STS TASK SIMULATOR

SALVATORE ALFANO, CAPTAIN, USAF

AUGUST 1985
FINAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO SPRINGS, CO 80840
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 APH 190-1, APR 12-30, and APR 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
Lt Col, USAF
Director of Research and
Computer Based Education
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.
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.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. RCS ACCELERATION MODELING</td>
<td>2</td>
</tr>
<tr>
<td>3. RCS THRUSTER FIRING SELECTION</td>
<td>8</td>
</tr>
<tr>
<td>4. DETERMINING POSITION AND ATTITUDE</td>
<td>14</td>
</tr>
<tr>
<td>5. GENERATION OF VISUAL DISPLAY</td>
<td>15</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>18</td>
</tr>
<tr>
<td>APPENDIX A - DAP PANEL FUNCTIONS</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B - TERMINAL RENDEZVOUS/Docking</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C - ATTITUDE DETERMINATION USING QUATERNIONS</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D - THRUSTERS PROGRAM LISTING</td>
<td>D-1</td>
</tr>
<tr>
<td>APPENDIX E - MAIN PROGRAM LISTING</td>
<td>E-1</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>RCS Thruster Data</td>
</tr>
<tr>
<td>2</td>
<td>Typical Orbiter Mass Properties</td>
</tr>
<tr>
<td>3</td>
<td>Response Matrix for RCS Thrusters</td>
</tr>
<tr>
<td>4</td>
<td>Vernier Jet Select Table</td>
</tr>
<tr>
<td>5</td>
<td>Primary Jet Select Table</td>
</tr>
<tr>
<td>6</td>
<td>Primary with +Z Thrusters Inhibited Jet Select Table</td>
</tr>
<tr>
<td>7</td>
<td>Primary with Additional +Z Thrusters Jet Select Table</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Visual Display Layout</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>DAP Panel</td>
<td>A-1</td>
</tr>
<tr>
<td>3</td>
<td>Clohessy-Wiltshire Coordinate System</td>
<td>B-1</td>
</tr>
</tbody>
</table>
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 \((ST_{\text{cg}}, BL_{\text{cg}}, WL_{\text{cg}})\), the torque produced by a particular jet is computed from the equation

\[
L_j = \bar{R}_j \times \bar{F}_j + C_j F_j
\]

where

\[
\bar{F}_j = \begin{bmatrix} F_{x_j} \\ F_{y_j} \\ F_{z_j} \end{bmatrix}
\]

\[
\bar{R}_j = \begin{bmatrix} -(ST_{j} - ST_{\text{cg}})/12 \\ (BL_{j} - BL_{\text{cg}})/12 \\ -(WL_{j} - WL_{\text{cg}})/12 \end{bmatrix}
\]

and where the values of \(F_{x_j}, F_{y_j}, F_{z_j}, ST_{j}, BL_{j}, WL_{j}\), and \(C_j\) are obtained from columns 3-9 of Table 1.
Table 1. RCS Thruster Data (with plume impingement)

<table>
<thead>
<tr>
<th>THRUSTER NO.</th>
<th>Fx (LB)</th>
<th>Fy (LB)</th>
<th>Fz (LB)</th>
<th>STA (IN)</th>
<th>BL (IN)</th>
<th>WL (IN)</th>
<th>C (FT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2F</td>
<td>-879.4</td>
<td>-26.2</td>
<td>119.9</td>
<td>306.72</td>
<td>14.65</td>
<td>392.96</td>
<td>0.0000</td>
</tr>
<tr>
<td>F3F</td>
<td>-879.5</td>
<td>0.0</td>
<td>122.7</td>
<td>306.72</td>
<td>0.00</td>
<td>394.45</td>
<td>0.0000</td>
</tr>
<tr>
<td>F1F</td>
<td>-879.4</td>
<td>26.2</td>
<td>119.9</td>
<td>306.72</td>
<td>-14.65</td>
<td>392.96</td>
<td>0.0000</td>
</tr>
<tr>
<td>F1L</td>
<td>-26.3</td>
<td>873.6</td>
<td>18.2</td>
<td>362.67</td>
<td>-69.50</td>
<td>373.73</td>
<td>0.0000</td>
</tr>
<tr>
<td>F3L</td>
<td>-21.0</td>
<td>870.3</td>
<td>0.5</td>
<td>364.71</td>
<td>-71.65</td>
<td>359.25</td>
<td>0.0000</td>
</tr>
<tr>
<td>F2R</td>
<td>-26.3</td>
<td>-873.6</td>
<td>18.2</td>
<td>362.67</td>
<td>69.50</td>
<td>373.73</td>
<td>0.0000</td>
</tr>
<tr>
<td>F4R</td>
<td>-21.0</td>
<td>-870.3</td>
<td>0.5</td>
<td>364.71</td>
<td>71.65</td>
<td>359.25</td>
<td>0.0000</td>
</tr>
<tr>
<td>F2U</td>
<td>-32.3</td>
<td>-11.7</td>
<td>874.4</td>
<td>350.93</td>
<td>14.39</td>
<td>413.46</td>
<td>0.0000</td>
</tr>
<tr>
<td>F3U</td>
<td>-31.9</td>
<td>0.0</td>
<td>873.5</td>
<td>350.92</td>
<td>0.00</td>
<td>414.53</td>
<td>0.0000</td>
</tr>
<tr>
<td>F1U</td>
<td>-32.3</td>
<td>11.7</td>
<td>874.4</td>
<td>350.93</td>
<td>-14.39</td>
<td>413.46</td>
<td>0.0000</td>
</tr>
<tr>
<td>F2D</td>
<td>-28.0</td>
<td>-616.4</td>
<td>-639.5</td>
<td>333.84</td>
<td>61.42</td>
<td>356.95</td>
<td>0.0000</td>
</tr>
<tr>
<td>F1D</td>
<td>-28.0</td>
<td>616.4</td>
<td>-639.5</td>
<td>333.84</td>
<td>-61.42</td>
<td>356.95</td>
<td>0.0000</td>
</tr>
<tr>
<td>F4D</td>
<td>-24.8</td>
<td>-612.6</td>
<td>-639.4</td>
<td>348.44</td>
<td>66.23</td>
<td>358.44</td>
<td>0.0000</td>
</tr>
<tr>
<td>F3D</td>
<td>-24.8</td>
<td>612.6</td>
<td>-639.4</td>
<td>348.44</td>
<td>-66.23</td>
<td>358.44</td>
<td>0.0000</td>
</tr>
<tr>
<td>R3A</td>
<td>856.8</td>
<td>0.0</td>
<td>151.1</td>
<td>1555.29</td>
<td>137.00</td>
<td>473.06</td>
<td>0.000</td>
</tr>
<tr>
<td>R1A</td>
<td>856.8</td>
<td>0.0</td>
<td>151.1</td>
<td>1555.29</td>
<td>124.00</td>
<td>473.06</td>
<td>0.000</td>
</tr>
<tr>
<td>L3A</td>
<td>856.8</td>
<td>0.0</td>
<td>151.1</td>
<td>1555.29</td>
<td>-137.00</td>
<td>473.06</td>
<td>0.000</td>
</tr>
<tr>
<td>L1A</td>
<td>856.8</td>
<td>0.0</td>
<td>151.1</td>
<td>1555.29</td>
<td>-124.00</td>
<td>473.06</td>
<td>0.000</td>
</tr>
<tr>
<td>L4L</td>
<td>0.0</td>
<td>870.5</td>
<td>-8.4</td>
<td>1516.06</td>
<td>-149.83</td>
<td>455.21</td>
<td>-0.5987</td>
</tr>
<tr>
<td>L2L</td>
<td>0.0</td>
<td>870.5</td>
<td>-8.4</td>
<td>1529.07</td>
<td>-149.83</td>
<td>455.21</td>
<td>-0.6061</td>
</tr>
<tr>
<td>L3L</td>
<td>0.0</td>
<td>870.5</td>
<td>-8.4</td>
<td>1542.07</td>
<td>-149.83</td>
<td>455.21</td>
<td>-0.6235</td>
</tr>
<tr>
<td>L1L</td>
<td>0.0</td>
<td>870.5</td>
<td>-8.4</td>
<td>1555.07</td>
<td>-149.83</td>
<td>455.21</td>
<td>-0.6410</td>
</tr>
<tr>
<td>R4R</td>
<td>0.0</td>
<td>-870.5</td>
<td>-8.4</td>
<td>1516.06</td>
<td>149.83</td>
<td>455.21</td>
<td>0.5887</td>
</tr>
<tr>
<td>R2R</td>
<td>0.0</td>
<td>-870.5</td>
<td>-8.4</td>
<td>1529.07</td>
<td>149.83</td>
<td>455.21</td>
<td>0.6061</td>
</tr>
<tr>
<td>R3R</td>
<td>0.0</td>
<td>-870.5</td>
<td>-8.4</td>
<td>1542.07</td>
<td>149.83</td>
<td>455.21</td>
<td>0.6235</td>
</tr>
<tr>
<td>R1R</td>
<td>0.0</td>
<td>-870.5</td>
<td>-8.4</td>
<td>1555.07</td>
<td>149.83</td>
<td>455.21</td>
<td>0.6410</td>
</tr>
<tr>
<td>L4U</td>
<td>29.0</td>
<td>72.0</td>
<td>870.0</td>
<td>1520.04</td>
<td>-116.51</td>
<td>481.65</td>
<td>-0.4615</td>
</tr>
<tr>
<td>L2U</td>
<td>29.0</td>
<td>72.0</td>
<td>870.0</td>
<td>1532.96</td>
<td>-116.54</td>
<td>481.65</td>
<td>-0.3725</td>
</tr>
<tr>
<td>L1U</td>
<td>29.0</td>
<td>72.0</td>
<td>870.0</td>
<td>1545.87</td>
<td>-116.58</td>
<td>481.65</td>
<td>-0.2836</td>
</tr>
<tr>
<td>R4U</td>
<td>29.0</td>
<td>-72.0</td>
<td>870.0</td>
<td>1520.04</td>
<td>116.51</td>
<td>481.65</td>
<td>0.4615</td>
</tr>
<tr>
<td>R2U</td>
<td>29.0</td>
<td>-72.0</td>
<td>870.0</td>
<td>1532.96</td>
<td>116.54</td>
<td>481.65</td>
<td>0.3725</td>
</tr>
<tr>
<td>R1U</td>
<td>29.0</td>
<td>-72.0</td>
<td>870.0</td>
<td>1545.87</td>
<td>116.58</td>
<td>481.65</td>
<td>0.2936</td>
</tr>
<tr>
<td>L4D</td>
<td>312.4</td>
<td>346.8</td>
<td>-545.7</td>
<td>1498.11</td>
<td>-101.47</td>
<td>420.49</td>
<td>1.7413</td>
</tr>
<tr>
<td>L2D</td>
<td>312.4</td>
<td>346.8</td>
<td>-545.7</td>
<td>1513.68</td>
<td>-100.61</td>
<td>424.63</td>
<td>1.4807</td>
</tr>
<tr>
<td>L3D</td>
<td>312.4</td>
<td>346.8</td>
<td>-545.7</td>
<td>1529.23</td>
<td>-99.79</td>
<td>428.76</td>
<td>1.2208</td>
</tr>
<tr>
<td>R4D</td>
<td>312.4</td>
<td>-346.8</td>
<td>-545.7</td>
<td>1498.11</td>
<td>101.47</td>
<td>420.49</td>
<td>-1.7413</td>
</tr>
<tr>
<td>R2D</td>
<td>312.4</td>
<td>-346.8</td>
<td>-545.7</td>
<td>1513.68</td>
<td>100.61</td>
<td>424.63</td>
<td>-1.4807</td>
</tr>
<tr>
<td>R3D</td>
<td>312.4</td>
<td>-346.8</td>
<td>-545.7</td>
<td>1529.23</td>
<td>99.79</td>
<td>428.76</td>
<td>-1.2208</td>
</tr>
<tr>
<td>R5D</td>
<td>-0.8</td>
<td>-17.0</td>
<td>-17.6</td>
<td>324.35</td>
<td>59.70</td>
<td>350.12</td>
<td>0.0000</td>
</tr>
<tr>
<td>F5L</td>
<td>-0.8</td>
<td>17.0</td>
<td>-17.6</td>
<td>324.35</td>
<td>-59.70</td>
<td>350.12</td>
<td>0.0000</td>
</tr>
<tr>
<td>F5R</td>
<td>0.0</td>
<td>-24.0</td>
<td>-0.6</td>
<td>1565.00</td>
<td>149.87</td>
<td>459.00</td>
<td>0.0000</td>
</tr>
<tr>
<td>L5L</td>
<td>0.0</td>
<td>24.0</td>
<td>-0.6</td>
<td>1565.00</td>
<td>-149.87</td>
<td>459.00</td>
<td>0.0000</td>
</tr>
<tr>
<td>R5D</td>
<td>0.0</td>
<td>0.0</td>
<td>-24.0</td>
<td>1565.00</td>
<td>118.00</td>
<td>455.44</td>
<td>0.0000</td>
</tr>
<tr>
<td>L5D</td>
<td>0.0</td>
<td>0.0</td>
<td>-24.0</td>
<td>1565.00</td>
<td>-118.00</td>
<td>455.44</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

(Reproduced from Reference 1)
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEIGHT</td>
<td>200017.0 LB</td>
</tr>
<tr>
<td>Ixx</td>
<td>887302.0 SLUG-FT²</td>
</tr>
<tr>
<td>Iyy</td>
<td>6386877.0 SLUG-FT²</td>
</tr>
<tr>
<td>Izz</td>
<td>6694367.0 SLUG-FT²</td>
</tr>
<tr>
<td>Iyz</td>
<td>-971.0 SLUG-FT²</td>
</tr>
<tr>
<td>Izx</td>
<td>247376.0 SLUG-FT²</td>
</tr>
<tr>
<td>Ixy</td>
<td>5622.0 SLUG-FT²</td>
</tr>
<tr>
<td>CG STA</td>
<td>1095.3 IN</td>
</tr>
<tr>
<td>CG BL</td>
<td>0.3 IN</td>
</tr>
<tr>
<td>CG WL</td>
<td>377.4 IN</td>
</tr>
</tbody>
</table>
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

$$\ddot{a}_j = \frac{32.174}{W} \begin{bmatrix} F_{xj} \\ F_{yj} \\ F_{zj} \end{bmatrix}$$

(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

$$\ddot{a} = [I]^{-1} \ddot{\mathbf{L}}$$

(5)

where

$$\ddot{\mathbf{a}} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

(6)
\[
L = \begin{bmatrix}
L_x \\
L_y \\
L_z
\end{bmatrix}
\]  \hspace{1cm} (7)

\[
[I] = \begin{bmatrix}
I_{xx} & I_{xy} & I_{xz} \\
I_{yx} & I_{yy} & I_{yz} \\
I_{zx} & I_{zy} & I_{zz}
\end{bmatrix}
\]  \hspace{1cm} (8)

and where \( L_j \) represents the individual torque vectors produced by the designated thruster as calculated from Equations (1) through (3). The \([I]\) 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

<table>
<thead>
<tr>
<th>CMD</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>+X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ROL</td>
<td>L5D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-ROL</td>
<td>R5D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+PCH</td>
<td>F5R</td>
<td>F5L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-PCH</td>
<td>L5D</td>
<td>R5D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+YAW</td>
<td>R5R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-YAW</td>
<td>L5L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5. Primary (P) Jet Select Table

<table>
<thead>
<tr>
<th>CMD</th>
<th>THRUSTERS TO BE FIRED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>+X</td>
<td>R1A</td>
</tr>
<tr>
<td>-X</td>
<td>F2F</td>
</tr>
<tr>
<td>+Y</td>
<td>F1L</td>
</tr>
<tr>
<td>-Y</td>
<td>F2R</td>
</tr>
<tr>
<td>+Z</td>
<td>F3U</td>
</tr>
<tr>
<td>-Z</td>
<td>F1D</td>
</tr>
<tr>
<td>+ROL</td>
<td>L4D</td>
</tr>
<tr>
<td>-ROL</td>
<td>L4U</td>
</tr>
<tr>
<td>+PCH</td>
<td>F1D</td>
</tr>
<tr>
<td>-PCH</td>
<td>F3U</td>
</tr>
<tr>
<td>+YAW</td>
<td>F1L</td>
</tr>
<tr>
<td>-YAW</td>
<td>F2R</td>
</tr>
</tbody>
</table>
Table 6. Primary with +Z Thrusters Inhibited (PZI) Jet Select Table

<table>
<thead>
<tr>
<th>CMD</th>
<th>THRUSTERS TO BE FIRED</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>+X</td>
<td>R1A</td>
<td>L1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-X</td>
<td>F2F</td>
<td>F1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Y</td>
<td>F1L</td>
<td>L4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Y</td>
<td>F2R</td>
<td>R4R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Z</td>
<td>F2F</td>
<td>F1F</td>
<td>R1A</td>
<td>L1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Z</td>
<td>F1D</td>
<td>F2D</td>
<td>L4D</td>
<td>L2D</td>
<td>R4D</td>
<td>R2D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ROL</td>
<td>L4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-ROL</td>
<td>R4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+PCH</td>
<td>F1D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-PCH</td>
<td>L4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+YAW</td>
<td>F1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-YAW</td>
<td>F2R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12
Table 7. Primary with Additional +Z Thrusters
(HIGH Z) Jet Select Table

<table>
<thead>
<tr>
<th>CMD</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>+X</td>
<td>R1A</td>
<td>L1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-X</td>
<td>F2P</td>
<td>F1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Y</td>
<td>F1L</td>
<td>L4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-Y</td>
<td>F2R</td>
<td>R4R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Z</td>
<td>F2U</td>
<td>F3U</td>
<td>F1U</td>
<td>L4U</td>
<td>L2U</td>
<td>L1U</td>
<td>R4U</td>
<td>R2U</td>
<td>R1U</td>
</tr>
<tr>
<td>-Z</td>
<td>F1D</td>
<td>F2D</td>
<td>L4D</td>
<td>L2D</td>
<td>R4D</td>
<td>R2D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ROL</td>
<td>L4D</td>
<td>R4U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-ROL</td>
<td>L4U</td>
<td>R4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+PCH</td>
<td>F1D</td>
<td>F2D</td>
<td>L4U</td>
<td>R4U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-PCH</td>
<td>F3U</td>
<td>L4D</td>
<td>R4D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+YAW</td>
<td>F1L</td>
<td>R4R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-YAW</td>
<td>F2R</td>
<td>L4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

\[
\begin{align*}
    x &= f_x' - 2u y' \\
    y &= f_y' + 3u^2 y + 2u x' \\
    z &= f_z' - u^2 z
\end{align*}
\]

where \( f_x', f_y', \) and \( f_z' \) are the Orbiter translational acceleration components (due to thrust), and \( u \) 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


4. Astro 451 Course Handout, Astro 451, Course Readings #2, Fall 1984, Chapter F, USAF Academy, Colorado, 1984

APPENDIX A

(This appendix was taken largely from Reference 3)

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 0.000873 radians per second, DAP B yields 0.000175 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 0.3 radians per second squared, DAP B yields 0.06 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 \( R \))
   e. \( R \) -- Vector to target in fixed frame
   f. \( \bar{R} \) -- Vector to interceptor in fixed frame
   g. \( \bar{p} = \bar{r} - R \) -- Position of the interceptor relative to the target.

2. Apply \( \sum F_{ext} = \frac{d}{dt}(\bar{m}v) \) 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.)

\[
\frac{d^2}{dt^2} \bar{F} = \frac{d^2}{dt^2} R + \bar{w} \times (\bar{w} \times \bar{w} \times (R) + \bar{w} \times \bar{w} \times (R) + \bar{w} \times \bar{w} \times \frac{d}{dt} R)
\]
b. When \( \{ \) is the position of the interceptor vehicle, \( \{x\} \), then

\[
\frac{d^2}{dt^2} \mathbf{T} = \frac{\sum F_{\text{ext}}}{m} = \mathbf{T} = \mathbf{T} - \mathbf{\ddot{w}} \times \mathbf{T} + \mathbf{\ddot{w}} \times \mathbf{\ddot{w}} \times \mathbf{T} + 2 \mathbf{\dddot{w}} \times \mathbf{T} \tag{2.16}
\]

\[
c. \text{Noting the following relationships}
\]

\[
\begin{align*}
(1) \quad \mathbf{\ddot{w}} &= -\mathbf{w} \cdot \dot{x} \\
(2) \quad \mathbf{\dddot{w}} &= -\mathbf{w} \cdot \dot{x} \\
(3) \quad \mathbf{\dddot{R}} &= \mathbf{x} \cdot \dot{x} + \mathbf{y} \cdot \dot{y} + \mathbf{z} \cdot \dot{z} \\
(4) \quad \mathbf{\ddot{R}} &= \mathbf{x} \cdot \dot{x} + \mathbf{y} \cdot \dot{y} + \mathbf{z} \cdot \dot{z} \\
(5) \quad \mathbf{\dddot{R}} &= \mathbf{x} \cdot \dot{x} + \mathbf{y} \cdot \dot{y} + \mathbf{z} \cdot \dot{z} \\
(6) \quad \mathbf{\dddot{R}} &= \mathbf{R} \cdot \dddot{x}
\end{align*}
\]

\[
d. \text{Developing the required cross products}
\]

\[
\begin{align*}
\mathbf{\ddot{w}} \times \mathbf{T} &= \mathbf{w} (\mathbf{R} + \mathbf{y}) \times \dot{x} - \mathbf{w} \times \dot{y} \\
\mathbf{\dddot{w}} \times \mathbf{T} &= \mathbf{w} (\mathbf{R} + \mathbf{y}) \times \ddot{x} - \mathbf{w} \times \dddot{y} \\
\mathbf{\ddot{w}} \times \mathbf{\ddot{w}} \times \mathbf{T} &= -\mathbf{w}^2 \times \dot{x} - \mathbf{w}^2 (\mathbf{R} + \mathbf{y}) \cdot \dot{y}
\end{align*}
\]

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

\[
\begin{align*}
f_x &= \mathbf{x} + \mathbf{w} (\mathbf{R} + \mathbf{y}) - \mathbf{w}^2 \mathbf{x} + 2 \mathbf{w} \mathbf{(R} + \mathbf{y}) \\
f_y &= \mathbf{\ddot{R}} + \mathbf{\dddot{y}} - \mathbf{w} \mathbf{x} - \mathbf{w}^2 (\mathbf{R} + \mathbf{y}) - 2 \mathbf{w} \mathbf{x} \\
f_z &= \mathbf{\dddot{z}}
\end{align*}
\]
f. Now, let $f'$ be the specific force other than the gravitational attraction of the central body (i.e., drag, thrust, solar pressure, etc.)

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

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

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

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

these are very non-linear, but exact equations of motion.

3. Linearize the equations to find an analytic solution:

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

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

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

$$r^3 \approx R^3 (1 + 2\frac{y}{R})^{3/2}$$
b. Then the last terms in the equations of 2.g. take on the form

\[ -\frac{\mu}{r^3} \approx -\frac{\mu}{R^3(1 + \frac{2\chi}{R^4})} \]

\[ 3/2 \]

c. Since the term \((\chi)\) is small (less than 0.00125), use the binomial expansion to begin to simplify these equations

\[ (1 + \chi)^{-3/2} = 1 - \frac{3}{2} \chi + \frac{15}{R^2} \chi^2 - \ldots \]

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

\[ -\frac{\mu x}{R^3(1 + \frac{2\chi}{R^4})}^{-3/2} \approx -\frac{\mu x}{R^3} + \frac{3\mu xy}{R^4} \]

so that

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

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

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

e. The only nonlinear terms are the last ones in the equations. These terms are smaller than the proceeding terms by \((\chi)\). Neglecting these terms results in

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

\[ \ddot{y} = f_y' - \dot{w} + (u^2 - \frac{\mu}{R^3}) x + (u^2 + \frac{2\mu}{R^3}) y + 2u \dot{x} \]
\[ \ddot{z} = f_z' - \frac{\mu z}{R^3} \]

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

a. Then

\[ u = \frac{v_{CS}}{R} = \sqrt{\frac{\mu}{R^3}}, \quad u^2 = \frac{\mu}{R^3}, \quad \dot{u} = 0, \quad \dot{R} = 0 \]

b. The equations of motion become

\[ \dddot{x} = f_x'' - 2uy \]
\[ \dddot{y} = f_y'' + 3u^2y + 2ux \]
\[ \dddot{z} = f_z'' - u^2z \]

**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:

$$q = a \frac{w}{t}$$

(1)

where

$$q = 1q_1 + jq_2 + kq_3 + q_4$$

$$w = 1w_x + jw_y + kw_z$$

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

Expansion of (1) yields the following form:

$$
\frac{d}{dt} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -w_y & w_x \\ w_y & 0 & -w_z \\ -w_x & w_z & 0 \\ -w_y & -w_z & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}
$$

(2)
The quaternion components in the body (X, Y, Z) coordinate frame are

\[ q_1 = \frac{w_x}{w} \sin \frac{ut}{2} \]
\[ q_2 = \frac{w_y}{w} \sin \frac{ut}{2} \]
\[ q_3 = \frac{w_z}{w} \sin \frac{ut}{2} \]
\[ q_4 = \cos \frac{ut}{2} \]  

where

\[ w = \sqrt{w_x^2 + w_y^2 + w_z^2} \]

A vector is transformed from body to reference coordinates by

\[ x^R = q x^B q^* \]  

where

\[ q^* = -q_1 - q_2 - q_3 + q_4 \]
\[ q q^* = q^* q = 1 \]

The equivalent equation in vector-matrix notation is

\[ x^R = \begin{bmatrix} T^R_B \end{bmatrix} x^B \]  

\( G-2 \)
If (4) and (5) are expanded and compared, it is seen that

\[
\begin{bmatrix}
q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 - q_3 q_4) & 2(q_1 q_3 + q_2 q_4) \\
2(q_1 q_2 + q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 - q_1 q_4) \\
2(q_1 q_3 - q_2 q_4) & 2(q_2 q_3 + q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2
\end{bmatrix}
\]

(6)

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

\[
q_1 = \frac{1}{4q_4} (T_{32} - T_{23})
\]

\[
q_2 = \frac{1}{4q_4} (T_{13} - T_{31})
\]

\[
q_3 = \frac{1}{4q_4} (T_{21} - T_{12})
\]

\[
q_4 = \frac{1}{2} \sqrt{1 + T_{11} + T_{22} + T_{33}}
\]

(7)
APPENDIX D

THRUSTERS PROGRAM LISTING

D-1
This program generates rotational and translational accelerations and associated fuel costs for each thruster and stores them in array "THRUST". To do this, it applies each thruster's force and position on shuttle body (stored in array "JET") to shuttle mass, CG, and moments of inertia.

For a detailed explanation of equations/parameters refer to TRW paper "Engineering Description of the CMS/RCS/DAP Models Used in the HP-9825A High Fidelity Relative Motion Program (HFRMP)" 6 Oct 78.

Or see Capt. Alfanof, DFAS.

Real mass, STACG, BLCG, WLCG, RX, RY, RZ
Real MOIT(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

Selection tables are as follows:

| TABLE(1,1,I), I=1,9 | 9*0 |
| TABLE(1,2,I), I=1,9 | 9*0 |
| TABLE(1,3,I), I=1,9 | 9*0 |
| TABLE(1,4,I), I=1,9 | 9*0 |
| TABLE(1,5,I), I=1,9 | 9*0 |
| TABLE(1,6,I), I=1,9 | 9*0 |
| TABLE(1,7,I), I=1,9 | 44,8*0 |
| TABLE(1,8,I), I=1,9 | 45,8*0 |
| TABLE(1,9,I), I=1,9 | 39,40,7*0 |
| TABLE(1,10,I), I=1,9 | 44,43,7*0 |
| TABLE(1,11,I), I=1,9 | 41,8*0 |
| TABLE(1,12,I), I=1,9 | 42,8*0 |

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

| TABLE(3,1,I), I=1,9 | 16,18,7*0 |
| TABLE(3,2,I), I=1,9 | 1,3,7*0 |
| TABLE(3,3,I), I=1,9 | 4,19,7*0 |
| TABLE(3,4,I), I=1,9 | 6,23,7*0 |
| TABLE(3,5,I), I=1,9 | 13,16,18,5*0 |
| TABLE(3,6,I), I=1,9 | 12,11,33,34,36,37,3*0 |
| TABLE(3,7,I), I=1,9 | 33,30,7*0 |
| TABLE(3,8,I), I=1,9 | 36,8*0 |
| TABLE(3,9,I), I=1,9 | 12,11,7*0 |
| TABLE(3,10,I), I=1,9 | 33,36,7*0 |
| TABLE(3,11,I), I=1,9 | 4,23,7*0 |
| TABLE(3,12,I), I=1,9 | 6,19,7*0 |

| TABLE(4,1,I), I=1,9 | 16,18,7*0 |
| TABLE(4,2,I), I=1,9 | 1,3,7*0 |
| TABLE(4,3,I), I=1,9 | 4,19,7*0 |
DATA (TABLE(4,6,I),I=1,9)/6.23,7*0/
DATA (TABLE(4,5,I),I=1,9)/6.9,10.27,28.29,30,31,32/
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)/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/

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,
*0.3.1071.1/
DATA (JET(3,I),I=1,9)/-879.4,26.2,119.9,306.72,-14.65,392.96,
*0.3.1071.1/
DATA (JET(4,I),I=1,9)/-26.3,387.6,18.2,362.67,-69.5,373.73,
*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,
*0.3.1071.1/
DATA (JET(7,I),I=1,9)/-21.0,-870.3,5.364.71,71.65,359.25,0.0,
*0.3.1071.1/
DATA (JET(8,I),I=1,9)/-32.3,-1.7,874.4,350.93,14.39,413.46,
*0.3.1071.1/
DATA (JET(9,I),I=1,9)/-31.9,0.0,-873.5,350.92,0.0,414.53,
*0.3.1071.1/
DATA (JET(10,I),I=1,9)/-32.3,11.7,874.4,350.93,-14.39,413.46,
*0.3.1071.1/
DATA (JET(11,I),I=1,9)/-28.0,-616.4,-639.5,333.84,61.42,356.95,
*0.3.1071.1/
DATA (JET(12,I),I=1,9)/-28.0,-616.4,-639.5,333.84,-61.42,356.95,
*0.3.1071.1/
DATA (JET(13,I),I=1,9)/-24.8,-612.6,-639.4,348.44,66.23,358.44,
*0.3.1071.1/
DATA (JET(14,I),I=1,9)/-24.8,-612.6,-639.4,348.44,-66.23,358.44,
*0.3.1071.1/
DATA (JET(15,I),I=1,9)/-28.0,-616.4,-639.5,333.84,151.1,1555.29,
*0.3.1071.1/
DATA (JET(16,I),I=1,9)/-28.0,-616.4,-639.5,333.84,-151.1,1555.29,
*0.3.1071.1/
DATA (JET(17,I),I=1,9)/-28.0,-616.4,-639.5,333.84,151.1,1555.29,
*0.3.1071.1/
DATA (JET(18,I),I=1,9)/-28.0,-616.4,-639.5,333.84,-151.1,1555.29,
*0.3.1071.1/
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,-8.4,1516.07,-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,
*0.5061,3.1071.2/
DATA (JET(25,1),I=1,9) /0.0,-870.5,-8.4,1542.07,149.83,455.21,  
*0.6235,3.1071,2/  
DATA (JET(26,1),I=1,9) /0.0,-870.5,-8.4,1555.07,149.83,455.21,  
*0.641,3.1071,2/  
DATA (JET(27,1),I=1,9) /0.0,29.0,72.0,870.0,1520.04,-116.51,481.65,  
**0.4615,3.1071,3/  
DATA (JET(28,1),I=1,9) /0.0,29.0,72.0,870.0,1532.96,-116.54,481.65,  
**0.3725,3.1071,3/  
DATA (JET(29,1),I=1,9) /0.0,29.0,72.0,870.0,1545.87,-116.58,481.65,  
**0.2836,3.1071,3/  
DATA (JET(30,1),I=1,9) /29.0,-72.0,870.0,1520.04,-116.51,481.65,  
*0.4615,3.1071,2/  
DATA (JET(31,1),I=1,9) /29.0,-72.0,870.0,1532.96,-116.54,481.65,  
*0.3725,3.1071,2/  
DATA (JET(32,1),I=1,9) /29.0,-72.0,870.0,1545.87,-116.58,481.65,  
*0.2836,3.1071,2/  
DATA (JET(33,1),I=1,9) /312.4,346.8,-545.7,1498.11,-101.47,420.49,  
*1.7413,3.1071,3/  
DATA (JET(34,1),I=1,9) /312.4,346.8,-545.7,1513.68,-100.61,424.63,  
*1.4807,3.1071,3/  
DATA (JET(35,1),I=1,9) /312.4,346.8,-545.7,1529.23,-99.79,428.76,  
**1.2208,3.1071,2/  
DATA (JET(36,1),I=1,9) /312.4,-346.8,-545.7,1498.11,101.47,420.49,  
**1.7413,3.1071,3/  
DATA (JET(37,1),I=1,9) /312.4,-346.8,-545.7,1513.68,100.61,424.63,  
**1.4807,3.1071,2/  
DATA (JET(38,1),I=1,9) /312.4,-346.8,-545.7,1529.23,99.79,428.76,  
**1.2208,3.1071,2/  
DATA (JET(39,1),I=1,9) /-0.8,-17.0,-17.6,324.35,59.7,350.12,  
*0.0,-0.923,1/  
DATA (JET(40,1),I=1,9) /-0.8,-17.0,-17.6,324.35,-59.7,350.12,  
*0.0,-0.923,1/  
DATA (JET(41,1),I=1,9) /0.0,-24.0,-0.6,1565.0,149.87,459.0,  
*0.0,-0.923,2/  
DATA (JET(42,1),I=1,9) /0.0,-24.0,-0.6,1565.0,-149.87,459.0,  
*0.0,-0.923,3/  
DATA (JET(43,1),I=1,9) /0.0,0.0,24.0,1565.0,118.0,455.44,  
*0.0,-0.923,3/  
DATA (JET(44,1),I=1,9) /0.0,0.0,24.0,1565.0,-118.0,455.44,  
*0.0,-0.923,3/  
REASSIGN VALUES TO JET

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

MASS=200017.0/32.2
MOIC(1,1)=887302.00
MOIC(2,2)=6386877.0
MOIC(3,3)=6694367.0
MOIC(1,2)=-5622.0
MOIC(2,1)=-5622.0
MOIC(2,3)=971.0
MOIC(3,2)=971.0
MOIC(1,3)=-247376.0
MOIC(3,1)=-247376.0
STACG=1095.3
BLCG=0.3
DO 175 I=1,44
RX=(JET(I,4)-STACG)/12.0
RY=(JET(I,5)-BLCG)/12.0
RZ=(JET(I,6)-WLCG)/12.0
JET(I,8)=RY*JET(I,3)-RZ*JET(I,2)+JET(I,1)*JET(I,7)
JET(I,9)=RZ*JET(I,1)-RX*JET(I,3)+JET(I,2)*JET(I,7)
JET(I,10)=RX*JET(I,2)-RY*JET(I,1)+JET(I,3)*JET(I,7)
175 CONTINUE

ESTABLISH UNCOMPENSATED ACCELERATION PROFILES
MOI=MOMENT OF INERTIA MATRIX
IMO=INVERSE OF MOI
VUNC( )=VERNIER/UNCOMPENSATED RESPONSES
PUNC( )=PRIMARY/UNCOMPENSATED RESPONSES
PSIUN( )=PRIMARY WITH Z INHIBITED/UNCOMPENSATED RESPONSES
PSHI( )=PRIMAR, WITH HIGH Z/UNCOMPENSATED RESPONSES

CALL INVERT(MOI,IMO)
CALL JETSEL(IMOI,Jet,MASS,THRUST,VUN,PUN,PSIUN,PSHI,TABLE)

OPEN FILES HERE

OPEN(UNIT=2, NAME='[ALFANO . DATA]RESPONSES . DAT', TYPE='NEW',
   * FORM='UNFORMATTED', INITIALSIZE=60)
OPEN(UNIT=3, FILE='RSP . TBL', STATU S='NEW',
   * IOSTAT=ISTAT)

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),(THRUST(I,J),J=1,9)
825 CONTINUE

DO 850 I=1,12
   WRITE(2),(TABLE(1,1,J),J=1,9)
   WRITE(2),(TABLE(2,1,J),J=1,9)
   WRITE(2),(TABLE(3,1,J),J=1,9)
   WRITE(2),(TABLE(4,1,J),J=1,9)
   WRITE(2),(TABLE(I),J=1,9)
   WRITE(2),(PUN(I))
   WRITE(2),(PZIUN(I))
   WRITE(2),(PSHI(I))

850 CONTINUE

CLOSE FILES

CLOSE(UNIT=3)

END

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

   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 (ABSCMAT(J,RC) .LE. 0.0000000001) GOTO 200
   FACTOR=MAT(J,RC)/MAT(RC,RC)
   DO 400 I=1,6
      MAT(J,I)=MAT(J,I)+MAT(RC,I)*FACTOR
   CONTINUE
200 CONTINUE
END

SUBROUTINE INVERT(MOI,IMOI)
   'THIS ROUTINE SETS UP A 3 BY 6 MATRIX (DUMMY). LEFT 3 BY 3 MATRIX IS THE MOMENT OF INERTIA MATRIX (MOI). THE RIGHT 3 BY 3 IS THE IDENTITY MATRIX. CALLS TO SUBROUTINE ROWOP PERFORM ELEMENTARY ROW OPERATIONS MAKING THE LEFT 3 BY 3 OF DUMMY THE IDENTITY MATRIX AND THE RIGHT 3 BY 3 THE INVERSE MOMENT OF INERTIA MATRIX (IMOI).

REAL MOI(3,3),IMOI(3,3),DUMMY(3,6)
INTEGER K,L

DO 100 K=1,3
   DO 300 L=1,3
      DUMMY(K,L)=MOI(K,L)
      DUMMY(K,L+3)=0.0
   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
         IOMI(K,L)=DUMMY(K,L+3)
350 CONTINUE
150 CONTINUE
END

SUBROUTINE JETSEL(IMOI,JET,MASS,THRUST,VUN,PUN,PZIUN,
   * PZIHI,TABLE)
   'SUMS FORCES AND TORQUES OF THRUSTERS, CONVERTS TO ACCELERATIONS
   * ASSIGNS PROPELLANT LOSSES FOR THRUSTER.

INTEGER I,J,TABLE(4,12,9),ROW,COL
REAL M(12),ALPHA(3),IMOI(3,3),JET(44,12),MASS,THRUST(44,9)
REAL VUN(12),PUN(12),PZIUN(12),PZIHI(12)

DO 100 I=1,44
   M(I)=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)

ASSIGNING OF PROPELLANT

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

CONVERT FORCES AND TORQUES TO ACCELERATIONS

M(1) = M(1)/MASS
M(2) = M(2)/MASS
M(3) = M(3)/MASS
DO 400 J = 1,3
   ALPHAC(J) = IMO1(J,1)*M(4) + IMO1(J,2)*M(5) + IMO1(J,3)*M(6)
400 CONTINUE
M(4) = ALPHAC(1)
M(5) = ALPHAC(2)
M(6) = ALPHAC(3)

ASSIGN THRUST ARRAY AS FOLLOWS
COL 1-3 TRANSLATIONAL ACCELERATION (FPS2)
COL 4-6 ROTATIONAL ACCELERATION (RAD/S2)
COL 7-9 FUEL USED (LBS/S) (7-FWD 8-RT 9-LT)

DO 200 J = 1,9
   THRUST(I,J) = M(J)
200 CONTINUE
100 CONTINUE

ASSIGN RATES CORRESPONDING TO COMMANDS FOR GIVEN THRUST PROFILE

DO 500 I = 1,12
   VUN(I) = 0.0
   DO 600 J = 1,9
      IF (TABLE(I,J) .EQ. 0) GOTO 500
      ROW = TABLE(I,J)
      COL = INT((I+1)/2)
      VUN(I) = VUN(I) + THRUST(ROW,COL)
600 CONTINUE
500 CONTINUE

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
CONTINUE

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)
   CONTINUE
550 CONTINUE

CONTINUE

DO 575 I=1,12
   PZIHI(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)
      PZIHI(I)=PZIHI(I)+THRUST(ROW,COL)
   CONTINUE
575 CONTINUE

END
EXECUTES THRUST COMMANDS FOR PROXIMITY OPERATIONS  

DAP PANEL MODE DEFINITIONS CAN BE FOUND IN:  

FLIGHT PROCEDURES HANDBOOK/PROXIMITY OPERATIONS  

PRELIMINARY NOV 11, 1982 NASA  

PAGES 3-16 TO 3-20  

QUESTIONS ????? CONTACT CAPT ALFANO, DFAS  

INCLUDE [ALFANO]PROCONST.FOR/NOLIST"  

REAL X(6), Y(3), Z(4), JET(4, 12), JETSEL(44, 9), RT, RVX  
REAL WX(3), MAXWX, MAXWY, MAXWZ, ALT, AB(3), AR(3), ANG(3)  
REAL FUEL(3), DELTAT, DVX, DVY, DVZ, DWX, DWY, DWZ, GAS(3), FX(6)  
REAL DBX, DBY, DBZ, SNAP(3), REF(4), DELT1, DELT2, INCL, TIME  
REAL TW(3, 3), TARG(3), AP(4), RANGE, OMEGA, APSTOP, OLBX(3)  
REAL UP(3), HAFWAY, RR, RATE, TFUEL, FUDGE, STOPIT  

INTEGER*4 SYS$SETEF, SYS$WAITFR, CHAN, COUNT  
INTEGER*2 IBUF(337), IOSB(4)  
INTEGER S(24), C(12), P(12), I, J, PICK-L(24), CO(12), COMM, LITE  
INTEGER DC(12), NLOOPS, TABLE(4, 12, 9), PCOUNT, APSW, RESET, FLAG  
INTEGER PIC, DS(24), DL(24), OVD(12), IOSTAT, TRKCNT, RESTOP  
CHARACTER*16 TIMBUF  
INTEGER*4 TIMADR  

AB(3) - ACCELERATION (BODY FRAME)  
ALT - ALTITUDE OF TARGET (KM)  
ANG(3) - ANGULAR ACCELERATION (BODY FRAME)  
APSTOP - RANGE FROM TARGET YOU WISH AUTOPILOT TO STOP  
APSW - AUTOPILOT SWITCH (S(3)) FROM LAST ITERATION  
AP(4) - UNIT VECTOR FROM TARGET TO SHUTTLE FOR AUTOPILOT  
ARC(3) - ACCELERATION (REF FRAME)  
CHAN - CHANNEL FOR PHYSICAL I/O WITH PS300  
COMM - I=RHIC IN NEUTRAL, 0=RHIC NOT IN NEUTRAL  
COUNT - COUNTER FOR FAST AUTOPILOT GRAPHICS  
DBX - X DEADBAND FOR ATTITUDE CONTROL  
DBY - Y DEADBAND FOR ATTITUDE CONTROL  
DBZ - Z DEADBAND FOR ATTITUDE CONTROL  
DELT1,2 - TIME COUNTERS TO SEE IF LOOP EXCEEDS .04 SEC  
DELTAT - TIME STEP  
DVX - X VELOCITY PULSE  
DVY - Y VELOCITY PULSE  
DVZ - Z VELOCITY PULSE  
DWX - X ROTATIONAL PULSE  
DWY - Y ROTATIONAL PULSE  
DWZ - Z ROTATIONAL PULSE  
FLAG - A PARAMETER TO SLOW DOWN TRANSMISSION RATE TO PS300  
FUDGE - FUDGE FACTOR TO CHANGE STS RESPONSIVENESS  
FUEL(3) - TOTAL FUEL EXPENDED FROM EACH TANK  
FX(6) - PREDICTED FUTURE POS/VEL OF SHUTTLE  
GAS(3) - FUEL CONSUMED FOR PRESENT ITERATION (FROM EACH TANK)  
HAFWAY - HALF THE DISTANCE SINCE LAST AUTOPILOT UPDATE  
(USED FOR BRAKING PURPOSES)
E-3

INCLIN - INCLINATION OF TARGET ORBIT (DEG)
IBUF - BUFFER FOR ALL MOVEMENT (I/O WITH PS300)
IOSB - UNKNOWN PARAMETER (?) FOR I/O WITH PS300
IOSTAT - FLAG FOR CHECKING I/O ON UNIT 20
JET[4,12] - RESPONSE TO COMMANDS
JETSEL[44,9] - PRECOMPUTED ACCEL AND FUEL FOR EACH THRUSTER
LITE - 1-LVLH+3 DISC MODES SELECTED, 0=NOT SELECTED
MAXR - DISC RATE FOR ROLL
MAXDY - DISC RATE FOR PITCH
MAXDZ - DISC RATE FOR YAW
NLOOPS - LOOP COUNTER
OLDX(3) - ORIGINAL SHUTTLE POSITION FROM TARGET IN C-W FRAME
OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH (RAD/SEC)
P(12) - PULSE COMMAND COUNTER
PCOUNT - A COUNTER FOR PRINT STATEMENTS
PIC - FLAG FOR PS300 GRAPHICS USAGE
PICK - NUMBER OF SELECTED RESPONSE MATRIX (JET)
Q(4) - QUATERNION
RANGE - RANGE FROM TARGET
REF(4) - QUATERNION (LVLH TO C-W FRAME)
RESET - COMMAND FLAG TO START OVER FROM ORIGINAL POSITION
RESTOP - COUNTER TO ALLOW STOPPING OF PROGRAM W/ RESET BUTTON
RRATE - RANGE RATE FROM TARGET
RT - RENDEZVOUS TIME FOR BANANA AUTOPILOT
RVX - FINAL RENDEZVOUS OFFSET DISTANCE (35 FT)
SNAP(3) - SNAPSHOT ANGLE (NEEDED FOR DISC RATE MODE)
STOPIT - MAXIMUM NUMBER OF MINUTES YOU ANTICIPATE RUNNING
T(3,3) - TRANSITION MATRIX (BODY TO REFERENCE FRAME)
TABLE(4,12,9) - LIST OF THRUSTERS FIRED VS COMMAND
TARG(3) - TARGET POSITION wRT AFT BAY WINDOW
TMBUF,TIMADR - USED TO SET ITERATIVE LOOP TO DELTAT
TIME - TIME (SEC)
TFUEL - TOTAL FUEL USED
TWC(3,3) - TRANSITION MATRIX (REF TO WINDOW FRAME)
TRKCNT - COUNTER FOR WRITING TO UNIT 20
UP(3) - UP VECTOR FOR PS300 AND ALIGNMENT SUBROUTINE
WBC(3) - ROTATION RATE (BODY FRAME)
X(6) - SHUTTLE POSITION/VELOCITY FROM TARGET IN C-W FRAME

ASSIGNMENT OF DAP PANEL SWITCHES, LIGHTS, AND THRUST COMMANDS

S( ) - SWITCH ARRAY (READ FROM DAP PANEL)
L( ) - LIGHT ARRAY (WRITTEN TO DAP PANEL)
C( ) - THRUST COMMAND ARRAY
OVHD( ) - COMMAND ARRAY FOR OVERHEAD LOS (READ FROM THC/RHC)
CO( ) - COMMAND ARRAY FROM LAST ITERATION
DC( ) - DUMMY COMMAND ARRAY

S(1) - SELECT A / PARAMETERS FOR FAR AWAY MANEUVERING (MAN MODE)
S(2) - SELECT B / PARAMETERS FOR CLOSE IN MANEUVERING (MAN MODE)
S(3) - AUTOPILOT
S(4) - MAN / MANUAL
S(5) - NORM RCS JETS
S(6) - VERNIER RCS JETS
S(7) - DISCRETE RATE ROLL
S(8) - DISCRETE RATE PITCH
S(9) - DISCRETE RATE YAW
E-4

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

L( ) - SAME AS S ARRAY
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

CO( ) - SAME AS C ARRAY
DC( ) - SAME AS C ARRAY

1=ON  0=OFF

INITIALIZATIONS

DATA TIMBUF/"0000 00:00:00.12"/
DATA (S(I),I=1,24)/24*0/
DATA (L(I),I=1,12)/1,0,0,1,1,0,3*0,3*1/
DATA (L(I),I=13,24)/6*0,3*1,3*0/
DATA (C(I),I=1,12)/12*0/
DATA (CO(I),I=1,12)/12*0/
DATA (P(I),I=1,12)/12*0/
DATA (X(I),I=1,6)/6*0/
DELTAT=.04
TIME=0.
FLAG=0

SET TIMER

CALL SYSSBINTIM(TIMBUF,TIMADR)

READ IN JETSEL, TABLE AND JET MATRICES

OPEN(UNIT=2,NAME="[ALFANO.DAT]RESPONSES.DAT", +
    TYPE="OLD",FORM=UNFORMATTED,READONLY)

DO 115 I=1,44
READ(2)(JETSEL(I,J),J=1,9)
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

CLOSE(2)

ECOCHECK DATA THAT WAS READ IN
CALL ECOCHECK(TABLE,JET)

DETERMINE USER REQUIREMENTS

PRINT */! DO YOU WISH :
PRINT */1 REAL TIME GRAPHIC SIMULATION
PRINT */2 FAST GRAPHIC AUTOPILOT DEMO
PRINT */3 FAST AUTOPILOT W/O GRAPHICS
READ */PIC

SET UP AUTOPILOT IF CHOSEN

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

READ IN FUDGE FACTOR

PRINT */! BY WHAT FACTOR DO YOU WISH TO MULTIPLY
PRINT */!STS RESPONSIVENESS
READ */FUDGE

APPLY FUDGE FACTOR TO THRUSTER DATA

DO 37 I=1,44
   DO 38 J=1,9
      JETSEL(I,J)=JETSEL(I,J)*FUDGE
38 CONTINUE
37 CONTINUE

READ IN MAX RUN TIME

STOPIT=1000.
IF (PIC .EQ. 1) THEN
   PRINT */! HOW LONG DO YOU ANTICIPATE RUNNING
   PRINT */! (IF OVER 10 MINUTES PLEASE COORDINATE WITH SEILER)
READ *,STOPIT
ENDIF

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

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

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

READ IN INITIAL SHUTTLE POSITION
PRINT *,"ENTER COORDINATES OF SHUTTLE FROM TARGET"
PRINT *,X,Y,Z CLOHESSEY-WILTSHEW FRAME (FEET)
READ *,OLDX(1),CLDX(2),OLDX(3)

ZERO VELOCITY AND ANGULAR MOTION ARE ASSUMED FOR SHUTTLE.
SHUTTLE ALIGNMENT IS INITIALIZED SUCH THAT TARGET STARTS IN CENTER OF SCREEN (-Z IN BODY FRAME)

INITIALIZE PS300 DISPLAY
INITIALIZE BUFFER FOR PS300 MEMORY ADDRESS

IF (PIC.EQ.1).OR.(PIC.EQ.2) THEN
CALL PS300(ALT,INCLIN)
CALL INITBUF(IBUF_CHAN,IOSB)
ENDIF

READ AND WRITE TO SHUTTLE MOCKUP

IF (PIC.EQ.1) CALL IOBUFF(1,S,L,OVHD,RESET)

OPEN UNIT 20 FOR WRITING AND SET FORMAT.
IF UNABLE, PRINT MESSAGE AND END PROGRAM

OPEN (UNIT=20, FILE=*STS_TRK*, STATUS=*NEW*, IOSTAT=ISTAT)
IF (IOSTAT.NE.3) THEN
PRINT *,'CAN NOT OPEN OUTPUT FILE'
PRINT *,'FILE = ', STS_TRK'
PRINT *,'ISTAT = ', IOSTAT
GOTO 999
ENDIF

444 FORMAT(9P',4(3X,F8.1),4(3X,F8.6))
445 FORMAT('L',4(3X,F8.1),4(3X,F8.6))

SET EVENT FLAG FOR WAIT COMMAND

CALL SYS$SETEF(7)

RESET BEGINS HERE

CONTINUE
INITIALIZE PARAMETERS

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

UP(1)=0.0001
UP(2)=1.0
UP(3)=0.0001

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

DO 175 I=1,12
C(I)=0
DC(I)=0

175 CONTINUE

MFAILWAY=10000000.

WRITE INITIAL POSITION, TIME, AND QUATERNION TO UNIT 20
WRITE(20,444)X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)

ALIIN SHUTTLE SO TARGET IS CENTERED IN WINDOW

CALL ALIGN(X(1),X(2),X(3),T,Q,UP)

INITIALIZE LVLM REF TO ALIGNMENT QUATERNION

REF(1)=Q(1)
REF(2)=Q(2)
REF(3)=Q(3)
REF(4)=Q(4)

INITIALIZE COUNTERS

NLOOPS=0
PCOUNT=999
DELT1=SECNDS(0.)
TRKCNT=C
COUNT=0

ITERATIVE LOOP BEGINS HERE

450 CONTINUE
UPDATE COUNTERS

NLOOPS = NLOOPS + 1
TIME = NLOOPS * DELTAT
PCOUNT = PCOUNT + 1
TRKCNT = TRKCNT + 1
COUNT = COUNT + 1
FLAG = FLAG + 1

PICK APPROPRIATE THRUST RESPONSES.
VALUES OF PICK ARE AS FOLLOWS:
1 - VERNIER
2 - PRIMARY
3 - PRIMARY WITH +Z INHIBITED
4 - PRIMARY WITH HIGH Z

CHOOSE THRUST RESPONSES FROM DAP PANEL LIGHTS

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

SELECT RENDEZVOUS CONSTRAINTS (FROM DAP PANEL LIGHTS)

IF ((L(2)+L(3)) .GT. 0) THEN
DVX = 0.01
DVY = 0.01
DVZ = 0.01
DWX = 0.000175
D Wy = 0.000175
DWZ = 0.000175
DBX = 0.02
DBY = 0.02
DBZ = 0.02
MAXWX = 0.06
MAXWY = 0.06
MAXWZ = 0.06
ELSE
DVX = 0.05
DVY = 0.05
DVZ = 0.05
DWX = 0.000873
D Wy = 0.000873
DWZ = 0.000873
DBX = 1
DBY = 1
DBZ = 1
MAXWX = 0.3
MAXWY = 0.3
MAXWZ = 0.3
ENDIF

CHECK PULSE MODE LIGHTS

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),DVY,
* 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))
* DELTAT*JET(PICK,7), DELTAT*JET(PICK,8))
* DELTAT*JET(PICK,9), DELTAT*JET(PICK,10))
* DELTAT*JET(PICK,11), DELTAT*JET(PICK,12))

CHECK LVLM MODE USING COMM AND LITE
COMM=0
LITE=0
IF (((C(7)+C(8)+C(9)+C(10)+C(11)+C(12)).EQ.0)) .EQ. 0) COMM=1
IF (((L(7)+L(8)+L(9)).EQ.4)) LITE=1
IF (((LITE+COMM).EQ.2)) THEN
   CALL LVLM(C,CO,REF,DBX,DBG,DBZ,
* WB*JET,DELTAT,MAXWX,MAXWY,MAXWZ,PICK,T)
ENDIF

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,X(7),Z(7),C(7),CO(7),DBX,Snap(1))
IF (L(8).EQ.1) CALL DISC(JET(PICK,9),JET(PICK,10),WB(2),
* DELTAT,MAXWY,X(8),Z(8),C(8),CO(8),DBY,Snap(2))
IF (L(9).EQ.1) CALL DISC(JET(PICK,11),JET(PICK,12),WB(3),
* DELTAT,MAXWZ,X(9),Z(9),C(9),CO(9),DBZ,Snap(3))
ENDIF

CHECK IF THE AUTOPILOT (A OR B) IS SELECTED
A = DIRECT APPROACH
B = CLOSED LOOP CLOMESSY-WILSHIRE

IF (L(3).EQ.1) THEN
ZERO OUT THC COMMANDS
C(1)=0
C(2)=0
C(3)=0
C(4)=0
C(5)=0
C(6)=0

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)

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

CHOOSE PROPER AUTOPILOT
IF (L(1).EQ.1) THEN
   CALL APLLOT(L/C/REF/X/FX/T/APS/W/S(3)/AP/DEL/T/A/1/7/STOP/JET/PICK)
ELSE
   CALL BANANA(FX/OMEGA/C/T/REF/L/S(3)/APS/W/Q/HAFWAY/RT/DEL/T/U/JET/PICK/RVX)
ENDIF
ENDIF

IF NO OTHER MODE IS CHOSEN, NORM/ACCEL MODE IS ASSUMED. COMMANDS REMAIN THE SAME, SO NO SUBROUTINE IS NEEDED.
APPLY COMMANDS (C) TO RESPONSE MATRIX (JETSEL)
CALL THRUST(JETSEL,AB,ANG,GAS,C,PICK/TABLE)
TRANSLATE BODY ACCELERATIONS (AB) TO REF FRAME (AR)
CALL BTOR(T,AB,AR)
APPLY ACCELERATIONS TO DETERMINE POSITION (X(1-3)) AND VELOCITY (X(4-6)) OF SHUTTLE RELATIVE TO TARGET USING CLOHESSY-WILTSHEIRE EQUATIONS.
CALL LINTEG(X/OMEGA/AR/DEL/TAT)
APPLY ANGULAR ACCELERATIONS TO FIGURE QUATERNIONS AND NEW T MAT.
CALL ROTATE(ANG/DL/TAT/WB/QT)
UPDATE FUEL USED
DO 900 I=1,3
   FUEL(I)=FUEL(I)+GAS(I)*DEL/T
900 CONTINUE
TFUEL=FUEL(1)+FUEL(2)*FUEL(3)

DETERMINE TARGET POSITION AND ATTITUDE AS SEEN THRU AFT BAY WINDOW. FOR REAL SIMULATION, DISPLAY ON THE PS300.
IF (PIC.EQ.1)
   CALL LOOK(T/X/OMEGA/TIME/TFUEL/IBUF/CHAN/IOSB)
IF ((PIC.EQ.2).AND. (COUNT.EQ.13)) THEN
   CALL LOOK(T/X/OMEGA/TIME/TFUEL/IBUF/CHAN/IOSB)
   COUNT=0
ENDIF
WRITE POSITION, TIME, AND ATTITUDE DATA TO UNIT 20 EVERY 10 SECONDS.
IF (TRKCNT.GE.250) THEN
   WRITE(20,445),X(1),X(2),X(3),TIME,Q(1),Q(2),Q(3),Q(4)
   TRKCNT=0
ENDIF
UPDATE OLD COMMAND ARRAY (CO)
DO 950 I=1,12
CO(I)=DC(I)
CONTINUE

UPDATE AUTOPILOT SWITCH

APSW=S(3)

PRINT TO SCREEN EVERY 1000 ITERATIONS IF PIC=3

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

ZERO OUT SWITCHES AND RESTORE THRUST COMMANDS,
SET MANUAL LIGHT FLASHING IF AUTOPILOT ON

DO 645 I=1,12
  S(I)=0
  S(I+12)=0
  C(I)=DC(I)
CONTINUE

IF ((L(3).EQ.1) .AND. ((TIME-INT(TIME)).GT..5)) L(4)=1

READ SWITCHES AND THRUST COMMANDS FROM SHUTTLE MOCKUP.
WRITE TO LIGHTS ON DAP PANEL OF SHUTTLE MOCKUP.
IF RESET BUTTON HELD FOR ONE SECOND, STOP PROGRAM
IF RESET COMMANDED, GO TO 135

IF ((PIC .EQ. 1) .AND. (FLAG .GE. 3)) THEN
  CALL IOBUFF(2,5,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
ENDIF

CHANGE INPUT TO REFLECT OVERHEAD LINE OF SIGHT

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)=OVHD(8)

C ASSIGN REAL COMMAND ARRAY (C) WITH DUMMY COMMANDED VALUES
THIS IS TO PRESERVE DATA READ FROM THC/RHC WHILE PROGRAM
CHANGES C ARRAY DUE TO VARIOUS SELECTED MODES

DO 475 I=1,12
C(I)=DC(I)
475 CONTINUE

ENDIF

IF AUTOPILOT ON, SET L(4) (MANUAL LIGHT) TO ZERO

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

APPLY SWITCH COMMANDS TO DAP PANEL LIGHTS

CALL SWITCH(S,L)

RETURN TO 450 FOR NEXT ITERATION

IF (NLOOPS .LT. (1500*STOPIT)) GOTO 450

COMPUTE AVERAGE LOOP TIME

DELT2=SECNTS(DELT1)
PRINT *' AVERAGE LOOP TIME = 'DELT2/NLOOPS,' SECONDS'

CLOSE UNIT 20

CLOSE(20)

999 CONTINUE

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

COMPUTES BODY AXIS COMPONENTS IN REFERENCE FRAME.
XBOD 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)=SQR(D(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=SQR(D(Q(1)+Q(2)+Q(3)+Q(4))
Q(1)=Q(1)/CONST
Q(2)=Q(2)/CONST
Q(3)=Q(3)/CONST
Q(4)=Q(4)/CONST
Q(3) = Q(3) / CONST
Q(4) = Q(4) / CONST

C

END
SUBROUTINE APILOT(L, REF, FX, T, OLD_SW, NEW_SW,
* AP, DELTAT, Q, APSTOP, JET, PICK)

C

C THIS AUTOPilot FLIES THE SHUTTLE DIRECTLY TO
C TARGET (DIRECT APPROACH) USING LVLH FOR ATTITUDE
C CONTROL. IT "THinks" ONE ITERATION IN ADVANCE
C USING PREDICTED POS/VEL DUE TO RHC THRUSTING
C AND ORBITAL DRIFT.
C
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), OLDSW, NEWSW, PICK

C

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

C

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

C

C INITIALIZE LIGHTS AND COMMANDS
C
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

C

C COMPUTE MAX DISTANCE CHANGE
C
MAXD=.005*(RANGE)*DELTAT
IF (RANGE .LT. 1.5*APSTOP) THEN
  MAXD=MAXD*(RANGE-APSTOP)/(.5*APSTOP)
ENDIF

C

C SWITCH TO LOW Z INSIDE 200 FEET
E-16

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

CHECK TO SEE IF AUTOPILOT JUST SELECTED

C IF (OLDSW .NE. NEWSW) THEN
C
C SET REF TO PRESENT QUATERNION (TO BE USED BY LVLH)
C
REF(1)=Q(1)
REF(2)=Q(2)
REF(3)=Q(3)
REF(4)=Q(4)

C SET REFERENCE VECTOR
C
AP(1)=X(1)/RANGE
AP(2)=X(2)/RANGE
AP(3)=X(3)/RANGE
AP(4)=RANGE*1.0

ENDIF

FIGURE DESIRED FUTURE POSITION (DP) (+2 ITERATIONS)

IF (ABS(RANGE-APSTOP) .LT. ABS(AP(4)-APSTOP)) THEN
  AP(4)=RANGE
ELSE
  MAXD=0.0
ENDIF

DP(1)=(AP(4)-2.0*MAXD)*AP(1)
DP(2)=(AP(4)-2.0*MAXD)*AP(2)
DP(3)=(AP(4)-2.0*MAXD)*AP(3)

COMPUTE REQUIRED VELOCITY (NEXT ITERATION)

REQV(1)=(DP(1)-FX(1))/DELTAT
REQV(2)=(DP(2)-FX(2))/DELTAT
REQV(3)=(DP(3)-FX(3))/DELTAT

FIGURE VELOCITY TO BE GAINED (DELTAV, TARGET FRAME)

DELTAV(1)=REQV(1)-FX(4)
DELTAV(2)=REQV(2)-FX(5)
DELTAV(3)=REQV(3)-FX(6)

TRANSFORM TO BODY FRAME

CALL RTOB(T,DELTAV,DELVB)

DETERMINE IF THRUST IS REQUIRED (VDB IS VELOCITY DEADBAND)

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

ENDD
SUBROUTINE ARM

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

INCLUDE 'PROCONST.FOR/NLIST'

REAL*4 V(3)

COMMANDS FOR WRIST ROLL

SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION

CALL PFN('WRROLL','Y ROTATE',ERR)
CALL PSNST('WR ROLL',1,'LABEL1',ERR)
CALL PCONN('DIALS',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','Y ROTATE',ERR)
CALL PSNST('WRYAW',1,'LABEL2',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
SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
CALL PFN('WRPITCH*/XROTATE*'/ERR)
CALL PSNST('WR PITCH*/1*/DLABEL3*/ERR)
CALL PCONN('DIALS*/3*/WRPITCH*/ERR)

CONNECT INPUTS TO ACCUMULATOR
CALL PFN('ACC3*/ACCUMULATE*/ERR)
CALL PCONN('WRPITCH*/1*/ACC3*/ERR)
CALL PSNEA(0.0,2*/ACC3*/ERR)
CALL PSNEA(0.1,3*/ACC3*/ERR)
CALL PSNEA(1.0,4*/ACC3*/ERR)
CALL PSNEA(120.0,5*/ACC3*/ERR)
CALL PSNEA(-120.0,6*/ACC3*/ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM
CALL PCONN('ACC3*/1*/,WRIST.PITCH*/ERR)

COMMANDS FOR ELBOW PITCH
SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
CALL PFN('ELPITCH*/XROTATE*'/ERR)
CALL PSNST('EL PITCH*/1*/DLABEL4*/ERR)
CALL PCONN('DIALS*/4*/ELPITCH*/ERR)

CONNECT INPUTS TO ACCUMULATOR
CALL PFN('ACC4*/ACCUMULATE*/ERR)
CALL PCONN('ELPITCH*/1*/ACC4*/ERR)
CALL PSNEA(0.0,2*/ACC4*/ERR)
CALL PSNEA(0.1,3*/ACC4*/ERR)
CALL PSNEA(1.0,4*/ACC4*/ERR)
CALL PSNEA(160.0,5*/ACC4*/ERR)
CALL PSNEA(-2.0,6*/ACC4*/ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM
CALL PCONN('ACC4*/1*/,ELBOW.PITCH*/ERR)

COMMANDS FOR SHOULDER YAW
SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION
CALL PFN('SHYAW*/ZROTATE*/ERR)
CALL PSNST('SH YAW*/1*/DLABEL5*/ERR)
CALL PCONN('DIALS*/5*/SHYAW*/ERR)

CONNECT INPUTS TO ACCUMULATOR
CALL PFN('ACC5*/ACCUMULATE*/ERR)
CALL PCONN('SHYAW', -1, 1, 'ACC5', 'ERR)
CALL PSNREA(0, 0, 2, 'ACC5', 'ERR)
CALL PSNREA(0, 1, 3, 'ACC5', 'ERR)
CALL PSNREA(1, 0, 4, 'ACC5', 'ERR)
CALL PSNREA(180, 0, 5, 'ACC5', 'ERR)
CALL PSNREA(-180, 0, 6, 'ACC5', 'ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM

CALL PCONN('ACC5', 1, 1, 'SHOULDER.YAW', 'ERR)

COMMANDS FOR SHOULDER PITCH

SEND LABEL TO DIAL AND CONNECT TO ROTATE FUNCTION

CALL PFN('SHPITCH', 'XROTATE', 'ERR)

CALL PSNST('SH PITCH', 1, 'DIALLABEL6', 'ERR)
CALL PCONN('DIALS', 6, 1, 'SHPITCH', 'ERR)

CONNECT INPUTS TO ACCUMULATOR

CALL PFN('ACC6', 'ACCUMULATE', 'ERR)

CALL PCONN('SHPITCH', 1, 1, 'ACC6', 'ERR)
CALL PSNREA(0, 0, 2, 'ACC6', 'ERR)
CALL PSNREA(0, 1, 3, 'ACC6', 'ERR)
CALL PSNREA(1, 0, 4, 'ACC6', 'ERR)
CALL PSNREA(2, 0, 5, 'ACC6', 'ERR)
CALL PSNREA(-145, 0, 6, 'ACC6', 'ERR)

CONNECT ACCUMULATOR OUTPUT TO PROGRAM

CALL PCONN('ACC6', 1, 1, 'SHOULDER.PITCH', 'ERR)

CREATE A 3D VECTOR (IN FEET) TO TRANSLATE ARM

CALL PFN('XYVEC', 'VEC', 'ERR)
CALL PFN('XYZVEC', 'VEC', 'ERR)
CALL PCONN('XYVEC', 1, 1, 'SHPITCH', 'ERR)
CALL PCONN('XYZVEC', 1, 1, 'SHPITCH', 'ERR)

BUILD A WRIST

CALL PBEGS('WRIST', 'ERR)
CALL PROTX('PITCH', 0, 0, 0, 'ERR)
V(1) = .5415
V(2) = 1.23
V(3) = .5415
CALL PSSCALE('CYLINDER', 'ERR)
V(1) = 0.0
V(2) = 1.23
V(3) = 0.0
CALL PTRANS('CYLINDER', 'ERR)
CALL PROT('YAW', 0, 0, 'ERR)
CALL PROT('RCLL', 0, 0, 'ERR)
V(1) = .5415
V(2) = 4.93
V(3) = .5415
CALL PSACLE("**", V, "CYLINDER", ERR)
CALL PENDS(ERR)

C BUILD A FOREARM (W/ ELBOW)

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 PSACLE("**", V, "CYLINDER", ERR)
V(1) = 0.0
V(2) = 23.1625
V(3) = 0.0
CALL PTRANS("**", V, "WRIST", ERR)
CALL PENDS(ERR)

C BUILD THE SHOULDER

CALL PBEGS("SHOULDER", ERR)
V(1) = 0.0
V(2) = 0.0
V(3) = 0.0
CALL PTRANS("TRAN", V, "**", ERR)
CALL PROTX("PITCH", 0.0, "CYL", ERR)
V(1) = .58
V(2) = 1.083
V(3) = .58
CALL PSACLE("CYL", V, "CYLINDER", ERR)
CALL PROTX("PITCH", 0.0, "**", ERR)
CALL PROTX("YAW", 0.0, "**", ERR)
V(1) = .5415
V(2) = 20.921
V(3) = .5415
CALL PSACLE("**", V, "CYLINDER", ERR)
V(1) = 0.0
V(2) = 20.921
V(3) = 0.0
CALL PTRANS("**", V, "ELBOW", ERR)
CALL PENDS(ERR)

C RESCALE SHOULDER FROM FEET TO KM AND NAME IT "ARM"

V(1) = .0003048
V(2) = .0003048
V(3) = .0003048
CALL PSACLE("ARM", V, "SHOULDER", ERR)

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

C THIS IS AN EXPLICIT GUIDANCE SCHEME.
C IT SOLVES THE CLOHESSY-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 (DELVB). 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
REAL T(3,3),DELTAV(3),DELVB(3),A,D,E,F
REAL RANGE,B-VZ, Q(4),HAFWAY,RT,DELTAT,UP(3)
REAL POS(3),BPOS(3),JET(4,12),RVX

INTEGER C(12),IL(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 DELVB - 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

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

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

C SET DAP PANEL LIGHTS

L(2)=1
L(3)=1
L(5)=1
L(7)=1
L(9)=1
L(16)=1
L(20)=1

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

UPDATE TIME TO RENDEZVOUS

RT=RT-DELTAT

CHECK TO SEE IF AUTOPILOT JUST SELECTED

IF (OLDSW .NE. NEWSW) THEN
  HAFWAY=1000000.
  RVX=0.
ENDIF

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

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

COMPUTE NEW REFERENCE QUATERNION BASED ON POSITION

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

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

X(1)=X(1)+RVX

COMPUTE DELTAV (TARGET FRAME)

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

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

\[ \begin{align*}
VY &= \frac{-E - F \cdot B/A}{B \cdot B/A + D} \\
VX &= \frac{-B \cdot VY - F}{A} \\
VZ &= -X(3) \cdot \Omega \cdot \varepsilon / \sqrt{w}
\end{align*} \]

C

DELTAV(1) = VX - X(4) \\
DELTAV(2) = VY - X(5) \\
DELTAV(3) = VZ - X(6)

C

TRANSFORM TO BODY FRAME

CALL RTOB(T, DELTAV, DELVB)

C

DETERMINE IF THRUST IS REQUIRED

VDB IS VELOCITY DEADBAND

VDB = 1.1 * DELTAT

C

IF (DELBV(1) .GT. VDB * JET(PICK1)) C(1) = 1 \\
IF (DELBV(1) .LT. VDB * JET(PICK2)) C(2) = 1 \\
IF (DELBV(2) .GT. VDB * JET(PICK3)) C(3) = 1 \\
IF (DELBV(2) .LT. VDB * JET(PICK4)) C(4) = 1 \\
IF (DELBV(3) .GT. VDB * JET(PICK5)) C(5) = 1 \\
IF (DELBV(3) .LT. VDB * JET(PICK6)) 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)

C

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, SA

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(0.5*WB*WB/JET1)+DB*.9
    N=INT(-WB/(JET2*DELTAT))
    SA=ABS(N*WB+JET2*DELTAT*N*(N-1)/2.0)*DELTAT+2*DB
    IF (ANG .LE. SA) C1=1
    SA=ABS(0.5*(WB+JET2*DELTAT)**2/JET1)
    SA=SA+2*DB+WB*DELTAT
    IF (ANG .GT. SA) C2=1
  ELSE
    SA=ABS(0.5*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(0.5*(WB+JET1*DELTAT)**2/JET2)
    SA=SA+2*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
ENDIF
ENDIF
ELSE
  IF (((WB+DELTAT+JET2) .LT. -MAX) C2=0
  IF (WB .LT. -MAX) C1=1
ENDIF
C
END
SUBROUTINE CROSS(A, B, C)

COMPUTES  \( A = B \times C \)

REAL A(3), B(3), C(3)

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)

END
SUBROUTINE DISC(JET1,JET2,W,T,DELTA,T,MAX,C1,C2,C01,C02,DS3,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,DELTA,T,MAX,W,T,DS3,SA,ANG
INTEGER C1,C2,C01,C02

ANG - ANGLE FROM WHEN SNAPSHOT TAKEN
SNAP - TRIGGER TO TAKE NEW SNAPSHOT
C01,2 - 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*DELTA

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

SNAP=0
IF ((C1+C2) .EQ. 0) THEN
IF ((C01-C1) .GT. 0) SNAP=1
IF ((C02-C2) .GT. 0) SNAP=1
IF (SNAP .EQ. 1) ANG=0.0

DETERMINE THRUST PROFILE
CALL CHECK(ANG,DS3,W,T,C1,C2,JET1,JET2,T,MAX,DELTA)

ELSE

COMPARE RATES TO MAX RATE AND SET THRUST COMMANDS

IF (WB .GT. 0.0) THEN
IF ((WB+DELTA*JET1) .GT. MAX) C1=0
IF (WB .GT. MAX) C2=1
ELSE
IF ((WB+DELTA*JET2) .LT. -MAX) C2=0
IF (WB .LT. -MAX) C1=1
ENDIF
ENDIF
END
SUBROUTINE ECOCHECK(TABLE,JET)

ECOCHECK OF TABLE AND JET

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

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

PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT *, 'ECOCHECK 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 *, 'ECOCHECK 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 *, 'ECOCHECK OF PII 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 175,'+ ROL*(TABLE(3,7,J),J=1,9)/JET(3,7)
PRINT 175,'- ROL*(TABLE(3,8,J),J=1,9)/JET(3,8)
PRINT 175,'+ PCH*(TABLE(3,9,J),J=1,9)/JET(3,9)
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 175,' HIT RETURN TO CONTINUE'
READ *
C
PRINT *,(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
PRINT * ECOCHECK OF PZHI THRUSTER TABLE AND ACCEL*
PRINT *
C
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 *
C
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 *
C
PRINT 175,'+ ROL*(TABLE(4,7,J),J=1,9)/JET(4,7)
PRINT 175,'- ROL*(TABLE(4,8,J),J=1,9)/JET(4,8)
PRINT 175,'+ PCH*(TABLE(4,9,J),J=1,9)/JET(4,9)
PRINT *
C
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 *
C
PRINT * HIT RETURN TO CONTINUE'
READ *
C
PRINT *(CHAR(027),CHAR(072),CHAR(027),CHAR(074))
C
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 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 PSVPSERR, DETERH, 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
DIFFERENT ERROR HANDLER SO AS NOT TO GET CAUGHT IN A RECURSIVE LOOP IF WE CONSISTENTLY GET AN ERROR WHEN ATTEMPTING TO DETACH.

CALL PDTACH (DETERM)
CLOSE (UNIT=10)
IF (ERRCOD .LT. PSPPAF) .OR.
& (ERRCOD .GT. PSPPPF)) GOTO 2

IDENTIFY VMS ERROR IF THERE WAS ONE
CALL LIB$STOP (XVAL (PSVMSERR ()))
GOTO 3

ELSE
STOP
END IF

RETURN
END

SUBROUTINE DETERM (ERRCOD)

MAIN ERROR HANDLER DETACH ERROR HANDLER:

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)

PIDCOD: IDENTIFY PROCEDURAL INTERFACE COMPLETION CODE.

INCLUDE 'PROCONST.FOR/NOLIST'
INTEGER*4 ERRCOD
INTEGER*4 VMSDEF*133, PIDEN*133
INTEGER*4 PSVMSERR
CHARACTER*1555, MMSG2*67
PARAMETER (MSSG1 = 'PS-W-UNRCOMCOD: PROCEDURAL'
& '/INTERFACE'
& '(GSR) COMPLETION')
EXTERNAL PSVMSERR

WRITE (10, *) 'PS-I-PROERRWAR: PROCEDURAL'
& '/INTERFACE WARNING/'
// "ERROR COMPLETION CODE WAS * RECEIVED."
WRITE (10, *) 'PS-W-DAMCHAR: BAD CHARACTER'.
& // "IN NAME WAS "
& // "TRANSLATED TO: """
GOTO 1000
C ELSE
1 IF (ERRCOD .NE. PSWBC) GOTO 1
WRITE (10, *) 'PS-W-BADNAMCHR: BAD NAME'.
& // "LONG NAME WAS "
& // "TRANSLATED TO "
WRITE (10, *) '256 CHARACTERS'.
GOTO 1000
C ELSE
2 IF (ERRCOD .NE. PSWNTL) GOTO 2
WRITE (10, *) 'PS-W-STRTOOLON: STRING TOO '
& // "LONG STRING "
& // "WAS TRUNCATED "
WRITE (10, *) 'TO 240 CHARACTERS'.
GOTO 1000
C ELSE
7 IF (ERRCOD .NE. PWAAD) GOTO 7
WRITE (10, *) 'PS-W-ATTALRDON: ATTACH '.
& // "ALREADY DONE."
& // "MULTIPLE CALL TO PATCH WITHOUT INTERVENING PATCH CALL IGNORED."
GOTO 1000
C ELSE
8 IF (ERRCOD .NE. PSWAKS) GOTO 8
WRITE (10, *) 'PS-W-ATTKEYSEE: ATTENTION KEY'.
& // "SEEN (DEPRESSED)."
CALL PIBMSP
GOTO 1000
C ELSE
9 IF (ERRCOD .NE. PSWBC) GOTO 9
WRITE (10, *) 'PS-W-BADGENCHR: BAD GENERIC '
& // "CHANNEL CHARACTER BAD."
WRITE (10, *) 'CHARACTER IN STRING SENT VIA: '
& // "PUTGX WAS TRANSLATED TO "
WRITE (10, *) 'A BLANK'.
CALL PIBMSP
GOTO 1000
C ELSE
10 IF (ERRCOD .NE. PSWBSC) GOTO 10
WRITE (10, *) 'PS-W-BADSTRCHR: BAD '.
& // "CHARACTER IN STRING WAS "
& // "TRANSLATED TO A BLANK."
CALL PIBMSP
GOTO 1000
C ELSE
11 IF (ERRCOD .NE. PSWBSC) GOTO 11
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-INVUXCHA: 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-PREOPEEXP: PREFIX '
   & // 'OPERATOR CALL WAS '
   & // 'EXPECTED.'
   GOTO 1000
C ELSE
16 IF (ERRCOD .NE. PSEFDE) 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. PSEMMLB) 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. PSEMVB) GOTO 22
   WRITE (10, *) 'PS-E-MISVECBE: MISSING ' &
   // 'VECTOR LIST BEGIN'
   & 'CALL: CALL TO PVCLIS'
   WRITE (10, *) 'OR PVCEG WITHOUT CALL '
   & 'TO: PVCEG.'
   GOTO 1000
C ELSE
22 IF (ERRCOD .NE. PSENUN) 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-BADCOMTY: 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 NODE '
   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. PSEND) 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, *) '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-PHDEVTYPE: MISSING ' 
& ' OR INVALID PHYSICAL ' 
& ' DEVICE TYPE'
WRITE (10, *) 'SPECIFIER IN CALL TO PATTCH.'
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 PATTCH.'
CALL PVAXSP
GOTO 1000
C ELSE
40 IF (ERRCOD .NE. PSFPAF) GOTO 42
WRITE (10, *) 'PS-F-PHYATTFAI: ' 
& ' PHYSICAL ATTACH OPERATION ' 
& ' FAILED.'
GOTO 1000
C ELSE
41 IF (ERRCOD .NE. PSFPDF) GOTO 43
WRITE (10, *) 'PS-F-PHYDETFAI: PHYSICAL ' 
& ' DETACH OPERATION ' 
& ' FAILED.'
GOTO 1000
C ELSE
42 IF (ERRCOD .NE. PSFPGF) GOTO 44
WRITE (10, *) 'PS-F-PHYGETFAI: PHYSICAL ' 
& ' GET OPERATION FAILED.'
GOTO 1000
C ELSE
43 IF (ERRCOD .NE. PSFBTL) GOTO 46
WRITE (10, *) 'PS-F-SUITTOCL: 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
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
WRITE (10, *) 'PS-F-PROTOCOLAR: 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
UNKNOWN ERROR MESSAGE ERROR MESSAGE.
C END IF
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
MSSG2 = MSSG1 // "FATAL ERROR *
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. PSFPFF) 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: ', PSVMERR()
C END IF
2000 WRITE (10, *)
  RETURN
END

SUBROUTINE PIBMSP
C
PIBMSP: WRITE THE "IBM VERSION SPECIFIC"
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
SPECIFIC" MESSAGE TO THE ERROR
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)
C
C THIS SUBROUTINE FILLS IBUF WITH DATA THAT DOES NOT CHANGE
DURING PROGRAM. THIS DATA IS USED TO ADDRESS PS300 MEMORY
LOCATIONS.
C
INTEGER*4 SYSSQIOW, SYSSSETF, SYSSWAITFR
INTEGER*4 CHAN, STATUS, SYSSASSIGN, SYSSQIOW, NAMADX(20)
INTEGER*2 IBUF(3), NAMES(5, 20), NAMADR(2, 20), IOSB(4), BUFSIZE
CHARACTER*4 UNIT
EQUIVALENCE (NAMADR, NAMADX)
DATA NAMES/'TA', 'RG', 'T', 'LO', 'OK', 'XY', 'ZV', 'C', 'LO', 'OK',
   'GL', 'OB', 'E', 'LO', 'OK', 'ST', 'AR', 'S', 'LO', 'OK',
   'GL', 'OB', 'E', 'TR', '0T', 'ST', 'AR', 'S', 'TR', '0T',
   'XY', 'PT', 'H', 'TR', 'N1', 'XY', 'PT', 'H', 'TR', 'N2',
   'XY', 'PT', 'H', 'TR', 'N3', 'XY', 'PT', 'H', 'TR', 'N4',
   'XY', 'PT', 'H', 'TR', 'N5',
   'IN', 'FO', 'R', 'AN', 'GE', 'IN', 'FO', 'R', 'RA', 'TE',
   'IN', 'FO', 'R', 'FU', 'EL', 'IN', 'FO', 'R', 'TI', 'ME',
   'XY', 'PT', 'H', 'SS', 'TR', 'XY', 'PT', 'H', 'RO', 'TZ',
   'XY', 'PT', 'H', 'RO', 'TX'/
DATA UNIT/'PIAO'/'

C
C WAIT
C
DO 5 J=1, 100000
CONTINUE
5

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(CHAN, %VAL(34), IOSB, /)

ATTACH:
33 --> ATTACH FUNCTION CODE

STATUS=SYSSQIOW(CHAN, %VAL(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(CHAN, %VAL(43), IOSB, /)
   %VAL(10), %VAL(1),
   IF (STATUS .EQ. 1.AND. IOSB(1).EQ. 1.AND.
E-41

1 (IOSB(3).OR.IOSB(4)).NE.0) GOTO 21

TYPE *,'BAD ENTITY FETCH!' <STAT/IOSB> -- <STATE/IOSB>

STOP

GET THE ADDRESS FROM OUT OF THE IO STATUS BLOCK (IOSB)

21 DO 24 J=1,2

NAMADR(J,I)=IOSB(J+2)

24 CONTINUE

OFFSET THE ADDRESSES TO GET PAST THE FIRST THREE FIELDS
OF THE NODE WHICH WE DO NOT WANT TO CHANGE.

DO 30 I=1,20

NAMADX(I)=NAMADX(I)+8

30 CONTINUE

OFFSET TEXT BY AN ADDITIONAL 16

DO 31 I=14,17

NAMADX(I)=NAMADX(I)+16

31 CONTINUE

BUFFER 1 SETUP

TRANSLATION NEEDS 7 ELEMENTS

ROTATION NEEDS 19 ELEMENTS

LOOKAT NEEDS 28 ELEMENTS

IBUF(1) = 20 ! TWENTY BLOCKS
IBUF(2) = NAMADR(1,1) ! BLOCK ONE ADDRESS - TARGT.LOOK
IBUF(3) = NAMADR(2,1) ! BLOCK ONE ADDRESS
IBUF(4) = 27 ! BLOCK ONE ADDRESS
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 ! BLOCK TWO ADDRESS
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 ! BLOCK 3 ADDRESS
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 ! BLOCK 4 ADDRESS
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 ! BLOCK 5 ADDRESS
IBUF(144) = NAMADR(1,6) ! BLOCK 6 ADDRESS - STARS.TROT
IBUF(145) = NAMADR(2,6) ! BLOCK 6 ADDRESS
IBUF(146) = 19 ! BLOCK 6 ADDRESS
IBUF(166) = 1 ! BLOCK 7 ADDRESS - GLOBE.EXT
E-42

IBUF(167) = NAMADR(2,7) ! BLOCK 7 ADDRESS
IBUF(158) = 19 ! WORD COUNT FOR BLOCK 7
IBUF(188) = NAMADR(1,8) ! BLOCK 8 ADDRESS
IBUF(189) = 19 ! WORD COUNT FOR BLOCK 8
IBUF(190) = NAMADR(2,8) ! BLOCK 9 ADDRESS - SATEL.
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.RME
IBUF(279) = NAMADR(2,17) ! BLOCK 17 ADDRESS
IBUF(280) = 3 ! WORD COUNT FOR BLOCK 17
IBUF(283) = 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, 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 SYSWAITFR (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 SYSWAITFR (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 = ((DIBUF(1) . XOR. DATAIN) . AND. 64) / 64
DATAOUT = "6000"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)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
270 CALL LPAIO (19, ISTAT, CALL, LSTAT, )
IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 270
CALL LPAIO (MODE IN, UNIT, DIFLAG, ICHAN, NCHAN, NFRAME, DIBUF,
1 ISTAT, ICALL, ISMODE)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
280 CALL LPAIO (17, ISTAT, CALL, LSTAT, )
IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 280
DATAIN=DI BUF(1).XOR.'FFFF'X)
S(7)=DATAIN .AND. 1
S(8)=(DATAIN .AND. 2)/2
S(9)=(DATAIN .AND. 4)/4
S(10)=(DATAIN .AND. 8)/8
S(11)=(DATAIN .AND. 16)/16
S(12)=(DATAIN .AND. 32)/32
S(13)=(DATAIN .AND. 64)/64
S(14)=(DATAIN .AND. 128)/128
S(15)=(DATAIN .AND. 256)/256
DATAOUT= 8000'X
DO 290 I=16,24
290 DATAOUT=DATAOUT+L(I)*2**(I-16)
DO 300 I=1,NFRAME
300 DOBUF(I)=DATAOUT .XOR. 'FFFF'X
CALL LPAIO (MODEOUT, UNIT, DIFLAG, ICHAN, NCHAN, NFRAME, DOBUF,
1 ISTAT, ICALL, ISMODE)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
310 CALL LPAIO (18, ISTAT, CALL, LSTAT, )
IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 310
CALL LPAIO (MODE IN, UNIT, DIFLAG, ICHAN, NCHAN, NFRAME, DIBUF,
1 ISTAT, ICALL, ISMODE)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
320 CALL LPAIO (17, ISTAT, CALL, LSTAT, )
IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 320
DATAIN=DI BUF(1).XOR.'FFFF'X)
S(16)=DATAIN .AND. 1
S(17)=(DATAIN .AND. 2)/2
S(18)=(DATAIN .AND. 4)/4
S(19)=(DATAIN .AND. 8)/8
S(20)=(DATAIN .AND. 16)/16
S(21)=(DATAIN .AND. 32)/32
S(22)=(DATAIN .AND. 64)/64
S(23)=(DATAIN .AND. 128)/128
S(24)=(DATAIN .AND. 256)/256
DATAOUT= 8000'X
DO 340 I=1,NFRAME
340 DOBUF(I)=DATAOUT .XOR. 'FFFF'X
CALL LPAIO (MODEOUT, UNIT, DIFLAG, ICHAN, NCHAN, NFRAME, DOBUF,
1 ISTAT, ICALL, ISMODE)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
350 CALL LPAIO (18, ISTAT, CALL, LSTAT, )
IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 350
CALL LPAIO (MODE IN, UNIT, DIFLAG, ICHAN, NCHAN, NFRAME, DIBUF,
1 ISTAT, ICALL, ISMODE)
IF (.NOT. ISTAT) GO TO 950
CALL SYSSWAITFR (XVAL(DIFLAG))
CALL LPAIO (17, SIST, ICALL, LSTAT)
IF ((ISTAT .NE. -1).AND. (LSTAT .NE. 1)) GO TO 360
DATAIN = (DIBUF(1) .XOR. 'F4F4'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

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)

C 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)
100 CONTINUE

END
SUBROUTINE LOOK(T,X,OMEGA,TIMEFUELIBUFCHAN, 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 'ALFANO/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),BMA(3,3),BVEC(3),AMAT(3,3)

INTEGER*4 CHANSYSQIO,SYS$QIOW
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.

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

CALL BTOR(T.UP.UPPS)

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

CONVERT AT, FM AND UPPS FROM FEET TO METERS

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

DETERMINE EARTH ROTATION (DEG)

EROT=TIME*.004178075

DETERMINE TARGET ROTATION (DEG)

TROT=OMEGA*57.29577951*TIME

DETERMINE SATELLITE SPIN (DEG)

RSAT=-TIME*30.0

SEND INFORMATION TO PS300

CALL P919CV(XYPTH,3,IBUF(287))
CALL ROT(90.1,AMAT)
CALL P919CV(AMAT,7,IBUF(291))
CALL TLHT(T,AMAT)
CALL P919CV(AMAT,9,IBUF(297))

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))

CALL LOOKAT(AT,FM,UPPS,3MAT,BVEC)
CALL P919CV(3MAT,7,IBUF(5))
CALL P919CV(BVEC,3,IBUF(25))
CALL P919CV(3MAT,9,IBUF(35))
CALL P919CV(BVEC,3,IBUF(55))

CALL PSNREA(RANGE*.01,1,"XYZVECSCL",ERR)
RESCALE TO DUS

DO 57 I=1,3
    AT(I)=AT(I)/6378135.
    FM(I)=FM(I)/6378135.
    UPPS(I)=UPPS(I)/6378135.
    CONTINUE

CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
CALL P919CV(BMAT,9,IBUF(65))
CALL P919CV(BVEC,3,IBUF(85))
CALL P919CV(BMAT,9,IBUF(95))
CALL P919CV(BVEC,3,IBUF(115))

CALL NUMBER(RANGE,IBUF(263))
CALL NUMBER(RRATE,IBUF(269))
CALL NUMBER(FUEL,IBUF(275))
CALL NUMBER(TIME,IBUF(281))

COMPUTE FUTURE POSITIONS FOR HEADS UP DISPLAY
AND SEND TO PS300 (DT IS TIME INCREMENT)

DT=300.
CALL PREDPATH(OMEGA,DT,X,NEWX)
CALL P919CV(NEWX,3,IBUF(213))
CALL PREDPATH(OMEGA,2.*DT,X,NEWX)
CALL P919CV(NEWX,3,IBUF(223))
CALL PREDPATH(OMEGA,3.*DT,X,NEWX)
CALL P919CV(NEWX,3,IBUF(233))
CALL PREDPATH(OMEGA,4.*DT,X,NEWX)
CALL P919CV(NEWX,3,IBUF(243))
CALL PREDPATH(OMEGA,5.*DT,X,NEWX)
CALL P919CV(NEWX,3,IBUF(253))

DO A WRITE SYNC
42 WRITE SYNC FUNCTION CODE
I0B DATA BUFFER (ACTUALLY ADDRESS OF BUFFER BY REFERENCE)
674 DATA BYTE COUNT (337 WORDS)
0 NOT CHARACTER DATA (1 = CHARACTER DATA)

SEND ALL DATA

IF (STATUS .NE. 1) THEN
    TYPE = 'BAD WRITE <STATUS,IOSB>,STATUS,IOSB'
STOP
ENDIF

END

SUBROUTINE ROT(ANGLE,IAxis,AMAT)

ROUTINE TO GENERATE ROTATION REQUESTS TO PS300
CALLING SEQUENCE:

CALL ROT(ANGLE, IAXIS, AMAT)

WHERE:

ANGLE IS THE REAL*4 ANGLE FOR ROTATION, IN DEGREES. NEED NOT BE LIMITED
TO A SINGLE CIRCLE.
IAXIS IS THE INTEGER*2 AXIS OF ROTATION (1=X, 2=Y, 3=Z).
AMAT IS THE REAL 3X3 MATRIX CALCULATED

INTEGER*2 IAXIS, I, J
REAL ANGLE, AMAT(3, 3), PI180
DATA PI180/0.0174531/

IDX(I)=MOD(I+2,3)+1 ! STATEMENT FUNCTION (NOT AN ARRAY)

IF(IAXIS.LT.1.OR.IAXIS.GT.3) STOP 'PSLIB--AXIS OUT OF BOUNDS'
DO 10 I=1, 3
DO 10 J=1, 3
10 AMAT(I, J)=0.E0
RADIAN = ANGLE * PI180
C=COS(RADIAN)
S=SIN(RADIAN)
AMAT(IAXIS, IAXIS)=1.E0
I=IDX(IAXIS-1)
J=IDX(IAXIS+1)
AMAT(I, J)=C
AMAT(J, J)=C
AMAT(I, J)=S
AMAT(J, I)=-S
RETURN
END

SUBROUTINE TLHT(T, AMAT)

ROUTINE TO CONVERT A RIGHT HANDED ROTATION MATRIX (T)
TO A LEFT HANDED ROTATION MATRIX (AMAT)

CALLING SEQUENCE:

CALL TLHT(T, AMAT)

REAL AMAT(3, 3), T(3, 3)

AMAT(1, 1)=T(1, 1)
AMAT(2, 1)=T(2, 1)
AMAT(3, 1)=-T(3, 1)
AMAT(1, 2)=T(1, 2)
AMAT(2, 2)=T(2, 2)
AMAT(3, 2)=-T(3, 2)
AMAT(1, 3)=-T(1, 3)
AMAT(2, 3)=T(2, 3)
AMAT(3, 3)=T(3, 3)
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 MANTISSES, 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 MANTISSES.

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
10 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
20 DWORD = PSMNOR(MATRIX(I), EDWORD)
BUFFER(2*I) = FWORD(2)
BUFFER(2*I+1) = FWORD(1)
CONTINUE
RETURN
END

SUBROUTINE LOOKATAT(FRAM, 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 <= INDEX <= 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
SUFFER IS THE RESULTING 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)
ELSE
   E(1) = E(1) - M*D(1)
   E(2) = E(2) - M*D(2)
   E(3) = E(3) - M*D(3)
ENDIF
ENDIF

MAG = SQRT(F(1)**2 + F(2)**2 + F(3)**2)

G(1) = F(1)/MAG
G(2) = F(2)/MAG
G(3) = F(3)/MAG

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)

T(1) = -FM(1)*H(1) - FM(2)*H(2) - FM(3)*H(3)
T(2) = -FM(1)*G(1)*F(2) + FM(2)*G(1)*F(3) - FM(3)*G(1)
T(3) = -FM(1)*G(2)*F(2) + FM(2)*G(2)*F(3) - FM(3)*G(2)

MAT(1,1) = H(1)
MAT(1,2) = H(2)
MAT(1,3) = H(3)
VEC(1) = T(1)

MAT(2,1) = G(1)
MAT(2,2) = G(2)
MAT(2,3) = G(3)
VEC(2) = T(2)

MAT(3,1) = D(1)
MAT(3,2) = D(2)
MAT(3,3) = D(3)
VEC(3) = T(3)

RETURN
END

SUBROUTINE P92OCVMATVECPBUFFER

THIS SUBROUTINE FILLS IN A 4 X 4 ARRAY INTO THE BUFFER.
YOU LOAD A 3 X 4 MAT, AND AN ARRAY VEC(3) IS THE FOURTH ROW.
IT FILLS 34 ELEMENTS IN THE BUFFER.

INTEGER*2 ITEMPI(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,1) = 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

SUBROUTINE NUMBER(X,BUFFER)
C
C THIS SUBROUTINE WILL TAKE ANY NUMBER FROM 9999.9 TO -999.9,
C ENCODE IT, AND PUT IT IN THE PROPER PLACE IN THE OUTPUT
C 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
ENCEDE(6,201,CHAR)X
201 FORMAT(F6.1)

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

REV A DESIGNED TO USE AST CALLS AT COMPLETION OF SWEEP (FILE NAME LPAIOA).

REV B MODIFIED TO USE EVENT FLAGS INSTEAD OF AST CALLS 03 FEB 83 (JML)

REV C MODIFIED TO ADD I/O MODE CALLING PARAMETER "ISMODE" AND TO ALLOW FOR 2 CHANNEL DIGITAL I/O WITH THE ADDITION OF MODES 7, 8, 17, 18.

CALLING SPECIFICATIONS:

MODE OF CALL WITH:

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

IUNIT UNIT NUMBER OF THE DESIRED LPA11 SUBSYSTEM:

IUNIT = 0 USES LAAO:
= 1 USES LABO:

IFLAG NUMBER OF THE EVENT FLAG WHICH IS TO BE SET A COMPLE

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)

IOBUF BUFFER FOR DATA (NFRAME * NCHAN 2 BYTE WORDS LONG)

RETURNED INFORMATION:

ARATE ACTUAL SAMPLE RATE USED (0 FOR ERROR)

ISTAT THREE WORD ARRAY WITH:

ISTAT = 0 ERROR IN CALL
= 1 SUCCESSFUL
= X VMS ERROR CODE
E-56

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

<table>
<thead>
<tr>
<th>ISTAT</th>
<th>LSTAT</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>-</td>
<td>NORMAL - BUFFER 0 DONE</td>
</tr>
<tr>
<td>-1 1</td>
<td>-</td>
<td>SWEEP TERMINATED OK</td>
</tr>
<tr>
<td>-1 X</td>
<td>X</td>
<td>X = LPA11 ERROR CODE (USER'S GD</td>
</tr>
</tbody>
</table>

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 MULTIPLE 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 DIGITAL 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
ALL SWITCHES ON S1 AND S2 ARE OFF, THE STATE OF JUMPERS W1-W6 IS A
DON'T CARE. JUMPERS W21-W23 SHOULD BE SET FOR APPROPRIATE POLARITY
OF THE INTERNAL DATA ACCEPT AND INTERNAL DATA READY LINES.

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

<table>
<thead>
<tr>
<th>MODE</th>
<th>INUM</th>
<th>IFLAG</th>
<th>DRATE</th>
<th>ICHAN</th>
<th>NCHAN</th>
<th>NFRAME</th>
<th>IOBUF</th>
<th>AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

***** PROGRAM DECLARATIONS *****

VARIABLE DEFINITION SECTION

IOBUF   DATA BUFFER AREA (INTEGER*2)
XYBUF   LPA11 CONTROL BLOCK (50 LONGWORDS)
XYSTAT  LPA11 COMPLETION STATUS (FORM LPASIGTBUF CALL)
IFLAG   LPA11 FLAG TO BE SET AT COMPLETION OF SWEEP
XYIOSB  I/O STATUS BLOCK FOR LPA11 (4 WORDS)
DXIOSZ  I/O STATUS BLOCK FOR DIGITAL I/O TO BUFFER 1 (CH A OR B)
XYMSKB  LPA11 SUBSYSTEM MASKS AND NUM BUFFER
ISTAT   LPA11 STATUS LONGWORD
LSTAT   LPA11 I/O COMPLETION STATUS BYTE
NBUF    NUMBER OF BUFFERS TO BE FILLED (LONGWORD)

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

VARIABLE TYPE SPECIFICATIONS

REAL    LPA$XRATE
INTEGER*4 SYSSCLREF

INTEGER*2 IOBUF(1),ADIOSB(4),DAIOSB(4)
INTEGER*2 DIIOSA(4),DOIOMSA(4),DIIOSB(4),DOIOMSB(4)
INTEGER*4 ADMSKB(2),DOMSBK(2)
INTEGER*4 DMSKB(2),DMSKB(2),DOMSBK(2)
INTEGER*4 ISTAT,BUFFNUM,NBUF,IFLAG
BYTE    IDSC,IEMC,LSTAT
INTEGER*2 IDSW,IEMW

SET AREA FOR CONTROl

INTEGER*4 ADBUF(50),DABUF(50)
INTEGER*4 DUBUF(50),DUBUF(50),DIBUF(50),DODBUF(50)
EQUIVALENCE (ADIOSB(1),ADBUF(1)),(DAIOSB(1),DABUF(1))
EQUIVALENCE (DIIOSA(1),DIUBUF(1)),(DOIOMSA(1),DODBUF(1))
EQUIVALENCE (DIIOSB(1),DIBUF(1)),(DOIOMSB(1),DODBUF(1))
E-59

C START OF PROGRAM ****
C DETERMINE MODE
C
C
C LSTAT = 0
GO TO (100,50,300,400,500,600,700,800,50,50,50,50,1300,1400,1500,1600,1700,1800)/MODE
C EXECUTION STARTS HERE IF MODE IS UNDEFINED
50 ICALL = 0
ISTAT = 0
GO TO 1950
C
C MODE = 1 ****
C LOAD LPA11 SPECIFIED BY IUNIT WITH MICROCODE FOR MULTIREQUEST MODE
C
100 CONTINUE
ICALL = 101
CALL LPASLOADMC (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
AINTRL = 1./IRATE
ICALL = 102
ACTUAL = LPASXRATE (AINTRL,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
**** MODE = 3 ****
START ANALOG-TO-DIGITAL INPUT SWEEP

CONTINUE

CLEAR A/D EVENT FLAG
ICALL = 300
ISTAT = SYS$CLREF (VAL(IFLAG))
IF (.NOT. ISTAT) GO TO 1950

INITIALIZE ADBUF ARRAY FOR SWEEP
ICALL = 301
CALL LPASETIBF (ADBUF, ISTAT, ADMSKB, IOBUF)
IF (.NOT. ISTAT) GO TO 1950

SET UP FOR LPA11 SUBSYSTEM NUMBER
ICALL = 302
CALL LPASLAMSKS (ADMSKB, IUNIT)

RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
ICALL = 303
CALL LPASRLSBUF (ADBUF, ISTAT, O)
IF (.NOT. ISTAT) GO TO 1950

START A/D SWEEP BY SPECIFYING ONLY ONE BUFFER
NPOINT = NFRAME * NCHAN
SPECIFY ONLY ONE BUFFER TO BE FILLED
NBUF = 1
IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETER
PROCEED WITH SWEEP START CALL
SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
ICALL = 304
CALL LPASADSWP (ADBUF, NPOINT, NBUF, ISMODE, VAL(IFLAG), ICHAN, 1 NCHAN, ISTAT)
IF (.NOT. ISTAT) GO TO 1950

RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF A/D SWEEP CALL
GO TO 1950
E-61

C *** MODE = 4 ***
C START DIGITAL-TO-ANALOG OUTPUT SWEEP
C
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,IOBUF)
IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
ICALL = 402
CALL LPASLAMSKS (DAMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
ICALL = 403
CALL LPASRLSBUF (DABUF,ISTAT,IO)
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 10 KHZ; 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),
LDELAY,ICHAN,NCHAN,ISTAT)
IF (.NOT. ISTAT) GO TO 1950
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 = SYS$CLREF (%VAL(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 LPASSETIBF (DIBUFA,ISTAT,DIMSFA,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
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
IEMC = 0
C
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
IDSW = -1
C
C SPECIFY EVENT MARK WORD MASK OF ALL BITS
IEMW = -1
ICALL = 503
CALL LPASLAMSKS (DIMSFA,IUNIT,0,IDSC,IEMC,IDS,W,0,EW--)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
ICALL = 504
CALL LPASRLSBUF (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
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
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
ICALL = 505
CALL LPASDISWP (DIBUFA,NPOINT,NBUF,ISMODE,%VAL(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
GO TO 1950
C ++++ MODE = 6 ++++
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
600 CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
ICALL = 600
ISTAT = SYSSCLREF (IVAL(IFLAG))
IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT MCHAN IS EQUAL TO ONE
ISTAT = 0
ICALL = 601
IF (MCHAN.NE.1) GO TO 1950
ISTAT = 1
C
C INITIALIZE DOBUFA ARRAY FOR SWEEP
C
ICALL = 602
CALL LPASSETIBF (DOBUFA,ISTAT,DOMSKA,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
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
IEMC = 0
C
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
IDSW = -1
C
C SPECIFY EVENT MARK WORD MASK OF ALL BITS
IEMW = -1
ICALL = 603
CALL LPASLAMSKS (DOMSKA,IUNIT,IDSC,IEMC,IDSW,IEMW)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
ICALL = 604
CALL LPASRLSBUF (DOBUFA,ISTAT,0)
IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
NPOINT = NFRAME
C
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 DO 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 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
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 = 605
CALL LPA&DOSWP (DOBUFA, MPOINT, NBUF, ISMODE, IDWELL, %VAL(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
GO TO 1950
C ****** MODE = 7 ******
C START DIGITAL INPUT SWEEP FOR "CHANNEL 8"
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 PITS
IDSW = -1
C SPECIFY EVENT MARK WORD MASK OF ALL BITS
IEMW = -1
ICALL = 703
CALL LPASLAMSKS (DIMSKB,UNIT,,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 RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT SWEEP CALL
GO TO 1950
C * Mode = 8 *
C Start Digital Output Sweep for "Channel B"

C 800 Continue
C
C Clear Digital Output Event Flag
ICALL = 800
ISTAT = SYSSCLARF (IVAL(IFLAG))
IF (.NOT. ISTAT) GO TO 1950
C
C Check that NCHAN is equal to one
ISTAT = 0
ICALL = 801
IF (NCHAN.NE.1) GO TO 1950
ISTAT = 1
C
C Initialize DOBUF Array for Sweep
ICALL = 802
CALL LPASSETIBF (DOBUF,ISTAT,DOMSKB,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
C Specify Event Mark Word Channel of Channel 0
IEMC = 0
C
C Specify Digital Start Word Mask of all Bits
IDSW = -1
C
C Specify Event Mark Word Mask of all Bits
IEMW = -1
ICALL = 803
CALL LPASLAMSKS (DOMSKB,UNIT,;IDSC,IEMC,IDSW,IEMW)
C
C Release the Buffer (Buffer Numbers are used rather than Names)
ICALL = 804
CALL LPASRLSBUF (DOBUF,ISTAT,0)
IF (.NOT. ISTAT) GO TO 1950
C
C For this mode, the number of points must equal number of frames
NPOINT = NFRAME
C
C Specify only one buffer to be filled
NBUF = 1
C
C In Rev C of this routine, ISMODE is specified in the calling parameter
C
C In DO 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 10 kHz; etc.
LDELAY = 1
IF (IRATE.EQ.1) LDELAY = 150
IF (IRATE.EQ.2) LDELAY = 15
IF (IRATE.EQ.3) LDELAY = 2
C
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 = 805
CALL LPASDOSWP (DOBUF, NPOINT, HBUF, ISMODE, IDWELL, %VAL(IFLAG),
1 DELAY, 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
E-68

**** MODE = 13 ****
C GET STATUS OF A/D SWEEP
C
1300 CONTINUE
C
ICALL = 1301
ISTAT = LPASIWTBUF(ADBUF)
LSTAT = IAND(ADIOSB(3),*FFOO*X)/256
GO TO 1950
C
**** MODE = 14 ****
C GET STATUS OF D/A SWEEP
C
1400 CONTINUE
C
ICALL = 1401
ISTAT = LPASIWTBUF(DABUF)
LSTAT = IAND(DA IOSB(3),*FFOO*X)/256
GO TO 1950
C
**** MODE = 15 ****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
1500 CONTINUE
C
ICALL = 1501
ISTAT = LPASIWTBUF(DIBUFA)
LSTAT = IAND(DI IOSA(3),*FFOO*X)/256
GO TO 1950
C
**** MODE = 16 ****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
1600 CONTINUE
C
ICALL = 1601
ISTAT = LPASIWTBUF(DOBUFA)
LSTAT = IAND(DO IOSA(3),*FFOO*X)/256
GO TO 1950
C
**** MODE = 17 ****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
1700 CONTINUE
C
ICALL = 1701
ISTAT = LPASIWTBUF(DIBUFB)
LSTAT = IAND(DII IOSB(3),*FFOO*X)/256
GO TO 1950
C
**** MODE = 18 ****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
1800 CONTINUE
C
ICALL = 1801
ISTAT = LPASIMTBUF(DOBUF)
LSTAT = IAND(DOIOSB(3),"FFOO"X)/256
GO TO 1950

C
C
C ***** ERROR SERVICE ROUTINE *****
C
1950 CONTINUE
C TRANSFER STATUS INFORMATION ON CALLING PARAMETER
C
RETURN
C
END
SUBROUTINE LVLH(C, COQ, REF, DBX, DBY, DBZ, WB, JET, DELTAT, MAXWX, MAXWY, MAXWZ, PICK, T)

C ATTEMPTS TO HOLD BODY AXES IN ALIGNMENT WITH LVLH AXES

INTEGER C(12), COQ(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(4)

C PI=2.0*ACOS(0.0)

C WAS RHC JUST MOVED TO NEUTRAL? IF SO, TAKE SNAPSHOT.
C REF(4) - QUATERNION AT TIME OF SNAPSHOT.
C SNAP - FLAG TO TAKE SNAPSHOT OF QUATERNION.

SNAP=0
DO 150 I=7,12
   IF (((CO(I)-C(I)) .GT. 0.0) SNAP=1
150 CONTINUE

IF (SNAP .EQ. 1) THEN
   REF(1)=Q(1)
   REF(2)=Q(2)
   REF(3)=Q(3)
   REF(4)=Q(4)
ENDIF

C COMPUTE FUTURE QUATERNION (NEXT ITERATION)
C FW3 - FUTURE BODY RATE
C FQ - FUTURE QUATERNION

ANG(1)=0.0
ANG(2)=0.0
ANG(3)=0.0

FQ(1)=Q(1)
FQ(2)=Q(2)
FQ(3)=Q(3)
FQ(4)=Q(4)

FWB(1)=WB(1)
FWB(2)=WB(2)
FWB(3)=WB(3)

CALL ROTATE(ANG, DELTAT, FWB, FQ, FT)

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

C COMPUTE TRANSFORMATION FROM FQ FRAME TO REF FRAME

DO 81 I=1,3
   DO 82 J=1,3
      QT(I,J)=REFT(I,1)*FT(1,J)+REFT(2,I)*FT(2,J)+
               REFT(3,I)*FT(3,J)
82   CONTINUE
CONTINUE

COMPUTE QUATERNIONS (FROM FG TO REF)

\[
QQ(4) = 1.0 + QT(1,1) + QT(2,2) + QT(3,3)
\]

IF (QQ(4) .LT. 1E-30) QQ(4) = 1E-30

\[
QQ(4) = \text{SORT}(QQ(4))/2.0
\]

\[
QQ(1) = (QT(3,2) - QT(2,3))/(4.0 + QQ(4))
\]

\[
QQ(2) = (QT(1,3) - QT(3,1))/(4.0 + QQ(4))
\]

\[
QQ(3) = (QT(2,1) - QT(1,2))/(4.0 + QQ(4))
\]

NORMALIZE QUATERNIONS

\[
\text{CONST} = \text{SORT}(QQ(1) + QQ(2) + QQ(3) + QQ(4))
\]

\[
QQ(1) = QQ(1)/\text{CONST}
\]

\[
QQ(2) = QQ(2)/\text{CONST}
\]

\[
QQ(3) = QQ(3)/\text{CONST}
\]

\[
QQ(4) = QQ(4)/\text{CONST}
\]

COMPUTE ANGLE OF ROTATION BETWEEN FRAMES

\[
WT = 2.0 \times \text{ACOS}(QQ(4))
\]

\[
S WT = \text{SIN}(WT/2.0)
\]

COMPUTE ANGULAR DIFFERENCE (OF FG W.R.T. REF)

IF (WT .LT. 1.0E-6) THEN

ROLL = 0.0
PITCH = 0.0
YAW = 0.0
ELSE

ROLL = QQ(1) * WT/WT
PITCH = QQ(2) * WT/WT
YAW = QQ(3) * WT/WT
ENDIF

DETERMINE THRUST PROFILE BY CHECKING ROLL, PITCH AND YAW ANGLES/RATES AGAINST ATTITUDE HOLD CRITERIA.

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)

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

C TAKES ONE COLUMN OF DAP PANEL MANUAL MODE AND
C FIGURES OUT WHAT LIGHTS TO TURN ON/OFF (LA, LB, LC)
C 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)

C
C THIS SUBROUTINE PREDICTS POSITION (X) RELATIVE TO
C TARGET GIVEN PRESENT POSITION AND SPEED (OLDX) AND
C FUTURE TIME (TIME)
C
REAL OMEGA, X(3), OLDX(6), TIME, CW, SW

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

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

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

C
C SCALE DOWN FOR HUD DISPLAY
C CONVERT TO LH SYSTEM
C
X(1)=X(1)/100.
X(2)=X(2)/100.
X(3)=-X(3)/100.

C END
SUBROUTINE PS300(ALT, INCL)

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]PROCOST.FOR/NLIST'

LOGICAL*1 POSLIN(73)
REAL ALT, INCL, AT(3), FM(3), UP(3), DSTAR
REAL HORD, HORT, HORD, LONG, ANG, ALT
REAL*4 V(3)
INTEGER I, NVEC
DIMENSION VECS(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
HORD - RADIUS OF EARTH HORIZON AS SEEN FROM TARGET (DU)
HORT - DISTANCE OF HORIZON FROM EARTH CENTER (DU)
DSTAR - DISTANCE OF STAR CLIPPING PLANE FROM TARGET (DU)

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

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

UNITS VARY FROM METERS TO DUS DEPENDING ON WHAT IS BEING DISPLAYED. THIS UNIT JUGGLING IS DONE TO MINIMIZE SCALE 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
UP(3) = 0.0

COMPUTE ATTITUDE DISPLAY DATA FOR HUD

CALL PBEGS("XYZV", ERR)
CALL PSEDCL("CLIP", FALSE, "", "", "", ERR)
CALL PVIEWP("", 1.0, 1.0, -1.0, 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 PScale("SCALE", V, "", "", "", "", ERR)
CALL PSECOL("", 180.0, 1.0, "XARROW", ERR)
CALL PSECOL("", 180.0, 1.0, "YARROW", ERR)
CALL PSECOL("", 180.0, 1.0, "ZARROW", ERR)
CALL PENDS(ERR)

COMPUTE PITCH AND FUTURE TRACK DATA FOR HUD

CALL PBEGS("XYPTH", ERR)
CALL PSEDCL("CLIP", FALSE, "", "", "", ERR)
CALL PVIEWP("", -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0, ERR)
CALL PWINDO("WND", 11.0, 11.0, 11.0, 11.0, 100., ERR)
CALL PLOOKA("LOOK", AT, FM, UP, "", "", ERR)
CALL PROTY("ROTY", 180., "", "", ERR)
CALL PSECOL("", 300.0, 1.0, "", ERR)
CALL PTRANS("TRN", V, "DIAMOND", ERR)
CALL PTRANS("TRN2", V, "DIAMOND", ERR)
CALL PTRANS("TRN3", V, "DIAMOND", ERR)
CALL PTRANS("TRN4", V, "DIAMOND", ERR)
CALL PTRANS("TRN5", V, "DIAMOND", ERR)
CALL PSECOL("", 180.0, 1.0, "XYAXIS", ERR)
CALL PTRANS("SSTR", V, "", ERR)
CALL PROTZ("ROTZ", 0., "", "", ERR)
CALL PROTX("ROTX", 90., "STICKSTS", ERR)
CALL PENDS(ERR)

COMPUTE PIPPER DISPLAY DATA

CALL PBEGS("GUNSITE", ERR)
CALL PSEDCL("CLIP", FALSE, "", "", "", ERR)
CALL PVIEWP("", 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, ERR)
CALL PWINDO("WND", 1.0, 1.0, 1.0, 1.0, 100., Err)
CALL PLOOKA("LOOK", AT, FM, UP, "", "", ERR)
CALL PSECW("", ERR)
CALL PCMSCA("", 80.0, 80.0, 80.0, 80.0, "", ERR)
CALL PCMS("", 80.0, 80.0, 80.0, 80.0, "", ERR)
CALL PCMS("", 79.0, 79.0, 1.0, 1.0, "RANGE (FT)", ERR)
CALL PCMS("", 77.0, 77.0, 1.0, 1.0, "RANGE RATE", ERR)
CALL PCMS("", 75.0, 75.0, 1.0, 1.0, "FUEL USED", ERR)
CALL PCMS("", 73.0, 73.0, 1.0, 1.0, "TIME (SEC)", ERR)

COMPUTE DISPLAY DATA FOR RANGE, RANGE RATE, FUEL AND TIME.

CALL PBEGS("INFO", ERR)
CALL PSEDCL("CLIP", FALSE, "", "", "", ERR)
CALL PVIEWP("", 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, ERR)
CALL PWINDO("WND", 0.0, 80.0, 80.0, 80.0, 100., ERR)
CALL PLOOKA("LOOK", AT, FM, UP, "", "", ERR)
CALL PSCHW("", ERR)
CALL PCMSCA("", 1.0, 1.0, "", ERR)
CALL PCMS("", 0.79, 1.0, 1.0, "RANGE (FT)", ERR)
CALL PCMS("", 0.77, 1.0, 1.0, "RANGE RATE", ERR)
CALL PCMS("", 0.75, 1.0, 1.0, "FUEL USED", ERR)
CALL PCMS("", 0.73, 1.0, 1.0, "TIME (SEC)", ERR)
CALL PCHS('RANGE', 12.79, 1.1, 0., 000000, 'ERR)
CALL PCHS('R RATE', 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')

C

COMPUTE DISPLAY DATA FOR SPINNING SATELLITE

CALL PBEGS('TARGET', 'ERR')

UNITS ARE IN METERS
CALL PSEXCL('CLIP', .FALSE., 'ERR')
CALL PVIEWP('C', -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 'ERR')
CALL PFOW('FOV', -35.0, 1.0, 3000.0, 'ERR')
CALL PLOOKA('LOOK', 'AT', 'FM', 'UP', 'ERR')
CALL PINST('WORLD', 'SATEL', 'ERR')
CALL PENDS('ERR')

C

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('p3OQ.O~1.0.0.*', 'ERR')
DO 150 I = 10, 360, 10

CONTINUE

CALL PROTX('LONG', 'RTDRS', 'ERR')
CALL PENDS('ERR')

C

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 PSEXCL('CLIP', .TRUE., 'ERR')
CALL PVIEWP('C', -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 'ERR')
CALL PFOW('FOV', -35.0, 1.0, 'ALT', +0.1, 'ERR')
CALL PLOOKA('LOOK', 'AT', 'FM', 'UP', 'ERR')
V(1) = 0.0
V(2) = (-1.0, 'ALT')
V(3) = 0.0
CALL PTRANS('V', 'ERR')
CALL PINST('HORIZON', 'ERR')
CALL PROTX('ROT', 0.0, 'ERR')
CALL PSEXCL('R240., 1.', 'ERR')
CALL PROTY('EROT', 0.0, 'ERR')
CALL PINST('WORLD', 'ERR')
CALL PSEXCL('R-240., 0.0', 'ERR')
CALL PINST(*"SHELL",*"SPHERE",*"ERR)
CALL PINST(*"SHELL",*"LATLINE",*"ERR)
CALL PENDS(*"ERR")

CREATE A HORIZON FROM A CIRCLE

CALL PBEGS(*"HORIZON",*"ERR")
UNITS ARE IN D\n
V(1)=0.0
V(2)=NORD
V(3)=0.0
CALL PTRANS(*"V",*"ERR")
CALL PROTX(*"90",*"ERR")
V(1)=HRR
V(2)=V(1)
V(3)=V(1)
CALL PSSCALE(*"V",*"CIRCLE",*"ERR")
CALL PENDS(*"ERR")

CREATE A STAR FIELD

CALL PBEGS(*"STARS",*"ERR")
UNITS ARE IN D\n
CALL PSDCLC(*"CLIP",*TRUE,*,"ERR")
CALL PVIEWP(*"-1.0,1.0,-1.0,1.0,1.0,1.0",*"ERR")
DST=HRR+9*SQRT(2.0*ALT+1.0)**2-1.0
CALL PPV(*"35.0,-1*DST,0*DST,2*STAR",*"ERR")
CALL PLOOKA(*"LOOK",*"AT,FM,UP",*"ERR")
V(1)=0.0
V(2)=1.0+ALT
V(3)=0.0
CALL PTRANS(*"V",*"ERR")
CALL PROTY(*"TROT",0.0,*,"ERR")
CALL PSECOL(*"180",*,"ERR")
CALL PROTY(*"90",*"STARS,TWINKLE",*"ERR")
CALL PROTX(*"90",*"STARS,TWINKLE",*"ERR")
V(1)=1.0+2.0*ALT
V(2)=V(1)
V(3)=V(1)
CALL PSSCALE(*"TWINKLE",*"STAR",*"ERR")
CALL PENDS(*"ERR")

BUILD A SPHERE FROM A CIRCLE

CALL PBEGS(*"SPHERE",*"ERR")
DO 200 I=10,180,10
LONG=I
CALL PROTY(*"LONG",*"CIRCLE",*"ERR")
200 CONTINUE
CALL PENDS(*"ERR")

BUILD A HEMISPHERE FROM A SEMICIRCLE

CALL PBEGS(*"HEMI",*"ERR")
DO 222 I=15,180,15
LONG=I
CALL PROTY(*"LONG",*"SEMI",*"ERR")
222 CONTINUE
CALL PENDS(*"ERR")
COMPUTE LINES OF LATITUDE USING CIRCLES

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)

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)-SINCANG-.174532925
CALL PTRANS('**/V',**/ERR)
V(1)=COS(ANG)
V(2)=V(1)
V(3)=0.0
CALL PSCALE('**/V/CIRCLE',ERR)
CONTINUE
20
CALL PENDS(ERR)

VECTOR LIST FOR CIRCLE

DO 10 I=1,73
ANG=5.0*(I-1)*.0174532925
VECS(1)=COS(ANG)
VECS(2)=SINCANG)
VECS(3)=0.0
VECS(4)=1.0
CONTINUE

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

VECTOR LIST FOR SEMI-CIRCLE

DO 12 I=1,13
ANG=(-90.0+15.0*(I-1))*.0174532925
VECS(1)=SIN(ANG)
VECS(2)=COS(ANG)
VECS(3)=0.0
VECS(4)=1.0
CONTINUE

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

COMMANDS FOR HUD ROTATION

SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
CALL PSNST('HUD', 'YROTHATE', 'ERR)
CALL PFNC('HUD', 'YROTATE', 'ERR)

CONNECT INPUTS TO ACCUMULATOR
CALL PFN('ACC1', 'ACCUMULATE', 'ERR)

CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM.
CALL PCONN('ACC1', 'HUD', 'ERR)
CALL PCONN('HUD', 'XYPTH', 'ROTY', 'ERR)

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

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

DETACH GRAPHICS DEVICE
CALL PDTACH('ERR)

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

C 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
SUBROUTINE QDOT(Q,WX,WY,WZ,DELTAT)

C FIGURES QUATERNION RATE (QD) AND LINEARLY INTEGRATES FOR TIME STEP DELTAT.

C REAL Q(4),QD(4),WX,WY,WZ,DELTAT,CONST
INTEGER J

QD(1) = (Q(2)*WZ-Q(3)*WY+Q(4)*WX)*.5
QD(2) = (-Q(1)*WZ+Q(3)*WX+Q(4)*WY)*.5
QD(3) = (Q(1)*WY-Q(2)*WX+Q(4)*WZ)*.5
QD(4) = (-Q(1)*WX-Q(2)*WY-Q(3)*WZ)*.5

DO 100 J=1,4
   Q(J) = Q(J) + DELTAT*QD(J)
100 CONTINUE

C NORMALIZE QUATERNIONS
CONST = SQRT(Q(1)^2+Q(2)^2+Q(3)^2+Q(4)^2)
Q(1) = Q(1)/CONST
Q(2) = Q(2)/CONST
Q(3) = Q(3)/CONST
Q(4) = Q(4)/CONST

C END
SUBROUTINE ROTATE(ANG,DELTAT,WB,Q,T)
C LINEARLY INTEGRATES TO FIND ROTATION IN BODY FRAME, THEN
TRANSFORMS TO REFERENCE FRAME.
REAL ANG(3),WB(3),DELTAT,Q(4),T(3,3)
C COMPUTE BODY RATES (WB)
(ANG IS ANGULAR ACCELERATION)
WB(1)=WB(1)+DELTAT*ANG(1)
WB(2)=WB(2)+DELTAT*ANG(2)
WB(3)=WB(3)+DELTAT*ANG(3)
C FIND QUATERNION RATE AND INTEGRATE
CALL QDOT(Q,WB(1),WB(2),WB(3),DELTAT)
C COMPUTE ROTATION MATRIX
CALL TRNSFR(T,Q)
END
SUBROUTINE RTOB(T, REF, BOD)
C
TRANSFORMS FROM REF TO BOD FRAME GIVEN TRANSFORMATION MATRIX T.
C
REAL T(3,3), REF(3), BOD(3)
C
BOD(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)
BOD(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(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))
CALL MODE(S(9), S(12), S(15), L(9), L(12), L(15))
SUBROUTINE THRUST(JETSEL,A,ANG,GAS,PICK,TABLE)
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 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 COMPUTES TRANSFORMATION MATRIX (T) FROM QUATERNIONS (Q)
C 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
SUBROUTINE UNITIZE(V)

UNITIZES V VECTOR

REAL V(3), MAGV

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

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
C INCLUDE 'C(ADFANO)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='C(ADFANO.DATA)'/NAME(:LENGTH)///.DAT'
OPEN ( UNIT=1, NAME=FILENAME, TYPE='OLD', READONLY)
READ ( 1, 910) VL, IPOS

910 FORMAT ( A1, I10 )
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, 3F12.3)
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=SQR(T(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