('','',-1.0,1.0,-1.0,1.0,1.0,1.0,'','',ERR)
CALL PFOV('','',35.0,.1*ALT,
*      HORT+.01,'','',ERR)
CALL PLOOKA('LOOK',AT,FM,UP,'','',ERR)
V(1)=0.0
V(2)=(-1.-ALT)
V(3)=0.0
CALL PTRANS('','',V,'','',ERR)
CALL PINST('','',HORIZON',ERR)
CALL PROTZ('TROT',0.0,'','',ERR)
CALL PSECCL('','',240.,1.,'','',ERR)
CALL PROTX('','',90.-INCLIN,'','',ERR)
CALL PROTY('EROT',0.0,'','',ERR)
CALL PINST('','',WORLD',ERR)
CALL PSECCL('','',240.,0.0,'','',ERR)

```

```

      CALL PINST('','SPHERE',ERR)
      CALL PINST('','LATLINE',ERR)
      CALL PENDS(ERR)

C
C      CREATE A HORIZON FROM A CIRCLE
C
      CALL PBEGS('HORIZON',ERR)
      UNITS ARE IN DU
      V(1)=0.0
      V(2)=HORD
      V(3)=0.0
      CALL PTRANS('','V','',ERR)
      CALL PROTX('','90.','','ERR)
      V(1)=HORD
      V(2)=V(1)
      V(3)=V(1)
      CALL PSCALE('','V','CIRCLE',ERR)
      CALL PENDS(ERR)

C
C      CREATE A STAR FIELD
C
      CALL PBEGS('STARS',ERR)
      UNITS ARE IN DU
      CALL PSEDCL('CLIP',.TRUE.,'',ERR)
      CALL PVIEWP('','-1.0,1.0,-1.0,1.0,1.0,1.0,','',ERR)
      DSTAR=HORD+.9*SQRT((2.*ALT+1.)*2-1.)
      CALL PFOV('',35.0,.1*DSTAR,DSTAR,'',ERR)
      CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
      V(1)=0.0
      V(2)=-1.-ALT
      V(3)=0.0
      CALL PTRANS('','V','',ERR)
      CALL PROTZ('TROT',0.0,'',ERR)
      CALL PSECOL('','180.,0.','','ERR)
      CALL PROTY('','90.,'STARS.TWINKLE',ERR)
      CALL PROTX('','90.,'STARS.TWINKLE',ERR)
      V(1)=1.+2.*ALT
      V(2)=V(1)
      V(3)=V(1)
      CALL PSCALE('TWINKLE',V,'STAR',ERR)
      CALL PENDS(ERR)

C
C      BUILD A SPHERE FROM A CIRCLE
C
      CALL PBEGS('SPHERE',ERR)
      DO 200 I=10,180,10
      LONG=I
      CALL PROTY('','LONG','CIRCLE',ERR)
200    CONTINUE
      CALL PENDS(ERR)

C
C      BUILD A HEMI-SPHERE FROM A SEMI-CIRCLE
C
      CALL PBEGS('HEMI',ERR)
      DO 222 I=15,180,15
      LONG=I
      CALL PROTY('','LONG','SEMI',ERR)
222    CONTINUE
      CALL PENDS(ERR)

```

```

C
C
C      COMPUTE LINES OF LATITUDE USING CIRCLES
C
CALL PBEGS('LATLINE',ERR)
CALL PROTX(' ',90.0,'CIRCLE',ERR)
CALL PROTX(' ',90.0,'LAT',ERR)
CALL PROTX(' ', -90.0,'LAT',ERR)
CALL PENDS(ERR)

C
C
CALL PBEGS('LAT',ERR)
DO 20 I=10,80,10
  ANG=I*.0174532925
  V(1)=0.0
  V(2)=0.0
  V(3)=SIN(ANG)-SIN(ANG-.174532925)
  CALL PTRANS(' ',V,' ',ERR)
  V(1)=COS(ANG)
  V(2)=V(1)
  V(3)=0.0
  CALL PSCALE(' ',V,'CIRCLE',ERR)
20  CONTINUE
CALL PENDS(ERR)

C
C
C      VECTOR LIST FOR CIRCLE
C
DO 10 I=1,73
  ANG=5.0*(I-1)*.0174532925
  VECS(1,I)=COS(ANG)
  VECS(2,I)=SIN(ANG)
  VECS(3,I)=0.0
  VECS(4,I)=1.0
10  CONTINUE

C
NVEC=73
CALL PVCBEG('CIRCLE',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
CALL PVCEND(ERR)

C
C
C      VECTOR LIST FOR SEMI-CIRCLE
C
DO 12 I=1,13
  ANG=(-90.0+15.0*(I-1))*0.0174532925
  VECS(1,I)=SIN(ANG)
  VECS(2,I)=COS(ANG)
  VECS(3,I)=0.0
  VECS(4,I)=1.0
12  CONTINUE

C
NVEC=13
CALL PVCBEG('SEMI',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
CALL PVCEND(ERR)

C
C
C      COMMANDS FOR HUD ROTATION
C
SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION

```

```

C      CALL PSNST(' HUD ',1,'DLABEL1',ERR)
C      CALL PFN('HUD','YROTATE',ERR)

```

```

C      CONNECT INPUTS TO ACCUMULATOR
C

```

```

C      CALL PFN('ACC1','ACCUMULATE',ERR)
C
C      CALL PCONN('DIALS',1,1,'ACC1',ERR)
C      CALL PSNREA(180.,2,'ACC1',ERR)
C      CALL PSNREA(1.,3,'ACC1',ERR)
C      CALL PSNREA(-15.,4,'ACC1',ERR)
C      CALL PSNREA(360.,5,'ACC1',ERR)
C      CALL PSNREA(0.,6,'ACC1',ERR)

```

```

C      CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION
C      AND THEN TO PROGRAM.
C

```

```

C      CALL PCONN('ACC1',1,1,'HUD',ERR)
C      CALL PCONN('HUD',1,1,'XYPTH.ROTY',ERR)
C

```

```

C      CALL NEEDED VECTOR LISTS
C

```

```

C      CALL VECTOR ('DIAMOND',7)
C      CALL VECTOR ('RTDRS',5)
C      CALL VECTOR ('STICKSTS',8)
C      CALL VECTOR ('XYAXIS',6)
C      CALL VECTOR ('XARROW',6)
C      CALL VECTOR ('YARROW',6)
C      CALL VECTOR ('ZARROW',6)
C      CALL VECTOR ('PIPPER',6)
C      CALL VECTOR ('WORLD',5)
C      CALL VECTOR ('STAR',4)

```

```

C      DISPLAY ALL
C

```

```

C      CALL PDISP('XYPTH',ERR)
C      CALL PDISP('GUNSITE',ERR)
C      CALL PDISP('TARGET',ERR)
C      CALL PDISP('GLOBE',ERR)
C      CALL PDISP('STARS',ERR)
C      CALL PDISP('INFO',ERR)
C      CALL PDISP('XYZVC',ERR)

```

```

C      DETACH GRAPHICS DEVICE
C

```

```

C      CALL PDTACH(ERR)
C

```

```

C      END

```

SUBROUTINE PULSE(COUNT1,COUNT2,C1,C2,DEL,JET1,JET2)

DETERMINES HOW MANY PULSES ARE NEEDED FOR THRUST COMMAND.
(COUNT1,COUNT2). RESETS CONTROL SWITCHES (C1,C2) TO
EXECUTE PROPER NUMBER OF PULSES REGARDLESS OF THC/RHC
POSITION.

INTEGER COUNT1,COUNT2,C1,C2
REAL JET1,JET2,DEL

COUNT DOWN ONE PULSE

COUNT1=COUNT1-1
COUNT2=COUNT2-1

IF JET1 OR JET2 IS ZERO, EXIT SUBROUTINE

IF (ABS(JET1) .LE. .00000001) GOTO 150
IF (ABS(JET2) .LE. .00000001) GOTO 150

SET COUNTERS IF NOT PULSING FROM PREVIOUS COMMAND.

DEL - DESIRED VELOCITY INCREMENT
JET1,2 - VELOCITY INCREMENT GAINED BY ONE PULSE

IF (COUNT1 .LT. 1) COUNT1=ABS(INT(C1*DEL/JET1))
IF (COUNT2 .LT. 1) COUNT2=ABS(INT(C2*DEL/JET2))

SET CONTROL SWITCHES BASED ON COUNTERS

IF (COUNT1 .GT. 0) C1=1
IF (COUNT2 .GT. 0) C2=1

150 END

```

C      SUBROUTINE QDOT(Q,WX,WY,WZ,DELTAT)
C      FIGURES QUATERNION RATE (QD) AND LINEARLY INTEGRATES
C      FOR TIME STEP DELTAT.
C
C      REAL Q(4),QD(4),WX,WY,WZ,DELTAT,CONST
C      INTEGER J
C
C      QD(1)=(Q(2)*WZ-Q(3)*WY+Q(4)*WX)*.5
C      QD(2)=(-Q(1)*WZ+Q(3)*WX+Q(4)*WY)*.5
C      QD(3)=(Q(1)*WY-Q(2)*WX+Q(4)*WZ)*.5
C      QD(4)=(-Q(1)*WX-Q(2)*WY-Q(3)*WZ)*.5
C
C      DO 100 J=1,4
C          Q(J)=Q(J)+DELTAT*QD(J)
100    CONTINUE
C
C      NORMALIZE QUATERNIONS
C
C      CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
C      Q(1)=Q(1)/CONST
C      Q(2)=Q(2)/CONST
C      Q(3)=Q(3)/CONST
C      Q(4)=Q(4)/CONST
C
C      END

```

```

C      SUBROUTINE ROTATE(ANG,DELTAT,WB,Q,T)
C
C      LINEARLY INTEGRATES TO FIND ROTATION IN BODY FRAME, THEN
C      TRANSFORMS TO REFERENCE FRAME.
C
C      REAL ANG(3),WB(3),DELTAT,Q(4),T(3,3)
C
C      COMPUTE BODY RATES (WB)
C      (ANG IS ANGULAR ACCELERATION)
C
C      WB(1)=WB(1)+DELTAT*ANG(1)
C      WB(2)=WB(2)+DELTAT*ANG(2)
C      WB(3)=WB(3)+DELTAT*ANG(3)
C
C      FIND QUATERNION RATE AND INTEGRATE
C
C      CALL QDOT(Q,WB(1),WB(2),WB(3),DELTAT)
C
C      COMPUTE ROTATION MATRIX
C
C      CALL TRNSFM(T,Q)
C
C      END

```

SUBROUTINE RTOB(T,REF,90D)

C
C TRANSFORMS FROM REF TO BOD FRAME GIVEN TRANSFORMATION MATRIX T
C

REAL T(3,3),REF(3),BOD(3)

C
90D(1)=REF(1)*T(1,1)+REF(2)*T(2,1)+REF(3)*T(3,1)
BOD(2)=REF(1)*T(1,2)+REF(2)*T(2,2)+REF(3)*T(3,2)
90D(3)=REF(1)*T(1,3)+REF(2)*T(2,3)+REF(3)*T(3,3)

C
END

SUBROUTINE SWITCH(S,L)

CONTROLS LIGHT PANEL FOR DAP BUTTONS

INTEGER S(24),L(24),FILLER

S - SWITCH POSITION (1=TRIPPED,0=UNTRIPPED)

L - LIGHT (1=ON,0=OFF)

FILLER - EMPTY VARIABLE TO BE USED AS FILLER WHEN CALLING MODE

CHECK SELECTION MODE

IF (S(1) .EQ. 1) L(1)=1

IF (S(1) .EQ. 1) L(2)=0

IF (S(2) .EQ. 1) L(1)=0

IF (S(2) .EQ. 1) L(2)=1

CHECK AUTOPILOT MODE

IF (S(3) .EQ. 1) L(3)=1

IF (S(3) .EQ. 1) L(4)=0

IF (S(4) .EQ. 1) L(3)=0

IF (S(4) .EQ. 1) L(4)=1

CHECK THRUSTER MODE

IF (S(5) .EQ. 1) L(5)=1

IF (S(5) .EQ. 1) L(6)=0

IF (S(6) .EQ. 1) L(5)=0

IF (S(6) .EQ. 1) L(6)=1

CHECK X,Y,Z TRANSLATION MODES

CHECK LOW Z MODE:

IF (S(17) .EQ. 1) THEN

L(17)=1

L(18)=0

L(21)=0

L(24)=0

ENDIF

CALL MODE(S(16),S(19),S(22),L(16),L(19),L(22))

CALL MODE(0,S(20),S(23),FILLER,L(20),L(23))

CALL MODE(S(18),S(21),S(24),L(18),L(21),L(24))

IF ((L(18)+L(21)+L(24)) .GT. 0) L(17)=0

CHANGE THRUSTER MODE IF HI/LO Z SELECTED

IF ((L(17)+L(18)) .GT. 0) THEN

L(5)=1

L(6)=0

ENDIF

CHECK ROTATION MODES

CALL MODE(S(7),S(10),S(13),L(7),L(10),L(13))

CALL MODE(S(8),S(11),S(14),L(8),L(11),L(14))

E-84

CALL MODE(S(9),S(12),S(15),L(9),L(12),L(15))

C

END

```

SUBROUTINE THRUST(JETSEL,A,ANG,GAS,C,PICK,TABLE)
C
C TAKES JETSEL MATRIX AND SUMS ROWS AS COMMANDED BY C.
C A - ACCEL X,Y,Z (FPS2,BODY FRAME)
C ANG - ANGULAR ACCEL X,Y,Z (RAD/S2,BODY FRAME)
C GAS - FUEL TAKEN FROM THREE TANKS
C      (1-FORWARD, 2-RIGHT AFT, 3-LEFT AFT)
C
REAL JETSEL(44,9),A(3),ANG(3),GAS(3)
INTEGER I,J,C(12),PICK,SELECT(44),ROW,TABLE(4,12,9)

C      DETERMINE WHAT THRUSTERS HAVE BEEN COMMANDED TO FIRE
C
DO 145 I=1,44
  SELECT(I)=0
145 CONTINUE
C
DO 175 I=1,12
  IF (C(I) .GT. 0) THEN
    DO 185 J=1,9
      ROW=TABLE(PICK,I,J)
      IF (ROW .GT. 0) SELECT(ROW)=1
185    CONTINUE
    ENDIF
175  CONTINUE
C
C      SUM COMMANDED ROWS OF JETSEL
C
DO 95 I=1,3
  A(I)=0.0
  ANG(I)=0.0
  GAS(I)=0.0
95 CONTINUE
C
DO 200 I=1,44
  IF (SELECT(I) .GT. 0) THEN
    A(1)=A(1)+JETSEL(I,1)
    A(2)=A(2)+JETSEL(I,2)
    A(3)=A(3)+JETSEL(I,3)
    ANG(1)=ANG(1)+JETSEL(I,4)
    ANG(2)=ANG(2)+JETSEL(I,5)
    ANG(3)=ANG(3)+JETSEL(I,6)
    GAS(1)=GAS(1)+JETSEL(I,7)
    GAS(2)=GAS(2)+JETSEL(I,8)
    GAS(3)=GAS(3)+JETSEL(I,9)
  ENDIF
200 CONTINUE
C
END

```

SUBROUTINE TRNSFM(T,Q)

C
C
C

COMPUTES TRANSFORMATION MATRIX (T) FROM QUATERNIONS (Q)

REAL T(3,3),Q(4)

C

T(1,1)=Q(1)*Q(1)-Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
T(2,1)=2.0*(Q(1)*Q(2)+Q(3)*Q(4))
T(3,1)=2.0*(Q(1)*Q(3)-Q(2)*Q(4))
T(1,2)=2.0*(Q(1)*Q(2)-Q(3)*Q(4))
T(2,2)=-Q(1)*Q(1)+Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
T(3,2)=2.0*(Q(2)*Q(3)+Q(1)*Q(4))
T(1,3)=2.0*(Q(1)*Q(3)+Q(2)*Q(4))
T(2,3)=2.0*(Q(2)*Q(3)-Q(1)*Q(4))
T(3,3)=-Q(1)*Q(1)-Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4)

C

END

C
C
C
C
SUBROUTINE UNITIZE(V)

UNITIZES V VECTOR

REAL V(3),MAGV

C
IF (ABS(V(1)) .LT. .1E-10) V(1)=.1E-10
IF (ABS(V(2)) .LT. .1E-10) V(2)=.1E-10
IF (ABS(V(3)) .LT. .1E-10) V(3)=.1E-10
MAGV=SQRT(V(1)*V(1)+V(2)*V(2)+V(3)*V(3))
V(1)=V(1)/MAGV
V(2)=V(2)/MAGV
V(3)=V(3)/MAGV
C

END

SUBROUTINE VECTOR (NAME,LENGTH)

```

C      THIS SUBROUTINE READS A VECTOR LIST FROM VAX
C      FILE AND PUTS IT IN USABLE FORM FOR THE PS300
C
      INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

      INTEGER*4 IPOS, LENGTH, CLASS
      REAL*4 POINTS (4,2000)
      LOGICAL*1 POSLIN (2000), PL, VL
      CHARACTER NAME*8, FILENAME*26

      FILENAME='[ALFANO.DATA]'/NAME(:LENGTH)/'.DAT'
      OPEN ( UNIT=1, NAME=FILENAME,TYPE='OLD',READONLY)
      READ ( 1, 910) VL,IPOS
      FORMAT ( A1,I10)
910      IF ((VL.EQ.'C') .OR. (VL.EQ.'C')) THEN
          CLASS=0
      ELSE IF ((VL.EQ.'D') .OR. (VL.EQ.'D')) THEN
          CLASS=1
      ELSE IF ((VL.EQ.'I') .OR. (VL.EQ.'I')) THEN
          CLASS=2
      ELSE IF ((VL.EQ.'S') .OR. (VL.EQ.'S')) THEN
          CLASS=3
      ENDIF
      DO 3410 I=1,IPOS
      READ ( 1, 911) PL,(POINTS(K,I),K=1,3)
911      FORMAT ( A1, 3F12.8)
      POINTS (4,I)=1
      POSLIN(I)=.FALSE.
      IF ((PL.EQ.'L') .OR. (PL.EQ.'L')) POSLIN(I)=.TRUE.
3410      CONTINUE
      CALL PVCBEG (NAME(:LENGTH),IPOS,.FALSE.,.FALSE.,3,CLASS,ERR)
      CALL PVCLIS (IPOS,POINTS,POSLIN,ERR)
      CALL PVCEND (ERR)
      CLOSE (UNIT=1)
      END

```

SUBROUTINE WINDOW(T,TW,X1,X2,X3,TARG)

THIS SUBROUTINE COMPUTES THE TRANSFORMATION MATRIX FROM REF TO WINDOW (TW) USING THE MATRIX FROM BOD TO REF (T). WINDOW IS ASSUMED TO HAVE -X,-Z BODY COMPONENTS. POSITION OF TARGET FROM WINDOW (TARG) IS COMPUTED KNOWING POSITION OF SHUTTLE FROM TARGET (X1,X2,X3).

WINDOW AXIS DEFINED AS FOLLOWS:

TARG(1) - INTO SCREEN

TARG(2) - LEFT

TARG(3) - UP

REAL T(3,3),TW(3,3),X1,X2,X3,TARG(3),C

C=SQRT(0.5)

COMPUTE TRANSFORMATION MATRIX

TW(1,1)=-C*(T(1,1)+T(1,3))

TW(1,2)=-C*(T(2,1)+T(2,3))

TW(1,3)=-C*(T(3,1)+T(3,3))

TW(2,1)=T(1,2)

TW(2,2)=T(2,2)

TW(2,3)=T(3,2)

TW(3,1)=C*(T(1,1)-T(1,3))

TW(3,2)=C*(T(2,1)-T(2,3))

TW(3,3)=C*(T(3,1)-T(3,3))

COMPUTE TARGET VECTOR

TARG(1)=-(X1*TW(1,1)+X2*TW(1,2)+X3*TW(1,3))

TARG(2)=-(X1*TW(2,1)+X2*TW(2,2)+X3*TW(2,3))

TARG(3)=-(X1*TW(3,1)+X2*TW(3,2)+X3*TW(3,3))

END

END

FILMED

2-86

DTIC