SYSTEMS TECHNOLOGY LABORATORY SERIES

STL-79-001

INTEL 8080 ORBIT PROPAGATION PROGRAM SYSTEM DESCRIPTION AND USER'S GUIDE

APRIL 1979

JUL 1979
RECEIVED
NASA STI FACILITY
ACCESS DEPT.

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

SYSTEMS TECHNOLOGY LABORATORY SERIES STL-79-001

INTEL 8080 ORBIT PROPAGATION PROGRAM SYSTEM DESCRIPTION AND USER'S GUIDE

APRIL 1979

NASA
National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Systems Technology Laboratory (STL) is a computational research facility located at the Goddard Space Flight Center of the National Aeronautics and Space Administration (NASA/GSFC). The STL was established in 1978 to conduct research in the area of flight dynamics systems development. The laboratory consists of a VAX-11/780 and a PDP-11/70 computer system, along with an image-processing device and some microprocessors. The operation of the Laboratory is managed by NASA/GSFC (Systems Development and Analysis Branch) and is supported by SYSTEX, Inc., Computer Sciences Corporation, and General Software Corporation.

The main goal of the STL is to investigate all aspects of systems development of flight dynamics systems (software, firmware, and hardware), with the intent of achieving system reliability while reducing total system costs. The flight dynamics systems include the following: (1) attitude determination and control, (2) orbit determination and control, (3) mission analysis, (4) software engineering, and (5) systems engineering. The activities, findings, and recommendations of the STL are recorded in the Systems Technology Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/TM-79/6080.

The primary contributor to this document was

Carl Rabbin	(Computer Sciences Corporation)

Other contributors include

Keiji Tasaki	(Goddard Space Flight Center)
Charles Goorevich	(Computer Sciences Corporation)

Single copies of this document can be obtained by writing to

Keiji Tasaki
Code 582.1
NASA/GSFC
Greenbelt, Maryland 20771

ABSTRACT

This document provides a system description and user's guide for the Intel 8080 orbit propagator developed on the Tektronix 8002 microprocessor development system. The Intel 8080 orbit propagator contains a force model up to 6 by 6 and includes the effects of the Sun and Moon. The propagator was specifically designed to handle the class of geosynchronous orbits. This document has been prepared in partial fulfillment of the requirements of Task Assignment 866 of Contract NAS 5-24300.

TABLE OF CONTENTS

Section 1 - Introduction 1-1

Section 2 - Computational Models 2-1

2.1 Coordinate System 2-1
2.2 Intel Orbit Propagation Model 2-2
2.2.1 Fourth-Order Runge-Kutta Integrator (Starter) 2-2
2.2.2 Adams Moulton Predictor Corrector 2-4
2.2.3 6-by-6 Earth Geopotential Model 2-6
2.3 Miscellaneous Models 2-8
2.3.1 Julian Date 2-9
2.3.2 Greenwich Hour Angle 2-9
2.3.3 Sun and Moon Positions 2-10

Section 3 - System Overview and User's Guide 3-1

3.1 System Overview 3-1
3.2 User's Guide to Execution of Orbit Program 3-3

Section 4 - Analysis of Orbit Propagator 4-1

4.1 IBM 360 Simulations 4-1
4.1.1 Conclusions of IBM Simulations 4-5
4.2 Intel Program Execution Results 4-5

Section 5 - System Maintenance 5-1

5.1 Compiling and Taskbuilding 5-1
5.2 Current Location of Software 5-3

Appendix A - Source Code Listings

References

LIST OF ILLUSTRATIONS

Figure

3-1 System Diagram of Intel Orbit Propagator 3-2
3-2 Example of Program Input 3-5
3-3 Example of Output Program 3-6

LIST OF TABLES

Table

4-1 Elements of Test Orbits 4-2
4-2 Analysis of TDRS Type Orbits 4-3
4-3 Analysis of SMM Type Orbits 4-4

SECTION 1 - INTRODUCTION

The Intel 8080 orbit propagator was designed to handle the class of geosynchronous orbits on a microprocessor. The perturbations affecting this class of orbits are the geopotential Earth terms up to 4 by 4 and the noncentral bodies of Sun and Moon. The code to handle this problem was first developed and verified on an IBM S/360 and then transported to the Tektronix 8002 microprocessor development system wherein differences in compilers were resolved and a checkout of the software completed. The microprocessor development system will be used to verify the hardware being specially built for this problem. The developed software will then be transferred to programmable read-only memory (PROM) contained in the specially built hardware. The final result will be a small microcomputer capable of propagating synchronous orbits.

This document reflects the status of the system to the point of verifying the software on the microprocessor development system. Section 2 describes the computational models used; Section 3 includes a system overview and user's guide. Section 4 describes the analysis performed on both the IBM simulations and the Tektronix development system. Section 5 gives information on how to maintain the system.

SECTION 2 - COMPUTATIONAL MODELS

This section includes the mathematical models used in the Intel 8080 orbit propagator and descriptions of the parameters used, coordinate systems, and analytical development. The mathematical models presented are as follows:

- Coordinate system
- Orbit propagation model
- Miscellaneous models

No attempt is made herein to show how the models are integrated into the propagator. Section 3 gives the necessary information on program structure and interfaces.

2.1 COORDINATE SYSTEM

In the orbit propagation program, the inertial system will be used to represent the orbital state (Cartesian coordinates) of the propagated satellite.

The coordinates of the inertial system are defined as follows:

Origin = center of the Earth

x = axis lying in the equatorial plane of the Earth and pointing towards Aries (Υ)[1]

z = axis pointing toward the North Pole

y = axis in the equatorial plane perpendicular to the x-axis and oriented so that (x, y, z) is a right-handed system

[1] True of Date System (TOD) since Aries will be located at spacecraft epoch.

Using the inertial system, a unit vector $\bar{r}$ may be defined in the following two ways:

- Inertial coordinates (r_1, r_2, r_3) of the vector $\bar{r}$
- Right ascension, α , and declination, δ , of the vector $\bar{r}$

$$0 \leq \alpha \leq 360^{\circ};\ -90^{\circ} \leq \delta \leq +90\ (\delta > 0 \text{ if } r_3 > 0)$$

The relationship between the inertial coordinates and right ascension and declination is given by the following formulas:

$$r_1 = \cos\delta\ \cos\alpha$$

$$r_2 = \cos\delta\ \sin\alpha$$

$$r_3 = \sin\delta$$

2.2 INTEL ORBIT PROPAGATION MODEL

The following components make up the propagator:

- Fourth-order Runge-Kutta integrator (starter) with modified Fehlberg coefficients
- Adams-Moulton predictor corrector (integrator)
- Up to 6-by-6 Earth geopotential model

Integration will be performed using the Cowell technique of integrating Cartesian coordinates of acceleration and velocity in TOD.

2.2.1 Fourth-Order Runge-Kutta Integrator (Starter)

The numerical integration technique used is taken from Reference 1. The integrator is a fixed-step, fourth-order Runge-Kutta, with modified Fehlberg

coefficients. The technique is designed to optimally approximate the solution to the initial value problem

$$\frac{dx}{dt} = f(t, x), \quad x_1 = x(t_1)$$

by the relation

$$x(t_1 + h) = x_1 + h \sum_{k=1}^{4} c_k f_k$$

where

$$f_1 = f(t_1, x_1)$$

$$f_k = f\left(t_1 + h\alpha_k, x_1 + h \sum_{\lambda=1}^{k-1} \beta_{k\lambda} f_\lambda\right), \quad k > 1$$

In the above relations, x and f are vectors; h is the stepsize; α, β, and c are the coefficients defining the locations of the functional evaluations of their weights. The following coefficients are used in the four function evaluations to be performed:

K \ λ	α_K	$\beta_{K\lambda}$			c_K
		1	2	3	
1	0				1/6
2	1/4	1/4			0
3	1/2	0	1/2		2/3
4	1	1	-2	2	1/6

2.2.2 Adams Moulton Predictor Corrector

The Intel orbit propagator uses the Adams-Bashford, Adams-Moulton (ABAM) predictor-corrector pair of numerical integrators. These integrators are part of the family of multistep methods used for solving initial value problems of ordinary differential equations. The problem is to find the function $y(x)$ satisfying $dy/dx = f(x, y)$, $y(x_o) = y_o$, on an interval containing x_o . The following equations are used for the integrator:

$$\text{(AB)} \quad \tilde{y}_p - \tilde{y}_{p-1} = \frac{h}{1440} \cdot (4277\, f_{p-1} - 7923\, f_{p-2} + 9982\, f_{p-3} - 7298\, f_{p-4} + 2877\, f_{p-5} - 475\, f_{p-6}) \tag{2-1}$$

$$\text{(AM)} \quad \tilde{y}_p - \tilde{y}_{p-1} = \frac{h}{1440} \cdot (475\, f_p + 1427\, f_{p-1} - 798\, f_{p-2} + 482\, f_{p-3} - 173\, f_{p-4} + 27\, f_{p-5}) \tag{2-2}$$

where h = stepsize between grid points x_q , x_{q+1}

$f_q = f(x_q, y(x_q))$ ($\tilde{y}(x_q)$ will usually replace $y(x_q)$)

$\tilde{y}_r$ = numerically computed value of solution at x_r (as opposed to exact value $y_r = y(x_r)$)

It should be noted that the starting values $f_{p-6}, \ldots, f_{p-1}$ must be obtained for the first use of the ABAM integrator. This is done in the Intel orbit propagator by (single-step) numerical integration with the Runge-Kutta-Fehlberg integrator described in the previous section.

The basic flow of operations is as follows. A new solution point, y_p , is predicted by Equation (2-1) as $\tilde{y}$. This value may then be used in computing $f_p = f(x_p, \tilde{y}_p)$, and a new corrected value of $\tilde{y}_p$ is computed as $\tilde{y}_p$ in Equation (2-2). This correction may be done more than once until a small difference between successive corrected values is obtained.

The ABAM predictor-corrector is used for numerically solving for the six Cartesian coordinates describing the satellite position and velocity. The three equations for the velocity are of the form

$$\ddot{x} = f(x, \dot{x}) \tag{2-3}$$

and the three equations for position are of the form

$$\dot{x} = g(x) \tag{2-4}$$

It should be noted that the general form of Equation (2-3) allows for the simple addition of perturbing effects (Earth nonsphericity, solar and lunar gravitational effects, etc.) onto the right-hand side of the equation. This technique places this integration scheme in the Cowell family of propagators.

The AB integrator herein is of the sixth order, and the AM is of the fifth. The following observations should be made for implementation of the ABAM pair, especially if accuracy problems are encountered:

- The coefficients for Equations (2-1) and (2-2) should be expressed as fractions (i.e., divided by 1440 first). This prevents a loss of digits which may occur when multiplying by a large coefficient and then dividing by a quantity within an order of magnitude of the coefficient.
- Arrange terms with the resulting reduced coefficients so that they will be summed in ascending order of the magnitudes of the corresponding coefficient. This will often prevent the loss of the digits of least significance, due to the shifting of digits during floating-point addition.

For further information concerning the derivation and error analyses of the Adams-Bashforth, Adams-Moulton predictor-corrector, Reference 2 should be consulted.

2.2.3 6-by-6 Earth Geopotential Model

Most solar system bodies are known to have figures which depart from the spherical model of the particle. The nonsphericity of the gravitational potential may give rise to a significant perturbation of satellite trajectories. Therefore, accurate orbit determination may require the inclusion of nonspherical terms. The model used in the target orbit propagator will contain up to a 6-by-6 Earth model, which will be modeled as described in Reference 3. The gravitational potential of the central planet in terms of spherical harmonics is expressed as

$$\psi(r, \phi, \lambda) = \frac{\mu}{r} + \frac{\mu}{r} \sum_{n=1}^{\infty} C_n^0 \left(\frac{R_e}{r}\right)^n P_n^0(\sin \phi)$$

$$+ \frac{\mu}{r} \sum_{n=1}^{\infty} \sum_{m=1}^{n} \left(\frac{R_e}{r}\right)^n P_n^m(\sin \phi) \left(S_n^m \sin m\lambda + C_n^m \cos m\lambda\right)$$

where

r = magnitude of the vector from the body's center of mass to the satellite

ϕ = the geocentric latitude

λ = the geocentric longitude (measured east from the prime meridian)

μ = the gravitational parameter of the central body (μ for the Earth = 398600.8 km^3/sec^2)

R_e = radius of the Earth (6378.14 km)

P_n^m = the associated Legendre function

S_n^m, C_n^m = harmonic coefficients (i.e., Zonal harmonics for $m = 0$; Sectorial harmonics for $m = n$; Tesseral harmonics for $n > m \neq 0$) (Note: $J_n = -C_n^0$)

The first term is the point mass potential for Keplerian motion, and the second and third terms are the nonspherical potential due to the sum of zonal and tesseral harmonics, respectively. The expansion to $n = m = 6$ is commonly refered to as a 6.6y - 6 Earth model.

The Legendre functions and the terms $\cos m\lambda$ and $\sin m\lambda$ are computed via recursion formulas as follows:

$$P_n^0(\sin \phi) = [(2n - 1) \sin \phi \, P_{n-1}^0(\sin \phi) - (n - 1) \, P_{n-2}^0(\sin \phi)]/n$$

$$P_n^m(\sin \phi) = P_{n-2}^m(\sin \phi) + (2n - 1) \cos \phi \, P_{n-1}^{m-1}(\sin \phi) \quad m \neq 0, \; m < n$$

$$P_n^n(\sin \phi) = (2n - 1) \cos \phi \, P_{n-1}^{n-1}(\sin \phi) \quad m \neq 0, \; m = n$$

where

$$P_0^0(\sin \phi) = 1$$

$$P_1^0(\sin \phi) = \sin \phi$$

$$P_1^1(\sin \phi) = \cos \phi$$

$$\sin m\lambda = 2 \cos \lambda \sin(m - 1) \lambda - \sin(m - 2) \lambda$$

$$\cos m\lambda = 2 \cos \lambda \cos(m - 1) \lambda - \cos(m - 2) \lambda$$

The spherical harmonic coefficients used to represent the mass distribution of the Earth are as follows:

Zonals	Sectorials and Tesserals	
$J2 = 0.108265 \times 10^{-2}$	$C21 = -0.1326739 \times 10^{-8}$	$S21 = -0.1374346 \times 10^{-7}$
$J3 = -.254503 \times 10^{-5}$	$C22 = 0.1566511 \times 10^{-5}$	$S22 = -0.8869932 \times 10^{-6}$
$J4 = -.1671499 \times 10^{-5}$	$C31 = 0.2161875 \times 10^{-5}$	$S31 = .2571596 \times 10^{-6}$
$J5 = -.2067209 \times 10^{-6}$	$C32 = 0.3172142 \times 10^{-6}$	$S32 = 0.2078203 \times 10^{-6}$
$J6 = 0.64006172 \times 10^{-6}$	$C33 = 0.1025055 \times 10^{-6}$	$S33 = 0.1949036 \times 10^{-6}$
	$C41 = -0.5052256 \times 10^{-6}$	$S41 = 0.4199062 \times 10^{-6}$
	$C42 = 0.7739965 \times 10^{-7}$	$S42 = 0.1515418 \times 10^{-6}$
	$C43 = 0.5901404 \times 10^{-7}$	$S43 = -.1275373 \times 10^{-7}$
	$C44 = -.3608512 \times 10^{-8}$	$S44 = .6386659 \times 10^{-8}$
	$C51 = -.51413652 \times 10^{-7}$	$S51 = -.84962448 \times 10^{-7}$
	$C52 = .10362962 \times 10^{-6}$	$S52 = -.49163518 \times 10^{-7}$
	$C53 = -.13284881 \times 10^{-7}$	$S53 = -.82141685 \times 10^{-8}$
	$C54 = -.22939341 \times 10^{-8}$	$S54 = .27915369 \times 10^{-9}$
	$C55 = .32954778 \times 10^{-9}$	$S55 = -.16403877 \times 10^{-8}$
	$C61 = -.62776599 \times 10^{-7}$	$S61 = -.24097526 \times 10^{-7}$
	$C62 = .58964246 \times 10^{-8}$	$S62 = -.43267643 \times 10^{-7}$
	$C63 = .75677410 \times 10^{-9}$	$S63 = -.1250492 \times 10^{-9}$
	$C64 = -.36861988 \times 10^{-9}$	$S64 = -.17000671 \times 10^{-8}$
	$C65 = -.21086437 \times 10^{-9}$	$S65 = -.42858549 \times 10^{-9}$
	$C66 = .64295796 \times 10^{-11}$	$S66 = -.56608237 \times 10^{-10}$

2.3 MISCELLANEOUS MODELS

This section presents the following miscellaneous models used by the orbit propagator:

- Julian date
- Greenwich hour angle
- Sun and Moon positions

2.3.1 Julian Date

The base times maintained in the program include

- T--time in seconds since epoch
- DJO--modified Julian date (days since 2400000) in universal time (UT)
- EJO--modified Julian date (days since 2400000) in ephemeris time (ET)

The epoch time of the satellite will be input in terms of year, month, day, hour, minute, and second (the second will be the smallest unit of input). The input will then be transferred to the modified Julian dates (DJO, EJO).

The relationship between DJO and EJO is as follows:

$$EJO = DJO + (38.66 + 2.5921E - 03\ (DJO - 40000.0))/86400.0$$

2.3.2 Greenwich Hour Angle

The Greenwich hour angle is calculated by the following equation:

$$GHA = 100^{\circ}.075542 + 0^{\circ}.98564735(\delta) + (2^{\circ}.9015)\ 10^{-13}(\delta)^2$$
$$+ W_e(t) + \text{(nutation in right ascension)}$$

W_e is the angular velocity (deg/sec) of the Earth and is given by

$$W_e = .0041780742/(1. + (5.21)\ 10^{-3}(\delta))$$

where δ = the number of whole days since 1950

t = the fraction of days (i.e., modified Julian date $33282.5 = \delta + t$)

For the microprocessor application, the nutation in right ascension will not be modeled.

2.3.3 Sun and Moon Positions

The formulas used in the Sun and Moon positions were obtained from the Explanatory Supplement to the Astronomical Ephemeris (Reference 4).

SECTION 3 - SYSTEM OVERVIEW AND USER'S GUIDE

This section presents the system overview and user's guide. Section 3.1 gives the baseline diagram of the Intel orbit propagator and a functional description of each module. The subroutine descriptions are given in the prolog to each routine (Appendix A). Section 3.2 gives user instructions on how to run the Intel orbit propagator on the microprocessor Tektronix 8002 development system.

3.1 SYSTEM OVERVIEW

Figure 3-1 gives the baseline diagram of the Intel orbit propagator system. Each module in the baseline diagram performs the following functions:

- Main Driver — Reads input and controls calls to DATE, ORBIT, and GHAX
- DATE — Calculates modified Julian date from year, month, day, hours, minutes, and seconds
- ORBIT — Performs integration (Runge-Kutta or Adams-Moulton)
- GHAX — Calculates Greenwich hour angle from Julian date
- ACCEL — Controls calls to perturbation models and sums accelerations
- SOL — Calculates position of the Sun
- LUNA — Calculates position of the Moon
- SPART — Calculates Earth geopotential field (up to 6 by 6)

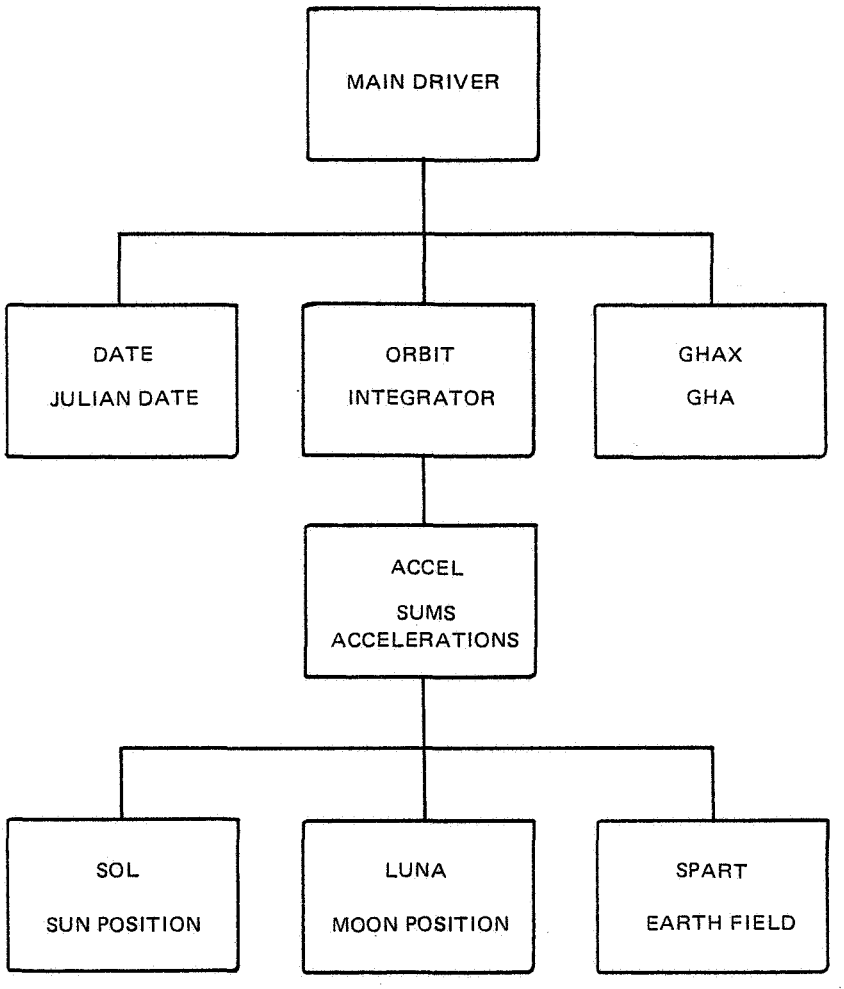

Figure 3-1. System Diagram of Intel Orbit Propagator

3.2 USER'S GUIDE TO EXECUTION OF ORBIT PROGRAM

The orbit program was developed and tested on the microprocessor Tektronix 8002 development system, an Intel 8080-based system. References 5 and 6 provide information on its use.

Executing the orbit program requires using a system disk in Disk Drive 0 and a disk containing the orbit object module ORBH in Drive 1. (Section 5.2 describes where these disks can be found.) The reset switch is used to enter the monitor, if the system is not in the monitor program already.

A series of commands is used to assign logical device unit numbers, load the object module into memory, and start program execution. These commands are all in the command file FORTLG0. To use this command file, the user should type after the prompt (↓ indicates carriage return).

```
≥FORTLG0  ORBH/1  CONO 2000↓
```

FORTLG0 is a command file on the system disk in Drive 0 allowing the user to specify certain arguments in the commands. In FORTLG0, the first argument is the object module to be loaded; the second is the device for output; the third is the starting address for the object module being used. The second argument is variable. It may be CONO, meaning output to the console (terminal), LPT1 for the line printer, or any other name for a disk file. If a /1 is appended to this disk file name, the file will be on the disk in Disk Drive 1; otherwise, it will be on Disk Drive 0. (Another reserved device name is CONI, meaning input from the console; however, this does not apply here.)

If a disk output file is being created and a file with that name is already on the disk, a modified version of the name will be used until the end of the program. Then, the old file will be deleted and the new file assigned the name. If, for example, the old file is ABC/1, the new one will be temporarily named *BC/1 and at the end it will be renamed ABC/1 and the old one deleted. This must

be known if the program is aborted because then the old file will not have been deleted. To abort the program, the user should press the ESC key twice. When the prompt >> appears, the user should type (after prompt)

```
>>ABO FORTLG0↓
```

This will abort whatever is being run and suppress continued reading of the command file FORTLG0.

The actual program begins by requesting data as shown in Figure 3-2. The underlined numbers are for the sample run generating the output of Figure 3-3. It should be noted that single- and double-precision floating-point numbers and integers must be entered as in the example. The data to be input is as follows:

```
INITIAL DATE (EPDATE(1→3)):  YEAR, MONTH, DAY (SINGLE-PRECISION
FLOATING POINT)
INITIAL TIME (EPDATE (4→6)):  HOUR, MINUTE, SECOND (SINGLE-PRECISION
FLOATING POINT)
INITIAL POSITION COORDINATES X, Y, Z(EPELEM(1→3):  (DOUBLE-PRECISION
FLOATING POINT)
INITIAL VELOCITY COORDINATES Ẋ, Ẏ, Ż(EPELEM 4→6):  (DOUBLE-PRECISION
FLOATING POINT)
TC:  DURATION OF ORBIT RUN (DOUBLE-PRECISION FLOATING POINT)
DT:  INTEGRATION STEPSIZE (DOUBLE-PRECISION FLOATING POINT)
PRINT:  INTERVAL FOR PRINTING DATA MUST BE AN INTEGRAL MULTIPLE
OF DT (DOUBLE-PRECISION FLOATING POINT)
ITYP = 0 FOR RUNGE-KUTTA INTEGRATION (INTEGER)
     = 1 FOR ADAMS-MOULTON INTEGRATION
MMAX:  MAXIMUM NUMBER OF ZONALS RANGES FROM 2 TO 6 BUT IS USUALLY 4
NMAX:  MAXIMUM NUMBER OF TESERRALS RANGES FROM 0 TO 6 BUT IS
       USUALLY 4
INC: = 0 IF NONCENTRAL BODY EFFECTS ARE IGNORED
     = 1 IF NONCENTRAL BODY (SUN AND MOON) EFFECTS ARE INCLUDED
CONVG:  CONVERGENCE TEST VALUE FOR ADAMS-MOULTON INTEGRATION
        (DOUBLE-PRECISION FLOATING POINT)
```

Output is of the form shown in Figure 3-3. First, the input is displayed for user verification. Next, the epoch Julian date and the Greenwich hour angle are displayed. Finally, a time history of position and velocity vectors at the requested time interval is displayed.

```
INPUT 3 NUMBERS FOR THE DATE (EPDATE(1->3)):
79.,10.,15.

INPUT 3 NUMBERS FOR THE TIME (EPDATE(4->6)):
0.,0.,0.

INPUT THE 3 POS. COORDINATES (EPELEM(1->3)):
13511.353170D0,-39737.344140D0,3876.289820D0

INPUT THE 3 VEL. COORDINATES (EPELEM(4->6)):
2.895251652D0,1.008673428D0,.2459405166D0

INPUT:   TC,DT,PRINT:
43.D5,430.D0,860.D0

INPUT:  ITYP,NMAX,MMAX,INC,CONVG:
1,4,4,1,1.D-12
```

Figure 3-2. Example of Program Input

INTEL ORBIT PROPAGATOR

EPOCH DATE: YR: 79. MO: 10. DAY: 15.
TIME: HR: 0. MN: 0. SEC: 0.

EPOCH ELEMENTS - CARTESIAN INERTIAL (TOE)

POSITION COMPUTED:

X	13511.353170 KM
Y	-39737.344140 KM
Z	3876.289820 KM

VELOCITY COMPONENTS -

XDOT	2.8952516520 KM/SEC
YDOT	1.0086734280 KM/SEC
ZDOT	.2459405166 KM/SEC

EPOCH JULIAN DATE: 2444161.5000000000

EPOCH GHA: .4002580047

TIME FROM EPOCH IN SEC:		INERTIAL POS. AND VEL. COORD.(KM): X / XDOT	Y / YDOT	Z / ZDOT
1)	860.0	15973.0489034673	-38792.2594971057	4080.0307242064
		2.8277458109	1.1884778447	.2277202074
2)	1720.0	18371.8833099660	-37694.5035218538	4267.7114093685
		2.7491104476	1.3636065321	.2086034233
3)	2580.0	20698.4150446494	-36448.3954979897	4438.5930510048
		2.6596552127	1.5333694015	.1886654831
4)	3440.0	22943.4884281991	-35058.8387006810	4592.0030759429
		2.5597325306	1.6970976333	.1679849486
5)	4300.0	25098.2668534521	-33531.3026793129	4727.3376271365
		2.4497360385	1.8541459964	.1466433159
6)	5160.0	27154.2705041058	-31871.8001206638	4844.0641541561
		2.3300992671	2.0038957882	.1247246945
7)	6020.0	29103.4088806768	-30086.8635782616	4941.7234431986
		2.2012938042	2.1457571441	.1023154704
8)	6880.0	30938.0127185963	-28183.5196648552	5019.9314266027
		2.0638274088	2.2791713654	.0795039622
9)	7740.0	32650.8642684802	-26169.2612697987	5078.3806968479
		1.9182420047	2.4036131417	.0563800715
10)	8600.0	34235.2257624085	-24052.0179406904	5116.8417148546
		1.7651115310	2.5185926323	.0330349258
11)	9460.0	35684.8659732305	-21840.1245364036	5135.1637092752
		1.6050396600	2.6236574040	.0095605178
12)	10320.0	36994.0847611765	-19542.2882775127	5133.2752632781
		1.4386573969	2.7183942175	-.0139506608
13)	11180.0	38157.7355090577	-17167.5543273419	5111.1845865037
		1.2666205718	2.8024306567	-.0374059852
14)	12040.0	39171.2453571519	-14725.2700420363	5068.9794712390
		1.0896072322	2.8754365938	-.0607130653
15)	12900.0	40030.6331579874	-12225.0480333685	5006.8269331517
		.9083149498	2.9371254846	-.0837801116
16)	13760.0	40732.5250806132	-9676.7281926848	4924.9725382131
		.7234580495	2.9872554880	-.1065162988

Figure 3-3. Example of Output Program

If output is sent to a disk file (e.g., ABC/1), this file can be written to the line printer or terminal using the COPY program. That is,

```
≥COPY  ABC/1  LPT1↓
≥COPY  ABC/1  CONO↓
```

SECTION 4 - ANALYSIS OF ORBIT PROPAGATOR

Section 4.1 and 4.2 describe the analysis performed on the IBM and Tektronix systems, respectively. Results show that the models developed compare, for synchronous orbits, with the Goddard Mission Analysis System (GMAS).

4.1 IBM 360 SIMULATIONS

The Intel orbit propogator described in this document was implemented in FORTRAN on the M&DO IBM S/360-95 computer so that timing and accuracy studies could be carried out against established orbit propagators. A switch was included in the Intel propagator to allow runs to be made using the Runge-Kutta integrator alone, or using the Adams-Moulton integrator after the initial steps had been obtained from the Runge-Kutta. Two satellite orbits were used in the study: a high altitude synchronous orbit for the TDRS satellite, and a circular moderately low-altitude orbit representative of the SMM satellite. The elements of these orbits are given in Table 4-1. Comparison orbits were generated using the GMAS Cowell propagator with a 4-by-4 Earth potential field. For the TDRS orbit, the effects of the Moon and the Sun were included; whereas, for the SMM orbit, the Moon and the Sun were not included, but the Harris-Priester drag option was turned on. The Cartesian true-of-date elements at epoch resulting from these runs were used as input to the Intel propagator model. Orbits were generated by the (360) Intel model using both the Runge-Kutta and Adams-Moulton options, employing the same force options as the GMAS run (4-by-4 Earth field, Moon-Sun for TDRS, drag for SMM). The comparisons of these orbits with the results from the GMAS Cowell integrator are given in Tables 4-2 and 4-3.

Table 4-1. Elements of Test Orbits

EPOCH KEPLERIAN OSCULATING ELEMENTS	TDRS	SMM
a	42166.75 KM	6933.740 KM
e	0.0004	0.001
i	7.0 DEG	33.0 DEG
Ω	240.0 DEG	270.0 DEG
ω	60.0 DEG	90.0 DEG
M	349.0 DEG	0.0 DEG
EPOCH	OCTOBER 15, 1979 $0^h\ 0^m\ 0.^s0$ UT	OCTOBER 15, 1979 $0^h\ 0^m\ 0.^s0$ UT

Table 4-2. Analysis[a] of TDRS Type Orbits

STEP (SEC)	$\lvert\Delta x\rvert$ (R-K)	$\lvert\Delta x\rvert$ (A-M)	$\Delta\alpha$ (R-K)	$\Delta\alpha$ (A-M)
ERROR AT 5 DAYS (4.3×10^5 SEC)				
$215.0\left(\frac{P^b}{400}\right)$	–	6.52×10^{-2}	–	8.9×10^{-5}
$430.0\left(\frac{P}{200}\right)$	2.26×10^{-2}	0.216	3.1×10^{-5}	3.0×10^{-4}
$860.0\left(\frac{P}{100}\right)$	0.543	2.62	-7.4×10^{-4}	3.6×10^{-3}
$1720.0\left(\frac{P}{50}\right)$	11.6	39.8	-1.6×10^{-2}	5.4×10^{-2}

STEP (SEC)	CPU (MIN) R-K	CPU (MIN) A-M	$\lvert\Delta x\rvert$ (R-K)	$\lvert\Delta x\rvert$ (A-M)	$\Delta\alpha$ (R-K)	$\Delta\alpha$ (A-M)
ERRORS AT 50 DAYS (4.3×10^6 SEC)						
$215.0\left(\frac{P}{400}\right)$	–	0.572	–	1.04	–	1.4×10^{-3}
$430.0\left(\frac{P}{200}\right)$	0.977	0.312	0.269	2.51	-3.5×10^{-4}	3.4×10^{-3}
$860.0\left(\frac{P}{100}\right)$	0.492	0.170	16.6	25.0	-2.2×10^{-2}	3.4×10^{-2}
$1720.0\left(\frac{P}{50}\right)$	0.254	0.091	484	259	–0.65	0.35

[a]TEST CASE SETUP:
ORBIT: INTEGRATED FOR APPROXIMATELY 50 DAYS (4.3×10^6 SECONDS); INCLUDES A 4-BY-4 EARTH FIELD, MOON/SUN, NO DRAG.

COMPARISON ORBIT: GMAS COWELL; INCLUDES A 4-BY-4 EARTH FIELD, MOON/SUN, NO DRAG.

ERROR MEASUREMENT: POSITION ERRORS ARE IN RMS KILOMETERS; RIGHT ASCENSION ERRORS IN DEGREES.

[b]P IS THE PERIOD OF ORBIT.

Table 4-3. Analysis[a] of SMM Type Orbits

STEP (SEC)	$\lvert\Delta x\rvert$ (R-K)	$\lvert\Delta x\rvert$ (A-M)	$\Delta\alpha$ (R-K)	$\Delta\alpha$ (A-M)
ERRORS AT ~ 1 PERIOD (5700 SEC)				
$10.0\left(\frac{P}{570}\right)$	0.018	0.018	1.6×10^{-4}	1.6×10^{-4}
$25.0\left(\frac{P}{228}\right)$	0.017	0.021	1.6×10^{-4}	1.9×10^{-4}
$100.0\left(\frac{P}{57}\right)$	0.140	0.820	1.4×10^{-3}	8.1×10^{-3}

STEP (SEC)	CPU (MIN) R-K	CPU (MIN) A-M	$\lvert\Delta x\rvert$ (R-K)	$\lvert\Delta x\rvert$ (A-M)	$\Delta\alpha$ (R-K)	$\Delta\alpha$ (A-M)
ERRORS AT ~20 PERIODS (1.14×10^5 SEC)						
$10.0\left(\frac{P}{570}\right)$	0.755	0.232	9.86	9.87	8.1×10^{-2}	8.1×10^{-2}
$25.0\left(\frac{P}{228}\right)$	0.315	0.103	9.85	9.92	8.1×10^{-2}	8.1×10^{-2}
$100.0\left(\frac{P}{57}\right)$	0.081	0.037	1.53	24.0	1.2×10^{-2}	0.197

[a]TEST CASE SETUP:

ORBIT: INTEGRATED FOR APPROXIMATELY 20 PERIODS (1.14×10^5 SECONDS); INCLUDES A 4-BY-4 EARTH FIELD, DRAG, NO MOON/SUN.

COMPARISON ORBIT: GMAS COWELL; INCLUDES A 4-BY-4 EARTH FIELD, HARRIS-PRIESTER DRAG, NO MOON/SUN.

ERROR MEASUREMENT: POSITION ERRORS ARE RMS KILOMETERS; RIGHT ASCENSION ERRORS IN DEGREES.

4.1.1 Conclusions of IBM Simulations

The following general conclusions can be drawn based on the results of Tables 4-2 and 4-3:

- The accuracy obtained with the Intel orbit propagator is a strong function of the stepsize. In comparison, tests run with the GMAS Cowel propagator which varied the stepsize through a considerable range showed very little effect on the results for these two orbits.
- The optimum stepsize for both cases appears to be about 1/200 of the orbital period. Longer stepsizes show a rapid decrease in accuracy, whereas shorter steps indicate that the point of diminishing returns has been reached.
- The Adams-Moulton integrator is usually less accurate than the Runge-Kutta. At the optimum stepsize, this amounts to a factor of 10 in Table 4-2, while nearly identical errors are obtained in Table 4-3. Attempts to force the accuracy of the Adams-Moulton to that of the Runge-Kutta by lowering the stepsize results in the diminishing returns noted above.
- At the optimum stepsize, the Adams-Moulton integrator is about three times faster than the Runge-Kutta integrator. It should be noted that at the same stepsize in Table 4-2, the GMAS Cowell integrator is faster than the Adams-Moulton. This is probably due to the calculations involved in the Moon-Sun routines compared with the ephemeris file used by GMAS. The GMAS Cowell is, of course, considerably larger.

4.2 INTEL PROGRAM EXECUTION RESULTS

A representative orbit propagation run (TDRS of IBM simulations) was performed on a Tektronix 8002 microcomputer using an Intel 8080 chip. The same

input data was used for a run on an IBM S/360-95 using similar FORTRAN code. The Intel program, whose input and output appears in Figures 3-1 and 3-2, used an Adams-Moulton integration procedure after a few initial points were obtained from a Runge-Kutta starter. Sun and Moon noncentral body effects were included, and output appeared every second integration step.

The wall clock timing of the run was as follows: The first four to five lines of data appeared after intervals of 1 minute or more each (Runge-Kutta starter). Soon, however, the computations began being printed at a regular interval of 40 to 45 seconds (Adams-Moulton integrator). Because the printouts occurred at every second point, the time of each step's computations was approximately 20 to 23 seconds. Because each step was equivalent to an orbit time of 430 seconds, 1 day's data (86,400 seconds) required approximately 60 minutes of computations. When the Sun and Moon computations were not added, the time required for computation of a point was approximately halved.

The storage requirements of the orbit propagation system are approximately 6000 Hex words or 24K of memory, of which 20K could be put on Programmable Read-Only Memory (PROM) and 4K would be the variable data Random Access Memory (RAM) area. The current test configuration is all in RAM with the program and constants going from 2000 (Hex) to 6024 (Hex) and the variables being stored down from 1FFF (Hex) to 102E (Hex).

A comparison of Tektronix data with IBM simulated data has a very slight but increasing difference in value for each point printed.

The difference after 1 day (86,400 seconds) is as follows:

Cartesian Coordinates	IBM Simulation	Textronix	Delta (TEK-IBM)
X (km)	13025.5060	13025.4643	-0.042
Y (km)	-39903.4974	-39903.5328	-0.035
Z (km)	3836.4665	3836.4635	-0.003

Cartesian Coordinates	IBM Simulation	Textronix	Delta (TEK-IBM)
XDOT (km/sec)	2.90705696	2.90705713	0.0000002
YDOT (km/sec)	0.97318158399	0.973177478	-0.0000041
ZDOT (km/sec)	0.2491992389	0.2491995403	0.0000003

SECTION 5 - SYSTEM MAINTENANCE

All programs for the orbit propagation system are written in FORTRAN and are stored as separate modules on Disk 1 (Section 5.2 describes obtaining this disk). The EDIT program should be used for maintenance and a /1 should be used after every filename to denote Disk Drive 1. The system disk should be in Disk Drive 0. The source code filenames are

BGNNG	COS	GHAX
ABS	ATAN2	LUNA
DMOD	SPART	ACCEL
SQRT	SOL	ORBIT
SIN	DATE	MAINDR

5.1 COMPILING AND TASKBUILDING

The FORTRAN compiler requires the set of routines that are to be linked together to be compiled from one large file. Hence, after editing a command file, ORBG/1 must be requested before compiling the Orbit system. ORBG copies each module into one large source file called ORB/1. (Actually it generates and deletes three intermediate files because all the module names can not fit on one line.) To request ORBG Drive 1, the following line should be typed after the prompt:

≥ ORBG/1↓

To compile the orbit system, command file may be used on the system disk, FORTB, which compiles the source file ORB/1 into a file called ORBH/1. The line to be typed is

≥ FORTB ORB/1 5↓

The 5 is merely one of five choices (1, 2, 3, 4, 5) of output as follows:

1. Not used
2. Object file (no source listing)
3. Not used
4. No object file (source listing)
5. Object file (source listing)

In FORTB, the source listing is output to CONO (console-teletype terminal). The five output options apply to all command files for compiling. Some other command files on the system disk are FORTC which generates object file ALLH/1 and a source listing to CONO and FORTP, which generates object file ALLH/1 and a source listing to LPT1 (line printer). By examining these command files through the use of the utility COPY, there should be little difficulty inventing other useful command files as needed.

To execute ORBH (or any other program), a load-and-go command file is needed. A useful one on the system disk is called FORTLG0. This file loads the program requested, assigns output to the device named in the request (to logical unit 2), and executes the program at its designated beginning. The beginning of the program is known from the first lines of the source listing, which are contained initially in file BGNNG/1. This states that the executable code begins at 2000 (Hex) and increases in core. The variable data is in core from IFFF (Hex) decreasing in core (see source listing). To run ORBH/1, the user should type

≥FORTLG0 ORBH/1 LPT1 2000↓

for output to the line printer. Output to the teletype would have CONO in place of LPT1. Output to a disk file would require entry of a name (e.g., XYZ/1).

In the source code of any FORTRAN program, four logical devices are available for I/O. Read (1) implies reading from logical Device 0 in the load-and-go-file which should be CONI (console input). In the FORTRAN code write (1, format statement number) implies writing to CONO. Writing to Device 2 is writing to the device named in FORTLG0's argument list. Device 3 is the third argument in a command file called FORTLG3. In all cases, the starting address is the final argument.

In the command file, it should be noted that the $1 is where the first argument goes, $2 the second, etc. Different combinations may be made as desired. Command files may also be put on Disk Drive 1. The command files already described will usually be enough for this system.

One problem occurs often enough to be mentioned herein. The disks often run out of directory space long before they run out of disk space. Whenever a file can't be created, a few deletions using DEL correct this difficulty.

5.2 CURRENT LOCATION OF SOFTWARE

In the microprocessor laboratory (Building 23, Room W-426) there is a binder of four floppy disks containing all necessary files for the Orbit Processor system. The first disk is the system disk normally used in Disk Drive 0. The second disk is the current version of the Orbit program and is the one usually used for execution and for program modification. The third disk is a backup disk containing the most recent working version of the source. The fourth disk contains the original source code obtained from the IBM S/360. To carefully update the backup disk, the DUP command may be used following the technique described for DUP in Reference 5.

APPENDIX A - SOURCE CODE LISTINGS

This appendix contains source code listings for the Intel 8080 orbit propagation program. The individual routines can be located as follows:

MAIN	A-2
DATE	A-6
GHAX	A-7
ORBIT	A-8
ACCEL	A-13
SPART	A-16
SOL	A-20
LUNA	A-22

The following support routines are also given:

ABS	A-24
DMOD	A-25
SQRT	A-26
SIN	A-27
COS	A-28
ATAN2	A-29

UTI FORT//80 COMPILER 3.2G

```
51F5 C      PURPOSE:
51F5 C         MAIN ROUTINE FOR SMALL ORBIT PROPAGATOR
51F5 C
51F5 C      METHOD:
51F5 C         1.  INPUT DATA
51F5 C         2.  CALCULATE MODIFIED JULIAN DATE AND GREENWICH HOUR
51F5 C             HOUR ANGLE AT EPOCH
51F5 C         3.  CALL SUBROUTINE ORBIT TO DO PROPAGATION
51F5 C         4.  PRINT RESULTS
51F5 C
51F5 C      USAGE:
51F5 C         INPUT DATA
51F5 C
51F5 C      NAME          TYPE      DESCRIPTION
51F5 C      ----          ----      -----------
51F5 C
51F5 C      EPDATE         R*4       EPOCH DATE -YR(79),MO(1-12),DAY(1-31),
51F5 C                                         HOUR(0-23),MIN(0-59),SEC(0-59)
51F5 C      EPELEM         R*8       EPOCH ELEMENTS - X,Y,Z,XDOT,YDOT,ZDOT
51F5 C                                            (KM AND KM/SEC)
51F5 C      TC             R*8       TIME IN SEC UNTIL END OF PROPAGATION
51F5 C      DT             R*8       INTEGRTION STEP SIZE
51F5 C      NMAX           I*2       MAX ZONALS (USUALLY 4)
51F5 C      MMAX           I*2       MAX TESSERALS (USUALLY 4)
51F5 C      PRINT          R*8       PRINT FREQUENCY OF INTERMEDIATE PRINTOUT
51F5 C      CONVG          R*8       CONVERGE FACTOR FOR ADAMS-MOULTON INTEGRATOR
51F5 C      ITYP           I*2       INTEGRATION METHOD
51F5 C                                =0, RUNGE-KUTTA
51F5 C                                -=1, ADAMS-MOULTON
51F5 C      INC            I*2       NON CENTRAL BODIES
51F5 C                                =0, NO NON-CENTRAL BODIES
51F5 C                                =1, INCLUDE SUN AND MOO N
51F5 C
51F5 C      SUBROUTINES CALLED:  DATE,GHAX,ORBIT
51F5 C
51F5 C      PROGRAMMER:  G. SNYDER
51F5 C      MODIFIED FOR INTEL BY:  C. RABBIN
51F5 C
51F5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
51F5 C
51F5        REAL*8 QA,QB,QC
51F5        REAL CS(48)
51F5        REAL*8 DJO,EJO,QJO
51F5        REAL*8 CONVG
51F5        REAL EPDATE(6)
51F5        REAL*8 EPELEM(6),PRELEM(6)
51F5 C
51F5 C
51F5        CS(1)=0.0
5201        CS(2)=-.1082650E-02
520F        CS(3)=0.2545030E-05
521C        CS(4)=0.1671500E-05
```

```
5229        CS(5)=0.2067210E-06
5236        CS(6)=-.6400620E-06
5244 C
5244        CS(7)=0.0E0
5251        CS(8)=-.1326739E-08
525F        CS(9)=0.2161875E-05
526C        CS(10)=-.5052256E-06
527A        CS(11)=-.5141365E-07
5288        CS(12)=-.627766E-07
5296 C
5296        CS(13)=-.5660824E-10
52A4        CS(14)=0.1566511E-05
52B1        CS(15)=0.3172142E-06
52BE        CS(16)=0.7739965E-07
52CB        CS(17)=0.1036296E-06
52D8        CS(18)=0.5896425E-08
52E5 C
52E5        CS(19)=-.4285855E-09
52F3        CS(20)=-.1640388E-08
5301        CS(21)=0.1025055E-06
530E        CS(22)=0.5901404E-07
531B        CS(23)=-.1328488E-07
5329        CS(24)=0.7567741E-09
5336 C
5336        CS(25)=-.1700067E-08
5344        CS(26)=0.2791537E-09
5351        CS(27)=0.6386659E-08
535E        CS(28)=-.3608512E-08
536C        CS(29)=-.2293934E-08
537A        CS(30)=-.3686199E-09
5388 C
5388        CS(31)=-.1250492E-09
5396        CS(32)=-.8214169E-08
53A4        CS(33)=-.1275373E-07
53B2        CS(34)=0.1949036E-06
53BF        CS(35)=0.3295478E-09
53CC        CS(36)=-.2108644E-09
53DA C
53DA        CS(37)=-.4326764E-07
53E8        CS(38)=-.4916352E-07
53F6        CS(39)=0.1515418E-06
5403        CS(40)=-.2078203E-06
5411        CS(41)=-.8869932E-06
541F        CS(42)=0.642958E-11
542C C
542C        CS(43)=-.2409753E-07
543A        CS(44)=-.8496245E-07
5448        CS(45)=-.4199062E-06
5456        CS(46)=0.2571596E-06
5463        CS(47)=-.1374346E-07
5471        CS(48)=0.0E0
547E C
```

```
547E 1      WRITE (1,3434)
5484 3434   FORMAT ('0INPUT 3 NUMBERS FOR THE DATE (EPDATE(1->3)):',//)
5484 3436   FORMAT ('0INPUT 3 NUMBERS FOR THE TIME (EPDATE(4->6)):',//)
5484        READ (1) QAA,QBB,QCC
54AD        EPDATE(1)=QAA
54B9        EPDATE(2)=QBB
54C6        EPDATE(3)=QCC
54D3 C
54D3        WRITE (1,3436)
54D9        READ (1) QAA,QBB,QCC
5502        EPDATE(4)=QAA
550E        EPDATE(5)=QBB
551B        EPDATE(6)=QCC
5528        WRITE (1,3435)
552E 3435   FORMAT ('0INPUT THE 3 POS. COORDINATES (EPELEM(1->3)):',//)
552E 3437   FORMAT ('0INPUT THE 3 VEL. COORDINATES (EPELEM(4->6)):',//)
552E        READ (1) QA,QB,QC
5557        EPELEM(1)=QA
5563        EPELEM(2)=QB
5570        EPELEM(3)=QC
557D        WRITE (1,3437)
5583        READ (1) QA,QB,QC
55AC        EPELEM(4)=QA
55B8        EPELEM(5)=QB
55C5        EPELEM(6)=QC
55D2 C
55D2        WRITE (1,4436)
55D8 4436   FORMAT ('0INPUT:   TC,DT,PRINT:',//)
55D8        READ (1) TC,DT,PRINT
5601 C
5601        WRITE (1,4437)
5606 4437   FORMAT ('0INPUT:  ITYP,NMAX,MMAX,INC,CONVG:',//)
5606        READ (1) ITYP,NMAX,MMAX,INC,CONVG
5631 C
5631        WRITE (2,2401)
5636 2401   FORMAT ('0',20X,'INTEL ORBIT PROPAGATOR',//)
5636        WRITE (2,2402)  (EPDATE(I),I=1,3)
564B 2402   FORMAT ('0EPOCH DATE:',5X,'YR: ',F6.0,'  MO: ',F6.0,
564B       # '  DAY: ',F6.0)
564B        WRITE (2,2403) (EPDATE(I),I=4,6)
5660 2403   FORMAT (7X,'TIME: ',5X,'HR: ',F6.0,'  MN: ',F6.0,
5660       # '  SEC: ',F6.0)
5660        WRITE (2,2404)
5665 2404   FORMAT ('-EPOCH ELEMENTS - CARTESIAN INERTIAL (TOE)')
5665        WRITE (2,2405)
566A 2405   FORMAT ('0POSITION COMPUTED:')
566A        WRITE (2,2475) EPELEM(1)
567C 2475   FORMAT (5X,'X',10X,F15.6,' KM')
567C        WRITE (2,2476) EPELEM(2)
568E 2476   FORMAT (5X,'Y',10X,F15.6,' KM')
568E        WRITE (2,2477) EPELEM(3)
56A0 2477   FORMAT (5X,'Z',10X,F15.6,' KM')
```

```
56A0        WRITE (2,2406)
56A5 2406   FORMAT ('-VELOCITY COMPONENTS - ')
56A5        WRITE (2,3477) EPELEM(4)
56B7        WRITE (2,2478) EPELEM(5)
56C9        WRITE (2,2479) EPELEM(6)
56DB 3477   FORMAT (5X,'XDOT',7X,F18.10,' KM/SEC')
56DB 2478   FORMAT (5X,'YDOT',7X,F18.10,' KM/SEC')
56DB 2479   FORMAT (5X,'ZDOT',7X,F18.10,' KM/SEC')
56DB C
56DB C      GET EPOCH JULIAN DATE (MODIFIED -2400000.0) EJO-EPHEMERIS
56DB C      AND CORRESPONDING GHA
56DB C
56DB        CALL DATE(EPDATE(1),EPDATE(2),EPDATE(3),EPDATE(4),EPDATE(5),
571C       *          EPDATE(6),DJO,EJO)
5751        CALL GHAX(DJO,GHA)
5774        QJO=DJO+2400000.D0
577F        WRITE (2,2408) QJO
578F 2408   FORMAT ('0EPOCH JULIAN DATE:',5X, F19.10)
578F        WRITE (2,2409)  GHA
579E 2409   FORMAT ('0EPOCH GHA:',5X,F19.10)
579E C
579E C      PROPAGATE ORBIT
579E C
579E        WRITE (2,2410)
57A3 2410   FORMAT ('0TIME FROM',10X,'INERTIAL POS. AND VEL. COORD.(KM):')
57A3        WRITE (2,2411)
57A8 2411   FORMAT (' EPOCH IN SEC:',11X,'X',20X,'Y',20X,'Z')
57A8        WRITE (2,2412)
57AD 2412   FORMAT (20X,'XDOT',17X,'YDOT',17X,'ZDOT',/)
57AD        IDFLAG=0
57B3        CALL ORBIT(EJO,GHA,DT,TC,EPELEM,PRELEM,PRINT,CONVG,ITYP,IDFLAG,
57F6       *           NMAX,MMAX,INC,CS)
5862 C
5862 C      PRINTOUT RESULTS
5862 C
5862        WRITE(2,2010)
5868 2010   FORMAT (1H0)
5868        GO TO 1
586B   999  WRITE(1,1000)
5870  1000  FORMAT ( '0NORMAL END --ALL INPUT PROCESSED')
5870  1001  STOP
5873        END
```

PROGRAM STORAGE: 2534

VARIABLE STORAGE: 0408

UTI FORT//80 COMPILER 3.2G

```
300C C PROGRAM NAME
300C C      DATE
300C C PURPOSE
300C C      TO CONVERT A CALENDAR DATE TO ITS MODIFIED JULIAN DATE
300C C      IN BOTH ET AND UT TIME (JULIAN DATE-2400000)
300C C
300C C CALLING SEQUENCE
300C C      CALL DATE(Y,XM,D,HR,TM,SEC,DJO,EJO)
300C C
300C C      Y -YEAR(GREGORIAN) XM -MONTH            D - DAY
300C C      HR- HOUR           TM -MINUTE           SEC - SECOND
300C C      DJO - UT MODIFIED JULIAN DATE
300C C      EJO - ET MODIFIED JULIAN DATE
300C C
300C        SUBROUTINE DATE(Y,XM,D,HR,TM,SEC,DJO,EJO)
300C        REAL*8 DJO,EJO
300C        REAL*8 FR,DJUL
300C        REAL*8 ZK,Z1,Z2,Z3,X1,Y1
300C C
300C        I=Y+1900.5
301A        J=XM+0.5
3028        K=D+0.5
3036        ZK=K
3040        FR=((HR-12.0)*3600.0+TM*60.+SEC)/86400.0
305F        Z1=I+4800+(J-14)/12
307F        Z1=1461.*Z1
308A        L1=Z1/1000.
3095        Y1=L1
309F        Y1=Y1*1000.
30AA        L1=Z1-Y1
30B5        X1=L1/4
30C6        Z1=X1+Y1/4.
30D5 C
30D5        Z2=J-2-(J-14)/12*12
3100        Z2=367.*Z2
310B        L1=Z2/3000.
3116        Y1=L1
3120        Y1=Y1*3000.
312B        L1=Z2-Y1
3136        X1=L1/12
3147        Z2=X1+Y1/12.
3156 C
3156        Z3=3*((I+4900+(J-14)/12)/100)/4
318A C
318A        DJUL=ZK-32075.+Z1+Z2-Z3-2.43D6
31A5        DJO=DJUL+FR+3.0D4
31B4        EJO=DJO+(38.66+2.5921D-03*(DJO-4.0D4))/86400.0
31CF        RETURN
31D1        END
```

PROGRAM STORAGE: 0453

UTI FORT//80 COMPILER 3.2G

```
31D1 C PROGRAM NAME
31D1 C      GHAX
31D1 C PURPOSE
31D1 C      COMPUTES GHA AT EPOCH(TIME DJO)
31D1 C CALLING SEQUENCE
31D1 C      CALL GHAX(DJO,GHA)
31D1 C
31D1 C      DJO - MODIFIED JULIAN DATE OF EPOCH
31D1 C      GHA - GHA AT EPOCH IN RADIANS
31D1 C
31D1 C ROUTINES CALLED
31D1 C      NONE
31D1 C
31D1 C
31D1        SUBROUTINE GHAX(DJO,GHA)
31D1        REAL*8 DJO,CE,CEE,FRAC,OMEGA,GHX,GHA
31D1 C
31D1        CE=DJO-33282.5D0
31DF        JCE=CE
31E6        CEE=JCE
31F0        FRAC=(CE-CEE)*86400.0D0
31FF        OMEGA=4.1780742D-03/(1.0D0+5.21D-13*CEE)
3212        GHX=100.075542D0+(.98564735D0+2.9015D-13*CEE)*CEE+
321D       * OMEGA*FRAC
3231        GHA=DMOD(GHX,360.0D0)/57.2957795130 8D0
3255        RETURN
3257        END
```

PROGRAM STORAGE: 0134

VARIABLE STORAGE: 0066

UTI FORT//80 COMPILER 3.2G

```
43D1 C PROGRAM NAME
43D1 C      ORBIT
43D1 C PURPOSE
43D1 C      NUMERICALLY INTEGRATE THE ORBITAL EQUATIONS OF MOTION TO
43D1 C      TIME TC.
43D1 C      THE INTEGRATION IS PERFORMED USING A FIXED-STEP,FOURTH-ORDER,
43D1 C      RUNGE-KUTTA INTEGRATOR WITH MODIFIED FEHLBERG COEFFICIENTS
43D1 C      RKT 2(4)
43D1 C      AT USER OPTION, AFTER THE 6TH STEP IS TAKEN THE PROGRAM WILL
43D1 C      SWITCH TO A  6TH ORDER ADAMS-MOULTON PREDICTOR/CORRECTOR
43D1 C
43D1 C CALLING SEQUENCE
43D1 C       CALL ORBIT(EJO,GHA,DT,TC,EPELEM,PRELEM,PRINT,CONVG,ITYP,IDFLAG,
43D1 C                  NMAX,MMAX)
43D1 C
43D1 C      EJO - MODIFIED JULIAN DATE
43D1 C      GHA - GHA AT EPOCH (RADIANS)
43D1 C      DT - STEP SIZE (SEC) ALWAYS POSITIVE
43D1 C      TC - LENGTH OFF PROPAGATION (SEC.) PLUS OR MINUS DEPENDING ON DIR.
43D1 C      EPELEM(6) - EPOCH CARTESIAN ELEMENTS
43D1 C      PRELEM(6) - PREDICTED CARTESIAN ELEMENTS
43D1 C      PRINT - PRINT INTERVAL (MUST BE INTEGER MULTIPLE OF STEPSIZE)
43D1 C      CONVG - CONVERGENCE TOLERANCE FOR A-M CORRECTOR
43D1 C      ITYP - TYPE OF INTEGRATOR = 0 RUNGE-KUTTA, = 1 ADAMS-MOULTON
43D1 C      NMAX - ORDER OF ZONALS
43D1 C      MMAX - ORDER OF TESERALS
43D1 C      INC - NON-CENTRAL BODIES FLAG. = 1, INCLUDE SUN AND MOON
43D1 C
43D1 C ROUTINES CALLED BY ORBIT
43D1 C      ACCEL
43D1 C
43D1 C ROUTINE CALLING ORBIT:
43D1 C      MAIN DRIVER
43D1 C
43D1 C  PROGRAMMER:  G. SNYDER
43D1 C  MODIFIED FOR INTEL BY:  C. RABBIN
43D1 C
43D1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
43D1 C
43D1        SUBROUTINE ORBIT (EJO,GHA,DT,TC,EPELEM,PRELEM,PRINT,CONVG,
43D1       *                  ITYP,IDFLAG,NMAX,MMAX,INC,CS)
43D1 C
43D1        REAL CS(48)
43D1        REAL*8 EJO,C1X,TERROR,CONVG
43D1        REAL*8 EPELEM(6),PRELEM(6)
43D1        REAL*4 C0,CP17,CP25,CP5,CP67,C1,C2,CM2
43D1        REAL*8 XC(3),XCDOT(3),XCDDOT(3),X(3),XDOT(3),
43D1       * XDDOT(3),F0(6),F1(6),F2(6),F3(6)
43D1        REAL*8 AM0(3),AM1(3),AM2(3),AM3(3),VM1(3),VM2(3),VM3(3),
43D1       *    PV(3),PR(3),ERROR(3)
43D1        REAL*8 AM4(3),AM5(3),VM4(3),VM5(3)
43D1        REAL*8 XDDNC(3),DPD
```

```
43D1          REAL*8 TD50,TT,T
43D1 C
43D1 C        START PROGRAM (INITIALIZATION)
43D1 C
43D1 C
43D1          NQ39=1
43D7          DPD=(DT/PRINT)/10.D0
43E9          C0=0.0
43F0          CP17=1.666667E-01
43F7          CP25=0.25
43FE          CP5=0.5
4405          CP67=6.666667E-01
440C          C1=1.0
4413          C2=2.0
441A          CM2=-2.0
4422 C
4422          DDT=TC
4429          T=C0
4430          C1X=DT
4437          ISTEP = 0
443E          DO 10 I=1,3
4442          X(I)=EPELEM(I)
4459          XDOT(I)=EPELEM(I+3)
4475          XC(I)=X(I)
448C          XCDOT(I)=XDOT(I)
44A3       10 CONTINUE
44B4 C
44B4 C        DETERMINE STEPSIZE TO BE USED
44B4 C
44B4       20 IF(DDT.LE.DT) GO TO 25
44C2          H=DT
44CC          DDT=DDT-H
44D7          IF (ITYP .EQ. 0) GO TO 30
44E4          IF (ISTEP .GE. 6) GO TO 200
44F0          GO TO 30
44F3       25 H=DDT
44FD          DDT=0.0
4504 C
4504 C        RUNGE-KUTTA INTEGRATOR
4504 C        INTEGRATE OVER THE STEPSIZE (H)
4504 C
4504       30 CONTINUE
4505          TD50=T
450F          CALL ACCEL(TD50,X,XDOT,XDDOT,EJO,GHA,IDFLAG,NMAX,MMAX,INC,CS)
458A          DO 40 I=1,3
4591          F0(I)=XDOT(I)
45A7          F0(I+3)=XDDOT(I)
45C2       40 CONTINUE
45D3 C
45D3          TT=T+CP25*H
45E5          D=CP25*H
45F0          DO 50 I=1,3
```

```
45F7        XC(I)=D*F0(I)+X(I)
461C        XCDOT(I)=D*F0(I+3)+XDOT(I)
4646     50 CONTINUE
4657        TD50=TT
4661        CALL ACCEL(TD50,XC,XCDOT,XCDDOT,EJO,GHA,IDFLAG,NMAX,MMAX,INC,CS)
46DC        DO 60 I=1,3
46E3        F1(I)=XCDOT(I)
46F9        F1(I+3)=XCDDOT(I)
4714     60 CONTINUE
4725 C
4725        TT=T+CP5*H
4737        D=0.0
473E        D1=CP5*H
4749        DO 70 I=1,3
4750        XC(I)=D*F0(I)+D1*F1(I)+X(I)
4784        XCDOT(I)=D*F0(I+3)+D1*F1(I+3)+XDOT(I)
47C1     70 CONTINUE
47D2        TD50=TT
47DC        CALL ACCEL(TD50,XC,XCDOT,XCDDOT,EJO,GHA,IDFLAG,NMAX,MMAX,INC,CS)
4857        DO 80 I=1,3
485E        F2(I)=XCDOT(I)
4874        F2(I+3)=XCDDOT(I)
488F     80 CONTINUE
48A0 C
48A0        TT=T+C1*H
48B2        D=C1*H
48BD        D1=CM2*H
48C8        D2=C2*H
48D3        DO 90 I=1,3
48DA        J=I+3
48E4        XC(I)=D*F0(I)+D1*F1(I)+D2*F2(I)+X(I)
4927        XCDOT(I)=D*F0(J)+D1*F1(J)+D2*F2(J)+XDOT(I)
496B     90 CONTINUE
497C        TD50=TT
4986        CALL ACCEL(TD50,XC,XCDOT,XCDDOT,EJO,GHA,IDFLAG,NMAX,MMAX,INC,CS)
4A01        DO 100 I=1,3
4A08        F3(I)=XCDOT(I)
4A1E        F3(I+3)=XCDDOT(I)
4A39    100 CONTINUE
4A4A C
4A4A C      COMPUTE ORBITAL STATE AT TIME T+H
4A4A C
4A4A        D=CP17*H
4A58        D1=0.0
4A5F        D2=CP67*H
4A6A        D3=CP17*H
4A75        DO 110 I=1,3
4A7C        J=I+3
4A86        XC(I)=D1*F1(I)+D*F0(I)+D3*F3(I)+D2*F2(I)+X(I)
4AD8        XCDOT(I)=D1*F1(J)+D*F0(J)+D3*F3(J)+D2*F2(J)+XDOT(I)
4B2B    110 CONTINUE
4B3C        GO TO 400
```

```
4B3F C
4B3F C      ADAMS-BASHFORTH PREDICTOR
4B3F C
4B3F   200 DO 210 I=1,3
4B45       PV(I)=XDOT(I)+C1X*(2.9701388889D0*AM0(I)-5.5020833333D0*AM1(I)
4B67      *+6.9319444444D0*AM2(I)-5.0680555556D0*AM3(I)+1.9979166667D0*AM4(I)
4B94      *         - 3.2986111111D-1*AM5(I))
4BB9       PR(I)=X(I)+C1X*(2.9701388889D0*XDOT(I)-5.5020833333D0*VM1(I)
4BDC      *+6.9319444444D0*VM2(I)-5.0680555556D0*VM3(I)+1.9979166667D0*VM4(I)
4C09      *         - 3.2986111111D-1*VM5(I))
4C2E   210 CONTINUE
4C3F C
4C3F C      COMPUTE NON-CENTRAL FORCES
4C3F C
4C3F       IFLG = IDFLAG + 4
4C49       TD50 = T + H
4C57       CALL ACCEL(TD50,PR,PV,XDDNC,EJO,GHA,IFLG,NMAX,MMAX,INC,CS)
4CD2 C
4CD2 C      ADAMS-MOULTON CORRECTOR
4CD2 C
4CD2       ICOUNT = 0
4CD9   250 CALL ACCEL(TD50,PR,PV,XCDDOT,EJO,GHA,3,NMAX,MMAX,INC,CS)
4D52       DO 251 I=1,3
4D59       XCDDOT(I) = XCDDOT(I) + XDDNC(I)
4D7A   251 CONTINUE
4D8B       DO 260 I=1,3
4D91       XCDOT(I)=XDOT(I)+C1X*(3.2986111111D-1*XCDDOT(I)+9.9097222222D-1
4DB3      * *AM0(I)-5.5416666667D-1*AM1(I)+3.3472222222D-1*AM2(I)
4DD1      *   -1.2013888888D-1*AM3(I)+ 1.875D-2*AM4(I))
4E05       XC(I)=X(I)+C1X*(3.2986111111D-1*XCDOT(I)+9.9097222222D-1*XDOT(I)
4E28      *   - 5.5416666667D-1*VM1(I) + 3.3472222222D-1*VM2(I)
4E46      * -1.2013888888D-1*VM3(I)            + 1.875D-2*VM4(I))
4E7A       ERROR(I) = XC(I) - PR(I)
4E9C       PV(I) = XCDOT(I)
4EB3       PR(I) = XC(I)
4ECA   260 CONTINUE
4EDB       TERROR = SQRT(ERROR(1)*ERROR(1) + ERROR(2)*ERROR(2)
4EF9      *                + ERROR(3)*ERROR(3))
4F2A       IF (ABS(TERROR) .LE. CONVG) GO TO 400
4F47       ICOUNT = ICOUNT + 1
4F51       IF (ICOUNT .LT.10) GO TO 250
4F5D       GO TO 900
4F60 C
4F60 C      CHECK TO SEE IF INTEGRATION IS COMPLETE
4F60 C
4F60   400 CONTINUE
4F60       IF(DDT.EQ.0.0) GO TO 500
4F6E       T=T+H
4F7C       DO 420 I=1,3
4F83       XDDOT(I)=XCDDOT(I)
4F99       X(I) = XC(I)
4FB0       AM5(I) = AM4(I)
```

```
4FC7        AM4(I) = AM3(I)
4FDE        AM3(I) = AM2(I)
4FF5        AM2(I) = AM1(I)
500C        AM1(I) = AM0(I)
5023        AM0(I) = XDDOT(I)
503A        VM5(I) = VM4(I)
5051        VM4(I) = VM3(I)
5068        VM3(I) = VM2(I)
507F        VM2(I) = VM1(I)
5096        VM1(I) = XDOT(I)
50AD        XDOT(I)=XCDOT(I)
50C4    420 CONTINUE
50D5        TMOD = T / PRINT
50E3        IMOD=TMOD
50EA        ZMOD=IMOD
50F4        XMOD=TMOD-ZMOD
50FF        IF (XMOD.NE.0.0D0)  GO TO 555
510A        WRITE (2,2412) NQ39,T,X(1),X(2),X(3)
5146        WRITE (2,3412) XDOT(1),XDOT(2),XDOT(3)
5172 2412   FORMAT (1H ,I3,') ',F10.1,2X,F18.10,2X,F18.10,2X,F18.10)
5172 3412   FORMAT (11X,F18.10,2X,F18.10,2X,F18.10)
5172        NQ39=NQ39+1
517C 555    ISTEP=ISTEP+1
5186        GO TO 20
5189    500 DO 505 I=1,3
518F        PRELEM(I)=XC(I)
51A7        PRELEM(I+3)=XCDOT(I)
51C4    505 CONTINUE
51D5        GO TO 999
51D8 C
51D8 C      ERROR - ADAMS-MOULTON DID NOT CONVERGE
51D8 C
51D8    900 WRITE (2,1000) TERROR,T
51F1        GO TO 400
51F4    999 CONTINUE
51F4        RETURN
51F5   1000 FORMAT (' ADAMS-MOULTON PREDICTOR/CORRECTOR DID NOT CONVERGE.'
51F5       * ,/,' ERROR= ' ,1X,F12.6,'  T= ',F14.6)
51F5        END
```

PROGRAM STORAGE: 3754

VARIABLE STORAGE: 0888

```
C PROGRAM NAME
C     ACCEL
C PURPOSE
C     COMPUTE THE SPACECRAFT ACCELERATIONS DUE TO THE EARTH
C     POTENTIAL (J2,J3,J4).
C
C CALLING SEQUENCE
C     CALL ACCEL(TT,X,XDOT,XDDOT,EJO,GHA,IDFLAG,NMAX,MMAX)
C
C     TT - TIME INTO PROPAGATION (SEC)
C     X(3) - POSITION (KM)
C     XDOT(3) - VELOCITY (KM/SEC)
C     XDDOT(3) - ACCELERATION (KM/SEC**2)
C     EJO - MODIFIED JULIAN DATE OF EPOCH
C     GHA - GHA AT EPOCH (RADIANS)
C     IDFLAG = PROCESS FLAG
C            = 0, COMPUTE FULL ACCELERATION WITHOUT DRAG
C            = 1, COMPUTE FULL ACCELERATION WITH DRAG (NOT APPLICABLE)
C            = 3, COMPUTE CENTRAL FORCE ONLY
C            = 4, COMPUTE NON-CENTRAL FORCES WITHOUT DRAG
C            = 5, COMPUTE NON-CENTRAL FORCES WITH DRAG (NOT APPLICABLE)
C     NMAX - ORDER OF ZONALS
C     MMAX - ORDER OF TESSERALS
C     INC = NON-CENTRAL BODIES FLAG
C         = 0, NO NON-CENTRAL BODIES
C         =1, INCLUDE SUN & MOON
C
      SUBROUTINE ACCEL(TT,XD,XDOTD,XDDOTD,EJO,GHA,IDFLAG,NMAX,MMAX,INC,
     # CS)
      REAL CS(48)
      REAL*8 EJO,TTX,TWOPI,ROTER,GHAC,GHAM,ALAM,TT,RASAT,GHA
      REAL*8 U,R,CONST
      REAL*8 AL1,AL11,EJT,XMAG2,XMAG3,XMAG4,XMAG5
      REAL*8 XD(3),XDOTD(3),XDDOTD(3)
      REAL*8 AGMD(3),AZD(3)
      REAL*8 X(3),XDOT(3),AZONAL(3)
      REAL*8 ANCD(3)
      REAL*8 POSM(3),POSS(3),RHO1(3),RHO2(3)
      REAL*8 X1,Y1,Z1
C
      U=398600.8D0
      ROTER=7.2921159D-5
      TWOPI=6.28318530717958647692D0
```

```
C
C
C     COMPUTE ACCELERATION DUE TO POINT-MASS GRAVIATIONAL FORCE
C
      DO 5 I=1,3
      AGMD(I) = 0.0D0
      AZONAL(I) = 0.0D0
      ANCD(I) = 0.0D0
      X(I)=XD(I)
      XDOT(I)=XDOTD(I)
    5 CONTINUE
      IF (IDFLAG .GT. 3) GO TO 11
      R = SQRT(XD(1)*XD(1) + XD(2)*XD(2) + XD(3)*XD(3))
      CONST=-U/(R*R*R)
      DO 10 I=1,3
   10 AGMD(I) = CONST * XD(I)
   11 CONTINUE
      IF (IDFLAG .EQ. 3) GO TO 15
      IF (INC .EQ. 0) GO TO 105
C
C     LOCATE POSITION OF MOON
C
      TTX = TT
      EJT = EJO + (TTX/86400.D0) + 2400000.D0
      X1=POSM(1)
      Y1=POSM(2)
      Z1=POSM(3)
      CALL LUNA(EJT,X1,Y1,Z1,AL1)
      POSM(1)=X1
      POSM(2)=Y1
      POSM(3)=Z1
      AL11 = 384399.06D0 / (1.D0 + AL1)
      DO 101 I=1,3
      POSM(I) = POSM(I) * AL11
  101 CONTINUE
C
C     LOCATE POSITION OF SUN
C
```

```
      X1=POSS(1)
      Y1=POSS(2)
      Z1=POSS(3)
      CALL SOL(EJT,X1,Y1,Z1,AL1)
      POSS(1)=X1
      POSS(2)=Y1
      POSS(3)=Z1
      AL11 = 0.149597871D+09 / (1.D0 + AL1)
      DO 102 I=1,3
      POSS(I) = POSS(I) * AL11
  102 CONTINUE
C
C     ADD 3RD BODY ACCELERATIONS
C
      DO 103 I=1,3
      RHO1(I) = POSM(I) - XD(I)
      RHO2(I) = POSS(I) - XD(I)
  103 CONTINUE
      XMAG2 = POSM(1)*POSM(1) + POSM(2)*POSM(2) +
     # POSM(3)*POSM(3)
      XMAG2=SQRT(XMAG2)
      XMAG2=XMAG2*XMAG2*XMAG2
      XMAG3 = POSS(1)*POSS(1) + POSS(2)*POSS(2) +
     # POSS(3)*POSS(3)
      XMAG3=SQRT(XMAG3)
      XMAG3=XMAG3*XMAG3*XMAG3
      XMAG4 = RHO1(1)*RHO1(1) + RHO1(2)*RHO1(2) +
     # RHO1(3)*RHO1(3)
      XMAG4=SQRT(XMAG4)
      XMAG4=XMAG4*XMAG4*XMAG4
      XMAG5 = RHO2(1)*RHO2(1) + RHO2(2)*RHO2(2) +
     # RHO2(3)*RHO2(3)
      XMAG5=SQRT(XMAG5)
      XMAG5=XMAG5*XMAG5*XMAG5
      DO 104 I=1,3
      ANCD(I) = 4902.778D0 * (RHO1(I)/XMAG4 - POSM(I)/XMAG2)
      ANCD(I) = 132715445.D3 * (RHO2(I)/XMAG5 - POSS(I)/XMAG3) + ANCD(I)
  104 CONTINUE
  105 CONTINUE
      IF (TT.NE.0.0D0)  GO TO 444
C
C     COMPUTE ACCELERATION DUE TO EARTH GEOPOTENTIAL FIELD
C
444   RASAT=ATAN2(X(2),X(1))
      GHAM=GHA+(ROTER*TT)
      GHAC=DMOD(GHAM,TWOPI)
      ALAM=RASAT-GHAC
      CALL SPART(ALAM,X,NMAX,MMAX,AZONAL,CS)
      IF (TT.NE.0.0D0)  GO TO 445
C
C     COMBINE ACCELERATIONS
C
445   CONTINUE
   15 DO 20 I=1,3
      AZD(I)=AZONAL(I)
   20 XDDOTD(I) = AGMD(I) + AZD(I) + ANCD(I)
      RETURN
      END
```

```
2652        SUBROUTINE SPART(ALAM,PR,NMAX,MMAX,DX,CS)
2652 C
2652 C      PURPOSE
2652 C          COMPUTE NON-SPHERICAL ACCELERATIONS(EARTH) UP TO 6X6 FIELD
2652 C
2652 C      CALLING SEQUENCE
2652 C
2652 C   NAME    I/O        DESCRIPTION
2652 C
2652 C   ALAM     I    GEOCENTRIC LONGITUDE OF SPACECRAFT
2652 C   PR       I    INERTIAL COORDINATES OF S/C IN TOD SYSTEM
2652 C   NMAX     I    N INDEX FOR THE HARMONIC COEFFICIENTS
2652 C   MMAX     I    M INDEX FOR THE HARMONIC COEFFICIENTS
2652 C   DX       O    RESULTANT ACCELERATION VECTOR IN TOD SYSTEM
2652 C
2652 C
2652 C
2652        REAL CS(48)  ,SINLAM(7),
2652       * COSLAM(7),PMN(48),TPSIM(7)
2652        REAL*8 RSQ,XYSQ
2652        REAL*8 ALAM,PR(3),DX(3)
2652 C
2652 C
2652 C
2652        AE=6378.14
265C        TGM=398600.8
2663        DO 211  I=1,48
266A        PMN(I)=0.0
267C 211    CONTINUE
268D C
268D C
268D C
268D C      INITIALIZE SUMATION ARRAYS
268D C
268D        INDEX2 = NMAX
2693        INDEX4 = MMAX + 2
269D        INDEX5 = MMAX + 1
26A7 C
26A7 C      SINE, COSINE AND TANGENT OF LATITUDE
26A7 C
26A7        XYSQ = PR(1)*PR(1) + PR(2)*PR(2)
26D5        RTXYSQ = SQRT(XYSQ)
26EE        RSQ = XYSQ + PR(3)*PR(3)
270B        R=SQRT(RSQ)
2724        SINP = PR(3)/R
2736        COSP = RTXYSQ/R
2741        TANP = SINP/COSP
274C        RINV = AE/R
2757 C
2757 C      POLYNOMIAL TERMS
2757 C
2757        PMN(1) = SINP
```

```
2764          PMN(2) = COSP
2771          CP3 = 3.*COSP
277C          PMN(9) = 1.5*SINP*SINP - .5
2795          PMN(10) = CP3*SINP
27A6          PMN(11) = CP3*COSP
27B7 C
27B7          TPSIM(1) = 0.
27C4          TPSIM(2) = TANP
27D1          TPSIM(3) = 2.*TANP
27E2 C
27E2 C        SINES, COSINES OF LONGITUDE
27E2 C
27E2          SINLAM(1) = 0.0
27EF          COSLAM(1) = 1.0
27FC          SINLAM(2)=SIN(ALAM)
281B          COSLAM(2) =COS(ALAM)
283A          CL2 = 2.*COSLAM(2)
284C          SINLAM(3)=CL2*SINLAM(2)
2860          COSLAM(3) = CL2*COSLAM(2) - 1.
2878 C
2878          DO 120 N=3,INDEX2
2885          F1 = N
288F          F2 = F1-1.0
289A          F3 = 2.*F1 - 1.
28A9          F4 = F3*COSP
28B4          N1 = N-1
28BF          N2 = N-2
28C9 C
28C9 C        ZONAL HARMONICS (M=0)
28C9 C
28C9          NA1=N*8-7
28D9          NA2=N1*8-7
28E9          NA3=N2*8-7
28F9          PMN(NA1) = (F3*SINP*PMN(NA2) - F2*PMN(NA3))/F1
292D          NX=N
2934          IF (INDEX4.LT.N)  NX=INDEX4
2948 C
2948 C        TESSERALS (M LESS THAN N)
2948 C
2948          DO 110 M=2,NX
2954          NA1=N*8-8+M
2969          NA2=N2*8-8+M
297E          NA3=N1*8-8+M-1
2997          PMN(NA1) = PMN(NA2) + F4*PMN(NA3)
29BB      110 CONTINUE
29CF          IF(NX.LT.N) GO TO 120
29DD          NN1 = N + 1
29E7 C
29E7 C        SECTORIAL (M=N)
29E7 C
29E7          NA1=N*8-8+NN1
29FC          NA2=N1*8-8+N
```

```
2A11        PMN(NA1) = F4*PMN(NA2)
2A2A        TPSIM(NN1) = TPSIM(N) + TANP
2A44        SINLAM(NN1) = CL2*SINLAM(N) - SINLAM(N1)
2A69        COSLAM(NN1) = CL2*COSLAM(N) - COSLAM(N1)
2A8E    120 CONTINUE
2AA2 C
2AA2 C      INITIALIZE SUMATIONS FOR PARTIALS
2AA2 C
2AA2        PS = 0.
2AAC        PLAMDA = 0.
2AB3        PPSI = 0.
2ABA        FN1 = 2.
2AC1        RN = RINV
2AC8 C
2AC8 C      SUMMATION FOR PARTIALS
2AC8 C
2AC8        DO 250 NC=2,INDEX2
2AD5        NS = 7 - NC
2AE3        RN = RN*RINV
2AF1        FN1 = FN1 + 1.
2AFC        DLAMDA = 0.
2B03 C
2B03        F1 = CS(NC)
2B12        NA1=NC*8-7
2B23        DR = F1*PMN(NA1)
2B34        DPSI = F1*PMN(NA1+1)
2B4A        IF(INDEX4.EQ.2) GO TO 210
2B57        FM = 1.
2B61        INDEX=NC+1
2B6C        IF (INDEX5.LT.NC+1)  INDEX=INDEX5
2B84        DO 200 MC=2,INDEX
2B90        MS = 10 - MC
2B9E        NA1=NC*8-8+MC
2BB3        P1 = PMN(NA1)
2BC0        NA1=MC*6-6+NC
2BD6        C1 = CS(NA1)
2BE4        NA1=MS*6-6+NS
2BFA        S1 = CS(NA1)
2C08 C
2C08        DLAMDA = DLAMDA + FM*P1*(S1*COSLAM(MC) - C1*SINLAM(MC))
2C35        F1 = C1*COSLAM(MC) + S1*SINLAM(MC)
2C56 C
2C56        DR = DR + F1*P1
2C65 C
2C65        NA1=NC*8-8+MC+1
2C7F        DPSI = DPSI + F1*(PMN(NA1) - TPSIM(MC)*P1)
2CA3        FM = FM + 1.
2CAE    200 CONTINUE
2CC2        PLAMDA = PLAMDA + DLAMDA*RN
2CD4 C
2CD4    210 PS=PS+DR*FN1*RN
2CEB        PPSI = PPSI + DPSI*RN
```

```
2CFA   250 CONTINUE
2D0E C
2D0E       GMR = TGM/R
2D1C C
2D1C C     PARTIAL WRT R
2D1C C
2D1C       PS = -GMR*PS/R
2D2C C
2D2C C     PARTIAL WRT LAMBDA
2D2C C
2D2C       PLAMDA = GMR*PLAMDA
2D37 C
2D37 C     PARTIAL WRT PSI
2D37 C
2D37       PPSI = GMR*PPSI
2D42 C
2D42 C     CONVERT TO RECTANGULAR COORDINATES
2D42 C
2D42       PRR = PS/R
2D4D       PLXY = PLAMDA/XYSQ
2D58       PPTP = PRR - PPSI*PR(3)/(RTXYSQ*RSQ)
2D76 C
2D76       DX(1) = PR(1)*PPTP - PLXY*PR(2)
2DA1       DX(2) = PR(2)*PPTP + PLXY*PR(1)
2DCC       DX(3) = PRR*PR(3) + PPSI*RTXYSQ/RSQ
2DF4 C
2DF4   999 RETURN
2DF6       END
```

PROGRAM STORAGE: 1956

VARIABLE STORAGE: 0484

```
C PROGRAM NAME
C     SOL
C
C PURPOSE
C     TO DETERMINE THE POSITION OF THE SUN WITH RESPECT TO THE EARTH
C
C METHOD
C     POSITION OF THE SUN WITH RESPECT TO THE EARTH IS DETERMINED
C     FROM THE MEAN MOTION OF THE SUN AS DESCRIBED IN THE SUPPLEMENT
C     TO THE NAUTICAL EPHEMERIS
C
C     SOLAR POSITION AND VELOCITY ROUTINE
C
C     POSITIONAL ACCURACY ABOUT .0005 RADIAN OR 75000 KM
C
C
C     INPUTS (ALL DOUBLE PRECISION)
C
C        AJD -   FULL JULIAN EPHEMERIS DATE
C
C
C
C     OUTPUTS (ALL DOUBLE PRECISION)
C
C        X1 -    X-COMPONENT OF GEOCENTRIC SOLAR POSITION VECTOR = X/R
C        Y1 -    Y-COMPONENT OF GEOCENTRIC SOLAR POSITION VECTOR = Y/R
C        Z1 -    Z-COMPONENT OF GEOCENTRIC SOLAR POSITION VECTOR = Z/R
C
C        AL1 -   SCALING FACTOR FOR GEOCENTRIC SOLAR DISTANCE
C
C
C     R = DISTANCE TO SUN IN KM = 149597871.D0/(1.D0+AL1-AL2*LAM0)
C
```

```
      SUBROUTINE SOL(AJD,X1,Y1,Z1,AL1)
      REAL*8 AJD,LAMO,N,X1,Y1,Z1,AL1,CDR,E,TWOPI,
     $GAMO,GAMD,GO,GD,EPSO,EPSD,D,EE,E1,E2,EPS,GAM,G,AL,R,CEPS,SEPS,
     $RASUN,DECSUN,SL,F
      TWOPI=6.283185307179586476920D0
      CDR=0.174532925199430D-01
      GAMO=281.220833D0*CDR
      GAMD=.0000470684D0*CDR
      GO=358.475845D0*CDR
      GD=0.985600267D0*CDR
      EPSO=23.452294D0*CDR
      EPSD=0.35626D-06*CDR
      D=AJD-2415020.0D0
      E=0.01675104D0-0.11444D-08*D
      E1=2.D0*E
      EPS=EPSO-D*EPSD
      GAM=GAMO+GAMD*D
      G=GO+D*GD
      Z2=SIN(G)
      AL=GAM+G+E1*Z2
      AL=DMOD(AL,TWOPI)
      X1=COS(AL)
      Z1=SIN(AL)
      CEPS=COS(EPS)
      SEPS=SIN(EPS)
      Y1=Z1*CEPS
      Z1=Z1*SEPS
      AL1=E*COS(G)
      RETURN
      END
```

```
C PROGRAM NAME
C     LUNA
C
C PURPOSE
C     TO DETERMINE THE POSITION OF THE MOON WITH RESPECT TO THE EARTH
C
C METHOD
C     THIS SUBROUTINE CALCULATES THE POSITION AND VELOCITY OF THE MOON
C     FROM MEAN FLEMENTS OBTAINED FROM THE SUPPLIMENT TO THE NAUTICAL
C     EPHEMERIS
C
C     LUNAR POSITION ROUTINE
C
C     POSITIONAL ACCURACY ABOUT .0005 RADIAN OR 200 KM
C        - TERMS WITH AMPLITUDES GREATER THAN 50 KM RETAINED
C
C     INPUTS (ALL DOUBLE PRECISION)
C
C        JDOB -  FULL JULIAN EPHEMERIS DATE
C
C     OUTPUTS (ALL DOUBLE PRECISION)
C
C        X1 -    X-COMPONENT OF GEOCENTRIC LUNAR POSITION VECTOR = X/R
C        Y1 -    Y-COMPONENT OF GEOCENTRIC LUNAR POSITION VECTOR = Y/R
C        Z1 -    Z-COMPONENT OF GEOCENTRIC LUNAR POSITION VECTOR = Z/R
C
C        AL1 -   SCALING FACTOR FOR GEOCENTRIC LUNAR DISTANCE
C
C
C     R = DISTANCE TO MOON IN KM = 384399.06D0/(1.D0+AL1-AL2*LAM0)
C
C
      SUBROUTINE LUNA (JDOB,X1,Y1,Z1,AL1)
      REAL*8 X1,Y1,Z1
      REAL*8 JDOB,OBLIQ,T,TWOPI,BL0,BLDOT,SL0,SLDOT,SLP0,SLPDO
     #T,F0,FDOT,DD0,DDOT,BL,SL,SLP,F,D,SOB,COB,B,R,D2,
     #AL1,EPS0,EPSD,BLR,SLR,SLPR,FR,DR,SSL,CSL,SSLP,CSLP,SD
     #,INCL,CDR,CD,SF,CF,S2SL,C2SL,S2D,C2D,S4D,C4D,S2F,C2F,SNLMD2,CLM2D,
     #SSLMF,SLPC2D,CLPS2D,SLCLP,CLSLP,SLC2F,CLS2F,SFC2D,CFS2D,SLCF,CLSF,
     #CLC2D,SLS2D,S2LCF,C2LSF,BLX,CSR,BLDDOT,SLDDOT,FDDOT,DDDOT
C
C      EPOCH= 1900 JAN. 0.5 = J.D. 2415020.0
C
      TWOPI=6.2831853071796D0
      CDR=0.17453292519943D-01
C
      EPS0=23.452294D0*CDR
      EPSD=0.35626D-06*CDR
      CSR=CDR/3600.D0
      INCL=0.089503D0
      BL0=(270.D0+26.0499D0/60.D0)*CDR
      BLDOT=(1336.D0*360.D0+307.D0+52.9886D0/60.D0)*CDR
      BLDDOT=-4.08D0*CSR
      SL0=(296.D0+6.2764D0/60.D0)*CDR
      SLDOT=(1325.D0*360.D0+198.D0+50.9465D0/60.D0)*CDR
      SLDDOT=33.09D0*CSR
      SLP0=(358.D0+28.55D0/60.D0)*CDR
      SLPDOT=(99.D0*360.D0+359.D0+2.985D0/60.D0)*CDR
      F0=(11.D0+15.0533D0/60.D0)*CDR
      FDOT=(1342.D0*360.D0+82.D0+1.509D0/60.D0)*CDR
      FDDOT=-11.56D0*CSR
      DD0=(350.D0+44.2492D0/60.D0)*CDR
      DDOT=(1236.D0*360.D0+307.D0+6.853D0/60.D0)*CDR
      DDDOT=-5.17D0*CSR
C
C      CONVERT RATES TO NATURAL UNITS (1/TWOPI DAY)
```

```
      T=(JDOB-2415020.0D0)/36525.0D0
      OBLIQ=     EPS0-EPSD*(JDOB-2415020.0D0)
      SOB=SIN(OBLIQ)
      COB=COS(OBLIQ)
      BL=DMOD(BL0+T*(BLDOT+BLDDOT*T),TWOPI)
      SL=DMOD(SL0+T*(SLDOT+SLDDOT*T),TWOPI)
      SLP=DMOD(SLP0+SLPDOT*T,TWOPI)
      F=DMOD(F0+T*(FDOT+FDDOT*T),TWOPI)
      D=DMOD(D0+T*(DDOT+DDDOT*T),TWOPI)
      SSL =SIN(SL)
      SSLP=SIN(SLP)
      CSL=COS(SL)
      CSLP=COS(SLP)
      SD=SIN(D)
      SF=SIN(F)
      CD=COS(D)
      CF=COS(F)
      S2SL=2.D0*SSL*CSL
      S2D =2.D0*SD *CD
      S2F =2.D0*SF *CF
      C2SL=CSL*CSL-SSL*SSL
      C2D =CD *CD -SD *SD
      C2F =CF *CF -SF *SF
      S4D=2.D0*S2D*C2D
      C4D=C2D*C2D-S2D*S2D
      SNLMD2=SSL*C2D-CSL*S2D
      CLM2D=CSL*C2D+SSL*S2D
      SLCF=SSL*CF
      CLSF=CSL*SF
      SSLMF=SLCF-CLSF
      X1=SNLMD2*CSLP
      X2=CLM2D*SSLP
      SLPC2D=SSLP*C2D
      CLPS2D=CSLP*S2D
      SLCLP=SSL*CSLP
      CLSLP=CSL*SSLP
      SLC2F=SSL*C2F
      CLS2F=CSL*S2F
      AL1=SF*C2D
      AL2=CF*S2D
      Z1=CSL*C2D
      Z2=SSL*S2D
      Y1=CSL*CSLP
      Y2=SSL*SSLP
      S2LCF=S2SL*CF
      C2LSF=C2SL*SF

      BL=BL+0.011490D0*S2D      -0.001996D0*S2F          -0.003238D0*SSLP
     #    -0.022236D0*SNLMD2      +0.109760D0*SSL      +0.003728D0*S2SL
     #-0.000607D0*SD-0.000267D0*(S2F*C2D-C2F*S2D)-0.000801D0*(SLPC2D
     # -CLPS2D)-0.000118D0*(SLPC2D+CLPS2D)+0.000138D0*(X1-X2)+0.000716D0
     #*(SLCLP-CLSLP)+0.000192D0*(SLC2F-CLS2F)-0.000186D0*(SSL*C4D-CSL*S4
     #D)+0.000931D0*(SSL*C2D+CSL*S2D)-0.000219D0*(SLC2F+CLS2F)-0.000999D
     #0*(X1+X2)-0.000532D0*(SLCLP+CLSLP)-0.000149D0*(S2SL*C4D-C2SL*S4D)
     #-0.001026D0*(S2SL*C2D-C2SL*S2D)+0.000175D0*(SSL*C2SL+CSL*S2SL)
      B=INCL*SF-0.003023D0*(AL1-AL2) +0.004897D0*(SLCF+CLSF)           +0
     #.004847D0*SSLMF +0.000569D0*(AL1+AL2) -0.000144D0*(SSLP*(CF*C2D+SF
     # *S2D)+CSLP*(AL1-AL2)) -0.000807D0*(SNLMD2*CF+CLM2D*SF)+0.000161D0
     #*(SSLMF*C2D+(CSL*CF+SSL*SF)*S2D)-0.000967D0*(SNLMD2*CF-CLM2D*SF)
     #+0.000301D0*(S2LCF+C2LSF)+0.000154D0*(S2LCF-C2LSF)
      AL1=          0.0082488D0*C2D      +0.0100247D0*(Z1+Z2)     +0.05450
     #08D0*CSL+0.0029700D0*C2SL+0.0009017D0*(Z1-Z2)+0.0005604D0*(CSLP*C2
     #D+SSLP*S2D)+0.0004219D0*(CLM2D*CSLP-SNLMD2*SSLP)+0.0003369D0*(Y1+Y
     #2)-0.0002773D0*(Y1-Y2)-0.0002086D0*(CSL*C2F+SSL*S2F)
     # +0.0001817D0*(CSL*C2SL-SSL*S2SL)
C
C     FROM NOW ON, SL=SIN(BL), F=SIN(B), D=COS(B)
C
      SL=SIN(BL)
      SLP=COS(BL)
      F=SIN(B)
      D=COS(B)
      X1      =D*SLP
      Y1      =D*SL*COB-F*SOB
      Z1      =D*SL*SOB+F*COB
      RETURN
      END
```

```
UTI FORT//80 COMPILER 3.2G

0000        COMPILER(1)=2000H
2006        COMPILER(3)=1FFFH
2006        REAL*8 FUNCTION ABS(X)
2006 C
2006        REAL*8 X
2006 C
2006        ABS=X
2010        IF (ABS.LT.0.0D0)  ABS=-ABS
2027        RETURN
202D        END

PROGRAM STORAGE: 0039

VARIABLE STORAGE: 0020
```

UTI FORT//80 COMPILER 3.2G

```
202D        REAL*8 FUNCTION DMOD(X,XM)
202D C
202D        REAL*8 X,XM,XMOD
202D C
202D        KMOD=X/XM
203B        XMOD=KMOD
2045        DMOD=X-XMOD*XM
2054        IF (DMOD.LT.0.0D0)  DMOD=DMOD+XM
206E        RETURN
2074        END
```

PROGRAM STORAGE: 0071

VARIABLE STORAGE: 0042

UTI FORT//80 COMPILER 3.2G

```
2074          REAL*8 FUNCTION SQRT(X)
2074 C
2074          REAL*8 X,SQ1,SQ2
2074 C
2074          IF (X.LT.1.D-6)  SQ1=5.D-13
208D          IF (X.GE.1.D-6)  SQ1=5.D-10
20A6          IF (X.GE.1.D-3)  SQ1=5.D-5
20BF          IF (X.GE.1.D-1)  SQ1=4.D-1
20D8          IF (X.GE.1.D0)   SQ1=1.D0
20F1          IF (X.GE.4.D0)   SQ1=2.5D0
210A          IF (X.GE.1.D1)   SQ1=5.D0
2123          IF (X.GE.5.D1)   SQ1=1.D1
213C          IF (X.GE.5.D2)   SQ1=1.D2
2155          IF (X.GE.5.D4)   SQ1=1.D3
216E          IF (X.GE.5.D6)   SQ1=1.D4
2187          IF (X.GE.5.D8)   SQ1=1.D5
21A0          IF (X.GE.5.D10)  SQ1=1.D6
21B9          IF (X.GE.5.D12)  SQ1=1.D7
21D2          IF (X.GE.5.D14)  SQ1=1.D8
21EB          IF (X.GE.5.D16)  SQ1=1.D9
2204          IF (X.GE.5.D18)  SQ1=1.D10
221D          IF (X.GE.5.D20)  SQ1=1.D11
2236          IF (X.GE.5.D22)  SQ1=1.D12
224F C
224F          J=0
2255 301      SQ2=0.5D0*(SQ1+X/SQ1)
226B          IF (ABS(SQ2-SQ1).LT.ABS(1.D-12*SQ2))  GO TO 310
2294          J=J+1
229E          IF (J.GT.30)  GO TO 310
22AA          SQ1=SQ2
22B4          GO TO 301
22B8 310      SQRT=SQ2
22C2          RETURN
22C9          END
```

PROGRAM STORAGE: 0597

VARIABLE STORAGE: 0038

```
      REAL*8 FUNCTION SIN(XA)
C
      REAL*8 X,XA,A,B,C,D,E,XSQ
      REAL*8 PI,PIO2,TWOPI
C
      PI=3.14159265358979323846D0
      PIO2=1.57079632679489661923D0
      TWOPI=6.28318530717958647692D0
C
      A=1.57079631847D0
      B=-.64596371106D0
      C=0.7968967928D-1
      D=-0.467376557D-2
      E=0.15148419D-3
      X=XA
      X=DMOD(X,TWOPI)
      IF (X.GT.PI) X=X-TWOPI
      IF (X.GT.PIO2)  X=PI-X
      IF (X.LT.-PIO2) X=-PI-X
      X=X/PIO2
      XSQ=X*X
      SIN=X*(A+XSQ*(B+XSQ*(C+XSQ*(D+XSQ*E))))
      IF (SIN.GT.1.0D0)  SIN=1.0D0
      IF (SIN.LT.-1.0D0)  SIN=-1.0D0
      RETURN
      END
```

```
UTI FORT//80 COMPILER 3.2G

2406        REAL*8 FUNCTION COS(X)
2406 C
2406        REAL*8 X,XX,PIO2
2406 C
2406        PIO2=1.5707963267948966192 3D0
2410        XX=PIO2-X
241B        COS=SIN(XX)
2434        RETURN
243B        END

PROGRAM STORAGE: 0053

VARIABLE STORAGE: 0036
```

UTI FORT//80 COMPILER 3.2G

```
243B          REAL*8 FUNCTION ATAN2(X1,X2)
243B C
243B          REAL*8 X1,X2,U,C0,C1,C2,C3,A
243B          REAL*8  PIO16,THRPIO16,TPIO16,T3PIO16,PIO2,SQT2M1
243B          REAL*8 B1,B2,B3,Z,Z1,Z2,Z3,P1,P2,Q1,Q2,F3,ATANX,Y,Y1,ALPHA
243B C
243B          PIO16=0.1963495408493D0
2445          THRPIO16=PIO16+PIO16+PIO16
2454          TPIO16=0.1989123673791D0
245B          T3PIO16=0.6681786379186D0
2462          PIO2=1.5707963267948966192 3D0
2469          SQT2M1=0.414213562373D0
2470 C
2470          C0=0.206885682818530 5D0
2477          C1=3.008682092051749D0
247E          C2=-3.497610177361543D0
2486          C3=-0.1343364285454181 8D0
248E          B1=5.182726637174420D0
2495          B2=2.619466421369197D0
249C          B3=1.319706666866303D0
24A3 C
24A3          IF (X2.EQ.0.0D0)  GO TO 209
24AE          U=X1/X2
24BC          IF (U.LT.0.0D0)  GO TO 101
24C7          IUSIGN=0
24CD          GO TO 102
24D0 101      U=-U
24DB          IUSIGN=1
24E2 102      IF (U.GT.1.D0)  GO TO 105
24F0          Y=U
24FA          A=0.D0
2501          KB=1
2508          GO TO 106
250B C
250B 105      Y=1.D0/U
2519          A=PIO2
2520          KB=-1
2527 106      IF (Y.GT.SQT2M1)  GO TO 109
2535          Y1=TPIO16
253F          ALPHA=PIO16
2546          GO TO 110
254A C
254A 109      Y1=T3PIO16
2554          ALPHA=THRPIO16
255B 110      X=(Y-Y1)/(1.0D0+Y*Y1)
2576          Z=X*X
2581          Z1=Z+B1
258C          Z2=Z+B2
2597          Z3=Z+B3
25A2          P1=C2*Z3
25AD          Q1=C3+Z2*Z3
25BC          P2=C1*Q1
```

```
25C7        Q2=P1+Z1*Q1
25D6        F3=C0+P2/Q2
25E5        ATANX=X*F3
25F0 C
25F0        ATAN2=ALPHA+ATANX
25FB        IF (KB.EQ.-1)  ATAN2=-ATAN2
2614        ATAN2=A+ATAN2
2622        IF (IUSIGN.EQ.1)  ATAN2=-ATAN2
263B        RETURN
2641 209    ATAN2=PIO2
264B        RETURN
2652        END
```

PROGRAM STORAGE: 0535

VARIABLE STORAGE: 0264

REFERENCES

1. Hall, David G. and Dale G. Bettis, "Optimal Runge-Kutta Method," (paper AA575-080 presented at the AAS/AIAA Astrodynamics Conference, Nassau, Bahamas, July 1975)

2. P. Henrici, Discrete Variable Methods in Ordinary Differential Equations. New York: John Wiley and Sons, 1962

3. CSC/SD-78/6131, System Description of the IMP-16C Microprocessor Orbit Determination Program, C. Shenitz, C. Rabbin, and G. Synder, October 1978

4. G. A. Wilkins, Editor, "Explanatory Supplement to the Astronomical Ephemeris," H. M. Stationery Office (1961), p. 98

5. Tektronix 8002 Microprocessor Laboratory System User's Manual, Tektronix, Inc., 1977

6. FORT/80 Language Manaual, Realistic Controls Corporation, Cleveland, Ohio

STL BIBLIOGRAPHY

Systems Technology Laboratory, STL-78-001, System Description of the IMP-16C Microprocessor Orbit Determination Program, C. Shenitz, C. Rabbin, and G. Snyder, October 1978

--, STL-78-002, Landsat/NAVPAC System Description and User's Guide, S. R. Waligora, December 1978

--, STL-79-001, Intel 8080 Orbit Propagation Program System Description and User's Guide, C. Rabbin, April 1979

--, STL-79-002, Evaluation of the IMP-16 Microprocessor Orbit Determination System Filter, C. Shenitz, September 1979

--, STL-79-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) MODCOMP Device and MAX IV Dependency Study, T. Weldon and M. McClellan, December 1979

--, STL-80-001, Orbit Determination Software Development for Microprocessor-Based Systems: Evaluation and Recommendations, C. M. Shenitz, July 1980

--, STL-80-002, The Two-Way TDRSS Observation Model for the LSI-11/23 Microcomputer, C. E. Goorevich, July 1980

--, STL-80-003, Automated Orbit Determination System (AODS) Requirements Definition and Analysis, S. R. Waligora, C. E. Goorevich, J. Teles, and R. S. Pajerski, September 1980

--, STL-80-004, Algorithms for Autonomous Star Identification, P. Gambardella, October 1980

--, STL-80-005, Autonomous Onboard Attitude Determination System Specifications and Requirements, M. D. Shuster, S. N. Ray, and L. Gunshol, December 1980

--, STL-81-001, Systems Technology Laboratory (STL) Library Methods and Procedures, W. J. Decker and P. D. Merwarth, September 1981

--, STL-81-002, Mathematical Specifications of the Onboard Navigation Package (ONPAC) Simulator (Revision 1), J. B. Dunham, February 1981

--, STL-81-003, Systems Technology Laboratory (STL) Compendium of Utilities, W. J. Decker, E. J. Smith, W. A. Taylor, P. D. Merwarth and M. E. Stark, July 1981

--, STL-81-004, Automated Orbit Determination System (AODS) Environment Simulator for Prototype Testing (ADEPT) System Description, S. R. Waligora, J. E. Fry, Jr., B. J. Prusiewicz, and G. N. Klitsch, August 1981

--, STL-81-104, Automated Orbit Determination System (AODS) Environment Simulator for Prototype Testing (ADEPT) System Description, S. R. Waligora, J. E. Fry, Jr., B. J. Prusiewicz, G. N. Klitsch, and Y. Ong, June 1982

--, STL-81-005, Preliminary Automated Orbit Determination System (AODS)/AODS Environment Simulator for Prototype Testing (ADEPT) User's Guide, S. R. Waligora, Y. Ong, J. E. Fry, Jr., and B. J. Prusiewicz, September 1981

--, STL-81-105, Preliminary Automated Orbit Determination System (AODS)/AODS Environment Simulator for Prototype Testing (ADEPT) User's Guide, S. R. Waligora, J. E. Fry, Jr., and Y. Ong, June 1982

--, STL-82-001, GPSPAC/Landsat-D Interface (GLI) System Description, J. B. Dunham, H. M. Sielski, and W. T. Wallace, April 1982

--, STL-82-002, GPSPAC/Landsat-D Interface (GLI) System User's Guide, H. M. Sielski, J. B. Dunham, and P. D. Merwarth, March 1982

--, STL-82-003, Autonomous Attitude Determination System (AADS), Volume 1: System Description, K. A. Saralkar, Y. G. Frenkel, G. N. Klitsch, K. Liu and E. Lefferts, April 1982

--, STL-82-007, System Description for the Global Positioning Subsystem Experiment Package (GPSPAC) Experiment Data Preprocessor (GEDAP), P. D. Merwarth and J. F. Cook, June 1982