Dynetics, Inc.

CONTRACT NAS8-37690

JANUARY 1987

(NASA-CR-144193) ATTITUDE PROFILE DESIGN
prelative final report (Dynetics) p4 1
CSCL 72A

PREPARATION
NATIONAL AEROSPACE ADMINISTRATION
MARSHALL SPACE
FINAL REPORT

ATTITUDE PROFILE DESIGN PROGRAM

CONTRACT NAS8-37850

JANUARY 1991

PREPARED FOR:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MARSHALL SPACE FLIGHT CENTER, AL 35812
# TABLE OF CONTENTS

1. INTRODUCTION ................................................................. 1-1
2. COORDINATE SYSTEMS ...................................................... 2-1
3. INPUT FORMAT ................................................................. 3-1
4. OUTPUT FORMAT ............................................................... 4-1
5. EXPLANATION OF PROGRAM ............................................... 5-1
   5.1 GENERAL METHODOLOGY ............................................ 5-1
   5.2 SUBROUTINE EXPLANATIONS ....................................... 5-2
6. SAMPLE INPUT FILE .......................................................... 6-1
APPENDIX A. FORTRAN LISTING OF APD PROGRAM ............... A-1
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>APD</td>
<td>Attitude Profile Design</td>
</tr>
<tr>
<td>ECI</td>
<td>earth-centered inertial</td>
</tr>
<tr>
<td>ECID</td>
<td>earth-centered inertial of date</td>
</tr>
<tr>
<td>SCOOT</td>
<td>Simplex Computation of Optimum Orbital Trajectories</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The Attitude Profile Design (APD) Program is designed to be used as a stand-alone addition to the Simplex Computation of Optimum Orbital Trajectories (SCOOT). The program uses information from a SCOOT output file and the user-defined attitude profile to produce time histories of attitude, angular body rates, and accelerations.

The APD program is written in standard FORTRAN 77 and should be portable to any machine that has an appropriate compiler. The input and output are through formatted files. The program reads the basic flight data, such as the states of the vehicles, acceleration profiles, and burn information, from the SCOOT output file. The user inputs information about the desired attitude profile during coasts in a high-level manner. The program then takes these high level commands and executes the maneuvers, outputting the desired information.
2. COORDINATE SYSTEMS

There are four coordinate systems that the user may utilize in specifying his attitude pointing commands. They are:

1. Earth-centered inertial coordinate system,
2. Solar coordinate system,
3. Local north-east-down coordinate system, and
4. Stellar coordinate system.

The earth-centered inertial (ECI) system is the same as the earth-centered inertial of date (ECID) system used in SCOOT. The +x-axis points along the vernal equinox. The +z-axis points north along the earth's spin axis. The y-axis completes the right-handed orthogonal system.

The solar coordinate system is a pure rotation of the ECI system at a particular time. The ECI system is rotated about its +z-axis, and then about its new +y-axis until the +x-axis lines up with the sun vector. The resulting right-handed orthogonal system is the solar coordinate system.

The local north-east-down system rotates with the vehicle. It's x- and y-axes are in the local horizontal plane and point north and east, respectively. The +z-axis points along the negative earth radius vector.

The stellar coordinate system is defined with the use of the two star vectors that the user inputs. The +x-axis lies along the vector to star#1. The +z-axis lies in the direction of the cross product of the +x-axis with the vector to star#2. The y-axis completes the right-handed orthogonal system.
3. **INPUT FORMAT**

The basic means of entering data for the user is through a formatted input file. An example of such a file is shown below:

NAME OF NAV FILE:
STAR #1 DECLINATION:
STAR #1 RIGHT ASCENSION:
STAR #2 DECLINATION:
STAR #2 RIGHT ASCENSION:
ROLL ATTITUDE DURING BURN:
BURN PRINT INTERVAL (SECONDS):
COAST PRINT INTERVAL (SECONDS):

 COAST #:
 MANEUVER #:
 NAME:
 COORDINATE SYSTEM (A,B,C,D):

POINTING ANGLES:
SLEW RATE, ACCELERATION:
ROLL RATE, ACCELERATION:

BODY RATE COMMANDS:
BODY ACCELERATION LIMITS:

TIME OF MANEUVER (MINUTES):

The first section above appears at the top of every file. The user enters the alpha-numeric name of the data file from SCOOT to be used. The next four entries pertain to star locations and should be entered in degrees immediately following the colon. Entering preceding blanks may cause unpredictable results and is discouraged throughout the input file. The next input is the roll attitude during burns. The angle entered is defined with respect to the projection of the sun vector onto the body yz plane. Zero is defined when the +y-axis is aligned with the projection. Positive rotation is clockwise. The next input is the desired printout interval in seconds for the burn and coast intervals.
The next section is the standard maneuver definition section. Each maneuver requires a section like this. The first entry is the coast number, and the second entry is the maneuver number. The program checks the user to make sure that he has numbered the coasts and maneuvers properly. The important thing here is to remember to enter them. The next entry is the name of the maneuver. This is purely for the user's benefit in keeping things straight. The next entry is the coordinate system. The user should enter the letter (A,B,C,or D) of the system in which he wishes to define the maneuver. The user should always enter something here.

If the maneuver is to be a pointing command (as opposed to a rate command) then the user should fill in the next subsection. A pointing command is defined with the use of three angles. The angles are defined in the following manner for all coordinate systems except the stellar system.

ANGLE #1 - the angle of rotation about the +z-axis

ANGLE #2 - the angle of rotation about the new +y-axis

ANGLE #3 - the roll angle with respect to the projection of the sun vector onto the body yz plane as measured to the body +y-axis (exactly as defined above for the roll attitude during burns)

For the stellar system, the pointing vector is always along the stellar +z-axis; therefore, the following definitions exist for a pointing command.

ANGLE #1 - the roll angle from the vector to star#1 to the body +y-axis. Positive rotation is toward star#2.

ANGLE #2 - the roll angle from the vector to star#2 to the body +y-axis. Positive rotation is away from star#1.

Only one of the first two angles may be entered at a time and the third angle is not used.

The user should enter three angles in degrees, separated by commas, unless the desired attitude is the beginning attitude of the upcoming burn. In this case, the user should enter a “B” in the space for the pointing angles. The program will then use the burn attitude as the target attitude.

If the command is a pointing command, then the user has the option of entering rates and accelerations that must be observed in achieving the desired attitude. The user should enter these in the appropriate spaces. If a rate or acceleration is left blank, then the program assumes that they are infinite.

If the user wants a rate-commanded maneuver, then the next section should be completed. The body-rate commands are to be entered in roll, pitch, and yaw order and separated by commas. An “X” may
be used instead of a number to indicate that no new command is to be entered for a particular axis. For example, a desired pitch rate of 2 deg/s would be achieved by the following entry:

**BODY RATE COMMANDS:** X, 2.0

The roll and yaw body rates would remain unchanged, but the pitch rate would go to 2 deg/s.

The user may also supply accelerations to be observed in achieving the desired rates. They are entered in the same manner as the rates.

The final entry in each maneuver definition is the time of maneuver, and it may contain three different types of entries:

1. A blank indicates that once the desired end condition of this maneuver has been achieved, then the vehicle should move on to the next maneuver,
2. A number indicates that once the desired end condition has been achieved, the vehicle should hold this condition for the indicated length of time,
3. A "+" indicates that the end condition is to be held for an undetermined length of time. The following maneuvers are included in this variable time calculation. The next maneuver reached that has a time entered here constrains the problem. For example:

**MAN #1 TIME OF MANEUVER:** +
**MAN #2 TIME OF MANEUVER:**
**MAN #3 TIME OF MANEUVER:** 10.0

This example says that from the beginning of maneuver #1 to the end of maneuver #3 should take 10 minutes. Since maneuver #1 has a "+" entered, the extra time is added in holding its end condition until maneuver #2 begins.

The other means of input to the APD program is through an input file from SCOOT. It contains several items:

1. Julian day of the start of the mission,
2. Number of legs in the mission,
3. Time history of vehicle position,
4. Time history of gravitational acceleration,
5. Time history of thrust acceleration,
6. Time history of burn/coast condition,
7. Time history of thrust vector direction.
4. **OUTPUT FORMAT**

There are six output files. They are:

1. **INACC.DAT** - mission time (s), inertial accelerations x,y,z (m/s/s),
2. **GRACC.DAT** - mission time (s), gravitational accelerations x,y,z (m/s/s),
3. **CONACC.DAT** - mission time (s), contact accelerations x,y,z (m/s/s),
4. **WBODY.DAT** - mission time (s), body angular rates p,q,r (rad/s),
5. **QUAT.DAT** - mission time (s), body attitude quaternions q₀,q₁,q₂,q₃
6. **APD.LOG** - a time history of events.

The first line of the first five output files above contains an integer indicating the number of data lines to follow.
5. **EXPLANATION OF PROGRAM**

5.1 **GENERAL METHODOLOGY**

During a burn interval, the user only has one degree of freedom through the use of the input file. The rest are defined through the navigation input file. The one input that the user can enter is roll attitude with respect to the sun during burns. Otherwise, the attitude of the vehicle at each timepoint of interest (namely, the output timepoints) is taken to be pointing along the thrust vector at all times. The body rates are calculated by taking numerical derivatives of the Euler angles at the desired timepoints to get Euler rates. These are then converted to body rates.

During a coast, the user may define either a pointing command or a body rate command—with one exception. The first maneuver of the first coast always defines the initial attitude of the vehicle and an error will occur if the user tries to do otherwise.

If the user enters a pointing command, the vehicle already has a given attitude and, possibly, body rotational rates. The first thing that is done is that any body rotational rates that are left over from the previous maneuver are nulled out. Any existing roll rate and slew rate are nulled simultaneously and independently according to the user-defined accelerations for rolling and slewing.

The next thing to be done is to roll the vehicle to the proper orientation with respect to the sun. This is done while obeying the user-defined roll rate and acceleration.

Next, the vehicle slews in a plane to the desired pointing vector, obeying the user-defined slew rate and acceleration. Simultaneously, the roll attitude is being changed so that at the time the slew maneuver is completed, the correct roll attitude is being reached also. If the total amount of roll required exceeds the physical limitations imposed by the user-defined roll rate and acceleration and the time limit imposed by the slew maneuver, then the vehicle rolls at its maximum, and the roll is completed as soon after the slew as possible. Otherwise, the roll rate is kept at the minimum rate required to achieve the above-stated condition.

Once the vehicle has achieved the desired attitude conditions, there may be a station-keeping requirement imposed by the user. If the user has used the north-east-down system to define the pointing command, then a station-keeping command requires that the inertial attitude continue to change to maintain the desired conditions. The new attitude is calculated for each timepoint and numerical differentiation is used to obtain body rates. If any of the other coordinate systems were used, then no movement is required because they are assumed inertial for the station-keeping length.

If the user has entered a rate command rather than a pointing command, then each axis is treated simultaneously and independently. The body rates are changed from the current rates to the desired rates.
while obeying the accelerations entered by the user. If a hold maneuver condition is called for, the desired body rates continue for that length of time and are integrated to determine position at necessary points.

When the vehicle is commanded to a particular attitude (either roll or pointing) the user-defined rates and accelerations are used and obeyed. If the user does not enter a rate, the vehicle assumes the desired attitude instantaneously. If a rate is entered without an acceleration, it is achieved instantaneously, maintained until the desired attitude is reached, and then nulled out instantaneously. If the user enters a rate and an acceleration, then the vehicle accelerates at the user-defined value until the rate is achieved. The rate is then maintained for the appropriate time and the vehicle decelerates until a rate of zero and the desired attitude are achieved at the same time. Sometimes the attitude change required is so small that the above scheme overshoots the desired attitude no matter how short the time at maximum rate. In this case, the vehicle accelerates to some sub-maximum rate and immediately begins decelerating to a rate of zero and the desired attitude at the same time.

5.2 SUBROUTINE EXPLANATIONS

Listed below are brief descriptions of each subroutine:

- **ANG**: converts any angle in radians to an angle between 0 and 2π
- **CONVERT**: converts a character string to a real number
- **GETPROJ**: determines the angle necessary to rotate the vehicle in order to align the body y-axis along the projection of the sun vector onto the body yz plane
- **GETSTATE**: determines (through interpolation) the state of the vehicle at a given time using the data read from the navigation input file
- **LVLH**: determines the coordinate transformation matrix and Euler angles for a given attitude in north-east-down coordinates
- **QUAT**: determines the four quaternions from a coordinate transformation matrix
- **QUATUP**: integrates body rates to obtain new quaternion values and the coordinate transformation matrix
- **POINTER**: determines the coordinate transformation matrix for an attitude defined in any system
- **RMAN**: reads a single maneuver from the user input file
- **ROLLER**: computes and executes a roll maneuver from an initial roll attitude to the desired roll attitude
- **ROTATE**: executes rotations about a body axis and computes the new coordinate transformation matrix
SLEWER - executes a slew maneuver with accompanying roll to a predetermined schedule
SUNV - determines the right ascension and declination of the sun at the desired time
OUTPUT - outputs the desired information to the appropriate files
6. **SAMPLE INPUT FILE**

The following is a maneuver-by-maneuver explanation of a sample input file that covers the capability of the Attitude Profile Design (APD) Program.

**COAST #1**

**MAN#1** - the initial maneuver always defines the initial conditions of the vehicle. The coordinate system is ECI, as indicated by the letter “A”. The initial conditions are a right ascension of 10.0° and a declination of 20.0°. The body y-axis makes an angle of 30.0° with the projection of the sun vector onto the body yz plane. This condition is held for 10.0 minutes before moving to the next maneuver.

**MAN#2** - the vehicle slews to the indicated pointing conditions in the solar coordinate system using slew and roll rates of 5.0 deg/s and 2.0 deg/s, respectively. Slew and roll accelerations are both 2.0 deg/s. This attitude is not held before moving to the next maneuver.

**MAN#3** - the vehicle achieves a clockwise roll of 3.0 deg/s using an acceleration of 2.0 deg/s. Once the desired rate is achieved, it is held for 25.0 minutes.

**MAN#4** - the vehicle achieves a counterclockwise roll of 3.0 deg/s and is held for 25.0 minutes.

**MAN#5** - the vehicle slews to the indicated attitude in north-east-down coordinates at the indicated rates and accelerations. The “+” in the time entry, along with the 10.0 minutes in the next maneuver time entry, indicates that the time elapsed from the beginning of maneuver#5 to the end of maneuver#6 is to be 10.0 minutes.

**MAN#6** - the vehicle slews to the desired solar orientation

**MAN#7 thru MAN#10** - repeats MAN#3 thru MAN#6

**MAN#11 thru MAN#14** - repeats MAN#3 thru MAN#6

**MAN#15 thru MAN#18** - repeats MAN#3 thru MAN#6

**MAN#19 thru MAN#22** - repeats MAN#3 thru MAN#6

**MAN#23** - vehicle slews to stellar pointing vector. The body y-axis makes an angle of 10.0° with the vector to star#1

**MAN#24** - vehicle rolls so that body y-axis makes an angle of -10.0° with the vector to star#1

**MAN#25** - vehicle rolls so that body y-axis makes an angle of 10.0° with the vector to star#2

**MAN#26** - vehicle rolls so that body y-axis makes an angle of -10.0° with the vector to star#2
MAN#27 - vehicle slews to the attitude for burn #1, as indicated by the "B" in the pointing angle entry.

COAST#2

MAN#1 - vehicle slews to the attitude for burn #2

COAST#3

MAN#1 - vehicle slews to the attitude for burn #3

The sample data input file associated with the previous explanation is presented on the following pages.
NAME OF NAV FILE: BRET NAV.DAT

STAR #1 DECLINATION: 15.2
STAR #1 RIGHT ASCENSION: 20.3
STAR #2 DECLINATION: 16.3
STAR #2 RIGHT ASCENSION: 22.4

ROLL ATTITUDE DURING BURN: 0.0

BURN PRINT INTERVAL (SECONDS): 1.
COAST PRINT INTERVAL (SECONDS): 20.

ROLL ATTITUDE DURING BURN: 0.0

COAST #: 1
MANEUVER #: 1
NAME: INERTIAL POINTING
COORDINATE SYSTEM (A,B,C,D): A

POINTING ANGLES: 10.0, 20.0, 30.0
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): 10.0

-----------------------------------------------

COAST #: 1
MANEUVER #: 2
NAME: SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES: 0.0, 40.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): 

-----------------------------------------------

COAST #: 1
MANEUVER #: 3
NAME: ROLL CLOCKWISE
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: 3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

-----------------------------------------------

COAST #: 1
MANEUVER #: 4
NAME: COUNTERCLOCKWISE ROLL
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: -3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

-----------------------------------------------

COAST #: 1
MANEUVER #: 5
NAME: SLEW TO NEGATIVE EARTH RADIUS VECTOR
COORDINATE SYSTEM (A, B, C, D) : C

POINTING ANGLES: 0.0, 90.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES):

COAST #: 1
MANEUVER #: 6
NAME: SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES: 0.0, 40.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): 10.0

COAST #: 1
MANEUVER #: 7
NAME: ROLL CLOCKWISE
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: 3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

COAST #: 1
MANEUVER #: 8
NAME: COUNTERCLOCKWISE ROLL
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: -3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

COAST #: 1
MANEUVER #: 9
NAME: SLEW TO NEGATIVE EARTH RADIUS VECTOR
COORDINATE SYSTEM (A, B, C, D) : C

POINTING ANGLES: 0.0, 90.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:
TIME OF MANEUVER (MINUTES) : 10.0

COAST # : 1
MANEUVER # : 10
NAME : SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES : 0.0, 40.0, 0.0
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) : 25.0

COAST # : 1
MANEUVER # : 11
NAME : ROLL CLOCKWISE
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES :
SLEW RATE, ACCEL :
ROLL RATE, ACCEL :

BODY RATE COMMANDS : 3.0
BODY ACCEL LIMITS : 2.0

TIME OF MANEUVER (MINUTES) : 25.0

COAST # : 1
MANEUVER # : 12
NAME : COUNTERCLOCKWISE ROLL
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES :
SLEW RATE, ACCEL :
ROLL RATE, ACCEL :

BODY RATE COMMANDS : -3.0
BODY ACCEL LIMITS : 2.0

TIME OF MANEUVER (MINUTES) : 25.0

COAST # : 1
MANEUVER # : 13
NAME : SLEW TO NEGATIVE EARTH RADIUS VECTOR
COORDINATE SYSTEM (A, B, C, D) : C

POINTING ANGLES : 0.0, 90.0, 0.0
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) : +

COAST # : 1
MANEUVER # : 14
NAME : SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A, B, C, D) : B

POINTING ANGLES : 0.0, 40.0, 0.0
SLEW RATE, ACCEL : 5.0, 2.0

6-5
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:

BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): 10.0

MANEUVER #: 1
NAME: ROLL CLOCKWISE
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: 3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

COAST #: 1
MANEUVER #: 15
NAME: COUNTERCLOCKWISE ROLL
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES:
SLEW RATE, ACCEL:
ROLL RATE, ACCEL:

BODY RATE COMMANDS: -3.0
BODY ACCEL LIMITS: 2.0

TIME OF MANEUVER (MINUTES): 25.0

COAST #: 1
MANEUVER #: 16
NAME: SLEW TO NEGATIVE EARTH RADIUS VECTOR
COORDINATE SYSTEM (A,B,C,D): C

POINTING ANGLES: 0.0, 90.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): +

COAST #: 1
MANEUVER #: 17
NAME: SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A,B,C,D): B

POINTING ANGLES: 0.0, 40.0, 0.0
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES): 10.0

COAST #: 1
MANEUVER #: 18
NAME: ROLL CLOCKWISE
COORDINATE SYSTEM (A,B,C,D) : B

POINTING ANGLES :
SLEW RATE, ACCEL :
ROLL RATE, ACCEL :

BODY RATE COMMANDS : 3.0
BODY ACCEL LIMITS : 2.0

TIME OF MANEUVER (MINUTES) : 25.0

COAST # : 1
MANEUVER #: 20
NAME : COUNTERCLOCKWISE ROLL
COORDINATE SYSTEM (A,B,C,D) : B

POINTING ANGLES :
SLEW RATE, ACCEL :
ROLL RATE, ACCEL :

BODY RATE COMMANDS : -3.0
BODY ACCEL LIMITS : 2.0

TIME OF MANEUVER (MINUTES) : 25.0

COAST # : 1
MANEUVER #: 21
NAME : SLEW TO NEGATIVE EARTH RADIUS VECTOR
COORDINATE SYSTEM (A,B,C,D) : C

POINTING ANGLES : 0.0, 90.0, 0.0
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

COAST # : 1
MANEUVER #: 22
NAME : SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A,B,C,D) : B

POINTING ANGLES : 0.0, 40.0, 0.0
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) : 10.0

COAST # : 1
MANEUVER #: 23
NAME : SLEW TO SOLAR ORIENTATION
COORDINATE SYSTEM (A,B,C,D) : D

POINTING ANGLES : 10.0, x, 0.0
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

6-7
COAST #: 1
MANEUVER #: 24
NAME: ROLL THROUGH STAR 1
COORDINATE SYSTEM (A,B,C,D) : D

POINTING ANGLES: X, 10.0
SLEW RATE, ACCEL : 
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

COAST #: 1
MANEUVER #: 25
NAME: ROLL THROUGH STARS 1 AND 2
COORDINATE SYSTEM (A,B,C,D) : D

POINTING ANGLES: X, 10.0
SLEW RATE, ACCEL : 
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

COAST #: 1
MANEUVER #: 26
NAME: ROLL THROUGH STAR 2
COORDINATE SYSTEM (A,B,C,D) : D

POINTING ANGLES: X, -10.0
SLEW RATE, ACCEL :
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

COAST #: 1
MANEUVER #: 27
NAME: SLEW TO BURN ATTITUDE
COORDINATE SYSTEM (A,B,C,D) : A

POINTING ANGLES: B
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0

BODY RATE COMMANDS :
BODY ACCEL LIMITS :

TIME OF MANEUVER (MINUTES) :

COAST #: 2
MANEUVER #: 1
NAME: SLEW TO BURN ATTITUDE
COORDINATE SYSTEM (A,B,C,D) : A

POINTING ANGLES: B
SLEW RATE, ACCEL : 5.0, 2.0
ROLL RATE, ACCEL : 3.0, 2.0
BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES):

COAST #: 3
MANEUVER #: 1
NAME: SLEW TO BURN ATTITUDE
COORDINATE SYSTEM (A,B,C,D): A

POINTING ANGLES: B
SLEW RATE, ACCEL: 5.0, 2.0
ROLL RATE, ACCEL: 3.0, 2.0

BODY RATE COMMANDS:
BODY ACCEL LIMITS:

TIME OF MANEUVER (MINUTES):
APPENDIX A. FORTRAN LISTING OF APD PROGRAM
FUNCTION ANG(X)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/COM3/PI,TWOPi,PIO2
ANG=X-TWOPi*FLOAT(INT(X/TWOPi))
IF(ANG) 1,2,2
1 ANG=ANG+TWOPi
2 RETURN
END
PROGRAM ATTITUDE
IMPLICIT REAL*8 (A-H,O-Z)

THIS PROGRAM REQUIRES INPUT FROM A SAMBO OUTPUT RUN AND INPUT FROM
A FORMATTED INPUT FILE THAT DEFINES PARAMETERS FOR THE ATTITUDE
PROFILE HISTORY

COMMON/COM1/DTIME, DX, DGACC, DMACC, IBRN,
1 OAB, DDEC
COMMON/COM2/PINT, TIME0
COMMON/COM3/P, TWOP, P102
COMMON/COM4/RBURN
COMMON/COM5/TRA, TR1, TR2, WPMAX
COMMON/COM7/SANG1, SANG2, STARROT1, STARROT2, SV1, SV2
COMMON/COM8/STARD, STADDR, A(3,3), B(3,3), R(3,3), ISYSZ(5),
1 STIMEZ(5), IPOINTZ(5,3), RATEZ(5,3), ACCELZ(5,3), ISLEWZ(5,2),
2 ROLLZ(5,2), DTIME(5000), DRA(20), DECB(20), TSTAY(50),
3 BBRE(3), BBRE(3), RBRATE(3), IACC(3), TACC(3), SACC(3), X(3),
1 DIM(3), DUN(3), TIMEB(20), TIMEC(20), D(3,3), AV(3,3),
2 SV1(3), SV2(3), SCANV(3), SV1M(3), SV2M(3)
DATA PI /3.14159265/
DATA TWOPI /6.28318531/
DATA PIO2 /1.57079633/
DATA TOL /0.1/
DATA TDIFF /0.01/
DATA TINT /0.1/
DATA TTOL /0.1/
WRITE (6, *) 'WELCOME TO THE ATTITUDE PROFILE HISTORY PROGRAM'
WRITE (6, *) 'PLEASE ENTER THE NAME OF THE INPUT DATA FILE'
READ (5, 99) NAME
FORMAT (A20)
OPEN INPUT FILE
OPEN (UNIT=13, FILE='AD.DLOG', STATUS='NEW')
OPEN (UNIT=20, FILE=NAME, STATUS='OLD')
OPEN (UNIT=31, FILE='INACC.DAT', STATUS='NEW')
OPEN (UNIT=32, FILE='GRACC.DAT', STATUS='NEW')
OPEN (UNIT=33, FILE='CONACC.DAT', STATUS='NEW')
OPEN (UNIT=34, FILE='WBODY.DAT', STATUS='NEW')
OPEN (UNIT=35, FILE='QUAT.DAT', STATUS='NEW')
READ SAMBO FILE NAME
READ (20, 100) SFILE
FORMAT (18X, A15)
OPEN (UNIT=21, FILE='SFILZ', STATUS='OLD')
READ JULIAN DAY
READ (20, 101) IYY, IMM, IDD, IHR, IMM, SS
FORMAT (12X, 5(12, LX), F6.3)
CHECK FOR MISTAKES
IF (IMM .LT. 0 .OR. IMM .GT. 12) THEN
WRITE (6,*) 'YOU HAVE ENTERED AN INVALID MONTH'
STOP
ELSE IF (IDD .LT. 0 .OR. IDD .GT. 31) THEN
WRITE (6,*) 'YOU HAVE ENTERED AN INVALID DAY'
STOP
ELSE IF (IDD .EQ. 31 .AND. (IMM .EQ. 2 .OR. IMM .EQ. 4 .OR. IMM
1 .EQ. 9 .OR. IMM .EQ. 11)) THEN
WRITE (6,*) 'THE MONTH YOU HAVE ENTERED DOES NOT HAVE 31 DAYS'
STOP
ELSE IF (IMM .EQ. 2 .AND. (IDD .EQ. 30 .OR. IDD .EQ. 29 .AND.
1 MOD(IY,4) .NE. 0)) THEN
WRITE (6,*) 'FEBRUARY DOES NOT HAVE THIS MANY DAYS THIS YEAR'
STOP
ELSE IF (IMM .LT. 0 .OR. IMM .GT. 59) THEN
WRITE (6,*) 'YOU HAVE ENTERED AN INVALID MINUTE'
STOP
ELSE IF (SS .LT. 0 .OR. SS .GE. 60.0) THEN
WRITE (6,*)'YOU HAVE ENTERED AN INVALID SECOND'
STOP
END IF
GET STAR COORDINATES
DO I = 1,2
READ (20,102) STARDEC(I)
STARDEC(I) = STARDEC(I) * PI / 180.0
END DO
READ (20,103) STARRA(I)
STARRA(I) = STARRA(I)*PI/180.0
END DO
SV1(1) = COS(STARDEC(1)) * COS(STARRA(1))
SV1(2) = COS(STARDEC(1)) * SIN(STARRA(1))
SV1(3) = SIN(STARDEC(1))
SV2(1) = COS(STARDEC(2)) * COS(STARRA(2))
SV2(2) = COS(STARDEC(2)) * SIN(STARRA(2))
SV2(3) = SIN(STARDEC(2))
SCANV(1) = SV1(2) * SV2(3) - SV1(3) * SV2(2)
SCANV(2) = SV1(3) * SV2(1) - SV1(1) * SV2(3)
SCANV(3) = SV1(1) * SV2(2) - SV1(2) * SV1(1)
XMAG = (SCANV(1)**2 + SCANV(2)**2 + SCANV(3)**2)**0.5
SCANV(1) = SCANV(1) / XMAG
SCANV(2) = SCANV(2) / XMAG
SCANV(3) = SCANV(3) / XMAG
SANG1 = ATAN2(SCANV(2),SCANV(1))
SANG2 = ASIN(SCANV(3))
DO I = 1,3
DO J = 1,3
IF (I .EQ. J) THEN
A(I,J) = 1.0
ELSE
A(I,J) = 0.0
END IF
END DO
END DO
CALL ROTATE (A,SANG1,-SANG2,0.0,3,2,0)
DO I = 1,3
SVIM(I) = A(I,1) * SV1(1) + A(I,2) * SV1(2) + A(I,3) * SV1(3)
SV2M(I) = A(I,1) * SV2(1) + A(I,2) * SV2(2) + A(I,3) * SV2(3)
END DO
STARROT1 = ATAN2(SV1M(3),SV1M(2))
STARROT2 = ATAN2 (SV2M(3), SV2M(2))
READ (20, 194) RBRN
RBRN = RBRN * PI/180.0
FORMAT (28X, F20.10)
READ (20, 195) PINTB
FORMAT (31X, F20.10)
READ (20, 196) PINTC
FORMAT (32X, F20.10)
C READ SAMBO STUFF
C
READ (21, *) TIME0, NLEGS
IBRNL = 0
ILEG = 1
DO I = 1, 5000
READ (21, *, END=200) DTIME(I), DX(I, 1), DX(I, 2), DX(I, 3),
1 DGACC(I, 1), DGACC(I, 2), DGACC(I, 3),
2 DMACC(I, 1), DMACC(I, 2), DMACC(I, 3),
3 IBRN(I), DRA(I), DDEC(I)
IF (IBRNL .EQ. 0 .AND. IBRN(I) .EQ. I) THEN
TIMZ0 = TMIIE(I)
RAB(ILEG) = DRA(I)
DECB(ILEG) = DDEC(I)
ELSE IF (IBRNL .EQ. 1 .AND. IBRN(I) .EQ. 0) THEN
TIMZ0 = DTIE(I-1)
ILEG = ILEG + 1
END IF
IBRN = IBRN(I)
END DO
200 TIMEC(ILEG) = DTIE(I-1)
NPNT = 1
DO I = 1, ILEG
IF (I .EQ. I) THEN
NPNT = NPNT + INT(TIMEB(I)/PINTC) +
1 INT((TIMEB(I)-TIMEB(I-1))/PINTC) +
2 ELSE
NPNT = NPNT + INT(((TIMEB(I)-TIMEC(I-1))/PINTB) +
1 INT(((TIMEC(I) - TIMEB(I))/PINTB) +
2 END IF
END DO
WRITE (31, *) NPNT
WRITE (32, *) NPNT
WRITE (33, *) NPNT
WRITE (34, *) NPNT
WRITE (35, *) NPNT
A(1, 1) = 1.0
A(2, 2) = 1.0
A(3, 3) = 1.0
Q0 = 1.0
IREAD = 0
C INITIALIZE COAST AND MANEUVER COUNTERS
C
ICOAST = 1
IMAN = 1
1000 PINT = PINTC
CALL QUAT (A, Q0, Q1, Q2, Q3)
SRA = RAB(ICOAST)
SDEC = DDEC(ICOAST)
999 IF (TIME .GE. TIMEB(ICOAST)) THEN
IF (PINT .EQ. PINTC) THEN
PINT = PINTB
TIME = TIMEB(ICOAST)
PTIME = TIME
CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
PSIM = RA
THTM = -DEC
DO I = 1, 3
   DO J = 1, 3
      IF (I .EQ. J) THEN
         A(I,J) = 1.0
         B(I,J) = 1.0
         C(I,J) = 1.0
      ELSE
         A(I,J) = 0.0
         B(I,J) = 0.0
         C(I,J) = 0.0
      END IF
   END DO
END DO
CALL ROTATE (A, RA, -DEC, 0.0, 3, 2, 0)
CALL GETPROJ (TIME, A, ROT)
CALL ROTATE (A, 0.0, 0.0, ROT+RBURN, 0, 0, 1)
PHIM = ROT+RBURN
CALL GETSTATE (TIME+TDIFF, X, DUM1, DUM2, RA, DEC)
CALL ROTATE (B, RA, -DEC, 0.0, 3, 2, 0)
CALL GETPROJ (TIME+TDIFF, B, ROT)
PSIP = RA
THTP = -DEC
PHIP = ROT+RBURN
PSI = (PSIP+PSIM) * 0.5
THT = (THTP+THTM) * 0.5
PHI = (PHIP-PHIM) * 0.5
PSID = (PSIP-PSIM) / TDIFF
THTD = (THTP-THTM) / TDIFF
PHID = (PHIP-PHIM) / TDIFF
WP = PHID - PSID * SIN(THT)
WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)
CALL OUTPUT (0, TIME, A, WP, WQ, WR, PTIME)
GOTO 999
END IF
IF (PTIME .GT. TIMEC(ICOAST)) THEN
   PINT = PINTC
   TIME = TIMEC(ICOAST)
   CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
   PSIM = RA
   THTM = -DEC
   DO I = 1, 3
      DO J = 1, 3
         IF (I .EQ. J) THEN
            A(I,J) = 1.0
            B(I,J) = 1.0
         ELSE
            A(I,J) = 0.0
            B(I,J) = 0.0
         END IF
      END DO
   END DO
   CALL ROTATE (B, RA, -DEC, 0.0, 3, 2, 0)
   CALL GETPROJ (TIME, B, ROT)
   PHIM = ROT + RBURN
   CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
   PSIP = RA
   THTP = -DEC
   CALL ROTATE (A, RA, -DEC, 0.0, 3, 2, 0)
   CALL GETPROJ (TIME, A, ROT)
   CALL ROTATE (A, 0.0, 0.0, ROT+RBURN, 0, 0, 1)
   PHIP = ROT+RBURN
   PSI = (PSIP+PSIM) * 0.5
ENDIF
THT = (THTP + THTM) * 0.5
PHI = (PHIP + PHIM) * 0.5
PSID = (PSIP - PSIM) / TDIFF
THTD = (THTP - THTM) / TDIFF
PHID = (PHIP - PHIM) / TDIFF
WP = PHID - PSID * SIN(THT)
WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)

CALL OUTPUT (0, TIME, A, WP, WQ, WR, PTIME)

ELSE
TIME = PTIME + TDIFF * 0.5
CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
PSIM = RA
THTM = -DEC
DO I = 1, 3
DO J = 1, 3
IF (I .EQ. J) THEN
A(I, J) = 1.0
B(I, J) = 1.0
C(I, J) = 1.0
ELSE
A(I, J) = 0.0
B(I, J) = 0.0
C(I, J) = 0.0
END IF
END DO
END DO
CALL ROTATE (B, RA, DEC, 0.0, 3, 2, 0)
CALL GETPROJ (TIME, B, ROT)
PHIM = ROT + RBURN
TIME = PTIME
CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
PSI = RA
THT = -DEC
CALL ROTATE (A, RA, DEC, 0.0, 3, 2, 0)
CALL GETPROJ (TIME, A, ROT)
CALL ROTATE (A, 0.0, 0.0, ROT + RBURN, 0.0, 1)
PHI = ROT + RBURN
CALL GETSTATE (TIME + TDIFF * 0.5, X, DUM1, DUM2, RA, DEC)
PSIP = RA
THTP = -DEC
CALL ROTATE (C, RA, DEC, 0.0, 3, 2, 0)
CALL GETPROJ (TIME + TDIFF, C, ROT)
PHIP = ROT + RBURN
PSID = (PSIP - PSIM) / TDIFF
THTD = (THTP - THTM) / TDIFF
PHID = (PHIP - PHIM) / TDIFF
WP = PHID - PSID * SIN(THT)
WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)
CALL OUTPUT (0, TIME, A, WP, WQ, WR, PTIME)
GOTO 999
ENDIF
ENDIF

C C READ A MANEUVER
C IF (IREAD .LE. 1) THEN
CALL RMAN (JCOAST, JMAN, NAME, SYS, POINT, IPOINT, SLEW, ISLEW, ROLL, 1 IROLL, RATE, IRATE, ACCEL, IACCEL, STIME, ITIME)

C C CHECK COAST AND MANEUVERS
C IF (ICOAST .EQ. 1 .AND. JMAN .EQ. 1 .AND. (JCOAST .NE. 1 .OR. JMAN

A-7
1 IF (.NE. 1)) THEN
   WRITE (6,*) 'THE FIRST MANEUVER MUST BE COAST #1 AND MANEUVER #
11' STOP
ELSE IF (ICOAST .EQ. JCOAST) THEN
   IF (IMAN .NE. JMAN) THEN
      WRITE (6,*) 'YOU HAVE MISNUMBERED YOUR MANEUVERS', JCOAST,
1   STOP
   END IF
ELSE IF (JCOAST .NE. ICOAST + 1 .AND. JMAN .EQ. 1) THEN
   WRITE (6,*) 'YOU HAVE MISNUMBERED YOUR COASTS OR NOT RESTART
1ED YOUR MANEUVER NUMBERS', JCOAST, JMAN
   STOP
END IF
IF (ICOAST .GT. NLEGS) THEN
   WRITE (6,*), 'YOU HAVE EXCEEDED THE NUMBER OF COASTS IN THIS MIS
1SION'
   STOP
END IF
IF (ITIME .EQ. 2 .OR. IREAD .EQ. 1) THEN
   IF (IREAD .EQ. 0) THEN
      IMANS = IMAN-1
      IND = 1
      DO I = 1,3
         DO J = 1,3
            ASV(I,J) = A(I,J)
         END DO
      END DO
      WP SV = WP
      WQ SV = WQ
      WRSV = WR
      TIMESV = TIME
      PTIMESV = PTIME
   END IF
   ITIMEF = 1
   NAMEZ(IND) = NAME
   ISYSZ(IND) = ISYS
   STIMEZ(IND) = STIME
   ITIMEZ(IND) = ITIME
   DO I = 1,3
      POINTZ(IND, I) = POINT(I)
      IPOINTZ(IND, I) = IPOINT(I)
      RATEZ(IND, I) = RATE(I)
      IRATEZ(IND, I) = IRATE(I)
      ACCELE(IND, I) = ACCELE(I)
      IACCELE(IND, I) = IACCELE(I)
      IF (I .NE. 3) THEN
         SLEWZ(IND, I) = SLEW(I)
         ISLEWZ(IND, I) = ISLEW(I)
         ROLZ(IND, I) = ROLL(I)
         IROLZ(IND, I) = IROLL(I)
      END IF
   END DO
   IND = IND + 1
   IREAD = 1
   IF (ITIME .EQ. 1) THEN
      IREAD = 2
   END IF
ELSE IF (IREAD .EQ. 2) THEN

A-8
IF (IMAN-IMANS .EQ. 1) THEN
  DO I = 1, 3
    DO J = 1, 3
      A(I,J) = ASV(I,J)
    END DO
  END DO
  WP = WPSV
  WQ = WQSV
  WR = WRSV
  TIME = TIM_SV
  PTIME = PTIM_SV
END IF

NAME = NAMEZ(IMAN-IMANS)
ISYS = ISYSZ(IMAN-IMANS)
STIME = STIMEZ(IMAN-IMANS)
ITIME = ITIMEZ(IMAN-IMANS)
DO I = 1, 3
  POINT(I) = POINTZ(IMAN-IMANS, I)
  IPOINT(I) = IPOINTZ(IMAN-IMANS, I)
  RATE(I) = RATEZ(IMAN-IMANS, I)
  ACCEL(I) = ACCELZ(IMAN-IMANS, I)
  IACCEL(I) = IACCELZ(IMAN-IMANS, I)
  IF (I .NE. 3) THEN
    SLEW(I) = SLEWZ(IMAN-IMANS, I)
    ISLEW(I) = ISLEWZ(IMAN-IMANS, I)
    ROLL(I) = ROLLZ(IMAN-IMANS, I)
    IROLL(I) = IROLLZ(IMAN-IMANS, I)
  END IF
END DO

IF (ITIME .EQ. 0 .AND. ABS(TINC) .LT. TTOL .AND. IMAN - IMANS .EQ. 1) THEN
  IREAD = 0
  ITIME = 0
END IF

IF (IMAN .EQ. 1) THEN
  WRITE (13,*) '--BEGINNING COAST ', ICOAST
  WRITE (6,*) '--BEGINNING COAST ', ICOAST
END IF

WRITE (13, *) '**BEGINNING MANEUVER ', IMAN
WRITE (6,*) '**BEGINNING MANEUVER ', IMAN

C CHECK 1ST MANEUVER

IF (ICOAST .EQ. 1 .AND. IMAN .EQ. 1) THEN
  IF (IPOINT(1) .EQ. 2) THEN
    CALL ROTATE (A, SRA, -SDEC, 0.0, 3, 2, 0)
    CALL GETPROJ (TIME, A, ROT)
    CALL ROTATE (A, 0.0, 0.0, ROT+RBURN, 0, 0, 1)
  ELSE IF (IPOINT(1) .EQ. 1 .AND. IPOINT(2) .EQ. 1 .AND. IPOINT(3) .EQ. 1) THEN
    CALL POINTER (TIME, POINT, IPOINT, ISYS, A)
  ELSE IF (ISYS .EQ. 4 .AND. IPOINT(1) .EQ. 1 .OR. IPOINT(2) .EQ. 1) THEN
    CALL POINTER (TIME, POINT, IPOINT, ISYS, A)
  ELSE
    WRITE (6,*) 'YOU HAVE ENTERED AN INCORRECT OR INCOMPLETE SET OF INITIAL CONDITIONS'
  END IF
  CALL QUAT (A, Q0, Q1, Q2, Q3)
END IF

IF (ICOAST .EQ. 1 .AND. IMAN .EQ. 1) THEN
  PSI = ATAN2 (A(1,2), A(1,1))
  THT = ASIN (-A(1,3))
  PHI = ATAN2 (A(2,3), A(3,3))
END IF
CALL POINTER (TIME+TDIFF, POINT, IPOINT, ISYS, B)
PSIP = ATAN2 (B(1,2), B(1,1))
THTP = ASIN (-B(1,1))
PHIP = ATAN2 (B(2,3), B(3,3))
PSID = (PSIP-PSI) / TDIFF
THTP = (THTP-THT) / TDIFF
PHID = (PHIP-PHI) / TDIFF
WP = PHID - PSID * SIN(THT)
WQ = THT * COS(PHI) + PSID * COS(THT) * SIN(PHI)
WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)
ELSE
WP = 0.0
WQ = 0.0
WR = 0.0
END IF
CALL OUTPUT (ITIMEF, TIME, WP, WQ, WR, PTIME)
END IF

C CHECK THE STATION KEEPING TIME FOR THIS MANEUVER

IF (ITIME .EQ. 2) THEN
XTIME = TIME
MAN = IMAN
STIME = TSTAY(IMAN)
IF (STIME .EQ. 0.0) IPRNT = 0
END IF

C THIS SECTION OF CODE HANDLES THE CASE WHERE A POINTING COMMAND HAS
C BEEN ISSUED AND NOW THE VEHICLE MUST STOP ANY CURRENT MOTION THAT IT
C HAS AND THEN SLEW TO THE COMMANDED ATTITUDE

IF (IPOINT(1) .NE. 0 .OR. IPOINT(2) .NE. 0 .OR. IPOINT(3) .NE. 0) THEN

C THE FIRST PART HERE IS THE STOPPING OF ANY ROLL AND/OR SLEW RATE
C THAT MAY BE PRESENT FROM THE PREVIOUS MANEUVER

WRITE (13,*) 'STOPPING MOTION TIME,WP,WQ,WR =', TIME, WP, WQ, WR
CALL QUAT (A, Q0, Q1, Q2, Q3)
IEVENT = 0
IF (IROLL(2) .EQ. 0) THEN
WP = 0.0
TTSR = 0.0
ELSE
IF (WP .LT. 0.0) THEN
SROLL = -ROLL(1)
SACCEL = -ROLL(2)
ELSE
SROLL = ROLL(1)
SACCEL = ROLL(2)
END IF
TTSR = WP / SACCEL
END IF
IF (ISLEW(2) .EQ. 0) THEN
WQ = 0.0
WR = 0.0
TTSS = 0.0
ELSE
SLEWR = (WQ**2 + WR**2) ** 0.5
IF (WR .NE. 0.0 .OR. WQ .NE. 0.0) THEN
SLEWA = ATAN2 (WR, WQ)
END IF
TTSS = SLEWR / SLEW(2)
END IF
IF (TTSS .EQ. 0.0 .AND. TTSR .EQ. 0.0) THEN

A-10
IEVENT = 3
ELSE IF (IEVENT .LE. 1) THEN
  IEVENT = 1
  IF (TTSR .GT. TTSS) THEN
    TEVENT = TTSS
    IHIGH = 1
  ELSE
    TEVENT = TTSR
    IHIGH = 2
  END IF
END IF
IF (IEVENT .NE. 3) THEN
  IF (TINT .LT. TEVENT) THEN
    DELT = TINT
  ELSE
    DELT = TEVENT
  END IF
ELSE
  DELT = TEVENT
  IF (IEVENT .EQ. 1) THEN
    IEVENT = 2
  ELSE
    TEVENT = TTSS
  END IF
END IF
END IF
IF (TIME + DELT .GT. PTIME) THEN
  IF (WP .NE. 0.0) THEN
    ANGTRAVR = WP * (PTIME-TIME) - 0.5 * SACCEL * (PTIME-TIME)**2
    WPP = WP - SACCEL * (PTIME-TIME)
  ELSE
    WPP = WP
  END IF
  IF (SLEWR .NE. 0.0) THEN
    SLEWR = SLEWR * (PTIME-TIME) - 0.5 * SLEW(2) * (PTIME-TIME)**2
    SLEWR = SLEWR - SLEW(2) * (PTIME-TIME)
    WRP = SLEWR * SIN(SLEWA)
    WQP = SLEWR * COS(SLEWA)
  ELSE
    WR = WR
    WQ = WQ
  END IF
  THTDX = (WF + WPP) * 0.5
  THTDX = (WQ + WQP) * 0.5
  THTDZ = (WR + WRP) * 0.5
  DELT = DELT - (PTIME-TIME)
  CALL QUATUP (THTDX, THTDY, THTDZ, PTIME-TIME, Q0, Q1, Q2, Q3, A)
  TEVENT = TEVENT - (PTIME-TIME)
  TIME = PTIME
  WP = WPP
  WQ = WQP
  WR = WRP
  CALL OUTPUT (ITIMEF, TIME, A, WP, WQ, WR, PTIME)
GOTO 10
ELSE
  TEVENT = TEVENT - DELT
  TIME = TIME + DELT
  IF (WP .NE. 0.0) THEN
    ANGTRAVR = WP * DELT - 0.5 * SACCEL * DELT**2
    WPP = WP - SACCEL * DELT
  ELSE
    WPP = WP
  END IF
IF (SLEWR .NE. 0.0) THEN
  ANGTRAVS = SLEWR * DELT - 0.5 * SLEW(2) * DELT ** 2
  SLEWR = SLEWR - SLEW(2) * DELT
  WRP = SLEWR * SIN(SLEWA)
  WQP = SLEWR * COS(SLEWA)
ELSE
  WRP = WR
  WQP = WQ
END IF
THTDX = (WR + WRP) * 0.5
THTDY = (WQ + WQP) * 0.5
THTDZ = (WR + WRP) * 0.5
CALL QUATUP (THTDX, THTDY, THTDZ, DELT, Q0, Q1, Q2, Q3, A)
IF (DELT .NE. TINT) THEN
  IF (:EVENT .EQ. 3) THEN
    WP = 0.0
    WQ = 0.0
    WR = 0.0
    SLEWR = 0.0
  ELSE IF (IHIGH .EQ. I) THEN
    WQ = 0.0
    WR = 0.0
    SLEWR = 0.0
    WP = WPp
  ELSE
    WP = 0.0
    WQ = WQP
    WR = WRP
  END IF
ELSE
  WP = WPp
  WQ = WQP
  WR = WRP
END IF
END IF
GOTO 10
END IF
ELSE
  WP = 0.0
  WQ = 0.0
  WR = 0.0
END IF
WRITE (13,*) 'MOTION STOPPED, TIME = ', TIME
C NOW THAT ANY MOTION LEFT OVER FROM THE PREVIOUS MANEUVER HAS BEEN
C NULLLED OUT, THE NEXT OBJECTIVE IS TO GET INTO THE PROPER ROLL ATTITUDE
IF (ISYS .NE. 4) THEN
  CALL ROLLER (TIME, PTIME, A, IPOINT, POINT, ROLL, IROLL, ITIMEF, WP, 1
  ISYS)
END IF
C NOW THAT THE DESIRED ROLL ATTITUDE HAS BEEN REACHED, THE SLEWING
C MOTION TAKES PLACE
RDELAY = 0.0
PTIME = TIME
TV1(1) = A(1, 1)
TV1(2) = A(1, 2)
TV1(3) = A(1, 3)
DO I = 1, 3
  DO J = 1, 3
    IF (I .EQ. J) THEN
      B(I, J) = 1.0
    ELSE
      B(I, J) = 0.0
    END IF
  END DO
END DO
A-12
IF (IPOINT(1) .EQ. 2) THEN
CALL ROTATE (B, SRA, -SDEC, 0.0, 0, 3, 2, 0)
ELSE IF (ISYS .EQ. 1) THEN
CALL ROTATE (B, POINT(1), -POINT(2), 0.0, 0, 3, 2, 0)
ELSE IF (ISYS .EQ. 2) THEN
CALL SUNV (FTIME, SUNRA, SUNDEC)
CALL ROTATE (B, SUNRA, -SUNDEC, 0.0, 0, 3, 2, 0)
ELSE IF (ISYS .EQ. 3) THEN
CALL GETSTATE (FTIME, X, DUM1, DUM2, T1, T2)
ANGI = ATAN2 (X(2), X(1))
ANG2 = ATAN2 (X(3), (X(1)**2 + X(2)**2)**0.5)
CALL ROTATE (B, ANGI, -ANG2 - 0.5*PI, 0.0, 0, 3, 2, 0)
CALL ROTATE (B, POINT(1), -POINT(2), 0.0, 0, 3, 2, 0)
ELSE IF (ISYS .EQ. 4) THEN
CALL ROTATE (B, SANG1, -SANG2, 0.0, 0, 3, 2, 0)
END IF
ENDIF

IF (T1 .GT. 1.0) T1 = 1.0
IF (T1 .LT. -1.0) T1 = -1.0
SLEWANG = ACOS (T1)

 ELSE IF (ISLEW(1) .EQ. 0.0) THEN
TTS = 0.0
WQ = 0.0
WT = 0.0
ELSE IF (ISLEW(1) .EQ. 1.0) THEN
TTS = SLEWANG / SLEW(1)
ELSE IF (ISLEW(1) .EQ. 1.0) THEN
TACCEL = 0.5 * SLEW(2) * SLEW(2)
DACCCEL = SLEW(1) * TACCEL - DACCCEL
DDECCEL = (SLEWANG .LT. DACCCEL + DDECCEL) THEN
TTS = (4.0 * SLEWANG / SLEW(2)) ** 0.5
T1 = TIME + TTS ** 0.5
T2 = T1
ELSE
TTS = 2.0 * TACCEL + (SLEWANG - DACCCEL - DDECCEL) ** 0.5
T1 = TIME + TACCEL
T2 = TIME + TTS - TACCEL
END IF
ENDIF

ENDIF
ELSE IF (ISYS .EQ. 1) THEN
IF (ABS(FTIME + RDELAY - FTIME) .GT. TOL) THEN
GOTO 50
ELSE
FTIME = TIME + TTS + RDELAY
GOTO 50
ENDIF
ELSE
FTIME = TIME + TTS
ENDIF
CI2M(2,3) = CI2M(3,1)*CI2M(1,2) - CI2M(3,2)*CI2M(1,1)

DO J = 2, 3
   ROLLM(2, J) = 0.0
   DO K = 1, 3
      ROLLM(2, J) = ROLLM(2, J) + CI2M(2, K) * A(J, K)
   END DO
END DO

ROLLTEMP = ATAN2 (ROLLM(2, 3), ROLLM(2, 2))

IF (TTS .EQ. 0.0) THEN
   CALL ROTATE (A, SLEWANG, 0.0, 0.0, 3, 0, 0)
   CALL ROTATE (A, 0.0, 0.0, -ROLLTEMP, 0.0, 1)
   GOTO 666
END IF

ROLLI = -ROLLTEMP

DO I = 1, 3
   CALL ROTATE (D, SLEWANG, 0.0, 0.0, 3, 0, 0)
END DO

IF (ISYS .NE. 4) THEN
   CALL GETPROJ (FTIME, B, ROT)
   IF (IPOINT(1) .EQ. 2) THEN
      CALL ROTATE (B, 0.0, 0.0, ROT+RBURN, 0, 0, 1)
   ELSE
      CALL ROTATE (B, 0.0, 0.0, ROT+POINT(3), 0, 0, 1)
   END IF
ELSE
   IF (IPOINT(1) .EQ. 1) THEN
      CALL ROTATE (B, 0.0, 0.0, STARROTI+POINT(1), 0, 0, 1)
   ELSE
      CALL ROTATE (B, 0.0, 0.0, STARROT2+POINT(2), 0, 0, 1)
   END IF
ENDIF

DO J = 2, 3
   ROLLM(2, J) = 0.0
   DO K = 1, 3
      ROLLM(2, J) = ROLLM(2, J) + D(2, K) * B(J, K)
   END DO
END DO

ROLLF = -ATAN2 (ROLLM(2, 3), ROLLM(2, 2))

DELTROLL = ROLLF - ROLLI
SROLL = ROLL(1)
SACCEL = ROLL(2)

IF (DELTROLL .GT. PI) DELTROLL = DELTROLL - 2.0*PI
IF (DELTROLL .LT. -PI) DELTROLL = DELTROLL + 2.0*PI
IF (DELTROLL .LT. 0.0) THEN
   SROLL = -SROLL
   SACCEL = -SACCEL
ENDIF

WPAVG = DELTROLL / TTS
TR0 = TIME
WPMAX = WPAVG
TR1 = TIME
TR2 = TIME+TTS
IF (IROLL(2) .EQ. 0) THEN
   IF (ABS(WPMAX) .GT. ABS(SROLL)) THEN
      WPMAX = SROLL
      TR2 = TIME+TTS
   END IF
ELSE
   TEMPI = TTS**2 - 4.0 * DELTROLL/SACCEL
   IF (TEMPI .GE. 0.0) THEN
      TACCI = 0.5 * (TTS - TEMPI**0.5)
      WPMAX = SACCEL * TACCI
   END IF
ENDIF

A-14
IF (ABS(WPMAX) .GT. ABS(SROLL)) THEN
  WPMAX = SROLL
  TACC1 = SROLL / SACCEL
  TR1 = TR0 + TACC1
  ANGT = (TTS-TACC1) * WPMAX
  TR2 = TIME+TTS- TACC1 + (DELTROLL-ANGT)/WPMAX
ELSE
  TR1 = TR0 + TACC1
  TR2 = TIME+TTS - TACC1
END IF
ELSE
  TACC1 = (DELTROLL/SACCEL)**0.5
  IF (TACC1 .GT. SROLL/SACCEL) THEN
    TACC1 = SROLL/SACCEL
    TR1 = TR0 + TACC1
    WPMAX = SROLL
    ANGT = (TTS-TACC1) * SROLL
    TR2 = TIME+TTS- TACC1 + (DELTROLL-ANGT)/SROLL
  ELSE
    TR1 = TR0 + TACC1
    WPMAX = SACCEL * TACC1
  END IF
TR2 = TR1
END IF
END IF
END IF
IF (ISYS .EQ. 2 .OR. ISYS .EQ. 3) THEN
  RDELAY = TR2+TR1-TR0-PTIME
  IF (RDELAY .GT. TOL) GOTO 50
END IF
WRITE (13, *) 'BEGIN SLEWING, TIME, SLEWANG, DELTROLL=', TIME, 
SLEWANG, DELTROLL
IF (ISLEW(1) .EQ. 1 .AND. ISLEW(2) .EQ. 0) THEN
  TTS = SLEWANG / SLEW(I)
  IF (TIME + TTS .GT. PTIME) THEN
    ANGTRAV = SLEW(I) * (PTIME-TIME-TDIFF*0.5)
    SLEWANG = SLEWANG - ANGTRAV
    CALL SLEWER (PTIME-.5*TDIFF, 
      CI2M, ROLLTEMP, ANGTRAV, POINT, IPOINT, ROLLI, 
      PSI, THT, PHI, A)
  ANGTRAV = SLEW(I) * TDIFF * 0.5
  SLEWANG = SLEWANG - ANGTRAV
  CALL SLEWER (PTIME, 
      CI2M, ROLLTEMP, ANGTRAV, POINT, IPOINT, ROLLI, 
      PSI, THT, PHI, A)
  DO I = 1,3
    DO J = 1,3
      B(I,J) = CI2M(I,J)
    END DO
  END DO
  ROLLTEMP = ROLLTEMP
  ANGTRAV = SLEW(I) * TDIFF * 0.5
  CALL SLEWER (PTIME+.5*TDIFF, 
    B, ROLLTEMP, ANGTRAV, POINT, IPOINT, ROLLI, 
    PSI, THT, PHI, C)
  PSID = (PSIP-PSIM) / TDIFF
  THTD = (THTP-THTM) / TDIFF
  PHID = (PHIP-PHIP) / TDIFF
  WP = PHID - PSID * SIN(THT)
  WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
  WR = PSID * COS(THT) - COS(PHI) - THTD * SIN(PHI)
  TIME = PTIME
  CALL OUTPUT (ITIMEF, TIMZ, A, WP, WQ, WR, PTIME)
GOTO 51
ELSE
  TIME = TIME + TTS
  CALL SLEWER (TIME,
C12M, ROLLSNP, SLEWANG, POIN'T, IPOINT, ROLLI,  
PSI, TH7, PHI, A) 

WQ = 0.0  
WR = 0.0  
IF (TIME .LE. TR2) THEN  
WP = WMAX  
ELSE IF (TIME .GE. TR2+TR1-TR0) THEN  
WP = 0.0  
ELSE  
WP = -WMAX / (TR1-TR0) * (TIME-TR2) + WMAX  
END IF 
END IF  
ELSE IF (ISLW(1) .EQ. 1 .AND. ISLW(2) .EQ. 1) THEN  
IF (TIME + TTS .GT. PTIME) THEN  
 IF (PTIME-TIME-TDIFF*0.5 .GE. 0.0) THEN  
 ISG = 1  
ELSE  
 ISG = -1  
END IF  
ANGTRAV = SLEWR * (PTIME-TIME-TDIFF*0.5) + 0.5  
* SLEW(2)*ISG*(PTIME-TIME-TDIFF*0.5)**2  
SLEWR = SLEWR + SLEW(2) * (PTIME-TIME-TDIFF*0.5)  
ELSE IF (T2 .GT. PTIME) THEN  
 IF (TIME .LT. T1) THEN  
 ANGTRAV = SLEW(1) * (T2-TIME) + 0.5  
 SLEWR = SLEW(1) * (T2-TIME)**2  
 ELSE  
 ANGTRAV = SLEW(1) * (PTIME-TIME-TDIFF*0.5)  
 END IF  
SLEWR = SLEW(1)  
ELSE  
IF (TIME .LT. T1) THEN  
 ANGTRAV = SLEW(1) * (T1-TIME) + 0.5  
 SLEWR = SLEW(1)  
 ELSE IF (TIME .LT. T2) THEN  
 ANGTRAV = SLEW(1) * (T2-TIME) - 0.5*SLEW(2) *  
 (PTIME-T2-TDIFF*0.5)**2  
 SLEWR = SLEW(1) - SLEW(2) * (PTIME-T2-TDIFF*  
 0.5)  
 ELSE  
 ANGTRAV = SLEWR * (PTIME-TIME-TDIFF*0.5) -  
 0.5 * SLEW(2) * (PTIME-TIME-TDIFF*  
 0.5)**2  
 SLEWR = SLEWR - SLEW(2) * (PTIME-TIME-TDIFF*  
 0.5)  
END IF  
END IF  
SLEWANG = SLEWANG - ANGTRAV  
CALL SLEWR (PTIME-0.5*TDIFF,  
C12M, ROLLSNP, ANGTRAV, POINT, IPOINT, ROLLI,  
PSI, TH7, PHI, A)  
IF (T1 .GT. PTIME) THEN  
IF (SLEWR .LT. 0.0) THEN  
TT0 = -SLEWR/SLEW(2)  
 ANGTRAV = - (SLEWR * TT0 + 0.5 * SLEW(2) * TT0  
**2) + 0.5 * SLEW(2) *(TDIFF*0.5-  
TT0)**2  
SLEWR = SLEW(2) * (TDIFF*0.5-TT0)
ELSE
  ANGTRAV = SLEWR * (TDIFF*0.5) + 0.5 * SLEW(2) * (TDIFF*0.5)**2
  SLEWR = SLEWR + SLEW(2) * (0.5*TDIFF)
END IF
ELSE IF (T2 .GT. PTIME) THEN
  ANGTRAV = SLEW(1) * (TDIFF*0.5) .
  SLEWR = SLEW(1)
ELSE
  ANGTRAV = SLEWR * (TDIFF*0.5) - 0.5 * SLEW(2) * (TDIFF*0.5)**2
  SLEWR = SLEWR - SLEW(2) * (TDIFF*0.5)
END IF
SLEWANG = SLEWANG - ANGTRAV
CALL SLEWER (PTIME,
   CI2M, ROLTEMP, ANGTRAV, POINT, IPOINT, ROLLI,
   PSI, THT, PHI, A)
DO I = 1,3
  DO J = 1,3
    B(I,J) = CI2M(I,J)
  END DO
END DO
ROLLTEMPS = ROLTEMP
IF (T1 .GT. PTIME) THEN
  ANGTRAV = SLEWR * (TDIFF*0.5) + 0.5 *
  SLEW(2) * (TDIFF*0.5)**2
ELSE IF (T2 .GT. PTIME) THEN
  ANGTRAV = SLEW(1) * (TDIFF*0.5)
ELSE
  ANGTRAV = SLEWR * (TDIFF*0.5) - 0.5 * SLEW(2) *
   (TDIFF*0.5)**2
END IF
CALL SLEWER (PTIME+.5*TDIFF,
   B, ROLTEMP, ANGTRAV, POINT, IPOINT, ROLLI,
   PSI, THT, PHI, A)
PSID = (PSIP-PSIM) / TDIFF
THTD = (THTP-THTM) / TDIFF
PHID = (PHIP-PHIM) / TDIFF
WP = PHID - PSID * SIN(THT)
WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL OUTPUT (ITIMEF, TIME, A, WP, WQ, WR, PTIME)
GOTO 60
ELSE
  TIME = TIME + TTS
  CALL SLEWER (TIME,
   CI2M, ROLTEMP, SLEWANG, POINT, IPOINT, ROLLI,
   PSI, THT, PHI, A)
WQ = 0.0
WR = 0.0
IF (TIME .LT. TR2) THEN
  WP = WPMAX
ELSE IF (TIME .GT. TR2+TR1-TR0) THEN
  WP = 0.0
ELSE
  WP = -WPMAX / (TR1-TR0) * (TIME-TR2) + WPMAX
END IF
END IF
END IF
WRITE (13,*) ' END SLEWING MOTION , TIME=TIME
GOTO 60
CALL ROLLER (TIME, PTIME, A, IPOINT, POINT, ROLL, INROLL, ITIMEF, WP,
   ISYS)
C AT THIS POINT THE DESIRED POINTING ATTITUDE SHOULD HAVE BEEN ACHIEVED
C AND THE CURRENT SLEW RATE OF 0 BUT A POSSIBLE NON-ZERO ROLL RATE
C THE NEXT THING TO CONSIDER IS STATION KEEPING. THE ONLY STATION
C KEEPING MODE WHERE THE ATTITUDE IS CHANGING IS THE LVLH STATION
C KEEPING, THE OTHERS ARE INERTIALLY FIXED. THE ATTITUDE WITH RESPECT
C TO THE SUN CHANGES SLIGHTLY WITH TIME, AND IF DRIFT FROM THE DESIRED
C ATTITUDE IS A PROBLEM, THE USER SHOULD ENTER PERIODIC MANEUVER
C CORRECTIONS TO MAINTAIN THE PROPER ATTITUDE. THE SLEW AND ROLL RATES
C INVOLVED IN MAINTAINING A PARTICULAR SOLAR ATTITUDE ARE SO SMALL THAT
C IT WAS FELT THAT PERIODIC CORRECTIONS WOULD BE MADE RATHER THAN TRY TO
C MOVE AT THESE VERY SMALL RATES.
IF (IPOINT(1) .EQ. 2) THEN
  STIME = TIMEB(ICOAST) - TIME
  ITIME = 1
END IF
IF (ITIMEF .EQ. 1 .AND. ITIME .EQ. 1) THEN
  TINC = STIME - (TIME-XTIME)
  IF (ABS(TINC) .GT. TTOL) THEN
    TSTAY(MAN) = TSTAY(MAN) + TINC
  ELSE
    ITIMEF = 0
  END IF
  IF (IMAN = MAN - 1
ELSE IF (ITIME .NE. 0) THEN
  IF (TIME + STIME .GT. PTIME) THEN
    STIME = STIME - (PTIME-TIME)
    TIME = PTIME
  ELSE
    TIME = TIME + STIME
    STIME = 0.0
  END IF
END IF
IF (ISYS .EQ. 3) THEN
  CALL LVLH(TIME-TDIFF*0.5, POINT, IPOINT, A, PSIM, THTM, PHIM)
  CALL LVLH(TIME, POINT, IPOINT, A, PSI, THT, PHI)
  DO I = 1,3
    DO J = 1,3
      B(I,J) = A(I,J)
    END DO
  END DO
  CALL LVLH(TIME+TDIFF*0.5, POINT, IPOINT, B, PSIP, THTP, PHIP)
  PSID = (PSIP-PSIM) / TDIFF
  THTD = (THTP-THTM) / TDIFF
  PHID = (PHIP-PHIM) / TDIFF
  WQ = THTD * COS(PHI) + PSID * COS(THT) * SIN(PHI)
  WR = PSID * COS(THT) * COS(PHI) - THTD * SIN(PHI)
ELSE
  WP = 0.0
  WQ = 0.0
  WR = 0.0
END IF
IF (TIME .EQ. PTIME) THEN
  CALL OUTPUT (ITIMEF, TIMB, A, WP, WQ, WR, PTIME)
END IF
IF (STIME .NE. 0.0) GOTO 61
END IF
C AT THIS POINT EVERYTHING IS DONE FOR THIS MANEUVER IF IT IS A
C POINTING COMMAND
ELSE
C HERE A RATE COMMANDED MANEUVER WILL BE HANDLED
WRITE (13,*) 'BEGIN RATE COMMAND, TIME=',TIME
7O
75
BKATE(1) = WP
BKATE(2) = WQ
BRATE(1) = WR
TSTOP = 9.99E9
DO I = 1, 3
IACC(I) = 0
IF (IRATE(I) .EQ. 1 .AND. IACCEL(I) .EQ. 0) THEN
BRATE(I) = RATE(I)
TACC(I) = 0.0
ELSE IF (IRATE(I) .EQ. 1 .AND. IACCEL(I) .EQ. 1) THEN
IF (BRATE(I) > RATE(I) ) THEN
SACC(I) = -ACCEL(I)
ELSE
SACC(I) = ACCEL(I)
END IF
A(TACC(I) = (KATE(I) - BPATE(I)) / SACC(I)
IACC(I) = 1
END IF
END IF
END DO
DELT = TINT
ITEMP = 0
DO I = 1, 3
IF (IACC(I) .EQ. 1 .AND. TACC(I) .LT. DELT) THEN
DELT = TACC(I)
ITEMP = I
END IF
END DO
IF (TSTOP - TIME .LT. DELT) THEN
DELT = TSTOP - TIME
END IF
IF (TIME .GT. PTIME) THEN
DO I = 1, 3
IF (IACC(I) .NE. 0) THEN
BRATEP(I) = BRATE(I) + SACC(I) * (PTIME - TIME)
ELSE
BRATEP(I) = BRATE(I)
END IF
END DO
THTDX = (BRATEP(1) + BRATE(1)) * 0.5
THTDY = (BRATEP(2) + BRATE(2)) * 0.5
THTDZ = (BRATEP(3) + BRATE(3)) * 0.5
CALL QUATUP (THTDX, THTDY, THTDZ, PTIME - TIME, Q0, Q1, Q2, Q3, A)
TIME = TIME + DELT
BKATEP = BKATE(I)
A - 19
BRATE(2) = BRATEP(2)
BRATE(3) = BRATEP(3)

IF (IACC(1).EQ.1.OR. IACC(2).EQ.1.OR. IACC(3).EQ.1)
  THEN
    IKEEP = 0
  GOTO 70
ELSE IF (ITIMEF .EQ. 1 .AND. ITIME .EQ. 1) THEN
  TINC = STIME - (TIME-XTIME)
  IF (ABS(TINC) .GT. TTOL) THEN
    TSTAY(MAN) = TSTAY(MAN) + TINC
  ELSE
    ITIMEF = 0
  END IF
  IMAN = MAN - 1
ELSE IF (ITIME .EQ. 1) THEN
  IF (IKEEP .EQ. 0) THEN
    IKEEP = 1
    TSTOP = TIME + STIME
  END IF
  IF (TIME .LT. TSTOP) GOTO 75
END IF
END IF

WP = BRATE(1)
WQ = BRATE(2)
WR = BRATE(3)

IF (TIME .GT. TIMEB(ICOAST)) THEN
  WRITE (6,*,'YOU HAVE DESIGNED COAST MANEUVERS DURING A BURN')
  STOP
END IF
GOTO 1000
END
SUBROUTINE CONVERT (STRNI, LEN, LOC, DNUM1)
CHARACTER STRNI (1: LEN)
CHARACTER NUMT (10)
CHARACTER SIGN
CHARACTER MINUS, PLUS, COMMA, DECIML, DOLLAR, BLANK, NULL
INTEGER LOC, POINT, LEN
INTEGER CNT2
DOUBLE PRECISION DNUM1, VAL, VAL2, AMT
NULL-CHAR (0)
BLANK=' '
MINUS=' -'
PLUS=' +'
COMMA=' ,'
DECIML=' .'
DOLLAR=' $'
NUMT (10) = '0'
NUMT (1) = '1'
NUMT (2) = '2'
NUMT (3) = '3'
NUMT (4) = '4'
NUMT (5) = '5'
NUMT (6) = '6'
NUMT (7) = '7'
NUMT (8) = '8'
NUMT (9) = '9'
CNT2 = 0
SIGN = PLUS
VAL = 0
VAL2 = 0
IF (LOC .LT. 1) GOTO 10
IF (LOC .GT. LEN) GOTO 10
GOTO 15
10 CONTINUE
DNUM1 = 0
RETURN
15 CONTINUE
POINT = LOC - 1
20 CONTINUE
POINT = POINT + 1
IF (POINT .GT. LEN) GOTO 10
IF (STRNI(POINT) .EQ. BLANK) GOTO 20
IF (SIGN = BLANK) GOTO 20
DO 60 I = 1, 10
A = 21
28 CONTINUE
SIGN = STRNI(POINT)
30 CONTINUE
POINT = POINT + 1
50 CONTINUE
IF (SIGN .EQ. COMMA) GOTO 40
IF (SIGN .EQ. DOLLAR) GOTO 40
IF (SIGN .EQ. BLANK) GOTO 40
DO 60 I = 1, 10
A-21
IF(STRN1(POINT) .EQ.NUMT(I)) GOTO 25

60 CONTINUE
61 VAL=VAL/10
62 IF (STN1(POINT) .NE.DECML) GOTO 100
70 CONTINUE
71 POINT=POINT+1
72 IF (POINT.GT.LEN) GOTO 85
73 DO 80 I=1,10
74 IF (STRN1(POINT) .EQ.NUMT(I)) GOTO 28
80 CONTINUE
81 IF (STRN1(POINT) .EQ.BLANK) GOTO 70
85 VAL2=VAL2/10
86 DO 90 I=1,CNT2
87 VAL2=VAL2/10
90 CONTINUE
95 CONTINUE
100 CONTINUE
101 IF (POINT.GT.LEN) GOTO 120
102 IF (STRN1(POINT) .EQ.BLANK) GOTO 105
103 IF (STRN1(POINT) .EQ.PLUS) GOTO 110
104 IF (STRN1(POINT) .EQ.MINUS) GOTO 110
105 GOTO 120
106 CONTINUE
107 POINT=POINT+1
108 IF (POINT.GT.LEN) GOTO 120
109 GOTO 100
110 CONTINUE
111 SIGN=STRN1(POINT)
112 CONTINUE
113 DNUM1=VAL+VAL2
114 IF (SIGN.EQ.MINUS) DNUM1=DNUM1* (-1.)
115 RETURN
END
SUBROUTINE GETPROJ (TIME, A, ROT)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(3,3), SUNI(3), SUNM(3)
CALL SUNV (TIME, RA, DEC)
SUNI (1) = COS (DEC) * COS (RA)
SUNI (2) = COS (DEC) * SIN (RA)
SUNI (3) = SIN (DEC)
DO I = 1, 3
   SUNM (I) = 0.0
   DO J = 1, 3
      SUNM (I) = SUNM (I) + A (I,J) * SUNI (J)
   END DO
END DO
ROT = ATAN2 (SUNM (3), SUNM (2))
RETURN
END
SUBROUTINE GETSTATE (T, XI, GI, THI, RAI, DECI)

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION XI (3), GI (3), THI (3), VAR(11, 3), VALUE(11)
COMMON /COMI/ DTIME(5000), DX(5000,3), DGACC(5000,3), DMACC(5000,3),
              IBRN(5000), DRA(5000), DDEC(5000)

IF (IFIRST .EQ. 0) THEN
  T1 = DTIME(1)
  T2 = DTIME(2)
  T3 = DTIME(3)
  TC = DTIME(4)
  IC = 4
END IF

10 IF (T .GT. T2 .AND. T-T1 .GT. TC-T) THEN
  T1 = T2
  T2 = T3
  T3 = TC
  IC = IC + 1
  TC = DTIME(IC)
  GOTO 10
END IF

DO I = 1, 11
  DO J = 1, 3
    IF (I .LE. 3) THEN
      VAR(I,J) = DX(IC-4+J, I)
    ELSE IF (I .LE. 6) THEN
      VAR(I,J) = DGACC(IC-4+J, I-3)
    ELSE IF (I .LE. 9) THEN
      VAR(I,J) = DMACC(IC-4+J, I-6)
    ELSE IF (I .EQ. 10) THEN
      VAR(I,J) = DKA(IC-4+J)
    ELSE
      VAR(I,J) = DDEC(IC-4+J)
    END IF
  END DO
END DO

IF (IBRN(IC-3) .EQ. 1 .AND. IBRN(IC-2) .EQ. 1 .AND. IBRN(IC-1) .EQ. 0) THEN
  IF (I .GE. 7 .AND. I .LE. 9 .AND. T .GT. DTIME(IC-2)) THEN
    A = 0.0
    B = (DMACC(IC, I-6)-DMACC(IC-1, I-6))
    C = DMACC(IC, I-6) - B * DTIME(IC)
  ELSE
    A = 0.0
    B = (VAR(I,1)-VAR(I,2)) / (DTIME(IC-3)-DTIME(IC-2))
    C = VAR(I,1) - B * DTIME(IC-3)
  END IF
  VALUE(I) = B * T + C
END IF
ELSE
  IF (IBRN(IC-3) .EQ. 0 .AND. IBRN(IC-2) .EQ. 1 .AND. IBRN(IC-1) .EQ. 1) THEN
    IF (I .GE. 7 .AND. I .LE. 9 .AND. T .LT. DTIME(IC-2)) THEN
      A = 0.0
      B = (DMACC(IC-3, I-6)-DMACC(IC-4, I-6))
      C = DMACC(IC-3, I-6) - B * DTIME(IC-3)
    ELSE
      A = 0.0
      B = (VAR(I,1)-VAR(I,2)) / (DTIME(IC-3)-DTIME(IC-2))
      C = VAR(I,1) - B * DTIME(IC-2)
    END IF
    VALUE(I) = B * T + C
  END IF
ELSE
  A = 2
DO I = 1, 11
   IF (I .GE. 7 .AND. I .LE. 9 .AND. IBRN(IC-3) .EQ. 0 .AND.
      IBRN(IC-2) .EQ. 0 .AND. IBRN(IC-1) .EQ. 1) THEN
      A = 0.0
      B = (VAR(I,2) - VAR(I,3)) / (DTIME(IC-2) - DTIME(IC-1))
      C = VAR(I,2) - B * DTIME(IC-2)
   ELSE IF (I .GE. 7 .AND. I .LE. 9 .AND. IBRN(IC-3) .EQ. 1
      .AND. IBRN(IC-2) .EQ. 0 .AND. IBRN(IC-1) .EQ. 0) THEN
      A = 0.0
      B = (VAR(I,1) - VAR(I,2)) / (DTIME(IC-3) - DTIME(IC-2))
      C = VAR(I,1) - B * DTIME(IC-3)
   ELSE
      A = ((VAR(I,1) - VAR(I,2)) * (DTIME(IC-3) - DTIME(IC-1)) -
           (VAR(I,1) - VAR(I,3)) * (DTIME(IC-3) - DTIME(IC-2))) / (DTIME(IC-1) - (DTIME(IC-3)**2 - DTIME(IC-2)**2))
      B = (VAR(I,1) - VAR(I,2) - A**2) / (DTIME(IC-3)**2 - DTIME(IC-2)**2)
      C = VAR(I,1) - A - B
   END IF
END DO
END IF
END DO
DO I = 1, 11
   IF (I .LE. 3) THEN
      XI(I) = VALUE(I)
   ELSE IF (I .LE. 6) THEN
      GI(I-3) = VALUE(I)
   ELSE IF (I .LE. 9) THEN
      THI(I-6) = VALUE(I)
   ELSE IF (I .EQ. 10) THEN
      RAI = VALUE(I)
   ELSE
      DECI = VALUE(I)
   END IF
END DO
RETURN
END
SUBROUTINE LVLH (TIME, POINT, IPOINT, A, PSI, THT, PHI)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION POINT (3), IPOINT (3), A(3, 3), X(3), DUM1 (3), DUM2 (3)
COMMON/COM3/PI, TWOP, PI02
IF (IPOINT(3) .EQ. 0) THEN
   CALL GETPROJ (TIME, A, ROT1)
END IF
DO I = 1, 3
   DO J = 1, 3
      IF (I .EQ. J) THEN
         A(I, J) = 1.0
      ELSE
         A(I, J) = 0.0
      END IF
   END DO
END DO
CALL GETSTATE (TIME, X, DUM1, DUM2, RA, DEC)
ANG1 = ATAN2(X(2), X(1))
ANG2 = ATAN2(X(3), (X(1)**2 + X(2)**2)**0.5)
CALL ROTATE (A, ANG1, -ANG2 - 0.5*PI, 0.0, 3, 2, 0)
CALL ROTATE (A, POINT(1), -POINT(2), 0.0, 3, 2, 0)
CALL GETPROJ (TIME, A, ROT)
IF (IPOINT(3) .EQ. 1) THEN
   CALL ROTATE (A, 0.0, 0.0, ROT + POINT(3), 0, 0, 1)
ELSE
   CALL ROTATE (A, 0.0, 0.0, ROT - ROT1, 0, 0, 1)
END IF
PSI = ATAN2 (A(1, 2), A(1, 1))
THT = ASIN (-A(1, 3))
PHI = ATAN2 (A(2, 3), A(3, 3))
RETURN
END
SUBROUTINE OUTPUT (ITIMEF, TIME, A, WP, WQ, WR, PTIME)
IMPLICIT REAL*8 (A-H, O-Z)
COMMON /COM2/ PINT, TIME0
DIMENSION A(3, 3), G(3), TH(3), DUM(3)
PTIME = PTIME + PINT
IF (ITIMEF .EQ. 0) THEN
CALL QUAT (A, Q0, Q1, Q2, Q3)
CALL GETSTATE (TIME, DUM, G, TH, T1, T2)
WRITE (31,100) TIME, G(1), TH(1), G(2), TH(2), G(3), TH(3)
100 FORMAT (' ',D14.8,3(1X,D16.9))
WRITE (32,100) TIME, G(1), G(2), G(3)
WRITE (33,100) TIME, TH(1), TH(2), TH(3)
WRITE (34,100) TIME, WP, WQ, WR
WRITE (35,101) TIME, Q0, Q1, Q2, Q3
101 FORMAT (' ',D14.8,4(1X,D16.9))
END IF
RETURN
END
SUBROUTINE POINTER (TIME, POINT, IPOINT, ISYS, A)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION POINT(3), A(3, 3), X(3), IPOINT(3), SV1(3), SV1M(3), SV2(3),
1 SV2M(3)
COMMON/COM3/P1, TWOPI, PI02
COMMON/COM7/SANG1, SANG2, STARROT1, STARROT2, SV1, SV2
DO I = 1, 3
  DO J = 1, 3
    IF (I .EQ. J) THEN
      A(I,J) = 1.0
    ELSE
      A(I,J) = 0.0
    END IF
  END DO
END DO
IF (ISYS .EQ. 1) THEN
  CALL ROTATE (A, POINT(1), -POINT(2), 0.0, 3, 2, 0)
  CALL GETPROJ (TIME, A, ROT)
ELSE IF (ISYS .EQ. 2) THEN
  CALL SUNV (TIME, SUNRA, SUNDEC)
  CALL ROTATE (A, SUNRA, -SUNDEC, 0.0, 3, 2, 0)
  CALL GETPROJ (TIME, A, ROT)
ELSE IF (ISYS .EQ. 3) THEN
  CALL GETSTATE (TIME, X)
  ANG1 = ATAN2 (X(2), X(1))
  ANG2 = ATAN2 (X(3), (X(1)**2 + X(2)**2)**0.5)
  CALL ROTATE (A, ANG1, -ANG2 - 0.5*PI, 0.0, 3, 2, 0)
  CALL GETPROJ (TIME, A, ROT)
ELSE IF (ISYS .EQ. 4) THEN
  CALL ROTATE (A, SANG1, -SANG2, 0.0, 3, 2, 0)
  IF (IPOINT(1) .EQ. 1) THEN
    CALL ROTATE (A, 0.0, 0.0, STARROT1 + POINT(1), 0.0, 1)
  ELSE
    CALL ROTATE (A, 0.0, 0.0, STARROT2 + POINT(2), 0.0, 1)
  END IF
END IF
END
RETURN
END
SUBROUTINE QUAT (COORD, Q0, Q1, Q2, Q3)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION COORD(3,3)
Q0 = 0.5 * (1.0 + COORD(1,1) + COORD(2,2) + COORD(3,3)) ** 0.5
Q1 = (Q0 * Q0 - 0.5 * (COORD(2,2) + COORD(3,3))) ** 0.5
IF (COORD(2,3) - COORD(3,2) .LT. 0.0) THEN
  Q1 = -Q1
END IF
Q2 = (Q0 * Q0 - 0.5 * (COORD(1,1) + COORD(3,3))) ** 0.5
IF (COORD(3,1) - COORD(1,3) .LT. 0.0) THEN
  Q2 = -Q2
END IF
Q3 = (Q0 * Q0 - 0.5 * (COORD(1,1) + COORD(2,2))) ** 0.5
IF (COORD(1,2) - COORD(2,1) .LT. 0.0) THEN
  Q3 = -Q3
END IF
RETURN
END
SUBROUTINE QUATUP (THTDX, THTDY, THTDZ, DELTA, Q0, Q1, Q2, Q3, A)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(3,3)
THTD = (THTDX**2 + THTDY**2 + THTDZ**2)**0.5
IF (THTD .NE. 0.0) THEN
Q5 = C0S (THTD*DELTA/ 2.0)
Q6 = S1N (THTD*DELTA/ 2.0) / TTH1D
Q0P = Q5 * Q0 - Q6 * (Q1 * THTDX + Q2 * THTDY + Q3 * THTDZ)
Q1P = Q5 * Q1 + Q6 * (Q0 * THTDX + Q2 * THTDZ - Q3 * THTDY)
Q2P = Q5 * Q2 + Q6 * (Q0 * THTDY - Q1 * THTDZ + Q3 * THTDX)
Q3P = Q5 * Q3 - Q6 * (Q0 * THTDZ + Q1 * THTDX - Q2 * THTDY)
Q0 = Q0P
Q1 = Q1P
Q2 = Q2P
Q3 = Q3P
DQ = 0.5 * (1.0 - Q0*Q0 - Q1*Q1 - Q2*Q2 - Q3*Q3)
Q0 = Q0 * (1.0 + DQ)
Q1 = Q1 * (1.0 + DQ)
Q2 = Q2 * (1.0 + DQ)
Q3 = Q3 * (1.0 + DQ)
END IF
A(1,1) = Q0 * Q0 + Q1 * Q1 - Q2 * Q2 - Q3 * Q3
A(1,2) = 2.0 * (Q1 * Q2 + Q3 * Q0)
A(1,3) = 2.0 * (Q1 * Q3 - Q2 * Q0)
A(2,1) = 2.0 * (Q1 * Q2 - Q3 * Q0)
A(2,2) = Q0 * Q0 - Q1 * Q1 + Q2 * Q2 - Q3 * Q3
A(2,3) = 2.0 * (Q2 * Q3 + Q1 * Q0)
A(3,1) = 2.0 * (Q1 * Q3 + Q2 * Q0)
A(3,2) = 2.0 * (Q2 * Q3 - Q1 * Q0)
A(3,3) = Q0 * Q0 - Q1 * Q1 - Q2 * Q2 + Q3 * Q3
RETURN
END
SUBROUTINE RMAN (JCOAST, JMAN, NAME, ISYS, POINT, IPOINT, SLEW, ISLEW, 
ROLL, IROLL, RATE, IRATE, ACCEL, IACCEL, TIME, ITIME)

IMPLICIT REAL*8 (A-H, O-Z)
COMMON/COM3/PI, TWOPI, PI02
CHARACTER*1 CTYPE, ATEMP
CHARACTER*40 NAME, TEMP
CHARACTER*60 ALPHA
DIMENSION POINT (3), IPOINT (3), SLEW (2), ISLEW (2), ROLL (2), IROLL (2), 
RATE (3), IRATE (3), ACCEL (3), IACCEL (3)
DO I = 1, 3
    POINT(I) = 0.0 
    RATE(I) = 0.0 
    ACCEL(I) = 0.0 
    IPOINT(I) = 0 
    IRATE(I) = 0 
    IACCEL(I) = 0 
    IF (I .NE. 3) THEN
        SLEW(I) = 0.0 
        ROLL(I) = 0.0 
        ISLEW(I) = 0 
        IROLL(I) = 0 
    END IF
END DO

C C READ A BLANK LINE
C
READ (20, *)

C C GET COAST NUMBER
C
READ (20, 110, END=77) JCOAST
110 FORMAT (9X, I10)
    IF (JCOAST .LE. 0) THEN 
        WRITE (6, *) 'YOU HAVE ENTERED A NON-POSITIVE COAST NUMBER : ', JCOAST 
   END IF
STOP 

READ (20, 111) JMAN
111 FORMAT (12X, I10)
    IF (JMAN .LE. 0) THEN 
        WRITE (6, *) 'YOU HAVE ENTERED A NON-POSITIVE MANEUVER NUMBER : ', JMAN 
    END IF
STOP

READ (20, 112) NAME
112 FORMAT (6X, A40)
READ (20, 113) CTYPE
113 FORMAT (29X, A1)
    IF (CTYPE .EQ. 'A' .OR. CTYPE .EQ. 'a') THEN 
        ISYS = 1
    ELSE IF (CTYPE .EQ. 'B' .OR. CTYPE .EQ. 'b') THEN 
        ISYS = 2
    ELSE IF (CTYPE .EQ. 'C' .OR. CTYPE .EQ. 'c') THEN 
        ISYS = 3
    ELSE IF (CTYPE .EQ. 'D' .OR. CTYPE .EQ. 'd') THEN 
        ISYS = 4
    ELSE 
        WRITE (6, *) 'YOU HAVE ENTERED AN IMPROPER COORDINATE SYSTEM : ', CTYPE 
        STOP
    END IF
READ (20, *)
READ (20, 114) ALPHA
114 FORMAT (17X, A60)
IF (ALPHA(1:1) .EQ. 'B' .OR. ALPHA(1:1) .EQ. 'b') THEN
  IPOINT(1) = 2
ELSE IF (ALPHA(1:1) .EQ. ' ') THEN
  IPOINT(1) = 0
  IPOINT(2) = 0
  IPOINT(3) = 0
ELSE
  IF (ALPHA(1:1) .EQ. 'X' .OR. ALPHA(1:1) .EQ. 'x') THEN
    IPOINT(1) = 0
    INUM = 3
  ELSE
    INUM = 2
    IF (ALPHA(INUM:INUM) .NE. ',' .AND. ALPHA(INUM:INUM) .NE. ' ') THEN
      INUM = INUM + 1
      IF (INUM .EQ. 61) GOTO 11
      GOTO 10
    END IF
    TEMP(1:40) = ' ';
    TEMP = ALPHA(INUM:INUM-1)
    CALL CONVERT (TEMP,40,1,T1)
    POINT(1) = T1
    IPOINT(1) = 1
    INUM = INUM + 1
    IF (INUM .GT. 60 .OR. ALPHA(INUM-1:INUM-1) .EQ. ' ') GOTO 40
  END IF
  IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
    IPOINT(2) = 0
    IPOINT(3) = 0
  ELSE
    IF (ALPHA(INUM:INUM) .EQ. 'X' .OR. ALPHA(INUM:INUM) .EQ. 'x') THEN
      IPOINT(2) = 0
      INUM = INUM + 2
    ELSE
      INUM = INUM
      INUM = INUM + 1
      IF (ALPHA(INUM:INUM) .NE. ',' .AND. ALPHA(INUM:INUM) .NE. ' ') THEN
        INUM = INUM + 1
        IF (INUM .EQ. 61) GOTO 21
        GOTO 20
      END IF
      TEMP(1:40) = ' ';
      TEMP = ALPHA(INUM:INUM-1)
      CALL CONVERT (TEMP,40,1,T1)
      POINT(2) = T1
      IPOINT(2) = 1
      INUM = INUM + 1
      IF (INUM .GT. 60 .OR. ALPHA(INUM-1:INUM-1) .EQ. ' ') GOTO 40
    END IF
    IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
      IPOINT(3) = 0
    ELSE
      TEMP(1:40) = ' ';
      TEMP = ALPHA(INUM:60)
      CALL CONVERT (TEMP,40,1,T1)
      POINT(3) = T1
      IPOINT(3) = 1
    END IF
  END IF
END IF
END IF
READ (20,114) ALPHA
IF (ALPHA(1:1) .EQ. ' ') THEN
  ISLEW(1) = 0
ISLEW(2) = 0
ELSE
    IF (ALPHA(1:1) .EQ. 'X' .OR. ALPHA(1:1) .EQ. 'x') THEN
        ISLEW(1) = 0
        INUM = 3
    ELSE
        INUM = 2
    END IF
IF (ALPHA(INUM:INUM) .NE. ',') .AND. ALPHA(INUM:INUM) .NE. ' ' ) THEN
    INUM = INUM + 1
END IF
IF (INUM .EQ. 61) GOTO 31
GOTO 30
END IF
ISLEW(2) = 0
ELSE
    TEMP(1:40) = ' '
    TEMP = ALPHA(1:INUM-1)
    CALL CONVERT (TEMP, 40, 1, T1)
    SLEW(1) = T1
    ISLEW(1) = 1
    IF (INUM .GT. 60 .OR. ALPHA(INUM:INUM) .EQ. ' ') GOTO 50
    IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
        ISLEW(2) = 0
    ELSE
        IROLL(1) = 0
        IF (ALPHA(1:1) .EQ. 'X' .OR. ALPHA(1:1) .EQ. 'x') THEN
            IROLL(1) = 0
            INUM = 3
        ELSE
            INUM = 2
        END IF
        IF (ALPHA(INUM:INUM) .NE. ',') .AND. ALPHA(INUM:INUM) .NE. ' ' ) THEN
            INUM = INUM + 1
        END IF
        IF (INUM .EQ. 61) GOTO 231
        GOTO 230
    END IF
    TEMP(1:40) = ' '
    TEMP = ALPHA(1:INUM-1)
    CALL CONVERT (TEMP, 40, 1, T1)
    ROLL(1) = T1
    IROLL(1) = 1
    IF (INUM .GT. 60 .OR. ALPHA(INUM:INUM) .EQ. ' ') GOTO 250
    IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
        IROLL(2) = 0
    ELSE
        TEMP(1:40) = ' '
        TEMP = ALPHA(INUM:60)
        CALL CONVERT (TEMP, 40, 1, T1)
        ROLL(2) = T1
        IROLL(2) = 1
    END IF
END IF
READ (20,*)
A-33
READ (20,116) ALPHA
116  FORMAT (20X,A60)
   IF (ALPHA(1:1) .EQ. ' ') THEN
      IRATE(1) = 0
      IRATE(2) = 0
      IRATE(3) = 0
   ELSE
      IF (ALPHA(1:1) .EQ. 'X' .OR. ALPHA(1:1) .EQ. 'x') THEN
         IRATE(1) = 0
         INUM = 3
      ELSE
         INUM = 2
      END IF
      IF (ALPHA(INUM:INUM) .NE. '.' .AND. ALPHA(INUM:INUM) .NE. ' ,') THEN
         INUM = INUM + 1
         IF (INUM .EQ. 61) GOTO 311
         GOTO 310
      END IF
      TEMP(1:40) = ' '  
      TEMP = ALPHA(1:INUM-1)
      CALL CONVERT (ALPHA,40,1,T1)
      RATE(1) = T1
      IRATE(1) = 1
      INUM = INUM + 1
      IF (INUM .GT. 60 .OR. ALPHA(INUM:INUM-1) .EQ. ' ') GOTO 340
   END IF
   IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
      IRATE(2) = 0
      IRATE(3) = 0
   ELSE
      IF (ALPHA(INUM:INUM) .EQ. 'X' .OR. ALPHA(INUM:INUM) .EQ. 'x') THEN
         IRATE(2) = 0
         INUM = INUM + 2
      ELSE
         INUM0 = INUM
         INUM = INUM + 1
         IF (ALPHA(INUM:INUM) .NE. '.' .AND. ALPHA(INUM:INUM) .NE. '(') THEN
            INUM = INUM + 1
            IF (INUM .EQ. 61) GOTO 321
            GOTO 320
         END IF
         TEMP(1:40) = ' '  
         TEMP = ALPHA(INUM0:INUM-1)
         CALL CONVERT (TEMP,40,1,T1)
         RATE(2) = T1
         IRATE(2) = 1
         INUM = INUM + 1
         IF (INUM .GT. 60 .OR. ALPHA(INUM:INUM-1) .EQ. ' ') GOTO 340
      END IF
      IF (ALPHA(INUM:INUM) .EQ. '(') THEN
         IRATE(3) = 0
      ELSE
         TEMP(1:40) = ' '  
         TEMP = ALPHA(INUM:60)
         CALL CONVERT (TEMP,40,1,T1)
         RATE(3) = T1
         IRATE(3) = 1
      END IF
   END IF
   END IF
340  READ (20,117) ALPHA
117  FORMAT (19X,A60)
   IF (ALPHA(1:1) .EQ. ' ') THEN
IACCEL(1) = 0
IACCEL(2) = 0
IACCEL(3) = 0
ELSE
IF (ALPHA(1:1) .EQ. 'X' .OR. ALPHA(1:1) .EQ. 'x') THEN
  IACCEL(1) = 0
  INUM = 3
ELSE
  INUM = 2
  IF (ALPHA(INUM:INUM) .NE. ',' .AND. ALPHA(INUM:INUM) .NE. ' ') THEN
    INUM = INUM + 1
  IF (INUM .EQ. 61) GOTO 411
  GOTO 410
END IF
410
TEMP(1:40) = ''
TEMP = ALPHA(1:INUM-1)
CALL CONVERT (TEMP,40,1,T1)
IACCEL(1) = T1
INUM = INUM + 1
IF (INUM .GT. 60 .OR. ALPHA(INUM-1:INUM-1) .EQ. ' ') GOTO 440
END IF
411
TEMP(I:40) = ' '
TEMP = ALPHA(INUM:INUM)
CALL CONVERT (TEMP,40,1,T1)
ACCEL(1) = T1
ELSE
IF (ALPHA(INUM:INUM) .EQ. 'X' .OR. ALPHA(INUM:INUM) .EQ. 'x') THEN
  INUM = INUM + 1
ELSE
  INUM0 = INUM
  IF (ALPHA(INUM:INUM) .NE. ',' .AND. ALPHA(INUM:INUM) .NE. ' ') THEN
    INUM = INUM + 1
  IF (INUM .EQ. 61) GOTO 421
  GOTO 420
END IF
420
TEMP(1:40) = ''
TEMP = ALPHA(INUM0:INUM-1)
CALL CONVERT (TEMP,40,1,T1)
ACCEL(2) = T1
IACCEL(2) = 1
ELSE
IF (ALPHA(INUM:INUM) .EQ. ' ') THEN
  IACCEL(3) = 0
ELSE
  TEMP = ALPHA(INUM:INUM)
  CALL CONVERT (TEMP,40,1,T1)
  ACCEL(3) = T1
  IACCEL(3) = 1
END IF
END IF
421
END IF
440
READ (20,*)
READ (20,120) ATMP
120 FORMAT (28X,A1)
IF (ATMP .EQ. ' ') THEN
  ITIME = 0
ELSE IF (ATMP .EQ. '+') THEN
  ITIME = 0
END IF
A-35
ITIME = 2
ELSE
BACKSPACE 20
READ (20, 121) TIME
121 FORMAT (28X, F20.8)
ITIME = 2
IF (TIME .LT. 0.0) THEN
WRITE (6, *) 'YOU HAVE ENTERED A NEGATIVE TIME : ', TIME
STOP
END IF
END IF
DO I = 1, 3
POINT(I) = POINT(I) * PI/180.0
RATE(I) = RATE(I) * PI/180.0
ACCEL(I) = ACCEL(I) * PI/180.0
IF (I .LE. 2) THEN
SLEW(I) = SLEW(I) * PI/180.0
ROLL(I) = ROLL(I) * PI/180.0
END IF
END DO
TIME = TIME * 60.0
RETURN
END
SUBROUTINE ROLLER (TIME, PTIME, A, IPOINT, POINT, ROLL, IROLL, IREAD, WP, ISYS)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON /COM3/ PI, TW0PI, PI02
COMMON /CM4/ PUN
COMMON /CM7/ SANG, SANG2, STABROT1, STABROT2, SV1(3), SV2(3)
DIMENSION A(3,3), IPOINT(3), POINT(3), ROLL(2), IROLL(2), SV1M(3),
1 SV2M(3)

IF (ISYS .NE. 4) THEN
  CALL GETPROJ (TIME, A, ROT)
ELSE IF (IPOINT(1) .EQ. 2) THEN
  ROLLANG1 = ROT + PBURN
  ELSE IF (IPOINT(2) .NE. 0) THEN
  ROLLANG1 = ROT + POINT(3)
  ELSE
  ROLLANG1 = 0.0
END IF
ELSE
  IF (IPOINT(1) .EQ. 1) THEN
    SV1M(2) = A(2,1) * SV1(1) + A(2,2) * SV1(2) + A(2,3) * SV1(3)
    SV1M(3) = A(3,1) * SV1(1) + A(3,2) * SV1(2) + A(3,3) * SV1(3)
    ROT = ATAN2 (SV1M(3), SV1M(2))
    ROLLANG1 = ROT + POINT(1)
  ELSE
    SV2M(2) = A(2,1) * SV2(1) + A(2,2) * SV2(2) + A(2,3) * SV2(3)
    SV2M(3) = A(3,1) * SV2(1) + A(3,2) * SV2(2) + A(3,3) * SV2(3)
    ROT = ATAN2 (SV2M(3), SV2M(2))
    ROLLANG1 = ROT + POINT(2)
  END IF
END IF
ROLL = ANG(ROLLANG1)
WRITE (13,*) 'ROLLING TO CORRECT ATTITUDE, TIME, ROLLANG=',
1 TIME, ROLLANG IF (ROLLANG .GT. PI) THEN
  ROLLANG = ROLLANG - TWOPI
  SROLL = -ROLL(1)
  SACCEL = -ROLL(2)
ELSE
  SROLL = ROLL(1)
  SACCEL = ROLL(2)
END IF
ELSE IF (IROLL(1) .EQ. 0 .AND. IROLL(2) .EQ. 0) THEN
  CALL ROTATE (A, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
ELSE IF (IROLL(1) .EQ. 1 .AND. IROLL(2) .EQ. 0) THEN
  TTS = ROLLANG / SROLL
  IF (TIME + TTS .GT. PTIME) THEN
    ANGTRAV = SROLL * (PTIME - TIME)
    CALL ROTATE (A, 0.0, 0.0, 0.0, ANGTRAV, 0.0, 0.1)
    ROLLANG = ROLLANG - ANGTRAV
    WP = SROLL
    TIME = PTIME
    CALL OUTPUT (IREAD, TIME, A, WP, WR, PTIME)
    GOTO 30
  ELSE
    TIME = TIME + TTS
    WP = 0.0
    CALL ROTATE (A, 0.0, 0.0, 0.0, ROLLANG, 0.0, 0.1)
  END IF
ELSE IF (IROLL(1) .EQ. 1 .AND. IROLL(2) .EQ. 1) THEN
  TEMP1 = (0.5 * WP**2 + ROLLANG * SACCEL)**0.5
  TACC1 = (-WP + TEMP1) / SACCEL
END IF

A-37
TACC2 = (-WP - TEMP1) / SACCEL
IF (TACC > TACC2) THEN
TACC = TACC1
ELSE
TACC = TACC2
END IF
IF (TACC .LT. 0.0) THEN
TACC = 0.0
ENDIF
WPMAX = WP + TACC * SACCEL
IF (ABS(WPMAX) .GT. ABS(SROLL)) THEN
TACC = (SROLL-WP) / SACCEL
WPMAX = SROLL
ENDIF
T1 = TIME + TACC
TDEC = TACC + WP/SACCEL
ANGT = WP * TACC + 0.5 * SACCEL * TACC**2 + 0.5 * SACCEL * TDEC ** 2
T2 = T1 + (ROLLANG-ANGT)/SROLL
TTS = T2 + TDEC - TME
TACCEL = ROLL(1) / ROLL(2)
ANGACC = 0.5 * ROLL(2) * TACCEL**2
ANGDEC = ROLL(2) * TACCEL - ANGACC
IF (ABS(ROLLANG) .LT. ANGACC + ANGDEC) THEN
TTS = (4.0 * ABS(ROLLANG) / ROLL(2)) ** 0.5
T1 = TIME + TTS * 0.5
T2 = T1
ELSE
TTS = 2.0 * TACCEL + (ABS(ROLLANG) - ANGACC - ANGDEC) / ROLL(1)
T1 = TIME + TACCEL
T2 = TIME + TTS - TACCEL
ENDIF
IF (T1 .GT. PTIME) THEN
ANGTRAV = WP * (PTIME-TIME) + 0.5 * SACCEL * (PTIME-TIME)**2
WP = WP + SACCEL * (PTIME-TIME)
ROLLANG = ROLLANG - ANGTRAV
CALL ROTATE (A,0,0,0.0,ANGTRAV,0,0,1)
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE IF (T2 .GT. PTIME) THEN
IF (TIME .LT. TI) THEN
ANGTRAV = WP * (T1-TIME) + 0.5 * SACCEL * (T1-TIME)**2 + WPMAX * (PTIME-TIME)
ELSE
ANGTRAV = WPMAX * (PTIME-TIME)
ENDIF
WP = WPMAX
ROLLANG = ROLLANG - ANGTRAV
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL ROTATE (A,0,0,0.0,ANGTRAV,0,0,1)
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE IF (TIME + TTS .GT. PTIME) THEN
IF (TIME .LT. TI) THEN
ANGTRAV = WP * (T1-TIME) + 0.5 * SACCEL * (T1-TIME)**2 + WPMAX * (PTIME-TIME)
ELSE
ANGTRAV = WPMAX * (PTIME-TIME)
ENDIF
WP = WPMAX
ROLLANG = ROLLANG - ANGTRAV
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL ROTATE (A,0,0,0.0,ANGTRAV,0,0,1)
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE IF (TIME + TTS .GT. PTIME) THEN
IF (TIME .LT. TI) THEN
ANGTRAV = WP * (T1-TIME) + 0.5 * SACCEL * (T1-TIME)**2 + WPMAX * (PTIME-TIME)
ELSE
ANGTRAV = WPMAX * (PTIME-TIME)
ENDIF
WP = WPMAX
ROLLANG = ROLLANG - ANGTRAV
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL ROTATE (A,0,0,0.0,ANGTRAV,0,0,1)
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE IF (TIME + TTS .GT. PTIME) THEN
IF (TIME .LT. TI) THEN
ANGTRAV = WP * (T1-TIME) + 0.5 * SACCEL * (T1-TIME)**2 + WPMAX * (PTIME-TIME)
ELSE
ANGTRAV = WPMAX * (PTIME-TIME)
ENDIF
WP = WPMAX
ROLLANG = ROLLANG - ANGTRAV
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL ROTATE (A,0,0,0.0,ANGTRAV,0,0,1)
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE IF (TIME .LT. T2) THEN
ANGTRAV = WPMAX * (T2-TIME) - 0.5 * SACCEL * (PTIME-TIME)**2
ENDIF
A-38
WP = WPMAX - SACCEL * (PTIME-T2)
ELSE
  ANGTRAV = WP * (PTIME-TIME) - 0.5 * SACCEL * (PTIME-TIME)**2
  WP = WP - SACCEL * (PTIME-TIME)
END IF
ROLLANG = ROLLANG - ANGTRAV
CALL ROTATE (A,0.0,0.0,ANGTRAV,0,0,1)
TTS = TTS - (PTIME-TIME)
TIME = PTIME
CALL OUTPUT (IREAD,TIME,A,WP,WQ,WR,PTIME)
GOTO 40
ELSE
  TIME = TIME + TTS
  CALL ROTATE (A,0.0,0.0,ROLLANG,0,0,1)
  WP = 0.0
END IF
END IF
WRITE (13,*) 'FINISHED ROLLING, TIME=',TIME
RETURN
END
SUBROUTINE ROTATE (A, ANG1, ANG2, ANG3, N1, N2, N3)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A(3, 3), B(3, 3), N(3), ANGLE(3), SANG(3), CANG(3)

ANGLE(1) = ANG1
ANGLE(2) = ANG2
ANGLE(3) = ANG3
N(1) = N1
N(2) = N2
N(3) = N3
DO I = 1, 3
  SANG(I) = SIN(ANGLE(I))
  CANG(I) = COS(ANGLE(I))
END DO

DO I = 1, 3
  IF (N(I) .EQ. 1) THEN
    B(1,1) = A(1,1)
    B(1,2) = A(1,2)
    B(1,3) = A(1,3)
    B(2,1) = A(2,1) * CANG(I) + A(3,1) * SANG(I)
    B(2,2) = A(2,2) * CANG(I) + A(3,2) * SANG(I)
    B(2,3) = A(2,3) * CANG(I) + A(3,3) * SANG(I)
    B(3,1) = -A(2,1) * SANG(I) + A(3,1) * CANG(I)
    B(3,2) = -A(2,2) * SANG(I) + A(3,2) * CANG(I)
    B(3,3) = -A(2,3) * SANG(I) + A(3,3) * CANG(I)
    DO J = 1, 3
      A(J, K) = B(J, K)
    END DO
  END IF
  ELSE IF (N(I) .EQ. 2) THEN
    B(1,1) = A(1,1) * CANG(I) - A(3,1) * SANG(I)
    B(1,2) = A(1,2) * CANG(I) - A(3,2) * SANG(I)
    B(1,3) = A(1,3) * CANG(I) - A(3,3) * SANG(I)
    B(2,1) = A(2,1)
    B(2,2) = A(2,2)
    B(2,3) = A(2,3)
    B(3,1) = A(1,1) * SANG(I) + A(3,1) * CANG(I)
    B(3,2) = A(1,2) * SANG(I) + A(3,2) * CANG(I)
    B(3,3) = A(1,3) * SANG(I) + A(3,3) * CANG(I)
    DO J = 1, 3
      A(J, K) = B(J, K)
    END DO
  END IF
  ELSE IF (N(I) .EQ. 3) THEN
    B(1,1) = A(1,1) * CANG(I) + A(2,1) * SANG(I)
    B(1,2) = A(1,2) * CANG(I) + A(2,2) * SANG(I)
    B(1,3) = A(1,3) * CANG(I) + A(2,3) * SANG(I)
    B(2,1) = -A(1,1) * SANG(I) + A(2,1) * CANG(I)
    B(2,2) = -A(1,2) * SANG(I) + A(2,2) * CANG(I)
    B(2,3) = -A(1,3) * SANG(I) + A(2,3) * CANG(I)
    B(3,1) = A(3,1)
    B(3,2) = A(3,2)
    B(3,3) = A(3,3)
    DO J = 1, 3
      A(J, K) = B(J, K)
    END DO
  END IF
END DO
END DO
END IF
END DO
RETURN
END
SUBROUTINE SLEWER (TIME, CI2M, ROLLTEMP, ANGTRAV, POINT, IPOINT, ROLL1, 
  PSI, THT, PHI, A)
IMPLICIT REAL*8 (A-H,O-Z)
COMMON/COM4/ RB\nCOMMON/COM6/TR0, TR1, TR2, WPMAX
DIMENSION A(3,3), POINT(3), IPOINT(3), B(3,3), CI2M(3,3)
CALL ROTATE (CI2M, ANGTRAV, 0.0, 0.0, 0.0, 0.0)
DO I = 1, 3
  DO J = 1, 3
    A(I,J) = CI2M(I,J)
  END DO
END DO
IF (TR1 .EQ. TR0) THEN
  ROLLT = (TIME-TR0) * WPMAX
ELSE
  IF (TIME .LT. TR1) THEN
    WPT = WPMAX / (TR1-TR0) * (TIME-TR0)
    ROLLT = 0.5 * (TIME-TR0) * WPT
  ELSE IF (TIME .LT. TR2) THEN
    ROLLT = 0.5 * (TRI-TR0) * WPMAX + WPMAX * (TIME-TR1)
  ELSE
    WPT = -WPMAX / (TR1-TR0) * (TIME-TR2) + WPMAX
    ROLLT = (TR2-TR0) * WPMAX - 0.5 * (TR2+TR1-TR0-TIME) * WPT
  END IF
END IF
ROLLT = ROLL1 + ROLLT
CALL ROTATE (A, 0.0, 0.0, ROLLT, 0, 0, 1)
PSI = ATAN2 (A(1,2), A(1,1))
THT = ASIN (-A(1,3))
PHI = ATAN2 (A(2,3), A(3,3))
RETURN
END
SUBROUTINE SUNV(TIME, RA, DEC)
IMPLICIT REAL*8 (A-H, O-Z)
COMMON/COM2/PINT, TIME0
COMMON/COM3/PI, TWOPI, PIO2
DIMENSION XSUN(3)

DATA DJUL0 /2433282.5/
DATA OBL0 /0.40920621/
DATA SOBL0 /0.39788120/
DATA COBL0 /0.91436959/
DATA OBLD /-0.6198E-8/
DATA PEQD /0.6675E-6/
DATA GHA0 /1.74664770/
DATA GHADI /0.0172027918/
DATA GHADF /0.3003881/

DATA ASUN0 /6.2482947/
DATA ASUND /0.01720197/
DATA ECCS0 /0.01730108/
DATA ECCSD /-0.1148E-8/
DATA XLPS0 /4.9232341/
DATA XLPSD /0.8217E-6/

DJUL = TIME0 + TIME/86400.
DAYS = DJUL - DJUL0
OBL = OBL0 + OBLD*DAYS
SOBL = SIN(OBL)
COBL = COS(OBL)
ECCSUN = ECCS0 + ECCSD*DAYS
XLPSUN = XLPS0 + XLPSD*DAYS
A = DMOD((ASUN0 + ASUND*(DJUL-DJUL0)), TWOPI)
E = A

1 B = ECCSUN*SIN(E) - A
IF(ABS(B) .LT. 1.E-5) GO TO 5
DBDE = 1. - ECCSUN*COS(E)
E = E - B/DBDE
GO TO 1

5 TN = SQRT(1. - ECCSUN**2) * SIN(E)
TD = COS(E) - ECCSUN
F = ATAN2(TN, TD)
ANG = XLPSUN + F
SANG = SIN(ANG)
CANG = COS(ANG)
XSUN(1) = CANG
XSUN(2) = SANG*COBL
XSUN(3) = SANG*SOBL
RA = ATAN2(XSUN(2), XSUN(1))
DEC = ATAN2(XSUN(3), (XSUN(1)**2 + XSUN(2)**2)**0.5)
RETURN
END