NPS55-89-10

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A SATELLITE SIMULATION PROGRAM

REX H. SHUDDE

SEPTEMBER 1989

Prepared for:

Naval War College
Wargaming Department
Newport, RI 02841-5010

This work was supported by the Naval War College and was funded by the Naval Postgraduate School.

This report was prepared by:

[illegible]
REX H. SHUDDE, Associate Professor
Department of Operations Research

Reviewed by:

[illegible]
PETER PURDUE, Chairman
Department of Operations Research

Released by:

[illegible]
KNEALE T. MARSHALL
Acting Dean of Information and Policy

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS55-89-10	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL *(If applicable)* Code 55	7a. NAME OF MONITORING ORGANIZATION Naval War College, Wargaming Department
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000	7b. ADDRESS *(City, State, and ZIP Code)* Newport, RI 02841-5010
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL *(If applicable)* 55Su	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER O&MN, Direct Funding
8c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
A SATELLITE SIMULATION PROGRAM

12. PERSONAL AUTHOR(S) Rex H. Shudde

13a. TYPE OF REPORT Technical Report	13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT 51

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Satellite, Satellite Simulation, Satellite Vulnerability, Satellite Tracking, Satellite Orbit, Satellite Culmination, Ground Track

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report presents a satellite simulation algorithm written for the IBM PC and clones in Microsoft QuickBASIC 4.5. The program can either generate an ephemeris of satellite ground-track position and location with respect to a fixed ground station, or determine the position and time the satellite is above the horizon of a specified ground station. Provision is made for the program to handle multiple satellites and multiple ground stations. Input is either by way of the computer keyboard or by separate files for satellites and ground stations. Output is written both on the screen and to an ASCII output file which can be accessed by a data based management program.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Rex H. Shudde	22b. TELEPHONE *(Include Area Code)* (408) 646-2203 22c. OFFICE SYMBOL Code 55Su

A SATELLITE SIMULATION PROGRAM

by

Rex H. Shudde
NAVAL POSTGRADUATE SCHOOL
Monterey, California

September 1989

DISCLAIMER

ABSTRACT

This report presents a satellite simulation algorithm written for the IBM PC and clones in Microsoft QuickBASIC 4.5. The program can either generate an ephemeris of satellite ground-track position and location with respect to a fixed ground station, or determine the position and time the satellite is above the horizon of a specified ground station. Provision is made for the program to handle multiple satellites and multiple ground stations. Input is either by way of the computer keyboard or by separate files for satellites and ground stations. Output is written both on the screen and to an ASCII output file which can be accessed by a data based management program.

CONTENTS

Intentionally blank.

I. INTRODUCTION

This report presents a satellite simulation algorithm written for the IBM PC and clones in Microsoft QuickBASIC 4.5. The program can either generate an ephemeris of satellite ground-track position and location with respect to a fixed ground station, or determine the position and time the satellite is above the horizon of a specified ground station. Provision is made for the program to handle multiple satellites and multiple ground stations. Input is either by way of the computer keyboard or by separate files for satellites and ground stations. Output is written both on the screen and to an ASCII output file which can be accessed by a data based management program.

Two versions of the satellite kinematics subroutine `pos.update` are presented. These algorithms were provided by NAVSPASUR [Ref. 1]. The first is a very accurate algorithm based upon a paper by Brouwer [Ref. 2] and is contained in NAVSPASUR's FORTRAN program called SHOWALL. A listing of the BASIC version of this algorithm is presented in Appendix E. The second algorithm is a much simplified approximation to the Brouwer model. Equations for the simplified model are contained in Appendix A, and the BASIC implemention is embedded in the listing of the full simulation program in Appendix D.

The next sections contain program operating instructions, structure of the input/ output files, and suggestions for future improvements. The computation of the times of culmination, rise and set are in Appendix B, and the computation of satellite heading is in Appendix C. Readers unfamiliar with the nomenclature of satellite orbital elements should consult any standard work such as Escobal [Ref. 3, Ch. 3].

II. OPERATING INSTRUCTIONS

A. Overview. The source code is named **SATSTA.BAS** and the executable code is named **SATSTA.EXE**. To run, enter SATSTA at the DOS prompt and press the ↵ key.

The first menu (Fig. 1) requests a data input mode. Pressing 1 or 2 will require the data for one satellite and one ground station to be input from the keyboard. Pressing 3 or 4 assumes that two data files are accessible: either **ELEMENTS.DAT** or **ELEMENTS.NSS** containing the orbital elements for one or more satellites, and **STATION.DAT** containing the location of one or more ground stations—the structure of these files is discussed in the next section. No matter which of these four options is chosen, a simulation starting time, ending time and time-step increment will be required input from the keyboard in the third menu.

The second menu (Fig. 2) requests a run mode. Option 1 creates an ephemeris of satellite geographic position and satellite position with respect to the ground station for equally spaced times for the entire simulation duration. Option 2 computes the satellite geographic position and satellite position with respect to the ground station only for those times during the simulation duration at which the satellite is above the horizon of the ground station.

The third menu (Fig. 3) requests the simulation-times, consisting of a starting time, an ending time, and a time-step increment (Δt).

The fourth and fifth menus (Figs. 4 and 5, respectively) appear only if the first menu (data input mode) options 1 or 2 were chosen. Menu four requests the satellite epoch and elements, and menu five requests the station coordinates.

No matter which options are chosen, output is directed to the screen and is appended to file `OUTPUT.DAT`. If the file does not exist, it is created. This ASCII file is structured so that it can be used as input to a data base management program, if desired. The structure of this file is discussed in a later section.

B. Input. Menu 1 requests the selection of a data input mode. Depending upon the source of satellite elements, there are two ways in which these data can be input. The first is using standard "epoch of date" elements in which the time of perigee passage is the epoch. The second is the NAVSPASUR "One-Line Charlie" elements in which the longitude (or right ascension) of the ascending node has been referenced to 0000 hours, 1 January 1985. The "One-Line Charlie" option has been added for those users who may obtain satellite elements from NAVSPASUR. An example of "epoch of date" input is shown in Fig. 3.

```
Select data input mode 1, 2, 3 or 4:

     Keyboard input for satellite & ground station:
          1. Epoch of date.
          2. NAVSPASUR Reference Date.

     Data file input for satellite & ground station:
          3. Epoch of date.
          4. NAVSPASUR One-Line Charlie.
```

Figure 1. Menu 1—Select data input mode.

If data file input Option 3 is chosen from Menu 1, two ASCII data files, `ELEMENTS.DAT` and `STATION.DAT`, must have been previously created. The `ELEMENTS.DAT` file must contain a single line of for each satellite of interest. Each line must contain 10 entries, each enclosed in double quotations and separated by a blank space (Fig. 6). The entries are: (1) a satellite identification consisting of at most five alphanumeric characters; (2) the epoch date (date of perigee passage) in the form `yyyy.mmdd`, where `yyyy` is the year, `mm` is the month, and `dd` is the day; (3) the epoch time (time of perigee passage) in the form `hh.mmss`, where `hh` is the hour, `mm` is the minutes, and `ss` is the seconds; (4) the orbital eccentricity; (5) the longitude (or right ascension) of the ascending node in degrees; (6) the orbital inclination in degrees; (7) the argument of perigee in degrees; (8) the mean anomaly in degrees; (9) the mean motion in revolutions per day; and (10) the orbital decay in revolutions per day-squared.

If data file input Option 4 is chosen from Menu 1, two ASCII data files, `ELEMENTS.NSS` and `STATION.DAT`, must have been previously created. The `ELEMENTS.NSS` file must contain a single line of for each satellite of interest in the NAVSPASUR format. Each line must contain 10 entries starting in column 1 and ending in column 65 (Fig. 7). The entries are: (1) a satellite identification consisting of at most five alphanumeric characters (columns 1–5); (2) the mean anomaly in revolutions (columns 6–13); (3) the mean motion in radians per Herg[1] (columns 14–21); (4) the orbital decay in radians per Herg-squared (columns 22–27); (5) the orbital eccentricity (columns 28–35); (6) the argument of perigee in revolutions (columns 36–43); (7) the longitude (or right ascension) of the ascending node in revolutions (columns 44–51); (8) the orbital inclination in revolutions (columns 52–59); and (9) the epoch date (date of perigee passage) in the form `yymmdd`, where `yyyy` is the year, `mm` is the month, and `dd` is the day (columns 60–65). A decimal point must be present in columns 6, 14, 22, 28, 36, 44 and 52.

```
Select run mode 1 or 2:

     1. Create an ephemeris.
     2. Find times satellite is above the horizon.
```

Figure 2. Menu 2—Select run mode.

The `STATION.DAT` file must contain a single line of for each ground station or geographical location of interest. Each line must contain 5 entries, each enclosed in double quotations and separated by a blank space (Fig. 8). The entries are: (1) a station identification consisting of at most three alphanumeric characters; (2) the station latitude in the form `dd.mmss` (minus if south), where `dd` is degrees, `mm` is minutes, and `ss` is seconds; (3) the station longitude in the form `ddd.mmss` (minus if west); (4) the station altitude in feet (minus if below sea level); and (5) the name of the station or station location (alphanumeric).

If a data file input option is chosen, then SATSTA will open the two files described above and compute either an ephemeris or the times that each satellite is above the horizon (Option 1 or 2 of the second menu) for each ground station. Otherwise, keyboard input for one satellite and one ground station will be requested from the fourth and fifth menues, respectively.

[1] The Herg—a unit of time named after the astronomer Paul Herget—is defined as the period, divided by 2π, of a fictitious satellite at zero altitude over a smooth spherical earth with no atmosphere.

```
Simulation-Time Data:

Starting date (yyyy.mmdd)    ? 1983.0201
Starting time (hh.mmss)      ? 00.00
Ending date (yyyy.mmdd)      ? 1983.0202
Ending time (hh.mmss)        ? 00.00
Time increment (minutes)     ? 01

Press any key to continue.
```

Figure 3. Menu 3—Select simulation times.

Five pieces of information regarding the simulation are requested from the third menu (Fig. 3): (1) the simulation starting date in the form **yyyy.mmdd**, (2) the simulation starting time in the form hh.mmss; (3) the simulation ending date in the form yyyy.mmdd; (4) the simulation ending time in the form hh.mmss; and (5) the simulation time-step increment in minutes. The screen will display sample data for each entry in the appropriate format. If any sample entry is to be changed, it should be *completely* overwritten in order for it to be read in properly. When each entry is correct, enter it into the computer by pressing the ↵ key.

If the keyboard entry option was chosen from the first menu, then two more menues will appear, otherwise the simulation will commence with input being obtained from the two data files.

```
Satellite Data:

Epoch date (yyyy.mmdd)       ? 1983.0201
Epoch time (hh.mmss)         ? 00.00

Input mean elements of epoch:
Satellite ID Number          ? 11111
Eccentricity                 ? 0.0005545
Long. ascending node (deg)   ? 272.43497
Inclination (deg)            ? 65.06057
Arg. of perigee (deg)        ? 295.41470
Mean anomaly (deg)           ? 258.10682
Mean motion (rev/day)        ? 15.44194
Decay (rev/day^2)            ? 0

Press any key to continue.
```

Figure 4. Menu 4—Input satellite data.

Menu 4 (Fig. 4) requires ten separate entries: (1) the epoch date (date of perigee passage) in the form yyyy.mmdd, where yyyy is the year, mm is the month, and dd is the day; (2) the epoch time (time of perigee passage) in the form hh.mmss, where hh is the hour, mm is the minutes, and ss is the seconds; (3) a satellite identification consisting of at most five alphanumeric characters; (4) the orbital eccentricity; (5) the longitude (or right ascension) of the ascending node in degrees; (6) the orbital inclination in degrees; (7) the argument of perigee in degrees; (8) the mean anomaly in degrees; (9) the mean motion in revolutions per day; and (10) the orbital decay in revolutions per day-squared. The screen will display sample data for each entry in the appropriate format. Again, if any sample entry is to be changed, it should be *completely* overwritten in order for it to be read in properly. When each entry is correct, enter it into the computer by pressing the ↵ key.

```
Station Coordinates:

Station Identification Number            ? 001
Station Location or Name                 ? Daisy, Tenn.
Station latitude  (dd.mmss, South minus) ? 35.12
Station longitude (ddd.mmss, West minus) ? -85.12
Station altitude  (feet)                 ? 500.0

Press any key to continue.
```

Figure 5. Menu 5—Input ground station data.

Menu 5 (Fig. 5) requires five separate entries: (1) a station identification consisting of at most three alphanumeric characters; (2) a station location or name; (3) the station latitude in the form `dd.mmss` (minus if south); (4) the station longitude in the form `ddd.mmss` (minus if west); and (5) the station altitude in feet (minus if below sea level).

C. Output. Output is written to the screen and is appended to the data file OUTPUT.DAT if it exists, otherwise file OUTPUT.DAT is created. A sample from the OUTPUT.DAT file is shown in Fig. 9—the structure of this file is discussed later.

Sample output from the "create an ephemeris" option screen is shown in Fig. 10. The apogee/perigee output is approximately valid only for the first orbit. In particular, the longitude is not corrected for the earth's rotation during the current orbit. Also, the ascending node and argument of perigee are not corrected for their rates of change during the current orbit. It should be further noted that for nearly circular orbits, the height of the satellite will probably get lower than the perigee value and higher than the apogee value at locations other than the perigee and apogee positions—this is an effect of the earth's oblateness.

The main body of the ephemeris output consists of date, time, satellite latitude, longitude and height, followed by the elevation angle, azimuth, and range (in kilometers) of the satellite from the station. The satellite "look angle" is the angle between the satellite "ground-zero" point (usually called the satellite "sub-point") and the station. The last column of output is the satellite heading.

A sample of the output from the "times satellite is above the horizon" option screen for a time-step (Δt) of 1 minute is shown in Fig. 11. First the time of culmination for the current orbit is computed and saved. The culmination time is then incremented and decremented by Δt until the satellite is below the horizon. The time of rise and set is

```
"11111" "1983.0201" "00.00" "0.0005545" "272.43497" "65.06057" "295.41470" "258.10682" "15.44194" "0"
"22222" "1983.0201" "00.00" "0.0023064" "55.552464" "64.94868" "202.277268" "126.670896" "16.06302" "4.281119d-3"
```

Figure 6. Sample `elements.dat` file.

```
11111.7169634.9060250.00000.0005545.8205964.7567638.1807238830201
22222.3518636.9424655.23456.0023064.5618813.1543124.1804130830201
```

Figure 7. Sample `elements.nss` file.

```
"001" "35.12" "-85.12" "500.0" "Daisy, Tenn."
"002" "19.44" "-155.05" "300.0" "Hilo, Hawaii"
```

Figure 8. Sample `station.dat` file.

```
"001" "11111" "1983" " 2" " 1" " 1" "13" "19.4" " 54.37" "263.10" " 450.1" "  0.00" "340.48" " 2436" "69.23" "135.43"
"001" "11111" "1983" " 2" " 1" " 1" "18" "42.1" " 38.12" "280.62" " 443.8" " 32.24" " 55.33" "  773" "52.35" "150.23"
"001" "11111" "1983" " 2" " 1" " 1" "24" " 3.8" " 19.99" "290.85" " 437.5" " -0.00" "132.74" " 2400" "69.25" "156.58"
"001" "11111" "1983" " 2" " 1" " 2" "49" "40.9" " 45.38" "250.95" " 446.7" "  0.00" "306.66" " 2428" "69.23" "145.41"
"001" "11111" "1983" " 2" " 1" " 2" "54" "12.9" " 30.63" "261.84" " 441.0" " 11.96" "251.45" " 1429" "66.15" "153.58"
"001" "11111" "1983" " 2" " 1" " 2" "58" "44.7" " 15.05" "269.44" " 436.2" " -0.00" "195.12" " 2394" "69.26" "157.45"
"001" "22222" "1983" " 2" " 1" " 1" "42" "46.3" " 19.63" "268.42" " 273.2" "  0.00" "201.99" " 1883" "73.38" " 23.59"
"001" "22222" "1983" " 2" " 1" " 1" "47" " 1.1" " 34.59" "276.49" " 277.0" " 58.93" "111.16" "  321" "29.58" " 28.27"
"001" "22222" "1983" " 2" " 1" " 1" "51" "14.7" " 48.49" "288.39" " 280.2" " -0.00" " 32.74" " 1910" "73.46" " 37.64"
"001" "22222" "1983" " 2" " 1" " 3" "16" "59.0" " 35.68" "254.51" " 277.3" "  0.00" "277.85" " 1902" "73.40" " 28.78"
"001" "22222" "1983" " 2" " 1" " 3" "19" "25.5" " 43.84" "260.89" " 279.3" "  3.08" "313.11" " 1595" "73.18" " 33.65"
"001" "22222" "1983" " 2" " 1" " 3" "21" "50.0" " 51.32" "269.26" " 280.7" " -0.00" "347.67" " 1910" "73.47" " 40.74"
"001" "22222" "1983" " 2" " 1" " 9" "36" "38.1" " 48.57" "287.14" " 264.2" "  0.00" " 30.30" " 1853" "73.93" "142.26"
"001" "22222" "1983" " 2" " 1" " 9" "37" "38.2" " 45.41" "290.53" " 261.9" "  0.43" " 44.49" " 1798" "73.96" "145.11"
"001" "22222" "1983" " 2" " 1" " 9" "38" "34.4" " 42.35" "293.35" " 259.8" " -0.00" " 57.82" " 1839" "73.99" "147.39"
"002" "11111" "1983" " 2" " 1" " 5" "57" "53.0" " 40.06" "208.38" " 444.6" "  0.00" "  7.44" " 2417" "69.34" "149.12"
"002" "11111" "1983" " 2" " 1" " 6" " 2" "14.5" " 25.51" "217.41" " 439.2" "  9.78" " 60.03" " 1556" "67.27" "155.24"
"002" "11111" "1983" " 2" " 1" " 6" " 6" "34.7" " 10.45" "224.16" " 435.2" " -0.00" "112.97" " 2395" "69.37" "158.01"
"002" "11111" "1983" " 2" " 1" " 7" "32" "51.5" " 34.44" "188.75" " 442.4" "  0.00" "319.38" " 2413" "69.35" "152.04"
"002" "11111" "1983" " 2" " 1" " 7" "38" " 0.4" " 16.85" "197.87" " 436.6" " 24.13" "250.20" "  936" "58.62" "157.16"
"002" "11111" "1983" " 2" " 1" " 7" "43" " 8.8" " -1.18" "205.16" " 433.8" " -0.00" "179.39" " 2385" "69.39" "158.52"
"002" "22222" "1983" " 2" " 1" " 4" "39" "23.2" "  9.94" "218.83" " 271.1" "  0.00" "123.17" " 1877" "73.53" " 22.18"
"002" "22222" "1983" " 2" " 1" " 4" "40" "29.5" " 13.95" "220.51" " 271.9" "  0.58" "107.75" " 1818" "73.52" " 22.64"
"002" "22222" "1983" " 2" " 1" " 4" "41" "36.5" " 17.98" "222.27" " 272.8" " -0.00" " 92.22" " 1885" "73.52" " 23.27"
"002" "22222" "1983" " 2" " 1" " 6" " 8" " 3.1" "  6.39" "194.69" " 270.5" "  0.00" "218.92" " 1873" "73.54" " 21.90"
"002" "22222" "1983" " 2" " 1" " 6" "12" " 9.5" " 21.23" "201.07" " 273.6" " 28.62" "296.75" "  536" "57.38" " 23.93"
"002" "22222" "1983" " 2" " 1" " 6" "16" "16.0" " 35.66" "209.09" " 277.3" " -0.00" " 11.90" " 1896" "73.54" " 28.77"
```

Figure 9. Sample `output.dat` file (some output expunged). Epoch of date.

```
Satellite: 11111     Station: 001   Daisy, Tenn.


Apogee:
Height     =   442.8  km.
Latitude   =   -55.1540
Longitude  =   100.2099

Perigee:
Height     =   450.4  km.
Latitude   =    55.1539
Longitude  =   280.2099


Tue 1 Feb 1983
                                           height                                 look
year  mo  da  hr  mn   sec     lat    long   (km)    elev    azim   range  angle   head

1983   2   1   0   0   0.0  -12.26  327.56  434.0  -31.60  123.30    7437  52.90  157.82
1983   2   1   0  10   0.0  -45.84  347.68  441.5  -50.59  136.39   10382  36.56  145.02
1983   2   1   0  20   0.0  -65.19   48.39  446.4  -69.07  151.85   12354  19.68   88.46
1983   2   1   0  30   0.0  -44.71  106.99  439.1  -83.25  221.39   13089   6.45   34.07
1983   2   1   0  40   0.0  -10.95  126.55  430.7  -70.51  302.57   12481  18.11   22.04
1983   2   1   0  50   0.0   24.08  141.07  435.9  -51.76  316.67   10574  35.26   24.38
1983   2   1   1   0   0.0   55.55  168.97  449.2  -32.06  324.49    7561  52.31   46.47
```

Figure 10. Output from ephemeris option. Epoch of date.

```
Satellite: 11111     Station: 001   Daisy, Tenn.

                                            height                                 look
year  mo  da  hr  mn   sec     lat    long    (km)    elev    azim   range  angle   head

1983   2   1   1  13  19.4   54.37  263.10   450.1    0.00  340.48    2436  69.23  135.43
1983   2   1   1  13  42.1   53.35  264.76   449.7    1.39  341.68    2285  69.19  136.92
1983   2   1   1  14  42.1   50.53  268.75   448.7    5.55  345.74    1892  68.57  140.46
1983   2   1   1  15  42.1   47.58  272.27   447.6   10.78  351.87    1514  66.79  143.50
1983   2   1   1  16  42.1   44.51  275.37   446.4   17.70    2.15    1167  63.08  146.09
1983   2   1   1  17  42.1   41.35  278.14   445.1   26.48   21.39     892  56.95  148.32
1983   2   1   1  18  42.1   38.12  280.62   443.8   32.24   55.33     773  52.35  150.23
1983   2   1   1  19  42.1   34.83  282.88   442.6   27.03   90.21     875  56.40  151.86
1983   2   1   1  20  42.1   31.49  284.94   441.3   18.10  110.45    1141  62.69  153.26
1983   2   1   1  21  42.1   28.11  286.84   440.1   10.97  121.17    1484  66.60  154.46
1983   2   1   1  22  42.1   24.69  288.61   438.9    5.61  127.48    1862  68.51  155.46
1983   2   1   1  23  42.1   21.25  290.27   437.9    1.36  131.58    2255  69.20  156.31
1983   2   1   1  24   3.8   19.99  290.85   437.5   -0.00  132.74    2400  69.25  156.58
```

Figure 11. Output from "times above horizon" option. $\Delta t = 1$ minute. Epoch of date.

iterated until the time at which the satellite is on the horizon. All of these times and positions are then output to the screen. Note that the information printed out in this option, with the exception of the apogee/perigee positions, is identical to that printed out in the "ephemeris" option. By making Δt large, say 10 minutes, in the "times above horizon" option, then only the times of satellite rise, culmination, and set are computed and output (see Fig. 12).

Each line in the OUTPUT.DAT file consists of the station identifier and the satellite identifier followed by the same data, and in the same order, as that printed to the screen. This can be seen by comparing the first six lines of Fig. 12 with the same lines in Fig. 9.

```
Satellite: 11111     Station: 001   Daisy, Tenn.

                                         height                        look
year mo da hr mn   sec    lat    long     (km)    elev    azim  range angle   head

1983  2  1  1 13 19.4  54.37 263.10    450.1   0.00 340.48   2436 69.23 135.43
1983  2  1  1 18 42.1  38.12 280.62    443.8  32.24  55.33    773 52.35 150.23
1983  2  1  1 24  3.8  19.99 290.85    437.5  -0.00 132.74   2400 69.25 156.58

1983  2  1  2 49 40.9  45.38 250.95    446.7   0.00 306.66   2428 69.23 145.41
1983  2  1  2 54 12.9  30.63 261.84    441.0  11.96 251.45   1429 66.15 153.58
1983  2  1  2 58 44.7  15.05 269.44    436.2  -0.00 195.12   2394 69.26 157.45

1983  2  1 14 47 13.6  16.76 285.33    433.6   0.00 150.29   2387 69.32  22.83
1983  2  1 14 50 58.0  29.64 291.62    437.9   6.44 106.44   1792 68.35  26.06
1983  2  1 14 54 43.3  42.02 300.04    443.2  -0.00  63.10   2419 69.28  32.12

1983  2  1 16 20 48.3  17.88 262.22    433.9   0.00 216.51   2389 69.32  23.02
1983  2  1 16 26 16.7  36.50 272.26    440.8  55.22 303.76    528 32.33  28.94
1983  2  1 16 31 46.6  53.37 289.30    448.2  -0.00  24.72   2431 69.26  43.11

1983  2  1 17 59 48.3  37.38 249.28    441.2   0.00 283.77   2414 69.29  29.38
1983  2  1 18  2 51.4  47.05 257.60    445.5   3.50 317.85   2066 69.01  36.02
1983  2  1 18  5 55.6  55.73 269.75    449.2  -0.00 351.87   2433 69.26  46.77
```

Figure 12. Output from "times above horizon" option. $\Delta t = 10$ minutes. Epoch of date.

Fig. 13 is a sample output obtained using the NAVSPASUR reference time. A comparison of Fig. 10 and Fig. 13 shows that the satellite latitude, height, and heading is the same but the longitude differs because of the difference of the earth's rotation between the two different reference times. The satellite-station relationships differ for the same reason. Figures 12 and 14 are completely dissimilar because of the differing satellite-station relationships.

```
Satellite: 11111     Station: 001   Daisy, Tenn.

Apogee:
Height     =   442.8  km.
Latitude   =   -55.1540
Longitude  =   230.8483

Perigee:
Height     =   450.4  km.
Latitude   =    55.1539
Longitude  =    50.8483

Tue 1 Feb 1983
                                         height                        look
year mo da hr mn   sec    lat    long     (km)    elev    azim  range angle   head

1983  2  1  0  0   0.0 -12.26   98.20    434.0 -77.96 351.78  12919 11.09 157.82
1983  2  1  0 10   0.0 -45.84  118.32    441.5 -79.40 232.53  12965 10.02 145.02
1983  2  1  0 20   0.0 -65.19  179.03    446.4 -60.98 210.44  11620 27.15  88.46
1983  2  1  0 30   0.0 -44.71  237.62    439.1 -41.32 205.69   9023 44.80  34.07
1983  2  1  0 40   0.0 -10.95  257.18    430.7 -20.38 203.40   5458 61.42  22.04
1983  2  1  0 50   0.0  24.08  271.70    435.9  12.73 194.92   1373 65.77  24.38
1983  2  1  1  0   0.0  55.55  299.61    449.2  -5.07  32.07   3062 68.64  46.47
```

Figure 13. Output from ephemeris option. NAVSPASUR reference time.

```
Satellite: 11111     Station: 001   Daisy, Tenn.

                                            height                           look
year mo da hr mn  sec    lat    long         (km)    elev    azim  range angle  head

1983  2  1  0 47 32.1  15.54 267.79        433.2   0.00 199.85  2385 69.32  22.63
1983  2  1  0 52 59.8  34.24 277.36        439.8  58.70 112.56   509 29.01  27.87
1983  2  1  0 58 28.6  51.41 293.03        447.4  -0.00  33.16  2429 69.27  40.56

1983  2  1  2 24 57.1  29.86 251.15        438.0   0.00 262.00  2405 69.30  26.14
1983  2  1  2 29 10.7  43.72 260.91        443.9   8.69 312.74  1641 67.65  33.31
1983  2  1  2 33 25.7  55.99 276.83        449.3  -0.00   3.07  2433 69.26  47.22

1983  2  1  8 58 59.6  56.01 272.80        450.7   0.00 356.74  2437 69.23 132.74
1983  2  1  9  3 16.0  43.68 288.80        446.0   8.83  47.32  1637 67.55 146.72
1983  2  1  9  7 31.7  29.71 298.62        440.7  -0.00  98.40  2412 69.24 153.92

1983  2  1 10 33 57.2  51.41 256.65        449.0   0.00 326.74  2433 69.23 139.44
1983  2  1 10 39 26.8  34.21 272.34        442.3  58.17 247.39   514 29.48 152.14
1983  2  1 10 44 55.7  15.45 281.93        436.3  -0.00 160.33  2394 69.26 157.39

1983  2  1 12 13 31.5  31.47 250.42        441.3   0.00 266.88  2414 69.24 153.27
1983  2  1 12 14 27.8  28.30 252.20        440.2   0.29 256.89  2378 69.24 154.39
1983  2  1 12 15 23.2  25.15 253.85        439.1  -0.00 247.03  2407 69.25 155.34
```

Figure 14. Output from "times above horizon" option. $\Delta t = 10$ minutes. NAVSPASUR reference time.

III. SOURCES

As mentioned in the introduction, the high resolution and simplified satellite kinematics models were obtained from NAVSPASUR [Ref. 1]. The high resolution model is based upon a paper by Brouwer [Ref. 2]. The code for the high resolution model is given Appendix E, while the code for the simplified model is contained in the `pos.update` subroutine in Appendix D. Both of these routines require a solution of Kepler's equation. A non-iterative state-of-the-art solution of Kepler's equation, based upon a paper by Mikkola [Ref. 6] is contained in subroutine `kepler` listed in Appendix D.

The ground track determination is essentially that given by Escobal [Ref. 3, Ch. 10.3.2] but with a minor improvement described by Shudde [Ref. 7].

The method of computation of the times of satellite culmination, rise, and set are given in Appendix B, and the computation of the satellite heading is contained in Appendix C. Additional references are given there.

The IAU (1976) System of Astronomical Constants [Ref. 10, pg. S7] is used in SATSTA as well as the J2000.0 reference frame [Ref. 10, pg. L1]. Satellite positions computed by NAVSPASUR's SHOWALL [Ref. 1] use the WGS 72 constants and the J1900.0 reference frame. For this reason, the agreement between SATSTA and SHOWALL is not exact.

IV. DISCUSSION

Although the position updating in SATSTA is based upon the satellite kinematics model of Brouwer [Ref. 2], other models are currently in use. Most notable is the work of Hoots and Roehrich [Ref. 5]. SATRAK [Ref. 4] is a FORTRAN program implementing their models and is available from the Aerospace Defense Command, Peterson AFB. Unfortunately, the source code is not available for modification.

The program presented here can be used as a basis for a more sophisticated model. The input scheme could be improved by using an input editor to catch improper types of entry and to allow corrections or changes to be made to previous entries. File management programs can be used to select satellite elements and station parameters from master files and insert them into "working" files for program input.

Various sensor types have not been modelled and no provision has been made for including them in the program. Also, satellite-satellite visablity has not been modelled. If such features are desirable, they could be included by modifying the existing program.

APPENDIX A. THE SIMPLIFIED KINEMATICS ALGORITHM.

The computation of the perturbed satellite orbit about the earth is performed using a method which is an approximation of that used by NAVSPASUR [Ref. 1]. According to NAVSPASUR, this approximation will yield satellite positons with a random error of about 10–15 nautical miles for a 20 day interval around epoch.

The mean elements at epoch, which must be input, are:

Eccentricity e_0

Right ascension of the ascending node Ω_0

Inclination i_0

Argument of perigee ω_0

Mean anomaly M_0

Mean motion M_1, and

Decay M_2

Using the mean elements, the following auxillary quantities are computed:

$$a_0 = M_1^{-2/3}, \tag{A-1}$$

$$a_0 = a_0 \left[1 + \frac{C(3\cos^2 i_0 - 1)}{a_0^2(1-e_0^2)^{3/2}} \right]^{2/3}, \tag{A-1a}$$

$$\dot{\omega} = C\frac{5\cos^2 i_0 - 1}{a_0^2(1-e_0^2)^2}, \tag{A-2}$$

$$\dot{\Omega} = -2C\frac{\cos i_0}{a_0^2(1-e_0^2)^2}, \quad \text{and} \tag{A-3}$$

$$\dot{a} = -\frac{4a_0}{3M_1}M_2, \tag{A-4}$$

where $C = {}^3/_4 J_2$ is related to the oblateness of the earth. NAVSPASUR states the following:

For optimal accuracy, a_0 from Equation A–1 is used in Equation A–1a to obtain a final value for a_0. This latter a_0 is then used throughout. Where program size is a major consideration and additional errors of some ten miles in satellite height can be tolerated the user may choose to leave out Equation A–1a and/or Equation A–4.

Using the mean values of the elements at epoch T_0 and the results of the above calculations, the mean values of the elements for any time t are:

$$M = M_0 + M_1(t - T_0) + M_2(t - T_0)^2,$$

$$e = e_0,$$

$$\omega = \omega_0 + \dot{\omega}(M - M_0),$$

$$\begin{aligned}\Omega &= \Omega_0 + \dot{\Omega}(M - M_0) - \omega_e(t - T_0),\\ i &= i_0, \quad \text{and}\\ a &= a_0 + \dot{a}(t - T_0),\end{aligned}$$

where ω_e is the earth's sidereal rotation rate.

APPENDIX B. THE TIME OF CULMINATION, RISE AND SET ALGORITHM.

In this program the earth is considered to be non-rotating, *i.e.*, the coördinates of a station are given by latitude and longitude with the longitude unaffected by the earth's rotation. Rotation is accomplished by subtracting the accrued rotation from the right ascension of the ascending node of the satellite. Although this departs from realism, the benefit is that the rectangular coördinates of the station need to be calculated only once.

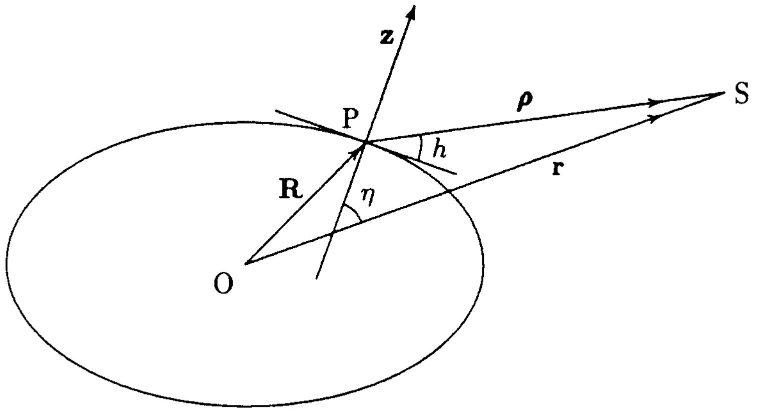

Fig. B–1. Satellite/Station Geometry.

Referring to Fig. B–1, culmination of the satellite, S, with respect to a ground station, P, occurs when the elevation angle, h, of the satellite is a maximum, which is equivalent to the minimization of the of the zenith angle, $90° - h$, which is also equivalent to the minimization of the angle η between **z**, the unit vector normal to the station location, and **r**, the vector from the earth's center, O, to the satellite. The equation for η is

$$\cos\eta = \frac{\mathbf{z}\cdot\mathbf{r}}{r}, \qquad \text{(B–1)}$$

where **z** is given by

$$\mathbf{z} = \begin{bmatrix} \cos\phi\cos\lambda \\ \cos\phi\sin\lambda \\ \sin\phi \end{bmatrix}, \qquad \text{(B–2)}$$

and ϕ and λ are the station geodetic latitude and longitude, respectively. The satellite position vector **r** is obtained from [Ref. 3, Equ. (3.57)]

$$\mathbf{r} = x_\omega\mathbf{P} + y_\omega\mathbf{Q}, \qquad \text{(B–3)}$$

where x_ω and y_ω measure the satellite position in the orbital plane.[1] **P** and **Q** are the first two columns of the Euler matrix [Ref. 3, Equs. (3.43) and (3.44)]—they transform the in-plane x_ω and y_ω coördinates to the equatorial rectangular system and are given by

$$\mathbf{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} \cos\omega\cos\Gamma - \sin\omega\sin\Gamma\cos i \\ \cos\omega\sin\Gamma + \sin\omega\cos\Gamma\cos i \\ \sin\omega\sin i \end{bmatrix} \tag{B–4a}$$

and

$$\mathbf{Q} = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix} = \begin{bmatrix} -\sin\omega\cos\Gamma - \cos\omega\sin\Gamma\cos i \\ -\sin\omega\sin\Gamma + \cos\omega\cos\Gamma\cos i \\ \cos\omega\sin i \end{bmatrix}, \tag{B–4b}$$

where ω is the argument of perigee and i is the orbital inclination; $\Gamma = \Omega - \omega_e(t - t_0)$, where Ω is the longitude of the ascending node at the time, t_0, of the last element update; ω_e is the earth's sidereal rotation rate and t is the current time.

The components of **r** vary with time. We will transform the equations in such a way that will replace time by the eccentric anomaly E as the independent variable. The inclination i is time independent and the elements Ω and ω will be considered to be independent of time over the period of a single orbit even though they are time dependent through perturbations. The components of **r** which will be considered to be time dependent are then x_ω, y_ω and Γ.

Write Equation (B–1) as

$$\mathbf{z} \cdot \mathbf{r} - r\cos\eta = 0.$$

Take the derivative with respect to E to obtain,

$$\mathbf{z} \cdot \frac{d\mathbf{r}}{dE} - \frac{dr}{dE}\cos\eta + r\sin\eta\frac{d\eta}{dE} = 0.$$

To find the value of E which minimizes η, set $d\eta/dE = 0$. Thus,

$$\mathbf{z} \cdot \frac{d\mathbf{r}}{dE} - \frac{dr}{dE}\cos\eta = 0,$$

or

$$\mathbf{z} \cdot \frac{d\mathbf{r}}{dE} - \frac{1}{r}\frac{dr}{dE}\mathbf{z} \cdot \mathbf{r} = 0.$$

Since this equation cannot be solved explicitly for E, set

$$F(E) = \mathbf{z} \cdot \frac{d\mathbf{r}}{dE} - \frac{1}{r}\frac{dr}{dE}\mathbf{z} \cdot \mathbf{r} \tag{B–5}$$

[1]The origin of the system is at the focus of the orbit. x_ω is positive in the direction of the perifocus. y_ω lies in the orbital plane and is orthogonal to x_ω.

and find E such that $F(E) = 0$ by the Newton-Raphson method. The Newton-Raphson procedure also requires $F'(E)$, the derivative of $F(E)$ with respect to E:

$$F'(E) = \mathbf{z} \cdot \frac{d^2\mathbf{r}}{dE^2} + \frac{1}{r^2}\left(\frac{dr}{dE}\right)^2 - \frac{1}{r}\frac{d^2 r}{dE^2}\,\mathbf{z}\cdot\mathbf{r} - \frac{1}{r}\frac{dr}{dE}\,\mathbf{z}\cdot\frac{d\mathbf{r}}{dE}. \tag{B–6}$$

The first and second derivatives of Equation (B–3) are

$$\frac{d\mathbf{r}}{dE} = \frac{dx_\omega}{dE}\mathbf{P} + x_\omega\frac{d\mathbf{P}}{dE} + \frac{dy_\omega}{dE}\mathbf{Q} + y_\omega\frac{d\mathbf{Q}}{dE} \tag{B–7a}$$

and

$$\frac{d^2\mathbf{r}}{dE^2} = \frac{d^2x_\omega}{dE^2}\mathbf{P} + 2\frac{dx_\omega}{dE}\frac{d\mathbf{P}}{dE} + x_\omega\frac{d^2\mathbf{P}}{dE^2} + \frac{d^2y_\omega}{dE^2}\mathbf{Q} + 2\frac{dy_\omega}{dE}\frac{d\mathbf{Q}}{dE} + y_\omega\frac{d^2\mathbf{Q}}{dE^2}, \tag{B–7b}$$

where [Ref. 3, Equ. (3.69)]

$$\begin{aligned} x_\omega &= a(\cos E - e) \quad \text{and} \\ y_\omega &= a\sqrt{1-e^2}\sin E, \end{aligned} \tag{B–8a}$$

with first and second derivatives

$$\begin{aligned} \frac{dx_\omega}{dE} &= -a\sin E, \\ \frac{dy_\omega}{dE} &= a\sqrt{1-e^2}\cos E, \end{aligned} \tag{B–8b}$$

and

$$\begin{aligned} \frac{d^2y_\omega}{dE^2} &= -a\sqrt{1-e^2}\sin E = -y_\omega, \\ \frac{d^2x_\omega}{dE^2} &= -a\cos E. \end{aligned} \tag{B–8c}$$

The first and second derivatives of Equations (B–4) are

$$\begin{aligned} \frac{dP_x}{dE} &= -P_y\frac{d\Gamma}{dE}, \\ \frac{dP_y}{dE} &= P_x\frac{d\Gamma}{dE}, \\ \frac{dP_z}{dE} &= 0, \\ \frac{dQ_x}{dE} &= -Q_y\frac{d\Gamma}{dE}, \\ \frac{dQ_y}{dE} &= Q_x\frac{d\Gamma}{dE}, \\ \frac{dQ_z}{dE} &= 0, \end{aligned} \tag{B–9a}$$

and

$$\begin{aligned}
\frac{d^2P_x}{dE^2} &= -P_x\left(\frac{d\Gamma}{dE}\right)^2 - P_y\frac{d^2\Gamma}{dE^2},\\
\frac{d^2P_y}{dE^2} &= -P_y\left(\frac{d\Gamma}{dE}\right)^2 + P_x\frac{d^2\Gamma}{dE^2},\\
\frac{d^2P_z}{dE^2} &= 0,\\
\frac{d^2Q_x}{dE^2} &= -Q_x\left(\frac{d\Gamma}{dE}\right)^2 - Q_y\frac{d^2\Gamma}{dE^2},\\
\frac{d^2Q_y}{dE^2} &= -Q_y\left(\frac{d\Gamma}{dE}\right)^2 + Q_x\frac{d^2\Gamma}{dE^2}, \quad \text{and}\\
\frac{d^2Q_z}{dE^2} &= 0.
\end{aligned} \tag{B–9b}$$

Also required is [Ref. 3, Equ. (3.67)]

$$r = a(1 - e\cos E) \tag{B–10a}$$

and it's derivatives

$$\begin{aligned}
\frac{dr}{dE} &= ae\sin E, \quad \text{and}\\
\frac{d^2r}{dE^2} &= ae\cos E.
\end{aligned} \tag{B–10b}$$

Finally, using Kepler's equation [Ref. 3, Equ. (3.79)],

$$t = \frac{E - e\sin E}{n} + T, \tag{B–11}$$

where n is the rate of orbital motion and T is the time of last perifocal passage, the equation for Γ can be written as

$$\Gamma = \Omega - \omega_e\left(\frac{E - e\sin E}{n} + T - t_0\right). \tag{B–12a}$$

Then,

$$\frac{d\Gamma}{dE} = -\omega_e\left(\frac{1 - e\cos E}{n}\right), \tag{B–12b}$$

and

$$\frac{d^2\Gamma}{dE^2} = \frac{-\omega_e(e\sin E)}{n}. \tag{B–12c}$$

To solve Equation (B–5), an initial estimate of E is made and the right-hand side of Equation (B–13) is evaluated. The result, the left-hand side of Equation (B–13), is an improved estimate of E,

$$E = E - \frac{F(E)}{F'(E)}. \tag{B–13}$$

This iteration is continued until $|F(E)/F'(E)|$ is less than some prescribed amount. The value of E obtained will either maximize or minimize η. The desired value of η will be the one for which $\cos\eta > 0$.

Once the eccentric anomaly for culmination, E_c, is found, the time of culmination is determined directly from Equation (B–11). Perturbing E_c by plus or minus about 10°, it is possible to obtain good starting values for the determination of the satellite set and rise times, respectively.

The procedure for finding the rise and set times is very similar to finding the time of culmination. The rise/set equations are developed by Escobal [Ref. 3, Sec. 5.4], but the algorithm presented here is somewhat simplified since, unlike Escobal, the time of culmination is available. In fact, many of the same equations can be used. Again, the eccentric anomaly will be used as the independent variable. Referring to Fig. B–1, satellite rise or set occurs when the elevation angle h is equal to zero, where h can be found from the equation

$$\sin h = \frac{\boldsymbol{\rho}\cdot\mathbf{z}}{\rho}, \tag{B–14}$$

where

$$\boldsymbol{\rho} = \mathbf{r} - \mathbf{R}. \tag{B–15}$$

In Equation (B–15), $\mathbf{R}$ is the station coordinate vector and $\boldsymbol{\rho}$ is the slant range vector from the station to the satellite. For the spheroid earth [Ref. 3, Equ. (5.54)],

$$\mathbf{R} = \begin{bmatrix} G_1\cos\phi\cos\lambda \\ G_1\cos\phi\sin\lambda \\ G_2\sin\phi \end{bmatrix}, \tag{B–16}$$

where ϕ and λ are the station geodetic latitude and longitude, respectively. G_1 and G_2 are factors which account for the earth's oblateness, and are given by

$$\begin{aligned} G_1 &= \frac{a_e}{\sqrt{1-(2f-f^2)\sin^2\phi}} + H \qquad \text{and} \\ G_2 &= \frac{(1-f^2)a_e}{\sqrt{1-(2f-f^2)\sin^2\phi}} + H, \end{aligned} \tag{B–17}$$

where

a_e = earth's equatorial radius,

f = earth's flattening factor, and

H = station elevation above the ellipsoid.

Write Equation (B–14) as

$$\boldsymbol{\rho} \cdot \mathbf{z} = \rho \sin h.$$

When the satellite is on the horizon, $h = 0$ and $\sin h = 0$. Since the resulting equation, $\boldsymbol{\rho} \cdot \mathbf{z} = 0$, cannot be solved explicitly for E, set

$$G(E) = \boldsymbol{\rho} \cdot \mathbf{z}$$

and find E such that $G(E) = 0$ by the Newton-Raphson method. Using Equation (B–15), write $G(E)$ as

$$G(E) = \mathbf{z} \cdot (\mathbf{r} - \mathbf{R}). \tag{B–18}$$

The Newton-Raphson procedure also requires $G'(E)$, the derivative of $G(E)$ with respect to E. Since the earth is being treated as non-rotating with the sidereal rotation subtracted from the satellite's ascending node, $\mathbf{R}$ is independent of E, hence

$$G'(E) = \mathbf{z} \cdot \frac{d\mathbf{r}}{dE}. \tag{B–19}$$

In Equations (B–18) and (B–19), $\mathbf{z}$, $\mathbf{r}$, $d\mathbf{r}/dE$, and the additional needed derivatives have all been given previously in this section.

To solve Equation (B–18), and initial estimate of E is made and the right-hand side of Equation (B–20) is evaluated. The result, the left-hand side of Equation (B–20), is an improved estimate of E,

$$E = E - \frac{G(E)}{G'(E)}. \tag{B–20}$$

The iteration is continued until $|G(E)/G'(E)|$ is less than some prescribed amount. The corresponding time of rise or set is then obtained from Equation (B–11).

APPENDIX C. THE SATELLITE HEADING COMPUTATION.

A satellite can be located by a position vector $\mathbf{x} = x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$ in a rectangular coordinate system with origin at the earth's center. In matrix form, and with x_1, x_2 and x_3 expressed in spherical coordinates,

$$\mathbf{x} = \begin{bmatrix} r\cos\phi\cos\lambda \\ r\cos\phi\sin\lambda \\ r\sin\phi \end{bmatrix}$$

where ϕ is the latitude, λ is the longitude at the point, and $r = |\mathbf{x}|$ is the distance of the point from the earth's center.

Let $\mathbf{z}$ be the unit vector in the direction of $\mathbf{r}$. Then

$$\mathbf{z} = \frac{\mathbf{x}}{|\mathbf{x}|} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \cos\phi\cos\lambda \\ \cos\phi\sin\lambda \\ \sin\phi \end{bmatrix}.$$

It is convenient to define two other unit vectors in the plane tangent to $\mathbf{z}$. These are local north $\mathbf{n}$ and local east $\mathbf{e}$, defined by

$$\mathbf{n} = \frac{\mathbf{z}_\phi}{|\mathbf{z}_\phi|} = \begin{bmatrix} -\sin\phi\cos\lambda \\ -\sin\phi\sin\lambda \\ \cos\phi \end{bmatrix} \qquad \text{and} \qquad \mathbf{e} = \frac{\mathbf{z}_\lambda}{|\mathbf{z}_\lambda|} = \begin{bmatrix} -\sin\lambda \\ \cos\lambda \\ 0 \end{bmatrix},$$

where

$$\mathbf{z}_\phi \equiv \frac{\partial \mathbf{z}}{\partial \phi} \qquad \text{and} \qquad \mathbf{z}_\lambda \equiv \frac{\partial \mathbf{z}}{\partial \lambda}.$$

The vectors $\mathbf{z}$, $\mathbf{n}$ and $\mathbf{e}$ provide the basis for a right-handed orthogonal coordinate system.

If a satellite's velocity vector $\mathbf{v} = v_1\mathbf{i} + v_2\mathbf{j} + v_3\mathbf{k}$ is known at position $\mathbf{x}$ then it is easily shown [Ref. 8] that

$$\mathbf{n}\cdot\mathbf{v} = \cos\alpha \quad \text{and}$$

$$\mathbf{e}\cdot\mathbf{v} = \sin\alpha,$$

where α is the course angle or heading. From these relations, α is found to be

$$\alpha = \operatorname{qatn}(\mathbf{e}\cdot\mathbf{v}, \mathbf{n}\cdot\mathbf{v})$$

where $\operatorname{qatn}(y,x)$ is $\arctan(y/x)$ corrected to the proper quadrant.

Since the position and velocity component's are in rectangular coordinates, it will be necessary to convert them to spherical coordinates in order to determine the components of $\mathbf{n}$ and $\mathbf{e}$. However, the spherical coordinate step can be circumvented as follows: First,

$$\cos\alpha = \mathbf{n}\cdot\mathbf{v} = -v_1\sin\phi\cos\lambda - v_2\sin\phi\sin\lambda + v_3\cos\phi.$$

Multiply both sides by $\cos\phi$ to give

$$\begin{aligned}\cos\phi\cos\alpha &= -v_1\sin\phi\cos\phi\cos\lambda - v_2\sin\phi\cos\phi\sin\lambda + v_3\cos^2\phi\\ &= -\sin\phi(z_1v_1 + z_2v_2) + v_3\cos^2\phi + z_3v_3\sin\phi - z_3v_3\sin\phi\\ &= -\sin\phi(z_1v_1 + z_2v_2 + z_3v_3) + v_3\cos^2\phi + z_3v_3\sin\phi\\ &= -z_3(\mathbf{z}\cdot\mathbf{v}) + v_3\cos^2\phi + v_3\sin^2\phi\\ &= v_3 - z_3(\mathbf{z}\cdot\mathbf{v}).\end{aligned}$$

Similarly,

$$\sin\alpha = \mathbf{e}\cdot\mathbf{v} = -v_1\sin\lambda + v_2\cos\lambda,$$

so that

$$\begin{aligned}\cos\phi\sin\alpha &= v_2\cos\phi\cos\lambda - v_1\cos\phi\sin\lambda\\ &= z_1v_2 - z_2v_1.\end{aligned}$$

Then,

$$\alpha = \mathrm{qatn}(\cos\phi\sin\alpha, \cos\phi\cos\alpha).$$

Since $\cos\phi > 0$ for all $\phi \in (-\pi/2, \pi/2)$, the $\cos\phi$ term effectively "cancels out" [Ref. 9] and so the heading is determined by

$$\begin{aligned}\alpha &= \mathrm{qatn}(\sin\alpha, \cos\alpha)\\ &= \mathrm{qatn}(z_1v_2 - z_2v_1, v_3 - z_3(\mathbf{z}\cdot\mathbf{v})).\end{aligned}$$

This equation is used by NAVSPASUR [Ref. 1] in SHOWALL. In SATSTA, the equivalent form,

$$\alpha = \mathrm{qatn}\left(\frac{x_1v_2 - x_2v_1}{r}, v_3 - \frac{x_3(\mathbf{x}\cdot\mathbf{v})}{r^2}\right),$$

is used.

APPENDIX D. THE SATSTA PROGRAM LISTING.

```
'SATSTA
' Satellite Track.  03-03-88.        Rev. 09-26-89 @ 0830.

DECLARE FUNCTION acos# (x#)
DECLARE FUNCTION asin# (x#)
DECLARE FUNCTION atan2# (y#, x#)
DECLARE FUNCTION decimal# (v$)
DECLARE FUNCTION dmod# (x#, m#)
DECLARE FUNCTION dot# (a#(), b#())
DECLARE FUNCTION gmst# (tu#, tm#)
DECLARE SUB apo.peri.gee (elem#(), elemu#())
DECLARE SUB culmnate (zv#(), we#, t0#, ea#, coseta#, elemu#())
DECLARE SUB euler (peri#, node#, incl#, psn#(), vel#())
DECLARE SUB ground.track (r#, sat.dc#, sat.lt#, sat.ht#)
DECLARE SUB hr.min.sec (time.of.day#, hr%, min%, sec#, n%)
DECLARE SUB jd.to.date (jd#(), m&, d&, y&, h#, mo$, day$)
DECLARE SUB julian.day.number (m&, d&, y&, ut#, jd#(), dj#, tj#)
DECLARE SUB kepler (m#, ec#, ea#)
DECLARE SUB output4 (k%, yr&, mo&, da&, time.of.day#, lt#, ln#, ht#, el#, _
          az#, rng#, langle#, head#, sta.id$, sat.id$)
DECLARE SUB pos.update (l%, elem#(), time#, sat.x#(), sat.xd#(), elemu#(), _
          r#, ea#)
DECLARE SUB rise.set (zv#(), sta.R#(), we#, t0#, ea#, elemu#())
DECLARE SUB sat.head (r#, sat.x#(), sat.xd#(), head#)
DECLARE SUB sat.sphere.coord (sat.x#(), r#, tj#, tm#, sat.dc#, sat.ln#, sat.ra#)
DECLARE SUB sta.p.coord (lt#, ln#, ht#, x#())
DECLARE SUB sta.sat (sta.x#(), sat.x#(), rng#, az#, el#, langle#)
DECLARE SUB time.update (k%, deltat#, tm#, jd#, jd#(), tj#, time.of.day#, _
          m&, d&, y&, h#, mo$, day$)
DECLARE SUB yrmoday (v$, yr&, mo&, dy&)

'COLOR 7, 1
'SCREEN 0, 1

DEFDBL A-Z
OPTION BASE 1
DIM jdepoch(2), jdstart(2), jdend(2), jd(2), sat.x(3), sat.xd(3), sta.x(3, 4)
DIM elem(9), elemu(9), pv(3), qv(3), sta.z(3), sta.R(3), jd.charlie(2)

' constants
CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi
CONST j2 = .0010826318#
CONST nmi.ft = .3048# / 1852#    ' n.mi/foot
CONST ae = 6378.14#              ' earth's equatorial radius (km.)
CONST GE = 398600.5#             ' geo. grav. const. (km)^3/(sec)^2
      k = 60# * SQR(GE / ae) / ae'(eru)^(3/2)/min
      Herg.min = k
CONST fl = 1# / 298.257#         ' earth's flattening factor
CONST e.sqr = (2# - fl) * fl     ' earth's ellipticity squared
CONST we = 1.002737909350795# * twopi / 1440#'earth's rotation (radians/minute)
CONST sixty = 60#

' Sample input (NORAD Project Spacetrack Rpt. #3, Dec. 1980):
amajor$ = "1.04050189": ecc$ = "0.0086731": aincl$ = "72.8435"
peri$ = "52.6988": anode$ = "115.9689": manomaly$ = "110.5714"
motion$ = "16.05824518": decay$ = "0"
epochdate$ = "1980.1001": epochtime$ = "23.41241138"
startdate$ = "1980.1001": starttime$ = "23.41241138"
enddate$ = "1980.1003": endtime$ = "0000"
deltat$ = "4"

' Sample input of arbitrary higher altitude, higher ecc orbit
ecc$ = "0.0200000": aincl$ = "64.94868"
peri$ = "202.277268": anode$ = "55.552464": manomaly$ = "126.670896"
```

```
 motion$ = "12.00000": decay$ = "4.281098d-3"
 epochdate$ = "1983.0201": epochtime$ = "00.00"
 startdate$ = "1983.0201": starttime$ = "00.00"
 enddate$ = "1983.0208": endtime$ = "00.00"
 deltat$ = "15"
 sta.lat$ = "35.12": sta.lon$ = "-85.12": sta.alt$ = "500.0"

 ' Sample input (NAVSPASUR: Sat. 22222 data from SHOWALL, private comm.
 '    Mr. Fred Lipp, July 1988):
 sat.id$ = "22222"
 ecc$ = "0.0023064": aincl$ = "64.94868"
 peri$ = "202.277268": anode$ = "55.552464": manomaly$ = "126.670896"
 motion$ = "16.06302": decay$ = "4.281119d-3"
 epochdate$ = "1983.0201": epochtime$ = "00.00"
 startdate$ = "1983.0201": starttime$ = "00.00"
 enddate$ = "1983.0203": endtime$ = "00.00"
 deltat$ = "01"
 sta.id$ = "001": sta.name$ = "Daisy, Tenn."
 sta.lat$ = "35.12": sta.lon$ = "-85.12": sta.alt$ = "500.0"

 ' Sample input (NAVSPASUR: Sat. 11111 data from SHOWALL, private comm.
 '    Mr. Fred Lipp, July 1988):
 sat.id$ = "11111"
 ecc$ = "0.0005545": aincl$ = "65.06057"
 peri$ = "295.41470": anode$ = "272.43497": manomaly$ = "258.10682"
 motion$ = "15.44194": decay$ = "0"
 epochdate$ = "1983.0201": epochtime$ = "00.00"
 startdate$ = "1983.0201": starttime$ = "00.00"
 enddate$ = "1983.0202": endtime$ = "00.00"
 deltat$ = "10"
 sta.id$ = "001": sta.name$ = "Daisy, Tenn."
 sta.lat$ = "35.12": sta.lon$ = "-85.12": sta.alt$ = "500.0"

 start:
 CLS
 PRINT "Select data input mode 1, 2, 3 or 4:"
 PRINT
 PRINT SPC(5); "Keyboard input for satellite & ground station:"
 PRINT SPC(10); "1. Epoch of date."
 PRINT SPC(10); "2. NAVSPASUR Reference Date."
 PRINT
 PRINT SPC(5); "Data file input for satellite & ground station:"
 PRINT SPC(10); "3. Epoch of date."
 PRINT SPC(10); "4. NAVSPASUR One-Line Charlie."
 data$ = ""
 WHILE data$ < "1" OR data$ > "4"
     data$ = INPUT$(1)
 WEND

 CLS
 PRINT "Select run mode 1 or 2:"
 PRINT
 PRINT SPC(5); "1. Create an ephemeris."
 PRINT SPC(5); "2. Find times satellite is above the horizon."
 case$ = ""
 WHILE case$ <> "1" AND case$ <> "2"
     case$ = INPUT$(1)
 WEND

 GOSUB input.sim.times

 SELECT CASE data$
 CASE "1", "2"
    CLS
    PRINT "Satellite Data:"
    PRINT
```

```
PRINT "Epoch date (yyyy.mmdd)      ? "; epochdate$
LOCATE 3, 29: INPUT v$: IF v$ = "" THEN v$ = epochdate$
epochdate$ = v$: CALL yrmoday(v$, yr&, mo&, dy&)

PRINT "Epoch time (hh.mmss)        ? "; epochtime$
LOCATE 4, 29: INPUT v$: IF v$ = "" THEN v$ = epochtime$
epochtime$ = v$: epochtime = decimal#(v$)
CALL julian.day.number(mo&, dy&, yr&, epochtime, jdepoch(), epoch, tjepoch)

LOCATE 6, 1: PRINT "Input mean elements of epoch:"

PRINT "Satellite ID Number         ? "; sat.id$
LOCATE 7, 29: INPUT v$: IF v$ = "" THEN v$ = sat.id$
sat.id$ = v$

PRINT "Eccentricity                ? "; ecc$
LOCATE 8, 29: INPUT v$: IF v$ = "" THEN v$ = ecc$
ecc$ = v$: ecc = VAL(v$)

PRINT "Long. ascending node (deg)  ? "; anode$
LOCATE 9, 29: INPUT v$: IF v$ = "" THEN v$ = anode$
anode$ = v$: anode = VAL(v$) * rad

PRINT "Inclination (deg)           ? "; aincl$
LOCATE 10, 29: INPUT v$: IF v$ = "" THEN v$ = aincl$
aincl$ = v$: aincl = VAL(v$) * rad

PRINT "Arg. of perigee (deg)       ? "; peri$
LOCATE 11, 29: INPUT v$: IF v$ = "" THEN v$ = peri$
peri$ = v$: peri = VAL(v$) * rad

PRINT "Mean anomaly (deg)          ? "; manomaly$
LOCATE 12, 29: INPUT v$: IF v$ = "" THEN v$ = manomaly$
manomaly$ = v$: manomaly = VAL(v$) * rad

PRINT "Mean motion (rev/day)       ? "; motion$
LOCATE 13, 29: INPUT v$: IF v$ = "" THEN v$ = motion$
motion$ = v$: motion = VAL(v$) * twopi / 1440#' convert to radians/minute

PRINT "Decay (rev/day^2)           ? "; decay$
LOCATE 14, 29: INPUT v$: IF v$ = "" THEN v$ = decay$
' convert to radians/minute^2:
decay$ = v$: decay = VAL(v$) * twopi / (1440#) ^ 2

LOCATE 16, 1: PRINT "Press any key to continue."
FOR i% = 1 TO 10: c$ = INKEY$: NEXT
c$ = INPUT$(1)

CLS
LOCATE 1, 1: PRINT "Station Coordinates:"
PRINT

PRINT "Station Identification Number            ? "; sta.id$
LOCATE 3, 42: INPUT v$: IF v$ = "" THEN v$ = sta.id$
sta.id$ = v$

PRINT "Station Location or Name                 ? "; sta.name$
LOCATE 4, 42: INPUT v$: IF v$ = "" THEN v$ = sta.name$
sta.name$ = v$: sta.lat = VAL(v$) * rad

PRINT "Station latitude  (dd.mmss, South minus) ? "; sta.lat$
LOCATE 5, 42: INPUT v$: IF v$ = "" THEN v$ = sta.lat$
sta.lat$ = v$: sta.lat = VAL(v$) * rad

PRINT "Station longitude (ddd.mmss, West minus) ? "; sta.lon$
```

```
      LOCATE 6, 42: INPUT v$: IF v$ = "" THEN v$ = sta.lon$
      sta.lon$ = v$: sta.lon = VAL(v$) * rad

      PRINT "Station altitude  (feet)                   ? "; sta.alt$
      LOCATE 7, 42: INPUT v$: IF v$ = "" THEN v$ = sta.alt$
      sta.alt$ = v$: sta.alt = VAL(v$)

      LOCATE 9, 1: PRINT "Press any key to continue."
      FOR i% = 1 TO 10: c$ = INKEY$: NEXT
      c$ = INPUT$(1)

      OPEN "output.dat" FOR APPEND AS #3

      GOSUB compute

   CASE "3"
      OPEN "station.dat" FOR INPUT AS #1
      OPEN "output.dat" FOR APPEND AS #3

      DO UNTIL EOF(1)
         INPUT #1, sta.id$, sta.lat$, sta.lon$, sta.alt$, sta.name$
         sta.lat = VAL(sta.lat$) * rad
         sta.lon = VAL(sta.lon$) * rad
         sta.alt = VAL(sta.alt$)

         OPEN "elements.dat" FOR INPUT AS #2

         DO UNTIL EOF(2)
            INPUT #2, sat.id$, epochdate$, epochtime$, ecc$, anode$, aincl$, _
                      peri$, manomaly$, motion$, decay$

            CALL yrmoday(epochdate$, yr&, mo&, dy&)
            epochtime = decimal#(epochtime$)
            CALL julian.day.number(mo&, dy&, yr&, epochtime, jdepoch(), epoch, _
                     tjepoch)
            ecc = VAL(ecc$)
            anode = VAL(anode$) * rad
            aincl = VAL(aincl$) * rad
            peri = VAL(peri$) * rad
            manomaly = VAL(manomaly$) * rad
            motion = VAL(motion$) * twopi / 1440#' convert to radians/minute
            decay = VAL(decay$) * twopi / (1440#) ^ 2  ' convert to rad/min^2

            jd(1) = jdstart(1)
            jd(2) = jdstart(2)
            jd = jd(1) + jd(2)
            time.of.day = starttime

            GOSUB compute

        LOOP
        CLOSE 2
     LOOP

  CASE "4"
     OPEN "station.dat" FOR INPUT AS #1
     OPEN "output.dat" FOR APPEND AS #3

     DO UNTIL EOF(1)
        INPUT #1, sta.id$, sta.lat$, sta.lon$, sta.alt$, sta.name$
        sta.lat = VAL(sta.lat$) * rad
        sta.lon = VAL(sta.lon$) * rad
        sta.alt = VAL(sta.alt$)

        OPEN "elements.nss" FOR INPUT AS #2
```

```
         DO UNTIL EOF(2)
            INPUT #2, c$

            sat.id$ = MID$(c$, 1, 5)
            manomaly$ = MID$(c$, 6, 9)
            motion$ = MID$(c$, 14, 9)
            decay$ = MID$(c$, 22, 7)
            ecc$ = MID$(c$, 28, 9)
            peri$ = MID$(c$, 36, 9)
            anode$ = MID$(c$, 44, 9)
            aincl$ = MID$(c$, 52, 9)
            epochdate$ = "19" + MID$(c$, 60, 2) + "." + MID$(c$, 62, 4)
            epochtime$ = "00.00"

            CALL yrmoday(epochdate$, yr&, mo&, dy&)
            epochtime = decimal#(epochtime$)
            CALL julian.day.number(mo&, dy&, yr&, epochtime, jdepoch(), epoch, _
                     tjepoch)
            ecc = VAL(ecc$)
            anode = VAL(anode$) * twopi
            aincl = VAL(aincl$) * twopi
            peri = VAL(peri$) * twopi
            manomaly = VAL(manomaly$) * twopi
            motion = VAL(motion$) * Herg.min      ' convert to radians/minute
            decay = VAL(decay$)
            IF decay > .5# THEN decay = decay - 1#
            decay = decay * .00001 * Herg.min ^ 2    ' convert to rad/min^2

            jd(1) = jdstart(1)
            jd(2) = jdstart(2)
            jd = jd(1) + jd(2)
            time.of.day = starttime

            GOSUB compute

         LOOP
         CLOSE 2
      LOOP
   END SELECT
   CLOSE

   END

   '     *         *         *         *         *         *         *

   compute:
   ' initialize times.
   tj = tjstart                                  'julian centuries from J2000.0
   begin.time = (startdate - epoch) * 1440#      'minutes
   end.time = (enddate - epoch) * 1440#          'minutes

        elem(1) = amajor        'semi-major axis
        elem(2) = ecc           'eccentricity
        elem(3) = tzero         'time of perifocal passage
        elem(4) = aincl         'inclination (radians)
        elem(5) = anode         'longitude of ascending node (radians)
        elem(6) = peri          'argument of perifocus (radians)
        elem(7) = motion        'mean motion (revolutions/day)
        elem(8) = manomaly      'mean anomaly (radians)
        elem(9) = decay         'decay constant (radians/minute^2)

   SELECT CASE data$
   CASE "1", "3"
       ' Since the earth is treated as non-rotating, elem(5) must be corrected for
       '   the accrued rotation from epoch to J2000.0
       elem(5) = elem(5) - 15# * gmst(tjepoch, 0#) * rad
```

```
 CASE "2", "4"
     ' NAVSPASUR one-line Charlie correction
     CALL julian.day.number(1, 1, 1985, 0, jd.charlie(), dummy, tj.charlie)
     wetc = (tjepoch - tj.charlie) * 36525# - (1# - gmst(tj.charlie, 0#) / 24#) _
                 * .9972695664#
     wetc = dmod(wetc * 1440# * we, twopi)
     elem(5) = elem(5) + wetc
     elem(5) = elem(5) - 15# * gmst(tjepoch, 0#) * rad
 END SELECT


 CALL sta.p.coord(sta.lat, sta.lon, sta.alt, sta.x())
 FOR i% = 1 TO 3
     sta.z(i%) = sta.x(i%, 2)
     sta.R(i%) = sta.x(i%, 1)
 NEXT

 CALL pos.update(1, elem(), begin.time, sat.x(), sat.xd(), elemu(), r, ea)
 CALL pos.update(2, elem(), begin.time, sat.x(), sat.xd(), elemu(), r, ea)

 CLS
 PRINT "Satellite: "; sat.id$; "     Station: "; sta.id$; "   "; sta.name$
 PRINT

 SELECT CASE case$
 '-------------------------------------------------------------------------------
 CASE "1"

 tm = starttime                          'UT in hours
 CALL apo.peri.gee(elem(), elemu())

 CALL jd.to.date(jd(), m&, d&, y&, h#, mo$, day$)
 PRINT
 PRINT day$; d&; mo$; y&

 CALL output4(0, yr&, mo&, da&, time.of.day, sat.lt, sat.ln, sat.ht, el, az, _
                 rng, langle, head, sta.id$, sat.id$)

 ' time-loop
 FOR time = begin.time TO end.time STEP delta.t

     CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)

 'compute spherical coordinates
     CALL sat.sphere.coord(sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)

 'satellite heading
     CALL sat.head(r, sat.x(), sat.xd(), head)

 'earth ground track
     CALL ground.track(r, sat.dc, sat.lt, sat.ht)

 'compute station-satellite relations
     CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)

     rng = rng * ae

 CALL output4(1, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, az, rng, _
                 langle, head, sta.id$, sat.id$)

 'update time
 CALL time.update(0, delta.t.hr, tm, jd, jd(), tj, time.of.day, m&, d&, y&, _
                 h#, mo$, day$)

 NEXT time
```

```
PRINT : PRINT "Press any key to continue."
FOR i% = 1 TO 10: c$ = INKEY$: NEXT
c$ = INPUT$(1)

'-----------------------------------------------------------------------------
CASE "2"

CALL output4(0, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, az, rng, _
             langle, head, sta.id$, sat.id$)

tm = 0
period = twopi / motion
ea = 0

time = begin.time
initialize:
CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
CALL culmnate(sta.z(), we, time, ea, coseta, elemu())
IF coseta < 0 THEN
    ea = ea + pi
    CALL culmnate(sta.z(), we, time, ea, coseta, elemu())
END IF
time = (ea - elem(2) * SIN(ea)) / motion + elemu(3)
dt.hr = (time - begin.time) / sixty
CALL time.update(0, dt.hr, tm, jd, jd(), tj, time.of.day, m&, d&, y&, h#, _
                 mo$, day$)
IF time > begin.time + period THEN
    time = time - period
    CALL time.update(1, -period, tm, jd, jd(), tj, time.of.day, m&, d&, y&, _
                     h#, mo$, day$)
    GOTO initialize
END IF

again:
FOR i% = 1 TO 2
    CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    CALL culmnate(sta.z(), we, time, ea, coseta, elemu())
    dt.min = (ea - elem(2) * SIN(ea)) / motion + elemu(3) - time
    time = time + dt.min
    CALL time.update(1, dt.min, tm, jd, jd(), tj, time.of.day, m&, d&, y&, _
                     h#, mo$, day$)
NEXT
time.cul = time

CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)
IF el > 0 AND time >= begin.time THEN
    CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    ea = ea - .335
    FOR i% = 1 TO 2
        CALL rise.set(sta.z(), sta.R(), we, time, ea, elemu())
        dt.min = (ea - elem(2) * SIN(ea)) / motion + elemu(3) - time
        time = time + dt.min
        CALL time.update(1, dt.min, tm, jd, jd(), tj, time.of.day, m&, d&, _
                         y&, h#, mo$, day$)
        CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    NEXT
    time.rise = time

    CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)
    CALL sat.sphere.coord(sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)
    CALL ground.track(r, sat.dc, sat.lt, sat.ht)
    CALL sat.head(r, sat.x(), sat.xd(), head)
    CALL jd.to.date(jd(), m&, d&, y&, h#, mo$, day$)
    CALL output4(1, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, _
                 az, rng * ae, langle, head, sta.id$, sat.id$)
    time.incr = dmod(time.cul - time.rise, delta.t)
    time = time + time.incr
    CALL time.update(1, time.incr, tm, jd, jd(), tj, time.of.day, m&, d&, _
                  y&, h#, mo$, day$)
    CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)
    CALL sat.sphere.coord(sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)
    CALL ground.track(r, sat.dc, sat.lt, sat.ht)
    CALL sat.head(r, sat.x(), sat.xd(), head)
    CALL jd.to.date(jd(), m&, d&, y&, h#, mo$, day$)
    CALL output4(1, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, _
              az, rng * ae, langle, head, sta.id$, sat.id$)

DO WHILE el > 0
     time = time + delta.t
     CALL time.update(0, delta.t.hr, tm, jd, jd(), tj, time.of.day, m&, d&, _
                   y&, h#, mo$, day$)
     CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
     CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)
     IF el < 0 THEN
          EXIT DO
     END IF
     CALL sat.sphere.coord(sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)
     CALL ground.track(r, sat.dc, sat.lt, sat.ht)
     CALL sat.head(r, sat.x(), sat.xd(), head)
     CALL jd.to.date(jd(), m&, d&, y&, h#, mo$, day$)
     CALL output4(1, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, _
               az, rng * ae, langle, head, sta.id$, sat.id$)
LOOP

save.time = time
FOR i% = 1 TO 2
    CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    CALL rise.set(sta.z(), sta.R(), we, time, ea, elemu())
    dt.min = (ea - elem(2) * SIN(ea)) / motion + elemu(3) - time
    time = time + dt.min
    CALL time.update(1, dt.min, tm, jd, jd(), tj, time.of.day, m&, d&, y&, _
                  h#, mo$, day$)
NEXT
    CALL pos.update(2, elem(), time, sat.x(), sat.xd(), elemu(), r, ea)
    CALL sta.sat(sta.x(), sat.x(), rng, az, el, langle)
    CALL sat.sphere.coord(sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)
    CALL ground.track(r, sat.dc, sat.lt, sat.ht)
    CALL sat.head(r, sat.x(), sat.xd(), head)
    CALL jd.to.date(jd(), m&, d&, y&, h#, mo$, day$)
    CALL output4(1, y&, m&, d&, time.of.day, sat.lt, sat.ln, sat.ht, el, _
              az, rng * ae, langle, head, sta.id$, sat.id$)
    PRINT

END IF
time = time.cul + period
dt.hr = (time - begin.time) / sixty - tm
CALL time.update(0, dt.hr, tm, jd, jd(), tj, time.of.day, m&, d&, y&, h#, _
              mo$, day$)

IF time < end.time THEN GOTO again
'-----------------------------------------------------------------------------
END SELECT
RETURN

input.sim.times:
   CLS
   PRINT "Simulation-Time Data:"
   PRINT

   PRINT "Starting date (yyyy.mmdd)   ? "; startdate$
```

```
    LOCATE 3, 29: INPUT v$: IF v$ = "" THEN v$ = startdate$
    startdate$ = v$: CALL yrmoday(v$, yr&, mo&, dy&)

    PRINT "Starting time (hh.mmss)     ? "; starttime$
    LOCATE 4, 29: INPUT v$: IF v$ = "" THEN v$ = starttime$
    starttime$ = v$: starttime = decimal#(v$)
    CALL julian.day.number(mo&, dy&, yr&, starttime, jdstart(), startdate, _
                           tjstart)
    jd(1) = jdstart(1)
    jd(2) = jdstart(2)
    jd = jd(1) + jd(2)
    time.of.day = starttime

    PRINT "Ending date (yyyy.mmdd)     ? "; enddate$
    LOCATE 5, 29: INPUT v$: IF v$ = "" THEN v$ = enddate$
    enddate$ = v$: CALL yrmoday(v$, yr&, mo&, dy&)

    PRINT "Ending time (hh.mmss)       ? "; endtime$
    LOCATE 6, 29: INPUT v$: IF v$ = "" THEN v$ = endtime$
    endtime$ = v$: endtime = decimal#(v$)
    CALL julian.day.number(mo&, dy&, yr&, endtime, jdend(), enddate, tjend)

    PRINT "Time increment (minutes)    ? "; deltat$
    LOCATE 7, 29: INPUT v$: IF v$ = "" THEN v$ = deltat$
    deltat$ = v$: delta.t = VAL(v$): delta.t.hr = delta.t / sixty

    LOCATE 9, 1: PRINT "Press any key to continue."
    FOR i% = 1 TO 10: c$ = INKEY$: NEXT
    c$ = INPUT$(1)

RETURN

'******************************************************************************
FUNCTION acos# (x#)     ' 02-10-88    Rev. 02-24-88
' The arccosine function derived from the ATN function with no singularities.
' The angle is returned in the (0,pi) interval.

'Note that in relational comparisons, -1 is "true" and 0 is "false".

CONST pi = 3.141592653589793#, eps = 1D-33

IF ABS(x#) <= 1# THEN
   acos# = ATN(SQR(1# - x# * x#) / (x# - eps * (x# = 0#))) - pi * (x# < 0#)
ELSE
   PRINT
   PRINT "Error in acos function. ABS(arg) > 1"
   PRINT "arg = "; x#
   PRINT
END IF
END FUNCTION

'******************************************************************************
SUB apo.peri.gee (elem#(), elemu#()) '  09-14-88.    Rev 09-14-88.
' Compute location and height of apogee and perigee above oblate earth.

' NOTE: (1) The output from this routine is approximately valid for the
'           current orbit only. In particular, the longitude is not corrected
'           for the earth's rotation during the current orbit. Also, the
'           ascending node and argument of perigee are not corrected for
'           their rates of change during the current orbit.
'       (2) For nearly circular orbits, the height of the satellite will
'           probably get lower than the perigee value and higher than the
'           apogee value at locations other than the perigee and apogee
'           positions---this is an effect of the earth's oblateness.
```

```
CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi

' spherical earth values:
apogee# = elemu#(1) * (1# - elem#(2))
perigee# = elemu#(1) * (1# + elem#(2))

' longitudes and declinations:
sincl# = SIN(elem#(4))
speri# = SIN(elemu#(6))
apo.dc# = asin#(speri# * sincl#)
apo.ln# = elemu#(5) + atan2#(COS(elem#(4)) * speri#, COS(elemu#(6)))
apo.ln# = dmod#(apo.ln#, twopi)
IF apo.ln# > pi THEN apo.ln# = apo.ln# - twopi
        ' next are approximations should update by peri- & nodal-rates
        '     for 1/2 revolution:
peri.dc# = -apo.dc#
peri.ln# = dmod(apo.ln# + pi, twopi)
IF peri.ln# > pi THEN peri.ln# = peri.ln# - twopi

CALL ground.track(apogee#, apo.dc#, apo.lt#, apo.ht#)
CALL ground.track(perigee#, peri.dc#, peri.lt#, peri.ht#)

PRINT
PRINT "Apogee:"
PRINT USING "& ####.# &"; "Height    = "; apo.ht#; " km. "
PRINT USING "& ###.####"; "Latitude  =  "; apo.lt# / rad
PRINT USING "& ###.####"; "Longitude =  "; dmod(apo.ln# / rad, 360#)
PRINT
PRINT "Perigee:"
PRINT USING "& ####.# &"; "Height    = "; peri.ht#; " km. "
PRINT USING "& ###.####"; "Latitude  =  "; peri.lt# / rad
PRINT USING "& ###.####"; "Longitude =  "; dmod(peri.ln# / rad, 360#)
PRINT

IF apo.ht# < 0 THEN
    PRINT "WARNING: Satellite is suborbital and will impact earth."
    PRINT "Do you want to continue?  Yes or No."
    PRINT
    WHILE c$ <> "y" AND c$ <> "n"
         c$ = LCASE$(INPUT$(1))
    WEND
    IF c$ = "n" THEN END
END IF
END SUB

'******************************************************************************
FUNCTION asin# (x#)    ' 02-10-88   Rev. 02-24-88
' The arcsine function derived from the ATN function with no singularities.
' The angle is returned in the (-pi,pi) interval.

'Note that in relational comparisons, -1 is "true" and 0 is "false".

CONST eps = 1D-33

IF ABS(x#) <= 1# THEN
   asin# = ATN(x# / (SQR(1# - x# * x#) - eps * (ABS(x#) = 1#)))
ELSE
   PRINT
   PRINT "Error in asin function. ABS(arg) > 1"
   PRINT "arg = "; x#
   PRINT
END IF
END FUNCTION

'******************************************************************************
FUNCTION atan2# (y#, x#)    '   02-10-88   Rev. 02-24-88
```

```
' The two argument quadrant determining arctangent function.
' The angle is returned in the (-pi,pi) interval.

'Note that in relational comparisons, -1 is "true" and 0 is "false".

CONST pi = 3.141592653589793#, eps = 1D-33

atan2# = ATN(y# / (x# - eps * (x# = 0#))) - pi * (x# < 0#) * (SGN(y#) _
                   - (y# = 0#))

END FUNCTION

'*****************************************************************************
SUB culmnate (zv#(), we#, t0#, ea#, coseta#, elemu#())

' Compute eccentric anomaly for the time of culmination of a satellite
' 01-30-89. Rev. 02-02-89 @ 1000.

' NOTE: This particular form of the culminate function was written from the
'       development of 30 Jan 89. It assumes a non-rotating earth so that the
'       station longitude is fixed. Rotation is accomplished by subtracting the
'       accumulated rotation from the right ascension of the ascending node of
'       the satellite. This is the method used to compute the ephemeris and is
'       also equivalent to the method used by NAVSPASUR.

'   lambda = station geodetic longitude
'   phi    = station geodetic latitude
'   we     = earth's rotation (sidereal rate of change)
'   t0     = time the elements, elemu(), were last updated
'   capt   = T -- time of latest perifocal passage
'   mot    = mean motion
'   ea     = eccentric anomaly E
'   ecc    = eccentricity e
'   amjr   = semimajor axis (cancels out in all formulas, so not included)

CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi

ecc# = elemu#(2)
capt# = elemu#(3)
incl# = elemu#(4)
node# = elemu#(5)
peri# = elemu#(6)
mot# = elemu#(7)

DIM pv#(3), pvd#(3), pvdd#(3), qv#(3), qvd#(3), qvdd#(3)
DIM rvec#(3), rvecd#(3), rvecdd#(3)
'   zv    = Z-vector, unit normal at the station
'   pv    = P-vector, qv = Q-vector (first two columns of the Euler matrix)
'   rvec = R-vector (satellite position), rvecd = dR/dE, rvecdd = d2R/dE2
'   rv    = r---magnitude of the R-vector, rvd = dr/dE, rvdd = d2r/dE2
'  gamma = node - accrued rotational motion

cosw# = COS(peri#)
sinw# = SIN(peri#)
cosi# = COS(incl#)
sini# = SIN(incl#)

corr# = 1
WHILE ABS(corr#) > .0001
    cosea# = COS(ea#)
    ecosea# = ecc# * cosea#
    sinea# = SIN(ea#)
    esinea# = ecc# * sinea#

    gamma# = node# - we# * ((ea# - esinea#) / mot# + capt# - t0#)
    gammad# = -we# * (1# - ecosea#) / mot#
```

```
          gammadd# = -we# * esinea# / mot#

          cosg# = COS(gamma#)
          sing# = SIN(gamma#)

          rtecc# = SQR(1# - ecc# * ecc#)

          rv# = 1# - ecosea#
          rvd# = esinea#
          rvdd# = ecosea#

          xw# = cosea# - ecc#
          xwd# = -sinea#
          xwdd# = -cosea#
          yw# = rtecc# * sinea#
          ywd# = rtecc# * cosea#
          ywdd# = -yw#

          pv#(1) = cosw# * cosg# - sinw# * sing# * cosi#
          pv#(2) = cosw# * sing# + sinw# * cosg# * cosi#
          pv#(3) = sinw# * sini#
          qv#(1) = -sinw# * cosg# - cosw# * sing# * cosi#
          qv#(2) = -sinw# * sing# + cosw# * cosg# * cosi#
          qv#(3) = cosw# * sini#

          pvd#(1) = -pv#(2) * gammad#
          pvd#(2) = pv#(1) * gammad#
          pvd#(3) = 0#
          qvd#(1) = -qv#(2) * gammad#
          qvd#(2) = qv#(1) * gammad#
          pvd#(3) = 0#

          pvdd#(1) = -pvd#(2) * gammad# - pv#(2) * gammadd#
          pvdd#(2) = pvd#(1) * gammad# + pv#(2) * gammadd#
          pvdd#(3) = 0#
          qvdd#(1) = -qvd#(2) * gammad# - qv#(2) * gammadd#
          qvdd#(2) = qvd#(1) * gammad# + qv#(2) * gammadd#
          pvdd#(3) = 0#

          FOR i% = 1 TO 3
              rvec#(i%) = xw# * pv#(i%) + yw# * qv#(i%)
              rvecd#(i%) = xwd# * pv#(i%) + xw# * pvd#(i%) + ywd# * qv#(i%) _
                           + yw# * qvd#(i%)
              temp# = xwdd# * pv#(i%) + 2# * xwd# * pvd#(i%) + xw# * pvdd#(i%)
              rvecdd#(i%) = temp# + ywdd# * qv#(i%) + 2# * ywd# * qvd#(i%) _
                            + yw# * qvdd#(i%)
          NEXT

          coseta# = dot#(zv#(), rvec#()) / rv#
          fe# = dot#(zv#(), rvecd#()) - rvd# * coseta#

          rrd# = rvd# / rv#
          fed# = dot#(zv#(), rvecdd#()) + coseta# * (rrd# - rvdd#)
          fed# = fed# - rrd# * dot#(zv#(), rvecd#())

          corr# = fe# / fed#
          IF corr# > pi THEN
               ea# = ea# + twopi
          ELSE
               ea# = ea# - corr#
               IF coseta# < 0 THEN ea# = ea# + pi
          END IF

     WEND
     END SUB
```

```
'*******************************************************************************
FUNCTION decimal# (v$)               '  03-21-88.    Rev 03-21-88.
' Convert ddd.mmssff or hh.mmssfff to decimal degrees or decimal hours.

ix = INSTR(v$, ".")
IF ix = 0 THEN
    decimal# = VAL(v$)
ELSE
    x# = VAL(LEFT$(v$, ix))
    sn = 0
    IF x# < 0# THEN
         sn = -1
         x# = -x#
    END IF
    w$ = v$ + "0000"
    y# = VAL(MID$(w$, ix + 1, 2))
    z# = VAL(MID$(w$, ix + 3, 2) + "." + RIGHT$(w$, LEN(w$) - ix - 4))
    x# = (z# / 60# + y#) / 60# + x#
    IF sn THEN x# = -x#
    decimal# = x#
END IF
END FUNCTION

'*******************************************************************************
FUNCTION dmod# (x#, m#)  '   02-10-88   Rev. 02-24-88
' Provides  x MOD m  for real-valued functions.

IF m# <> 0# THEN
    dmod# = x# - m# * INT(x# / m#)
ELSE
    PRINT
    PRINT "Error in dmod function. Modulus cannot be zero."
    PRINT
END IF
END FUNCTION

'*******************************************************************************
FUNCTION dot# (a#(), b#())
' Dot product of two vectors.

dot# = a#(1) * b#(1) + a#(2) * b#(2) + a#(3) * b#(3)

END FUNCTION

'*******************************************************************************
SUB euler (peri#, node#, incl#, psn#(), vel#())
' 09-05-88. Rev. 01-16-89.

'Compute the first two columns (p & q) of the Euler matrix.
'Rotate in-plane position and velocity components to rectangular equatorial
'   coordinates.
'psn() and vel() are both input and output vectors. It is assumed that the
'   third component of both psn() and vel() is zero on input.

cw# = COS(peri#)
sw# = SIN(peri#)
cn# = COS(node#)
sn# = SIN(node#)
ci# = COS(incl#)
si# = SIN(incl#)
px# = cw# * cn# - sw# * sn# * ci#
py# = cw# * sn# + sw# * cn# * ci#
pz# = sw# * si#
qx# = -sw# * cn# - cw# * sn# * ci#
qy# = -sw# * sn# + cw# * cn# * ci#
qz# = cw# * si#
```

```
t1# = psn#(1)
t2# = psn#(2)
psn#(1) = px# * t1# + qx# * t2#
psn#(2) = py# * t1# + qy# * t2#
psn#(3) = pz# * t1# + qz# * t2#
t1# = vel#(1)
t2# = vel#(2)
vel#(1) = px# * t1# + qx# * t2#
vel#(2) = py# * t1# + qy# * t2#
vel#(3) = pz# * t1# + qz# * t2#

END SUB

'*********************************************************************************
FUNCTION gmst# (tu#, tm#)  '  03-03-88.    Rev 03-03-88.
' Compute Greenwich mean sidereal time.
' tu# = number of centuries of 36525 days of universal time elapsed since
'       2000 Jan 1, 12h UT1 (JD 2451545 UT1).
' tm# = time of day
' The Astronomical Almanac, 1984, pp S13-S15.
' Almanac for Computers 1988, pp B2-B3.

CONST pi = 3.141592653589793#, rad = pi / 180#
CONST c0 = 24110.54841#, c1 = 8640184.812866#, c2 = 9.310400000000001D-02
CONST c3 = -.0000062#, hourpersec = 1# / 3600#, meantoapp = -.00029#
CONST d0 = 125.004452#, d1 = -.0529538#, d2 = .002071#

t# = (((c3 * tu# + c2) * tu# + c1) * tu# + c0) * hourpersec + tm#' GMST
     'The following two lines will convert GMST to GAST
     'lunarnode# = (d2 * tu# + d1) * tu# + t0#
     't# = t# + meantoapp * SIN(lunarnode# * rad)
gmst# = dmod#(t#, 24#)

END FUNCTION

'*********************************************************************************
SUB ground.track (r#, sat.dc#, sat.lt#, sat.ht#) '09-14-88.    Rev 09-14-88.

CONST ae = 6378.14#            ' earth's equatorial radius (km.)
CONST f1 = 1# / 298.257#       ' earth's flattening factor
CONST e.sqr = (2# - f1) * f1   ' earth's ellipticity squared
CONST f2 = (1# - f1) * (1# - f1)

    rkm# = r# * ae
    ps# = sat.dc#
el:     cp# = COS(ps#)
        rc# = ae * SQR((1# - e.sqr) / (1# - e.sqr * cp# * cp#))
        rcr# = rc# / rkm#
        sat.lt# = ATN(TAN(ps#) / f2)    ' satellite latitude
        sn# = SIN(sat.lt# - ps#)
        hr# = SQR(1# - rcr# * rcr# * sn# * sn#) - rcr# * COS(sat.lt# - ps#)
        pn# = sat.dc# - asin(hr# * sn#)
        IF (ABS(pn# - ps#) > .0000001) THEN ps# = pn#: GOTO el
    sat.ht# = hr# * rkm#                    ' satellite height

END SUB

'*********************************************************************************
SUB hr.min.sec (x#, hr%, min%, sec#, n%)
' Convert decimal hours to:    hh mm ss      if n% = 0,
'                              hh mm ss.f    if n% = 1,
'                              hh mm ss.ff   if n% = 2,
'                              hh mm ss.fff  if n% = 3.

CONST t0 = 1#, t1 = 10#, t2 = 100#, t3 = 1000#
CONST r0 = 1# / 7200#, r1 = r0 / t1, r2 = r1 / t2, r3 = r2 / t3

SELECT CASE n%
     CASE 0
          round = r0
          trunc = t0
     CASE 1
          round = r1
          trunc = t1
     CASE 2
          round = r2
          trunc = t2
     CASE 3
          round = r3
          trunc = t3
     CASE ELSE
          PRINT
          PRINT "Error in hr.min.sec subroutine. Must have 0 <= n% <= 3."
          PRINT "n% = "; n%
          PRINT
          EXIT SUB
END SELECT

v# = ABS(x#) + round
hr% = INT(v#)
v# = (v# - hr%) * 60#
min% = INT(v#)
sec# = INT((60# * (v# - min%) * trunc)) / trunc

END SUB

'******************************************************************************
SUB jd.to.date (jd#(), m&, d&, y&, h#, mo$, day$) ' 02-24-88   Rev. 03-02-88
' Convert Julian Day Number to month number, day, year, hour, month name and
'    day of the week. Limited to the Gregorian Calendar.
' Fliegel & VanFlandern, Comm. ACM, Vol. 11, No. 10, Oct. 1968, pg. 657.

CONST w$ = "MonTueWedThuFriSatSun"
CONST dd$ = "JanFebMarAprMayJunJulAugSepOctNovDec"

j4& = INT(jd#(1) + jd#(2) + .5)
h# = ((jd#(1) - j4&) + jd#(2) + .5) * 24#

l& = INT(j4& + 68569)
n& = INT(4 * l& / 146097)
l& = l& - INT((146097 * n& + 3) / 4)
y& = INT(4000 * (l& + 1) / 1461001)
l& = l& - INT(1461 * y& / 4) + 31
m& = INT(80 * l& / 2447)
d& = l& - INT(2447 * m& / 80)
l& = INT(m& / 11)
m& = m& + 2 - 12 * l&
y& = 100 * (n& - 49) + y& + l&

wn% = j4& - 7 * INT(j4& / 7) + 1
day$ = MID$(w$, 3 * wn% - 2, 3)
mo$ = MID$(dd$, 3 * m& - 2, 3)

END SUB

'******************************************************************************
SUB julian.day.number (m&, d&, y&, ut#, jd#(), dj#, tj#)
'                02-25-88    Rev. 03-23-88
' Conversion of calendar date: m& = month, d& = day, y& = 4-digit year,
'  and ut# = Universal Time, to Julian Day Number. Year must be between 1801
'  and 2099.
' Almanac for Computers, 1988, pg. B-2.

' Note that jd# is a subscripted double precision array. At any time, the
'    Julian Day Number is given by jd#(1) + jd#(2).
' For ease of programming, the user may put the entire epoch in jd#(1) and set
'    jd#(2) = 0#.
' For maximum accuracy, set jd#(1) = the most recent midnight at or before the
'    epoch and set jd#(2) = the fractional part of a day elapsed between jd#(1)
'    and the epoch. As an example, compute the number of days from epoch
'    J2000.0 = JD 2451545.0 using the code:
'                  days# = (jd#(1) - 2451545.0#) + jd#(2)
'    Note, particularly, the extra pair of parenthesis.
'
' dj# = days and fraction from J2000.0. tj# = julian centuries from J2000.0


CONST c1 = 1721013.5#, c2 = 190002.5#, jcent = 36525#

IF (1801 <= y&) AND (y& <= 2099) THEN
  jd#(1) = 367 * y& - INT(7 * (y& + INT((m& + 9) / 12)) / 4) _
         + INT((275 * m&) / 9) + d& + c1 - .5# * SGN(100 * y& + m& - c2) + .5#
  jd#(2) = ut# / 24
  dj# = (jd#(1) - 2451545#) + jd#(2)
  tj# = dj# / jcent
ELSE
   PRINT
   PRINT "Error in julian.day.number. Year is not between 1801 and 2099."
   PRINT "Year = "; y&
   PRINT
END IF
END SUB

'*****************************************************************************
SUB kepler (m#, ec#, ea#) ' 02-25-88.  Rev 02-26-88

'High precision, non-iterative, solution to Kepler's equation for the ellipse.
'Accuracy is 10E-18 using all terms or 10E-15 if the last term is omitted.
'Although not as compact as the Newton-Raphson procedure, one square root, one
'cube root, and only two trigonometric functions are evaluated thus making this
'procedure extremely fast. Can be extended to the hyperbolic case - see Ref.
'Ref.: S. Mikkalo, Cel. Mech., Vol. 40, 1987, pp 329-334.

'm# = mean anomaly, ec# = eccentricity and ea# = eccentric anomaly.

CONST pi = 3.141592653589793#, twopi = pi + pi, third = 1# / 3#
CONST r3 = 1# / 6#, r4 = 1# / 24#, r5 = 1# / 120#

IF ec# >= 1# OR ec# < 0# THEN
    PRINT
    PRINT "Error in Kepler. The eccentricity is not in the [0,1) interval."
    PRINT "Eccentricity = "; ec#
    PRINT
ELSE
    m# = m# - twopi * INT(m# / twopi)' assure that m# is
    IF m# > pi THEN m# = m# - twopi'    in the (-pi,pi) interval

    'Compute the initial approximation:
    g# = 1 / (4 * ec# + .5)
    a# = (1 - ec#) * g#
    b# = .5 * m# * g#
    arg# = ABS(b#) + SQR(b# * b# + a# * a# * a#)
    z# = arg# ^ third
    IF b# < 0 THEN z# = -z#
    s# = z# - a# / z#
    s# = s# - .078 * s# ^ 5 / (1 + ec#)
    ea# = m# + ec# * s# * (3 - 4 * s# * s#)'  abs(dea/ea) <= 0.002
```

```
     'Refine the approximation:
     f2# = ec# * SIN(ea#)
     f3# = ec# * COS(ea#)
     f0# = ea# - f2# - m#
     f1# = 1# - f3#
     f4# = -f2#
     f5# = -f3#
     'This next correction term is sufficient if ecc < 0.25
     d# = -f0# / f1#
     'Use more of the following terms to increase accuracy
     d# = -f0# / (f1# + .5# * f2# * d#)
     d# = -f0# / (f1# + d# * (.5# * f2# + d# * r3 * f3#))
     d# = -f0# / (f1# + d# * (.5# * f2# + d# * (r3 * f3# + d# * r4 * f4#)))
     d# = -f0# / (f1# + d# * (.5# * f2# + d# * (r3 * f3# + d# _
           * (r4 * f4# + d# * r5 * f5#))))
     ea# = ea# + d#
 END IF
 END SUB

'*****************************************************************************
SUB output4 (k%, yr&, mo&, da&, time.of.day, lt, ln, ht, el, az, rng, _
             langle, head, sta.id$, sat.id$)

CONST pi = 3.141592653589793#, rad = pi / 180#, tp = pi + pi
c$ = CHR$(34)

SELECT CASE k%
   CASE 0
              PRINT "                                            height         ";
              PRINT "            look"
              PRINT "year mo da hr mn  sec   lat   long    (km)   elev    ";
              PRINT "azim  range angle  head"
              PRINT
   CASE 1
         CALL hr.min.sec(time.of.day#, hr%, min%, sec#, 1)
         lt = lt / rad
         ln = dmod(ln / rad, 360#)
         la = langle
         hd = head / rad
         PRINT USING "#### ## ## ## ## ##.# "; yr&; mo&; da&; hr%; min%; sec;
         PRINT USING "###.## ###.## #####.# ###.## "; lt; ln; ht; el;
         PRINT USING "###.## ##### ##.## ###.##"; az; rng; la; hd
         PRINT #3, USING "!&! !&! "; c$; sta.id$; c$; c$; sat.id$; c$;
         PRINT #3, USING "!####! !##! "; c$; yr&; c$; c$; mo&; c$;
         PRINT #3, USING "!##! !##! "; c$; da&; c$; c$; hr%; c$;
         PRINT #3, USING "!##! !##.#! "; c$; min%; c$; c$; sec; c$;
         PRINT #3, USING "!###.##! !###.##! "; c$; lt; c$; c$; ln; c$;
         PRINT #3, USING "!#####.#! !###.##! "; c$; ht; c$; c$; el; c$;
         PRINT #3, USING "!###.##! !#####! "; c$; az; c$; c$; rng; c$;
         PRINT #3, USING "!##.##! !###.##!"; c$; la; c$; c$; hd; c$
  CASE ELSE
         PRINT
         PRINT "Error in output4 subroutine. Must have 0 <= k% <= 1."
         PRINT "k% = "; k%
END SELECT
END SUB

'*****************************************************************************
SUB pos.update (l%, elem#(), time#, sat.x#(), sat.xd#(), elemu#(), r#, ea#) _
                STATIC
'                                               09-07-88. Rev. 03-09-89.

' Case 1: Initialize orbital elements for time of epoch.
' Case 2: Update orbital elements and compute in-plane position & velocity.
' Simplest of NAVSPASUR methods. Accurate to 10-15 n.mi. within 20 days
'      of epoch.

'     elem(1) = amajor        semi-major axis
'     elem(2) = ecc           eccentricity
'     elem(3) = tzero         time of perifocal passage
'     elem(4) = aincl         inclination (radians)
'     elem(5) = anode         longitude of ascending node (radians)
'     elem(6) = peri          argument of perifocus (radians)
'     elem(7) = motion        mean motion (revolutions/day)
'     elem(8) = manomaly      mean anomaly (radians)
'     elem(9) = decay         1st decay constant (radians/minute^2)

CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi
CONST c = .00081197385#  '  0.75*j2
CONST twothird = 2# / 3#
CONST ae = 6378.14#              ' earth's equatorial radius (km.)
CONST GE = 398600.5#             ' geo. grav. const. (km)^3/(sec)^2
      k = 60# * SQR(GE / ae) / ae'(eru)^(3/2)/min
CONST we = 1.002737909350795# * twopi / 1440#'sidereal period in radians/min

SELECT CASE 1%
    CASE 1
         elem#(1) = (k / elem#(7)) ^ twothird
         a2# = elem#(1) ^ 2
         ecc2# = 1# - elem#(2) ^ 2
         ci# = COS(elem#(4))
         c2i# = ci# * ci#
         elem#(1) = elem#(1) * (1# + c * (3# * c2i# - 1#) / _
                    (a2# * ecc2# ^ 1.5#)) ^ twothird
         elem#(3) = time# - elem#(8) / elem#(7)
         aecc22# = (elem#(1) * ecc2#) ^ 2
         peridot# = c * (5# * c2i# - 1#) / aecc22#
         nodedot# = -2# * c * ci# / aecc22#
         adot# = -2# * twothird * elem#(1) * m2# / elem#(7)
         FOR i% = 1 TO 9
              elemu#(i%) = elem#(i%)
         NEXT
    CASE 2
         ma.m0# = (elem#(9) * time# + elem#(7)) * time#
         ma# = ma.m0# + elem#(8)
         elemu#(6) = elem#(6) + peridot# * ma.m0#
         elemu#(5) = elem#(5) + nodedot# * ma.m0# - we * time#
         elemu#(1) = elem#(1) + adot# * time#
         CALL kepler(ma#, elem#(2), ea#)
           s0# = SIN(ea#)
           c0# = COS(ea#)
         elemu#(8) = ma#
         elemu#(3) = time# - ma# / elem#(7)


         ' in-plane position and velocity
         sat.x#(1) = elemu#(1) * (c0# - elem#(2))
         sat.x#(2) = elemu#(1) * SQR(ecc2#) * s0#
         sat.x#(3) = 0#
         r# = elemu#(1) * (1# - elem#(2) * c0#)
         ed# = elemu#(1) * elem#(7) / r#
         sat.xd#(1) = -elemu#(1) * ed# * s0#
         sat.xd#(2) = elemu#(1) * ed# * SQR(ecc2#) * c0#
         sat.xd#(3) = 0#

         ' equatorial rectangular coordinates
         CALL euler(elemu(6), elemu(5), elem(4), sat.x(), sat.xd())

         ' correct velocity for earth's rotation
         sat.xd#(1) = sat.xd#(1) + we * sat.x(2)
         sat.xd#(2) = sat.xd#(2) - we * sat.x(1)
    CASE ELSE
         PRINT
         PRINT "Error in pos.update. First argument must be 1 or 2 only."
         PRINT "Argument = "; i%
         PRINT
END SELECT
END SUB

'*******************************************************************************
SUB rise.set (zv#(), sta.R#(), we#, t0#, ea#, elemu#())

' Compute eccentric anomaly for the time of rise or set of a satellite
' 02-04-89. Rev. 02-08-89 @ 1700.

' NOTE: To use this routine, the time of culmination MUST be computed first,
'   The entering ea# argument should be ea.cul# -/+ 0.2 radians (about 11
'   degrees) for rise/set.

' See the introductory comments in SUB culminate for nomenclature and
'     notation.

CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi

amaj# = elemu#(1)
ecc# = elemu#(2)
capt# = elemu#(3)
incl# = elemu#(4)
node# = elemu#(5)
peri# = elemu#(6)
mot# = elemu#(7)

DIM pv#(3), pvd#(3), qv#(3), qvd#(3), rvecd#(3), rho#(3)

cosw# = COS(peri#)
sinw# = SIN(peri#)
cosi# = COS(incl#)
sini# = SIN(incl#)

corr# = 1
DO
    cosea# = COS(ea#)
    ecosea# = ecc# * cosea#
    sinea# = SIN(ea#)
    esinea# = ecc# * sinea#

    gamma# = node# - we# * ((ea# - esinea#) / mot# + capt# - t0#)
    gammad# = -we# * (1# - ecosea#) / mot#

    cosg# = COS(gamma#)
    sing# = SIN(gamma#)

    rtecc# = amaj# * SQR(1# - ecc# * ecc#)

    xw# = amaj# * (cosea# - ecc#)
    xwd# = -amaj# * sinea#
    yw# = rtecc# * sinea#
    ywd# = rtecc# * cosea#

    pv#(1) = cosw# * cosg# - sinw# * sing# * cosi#
    pv#(2) = cosw# * sing# + sinw# * cosg# * cosi#
    pv#(3) = sinw# * sini#
    qv#(1) = -sinw# * cosg# - cosw# * sing# * cosi#
    qv#(2) = -sinw# * sing# + cosw# * cosg# * cosi#
    qv#(3) = cosw# * sini#
    pvd#(1) = -pv#(2) * gammad#
    pvd#(2) = pv#(1) * gammad#
    pvd#(3) = 0#
    qvd#(1) = -qv#(2) * gammad#
    qvd#(2) = qv#(1) * gammad#
    qvd#(3) = 0#

    FOR i% = 1 TO 3
        rho#(i%) = xw# * pv#(i%) + yw# * qv#(i%) - sta.R#(i%)
        rvecd#(i%) = xwd# * pv#(i%) + xw# * pvd#(i%) + ywd# * qv#(i%) _
                     + yw# * qvd#(i%)
    NEXT

    fe# = dot#(zv#(), rho#())
    fed# = dot#(zv#(), rvecd#())

    corr# = fe# / fed#
    ea# = ea# - corr#

LOOP UNTIL ABS(corr#) < .00001#
END SUB

'******************************************************************************
SUB sat.head (r#, sat.x#(), sat.xd#(), head#) '  01-18-89. Rev. 03-12-89.
'satellite heading

CONST pi = 3.141592653589793#, twopi = pi + pi

    shead# = (sat.x#(1) * sat.xd#(2) - sat.x#(2) * sat.xd#(1)) / r#
    chead# = sat.xd#(3) - (sat.x#(3) * dot#(sat.x#(), sat.xd#())) / r# ^ 2
    head# = atan2#(shead#, chead#)
    IF head# < 0# THEN head# = head# + twopi

END SUB

'******************************************************************************
SUB sat.sphere.coord (sat.x(), r, tj, tm, sat.dc, sat.ln, sat.ra)
'compute spherical coordinates

CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi

    sat.dc = asin(sat.x(3) / r)          ' satellite declination
    sat.ln = atan2(sat.x(2), sat.x(1))   ' satellite longitude
    ru = 15# * gmst(tj, tm) * rad        ' Greenwich mean sidereal time
    sat.ra = dmod(sat.ln + ru, twopi)    ' satellite right ascension

END SUB

'******************************************************************************
SUB sta.p.coord (lt#, ln#, ht#, x#()) '  08-06-88.   Rev 02-04-89.
' convert station, lat, long & height (ft.) from geographic
'    to geocentric rectangular - x(.)

CONST nmi.ft = .3048# / 1852#  ' n.mi./foot
CONST km.ft = .3048# / 1000#   ' km./foot
CONST ae = 6378.14#            ' earth's equatorial radius (km.)
CONST fl = 1# / 298.257#       ' earth's flattening factor
CONST e.sqr = (2# - fl) * fl   ' earth's ellipticity squared

ht.eru# = ht# * km.ft / ae
slt# = SIN(lt#)
clt# = COS(lt#)
sln# = SIN(ln#)
cln# = COS(ln#)
n# = 1# / SQR(1# - e.sqr * slt * slt)
g1# = n# + ht.eru#
```

```
g2# = n# * (1# - e.sqr) + ht.eru#

'radius vector for flattened earth
x#(1, 1) = g1# * clt# * cln#
x#(2, 1) = g1# * clt# * sln#
x#(3, 1) = g2# * slt#

'Spherical earth unit vectors:
'1) local perpendicular vector
x#(1, 2) = clt# * cln#
x#(2, 2) = clt# * sln#
x#(3, 2) = slt#

'2) east vector
x#(1, 3) = -sln#
x#(2, 3) = cln#
x#(3, 3) = 0#

'3) north vector
x#(1, 4) = -slt# * cln#
x#(2, 4) = -slt# * sln#
x#(3, 4) = clt#

END SUB

'****************************************************************************
SUB sta.sat (sta.x#(), sat.x#(), rng#, az#, el#, langle#)
'                                                 08-06-88. Rev. 08-31-88.

'compute station-satellite relations. Range, azimuth, elevation and satellite
' "look-angle" (angle between satellite sub point and station).

CONST pi = 3.141592653589793#, rad = pi / 180#, twopi = pi + pi

'rho() = sat coords minus sta coords. rhodots are the rho components projected
'  in the station local north-east-radial system,
DIM rho#(3), rhodots#(3)

rng# = 0#
FOR i% = 1 TO 3
    r# = sat.x#(i%) - sta.x#(i%, 1)
    rho#(i%) = r#
    rng# = rng# + r# * r#
NEXT
rng# = SQR(rng#)
FOR i% = 1 TO 3
    rho#(i%) = rho#(i%) / rng#
NEXT

FOR i% = 1 TO 3
    a# = 0#
    FOR j% = 1 TO 3
         a# = a# + rho#(j%) * sta.x#(j%, i% + 1)
    NEXT
    rhodots#(i%) = a#
NEXT

az# = atan2#(rhodots#(2), rhodots#(3)) / rad
IF az# < 0# THEN az# = az# + 360#

a# = rhodots#(1)
el# = ATN(a# / SQR(1# - a# * a#)) / rad

a# = 0#
b# = 0#
FOR i% = 1 TO 3
```

```
    c# = sat.x#(i%)
    a# = a# + rho#(i%) * c#
    b# = b# + c# * c#
NEXT
langle# = acos#(a# / SQR(b#)) / rad

END SUB

'*******************************************************************************
SUB time.update (k%, delta.t#, tm#, jd#, jd#(), tj#, time.of.day, m&, d&, _
                 y&, h#, mo$, day$)

CONST twentyfour = 24#, sixty = 60#, jul.cent = 36525#

t# = delta.t#                      'delta.t# in hours if k% = 0, t# in hours.
IF k% > 0 THEN t# = t# / sixty     'delta.t# in minutes if k% > 0, t# in hours.
tm# = tm# + t#
jd#(2) = jd#(2) + t# / twentyfour
jd# = jd#(1) + jd#(2)
tj# = tj# + t# / twentyfour / jul.cent
time.of.day = time.of.day + t#
IF time.of.day >= twentyfour THEN
    time.of.day = time.of.day - twentyfour
    jd#(1) = jd#(1) + 1#: jd#(2) = jd#(2) - 1#
    CALL jd.to.date(jd#(), m&, d&, y&, h#, mo$, day$)
END IF
IF time.of.day < 0# THEN
    time.of.day = time.of.day + twentyfour
    jd#(1) = jd#(1) - 1#: jd#(2) = jd#(2) + 1#
    CALL jd.to.date(jd#(), m&, d&, y&, h#, mo$, day$)
END IF

END SUB

'*******************************************************************************
SUB yrmoday (v$, yr&, mo&, dy&) '  03-21-88.    Rev 03-21-88.
' Convert yyyy.mmdd to yr&=yyyy, mo&=mm, and dy&=dd

IF INSTR(v$, ".") <> 5 OR LEN(v$) <> 9 THEN
    PRINT
    PRINT "Error in yrmoday. v$ must be of the form yyyy.mmdd"
    PRINT "v$ = "; v$
    PRINT
ELSE
    yr& = VAL(LEFT$(v$, 4))
    mo& = VAL(MID$(v$, 6, 2))
    dy& = VAL(RIGHT$(v$, 2))
END IF
END SUB
```

APPENDIX E. THE HIGH ACCURACY KINEMATICS ROUTINE.

This appendix contains a listing of the alternate `pos.update` routine. It is a BASIC version of the highly accurate PPT2 subroutine provided by NAVSPASUR [Ref. 1] with their SHOWALL program. It is believed that it has been accurately translated, however any errors are solely those of the author. Known discrepacies are the difference in the WGS-72 constants used in SHOWALL and the IAU 1976 constants used here [see section III. SOURCES, in this report for details].

```
'*************************************************************************
SUB pos.update (ind%, elem#(), time#, sat.x#(), sat.xd#(), elemu#(), _
                 r#, ea#) STATIC    Rev. 09-26-89 @ 0900

' Case 1: Initialize orbital elements for time of epoch.
' Case 2: Update orbital elements and compute in-plane position & velocity.

' Comments from original FORTRAN version:
'      implicit real*8(a-h,o-z)
'***  model reference-Astronomical Journal 64,No 1274,Nov.,'59
'***  solution of the problem of artificial satellite theory
'***  with out drag...Dirk Brouwer.
'      840329  confirmed use of w=u x v  vice  w=u x vel
'      840329  inserted cdh,sdh, cdt,sdt re osc vice mean els at t
'      for use only with dcext2,dc1,dcia
'      frame for u,v,w is the rotated inertial w/o coriolis
'      830127  new partials re (a,x,y,z,p,q)
'      830127  improved (polynomial) model for singularity in Brouwer

DIM a(25)
DIM f(25), osc(10), kf(10), cf(10), bs(3, 4), u(3), v(3), w(3), vel(3)
DIM ar(6, 8), br(3, 6), cr(3, 6), dv(3)

'..............................................
'***  The commented beta value is the WGS-72 value,however using this
'***  value, one obtains a value for a(9) different then what we had
'***  been using, thus causing a problem with time.therefore to retain
'***  previous value of a(9),one solves for beta using the old value
'***  of a(9).
'***  beta1=398600.5d0
'      beta1 = 398597.62579588#, erkm = 6378.135#, flat = 298.26#
'      k = .0743669161#

' IAU 1976:
CONST twopi = 6.283185307179586#
CONST we = 1.002737909350795# * twopi / 1440#'sidereal period in radians/min
CONST ae = 6378.14#, erkm = ae          ' earth's equatorial radius (km.)
CONST beta1 = 398600.5#                 ' geo. grav. const. (km)^3/(sec)^2
      k = 60# * SQR(beta1 / ae) / ae    '(eru)^(3/2)/min
CONST flat = 1# / 298.257#              ' earth's flattening factor

'     k-terms      [Note: These may or may not agree with IAU 1976]
CONST c20 = -.0004841605#, c30 = .00000095958#
CONST c40 = .00000055199#, c50 = .000000065875#
'..............................................
CONST twotrd = .666666666667#, fortrd = 1.333333333333#
CONST tentrd = 3.333333333333#, ar = 48 * 0#

'IF ind% <> 1 THEN GOTO secondcase
SELECT CASE ind%
CASE 1

kz% = 1
a(1) = 0#'  [Note: = 85-01-01 in SHOWALL. Set to zero here.]
```

```
a(3) = -.5# * c20 * SQR(5#)
a(4) = c30 * SQR(7#)
a(5) = .375 * c40 * SQR(9#)
a(6) = c50 * SQR(11#)
'     two pi
a(7) = twopi
a(16) = 1# / a(7)
'     hergs/day , secs/herg, mins/herg
a(9) = erkm * SQR(erkm / beta1)
a(8) = 86400# / a(9)
a(17) = 1440# / a(8)
'     we(rad/day), we- 2 pi, we(rad/herg)
a(11) = 6.3003880987#
a(10) = a(11) - a(7)
a(12) = a(11) / a(8)
'     earth flattening
a(13) = (2# - 1# / flat) / flat
'     sm/er, km/er, nm/er
a(20) = erkm / 1.609344#
a(21) = erkm
a(22) = erkm / 1.852#
'     deg/rad
a(23) = 360# / twopi
'     range rate/er/herg to cycles/second - conversion
a(24) = a(21) * 216980000# / (a(9) * 299792.5#)
'     fence plane displac from earth center
a(25) = .0031#

' Patch to interface PPT2 and SATSTA variables:
f(1) = elem(8)          'mean anomaly
f(2) = elem(7) * a(17) 'mean motion
f(3) = elem(9) * a(17) ^ 2'first decay constant
f(4) = 0#               'second decay constant
f(5) = elem(2)          'eccentricity
f(6) = elem(6)          'arg of perigee
f(7) = elem(5)          'node
f(8) = COS(elem(4))     'cosine inclination
f(9) = 0#     ' epoch in PPT2, time from epoch here (? but check ?)

'  Used only once to "recover" the remaining elements
h1 = 0#
'***  f(8) is cosine of inclination
t9 = f(8) * f(8)
t1 = 1# - 5# * t9
'     t2=(1.0d0-exp(-100.0d0*t1^2))/t1
'     revised handling of divisor per navspasur o2t memo nov '83
beta = 100# / 2# ^ 11
p3 = beta * t1 ^ 2
p1 = EXP(-p3)
p2 = 1# + p1
p1ex = p1 ^ 2
p4 = 1#
p3ex = -.5# * p3
FOR n% = 2 TO 13
    IF n% <= 11 THEN p2 = p2 * (1# + p1ex)
    p1ex = p1ex * p1ex
    p4 = p4 + p3ex
    p3ex = -p3 * p3ex / (n% + 1)
NEXT
t2 = p2 * p4 * beta * t1
cio2 = SQR(.5# + .5# * f(8))
sio2 = SQR(.5# - .5# * f(8))
t12 = 3# * t9 - 1#
t13 = t9 * t9
t14 = t2 * t13
t15 = 1# - t9
```

```
'***  f(5) is eccentricity
esq0 = f(5) * f(5)
eta20 = 1# - esq0
eta0 = SQR(eta20)
t6 = eta0 * eta20
t7 = eta20 * eta20
t8 = eta0 + 1# / (1# + eta0)
f(13) = f(2) ^ (-twotrd)
elem(1) = f(13)
'***  f(2) is basically mean motion but also contains the
'***  secular term in mean anomaly. therefore f(13) is (almost) the
'***  semi major axis.

'***  This loop is to remove the contribution of the additional
'***  term in mean motion for improving values of semi-major axis
'***  and seculars.
'***  a(3) to a(6) are 'k' terms wgs 72
FOR i% = 1 TO 5
      fs13 = f(13) * f(13) * t7
      fs11 = a(3) / fs13
      fs13 = a(5) / (fs13 * fs13)
      tt = 1# + 1.5# * fs11 * eta0 * t12
      uu = -15# + 16# * eta0 + 25# * eta20
      uu = uu + (30# - 96# * eta0 - 90# * eta20) * t9
      uu = uu + (105# + 144# * eta0 + 25# * eta20) * t13
      tt = tt + .09375# * fs11 ^ 2 * eta0 * uu
      tt = tt + .9375# * fs13 * eta0 * esq0 * (3# - 30# * t9 + 35# * t13)
      f(13) = (tt / f(2)) ^ twotrd
NEXT
'***  f(11) will be rate of change of argument of perigee
uu = -35# + 24# * eta0 + 25# * eta20
uu = uu + (90# - 192# * eta0 - 126# * eta20) * t9
uu = uu + (385# + 360# * eta0 + 45# * eta20) * t13
tt = -1.5# * fs11 * t1 + .09375# * fs11 ^ 2 * uu
uu = 21# - 9# * eta20 + (-270# + 126# * eta20) * t9
uu = uu + (385# - 189# * eta20) * t13
f(11) = tt + .3125# * fs13 * uu
'***  f(12) will be rate of change of right ascention
uu = -5# + 12# * eta0 + 9# * eta20
uu = uu - (35# + 36# * eta0 + 5# * eta20) * t9
tt = -3# * fs11 * f(8) + .375# * fs11 ^ 2 * f(8) * uu
f(12) = tt + 1.25# * fs13 * f(8) * (5# - 3# * eta20) * (3# - 7# * t9)
'***  f(15)=sine inclination,f(14)is a dot,f(16) is eccen.dot
f(15) = SQR(t15)
f(14) = -fortrd * f(3) * f(13) / f(2)
f(16) = f(14) * f(5) * eta20 / f(13)
q1 = f(2) * f(13) ^ 1.5#
f(11) = f(11) / q1
f(12) = f(12) / q1
fs12 = a(4) / (a(3) * f(13) * eta20)
fs13 = fs13 / fs11 * tentrd
fs14 = a(6) / (a(3) * f(13) ^ 3 * eta20 * t7)
q1 = .125# * (fs11 * (1# - 11# * t9 - 40# * t14) - fs13 _
           * (1# - 3# * t9 - 8# * t14))
p5 = 1# + t2 * (8# * t9 + 20# * t14)
p2 = 1# + 2# * p5
q2 = .125# * esq0 * f(8) * (fs11 * (1# + 10# * p5) - fs13 * p2)
p2 = .46875# * p2 * f(5) * f(8) * f(15) * (4# + 3# * esq0) * fs14
p3 = fs14 * (1# - 9# * t9 - 24# * t14)
q5 = .25# * (fs12 + .3125# * (4# + 3# * esq0) * p3)
p3 = .15625# * f(5) * f(15) * p3
p4 = .030381944# * f(5) * fs14 * (1# - 5# * t9 - 16# * t14)
p5 = .060763889# * f(5) * esq0 * f(8) * f(15) * fs14 * (1# + 4# * p5)
vle1 = f(5) * eta20 * q1
vlh1i = -f(15) * q2
tt = (t6 - 1#) * q1 - q2 + 25# * esq0 * t14 * t9 * t2 * (fs11 - .2# * fs13)
```

```
vls1 = tt - .0625# * esq0 * (fs11 * (1# - 33# * t9 - 200# * t14) _
          - fs13 * (1# - 9# * t9 - 40# * t14))
vle2 = eta20 * f(15) * q5
vlh2i = f(5) * f(8) * q5 + f(15) * p2
vls2 = f(5) * f(15) * (t8 + f(8) / (1# + f(8))) * q5 _
            + (11# + 3# * esq0 - 3# * t6) * p3 + (1# - f(8)) * p2
vle3 = -3# * f(5) * eta20 * f(15) * p4
vlh3i = -esq0 * f(8) * p4 - f(15) * p5
vls3 = f(15) * (3# * t6 - 3# - 2# * esq0 - esq0 * f(8) / (1# + f(8))) * p4 _
            - (1# - f(8)) * p5
vll2 = vle2 + 3# * f(5) * eta20 * p3
'     precomps for the partials
'***  f(6) is argument of perigee,f(7) is right ascension
sinh0 = SIN(f(7))
cosh0 = COS(f(7))
sint0 = SIN(f(6) + f(7))
cost0 = COS(f(6) + f(7))
'     kepler's law
dadm = -twotrd * f(13) / f(2)
'     variations thru the seculars
secul = a(3) / (f(13) * eta20) ^ 2
dgddm = 3# * secul * t1 / f(13) * dadm
dgdde = -6# * secul * t1 * f(5) / eta20
dgddi = -15# * secul * f(8) * f(15)
dhddm = 6# * secul * f(8) / f(13) * dadm
dhdde = -12# * secul * f(8) * f(5) / eta20
dhddi = 3# * secul * f(15)
dtddx = (dgdde + dhdde) * cost0
dtddy = (dgdde + dhdde) * sint0
dtddp = (dgddi + dhddi) * 2# * cosh0 / cio2
dtddq = (dgddi + dhddi) * 2# * sinh0 / cio2
dhddx = dhdde * cost0
dhddy = dhdde * sint0
dhddp = dhddi * 2# * cosh0 / cio2
dhddq = dhddi * 2# * sinh0 / cio2
'     variations thru m2
daddm1 = fortrd * f(3) * (f(13) / f(2) - dadm) / f(2)
daddm2 = -fortrd * f(13) / f(2)
dedde = -fortrd * (1# - 3# * esq0) * f(3) / f(2)
deddx = dedde * cost0
deddy = dedde * sint0
deddm1 = fortrd * f(5) * eta20 * f(3) / f(2) ^ 2
deddm2 = -fortrd * f(5) * eta20 / f(2)
FOR i% = 1 TO 9
     elemu#(i%) = elem#(i%)
NEXT

'secondcase:
CASE ELSE

tm = time# / a(17)   'Convert minutes to Hergs
h1 = tm - f(9)
'***  compute position as a function of time (in call line)
osc(9) = h1 * (f(2) + h1 * (f(3) + h1 * f(4)))
osc(3) = dmod(f(1) + osc(9), twopi)
tt = f(5) + f(16) * h1
IF tt < 0# THEN tt = 0#
IF tt > .99999# THEN tt = .99999#
osc(4) = tt
esq = osc(4) ^ 2
eta2 = 1# - esq
eta = SQR(eta2)
CALL kepler(osc(3), osc(4), c3)
osc(1) = COS(c3)
c1 = 1# - osc(4) * osc(1)
osc(1) = (osc(1) - osc(4)) / c1
```

```
 osc(2) = eta * SIN(c3) / c1

 ts4 = 1# + osc(4) * osc(1)
 cf3 = 0#
 IF kz% <> 0 THEN cf3 = a(12)
 tt = f(13) + f(14) * h1
 IF tt < 1# THEN tt = 1#
 osc(8) = tt
 osc(7) = f(8)
 fsi = f(15)
 osc(6) = dmod(f(7) + f(12) * osc(9), twopi)
 osc(5) = dmod(f(6) + f(11) * osc(9), twopi)

 r = osc(8) * eta2 / ts4
 osc(6) = osc(6) - cf3 * (tm - a(1))
 w1 = COS(osc(5))
 w2 = SIN(osc(5))
 w7 = COS(osc(6))
 w8 = SIN(osc(6))
 agda = 0#
 agde = 0#
 agdi = 0#
 agdl = 0#
 agdg = 0#
 agdh = 0#

 w3 = w1 * w1 - w2 * w2
 w4 = 2# * w1 * w2
 w5 = w3 * w1 - w4 * w2
 w6 = w4 * w1 + w3 * w2
 w9 = osc(1) * osc(1) - osc(2) * osc(2)
 w10 = 2# * osc(1) * osc(2)
 w11 = w3 * osc(1) - w4 * osc(2)
 w12 = w4 * osc(1) + w3 * osc(2)
 w13 = w3 * w9 - w4 * w10
 w14 = w4 * w9 + w3 * w10
 w15 = w13 * osc(1) - w14 * osc(2)
 w16 = w14 * osc(1) + w13 * osc(2)
 w17 = atan2(osc(2), osc(1)) + osc(4) * osc(2) - dmod(osc(3), twopi)
 w18 = COS(osc(3))
 w19 = SIN(osc(3))
 w20 = osc(1) * (3# + osc(4) * osc(1) * (3# + osc(4) * osc(1)))
 w21 = 3# * w14 + 3# * osc(4) * w12 + osc(4) * w16
 w22 = ts4 * (1# + ts4) / eta20
 osc(8) = osc(8) * (1# + fs11 / eta20 * (t12 * (ts4 ^ 3 - t6) _
            + 3# * t15 * ts4 ^ 3 * w13)) + agda
 de = vle1 * w3 + vle2 * w2 + vle3 * w6
 di = sio2 + .5# * cio2 * (.5# * fs11 * f(8) * f(15) * (3# * w13 + 3# * f(5) _
     * w11 + f(5) * w15) - de * f(5) * f(8) / f(15) / eta20) + .5# * cio2 * agdi
 de = osc(4) + .5# * fs11 * (t12 * (w20 + f(5) * t8) + 3# * t15 * (w20 + f(5)) _
             * w13 - eta20 * t15 * (3# * w11 + w15)) + de + agde
 dh = .5# / cio2 * (-.5# * fs11 * f(8) * f(15) * (6# * w17 - w21) + vlh1i * w4 _
          + vlh2i * w1 + vlh3i * w5) + .5# * agdh / cio2
 dl = -.25# * t6 * fs11 * (2# * t12 * (w22 + 1#) * osc(2) + 3# * t15 * ((1# _
            - w22) * w12 + (.333333333# + w22) * w16))
 os = osc(3) + osc(5) + osc(6) - dl * f(5) * (t8 - 1#) / t6 - .25# * fs11 * (6# _
             * w17 * (t1 + 2# * f(8)) - w21 * (t1 + 2# + 2# * f(8))) _
             + vls1 * w4 + vls2 * w1 + vls3 * w5 + agdg
 dl = dl + eta0 * (vle1 * w4 - vll2 * w1 - vle3 * w5) + agdl
 esq = de * de + dl * dl
 osc(4) = SQR(esq)
 osc(3) = atan2(de * w19 + dl * w18, de * w18 - dl * w19)
 osc(7) = 1# - 2# * (di * di + dh * dh)
 fsi = SQR(1# - osc(7) * osc(7))
 osc(6) = atan2(di * w8 + dh * w7, di * w7 - dh * w8)
 osc(5) = os - osc(3) - osc(6)
 eta2 = 1# - esq
 eta = SQR(eta2)

 CALL kepler(osc(3), osc(4), c3)
 osc(1) = COS(c3)
 c1 = 1# - osc(4) * osc(1)
 osc(1) = (osc(1) - osc(4)) / c1
 osc(2) = eta * SIN(c3) / c1

 ts4 = 1# + osc(4) * osc(1)
 r = osc(8) * eta2 / ts4
 w7 = COS(osc(6))
 w8 = SIN(osc(6))
 w1 = COS(osc(5))
 w2 = SIN(osc(5))
 ubs = w2 * osc(1) + w1 * osc(2)
 ubc = w1 * osc(1) - w2 * osc(2)
 u(1) = w7 * ubc - w8 * ubs * osc(7)
 u(2) = w8 * ubc + w7 * ubs * osc(7)
 u(3) = ubs * fsi
 v(1) = -w7 * ubs - w8 * ubc * osc(7)
 v(2) = -w8 * ubs + w7 * ubc * osc(7)
 v(3) = ubc * fsi
 w(1) = w8 * fsi
 w(2) = -w7 * fsi
 w(3) = osc(7)
 ts2 = eta * SQR(osc(8))
 FOR i% = 1 TO 3
     vel(i%) = (osc(4) * osc(2) * u(i%) + ts4 * v(i%)) / ts2
 NEXT
 vel(1) = vel(1) + cf3 * r * u(2)
 vel(2) = vel(2) - cf3 * r * u(1)

 '  Patch for PPT2 to SATSTA output
 elemu(1) = osc(8)'semimajor axis
 elemu(2) = osc(4)'eccentricity
 'elemu(3)=            'time of last perigee passage
 elemu(4) = acos(osc(7))'inclination
 elemu(5) = osc(6)'node
 elemu(6) = osc(5)'arg of perigee
 elemu(8) = osc(3)'mean anomaly

 ma# = elemu(8)
 CALL kepler(ma#, elem#(2), ea#)
   s0# = SIN(ea#)
   c0# = COS(ea#)
 'elemu#(8) = ma#
 elemu#(3) = time# - ma# / elem#(7)

 ' in-plane position and velocity
 ecc2# = 1# - elemu#(2) ^ 2
 sat.x#(1) = elemu#(1) * (c0# - elem#(2))
 sat.x#(2) = elemu#(1) * SQR(ecc2#) * s0#
 sat.x#(3) = 0#
 r# = elemu#(1) * (1# - elem#(2) * c0#)
 ed# = elemu#(1) * elem#(7) / r#
 sat.xd#(1) = -elemu#(1) * ed# * s0#
 sat.xd#(2) = elemu#(1) * ed# * SQR(ecc2#) * c0#
 sat.xd#(3) = 0#

 ' equatorial rectangular coordinates
 CALL euler(elemu(6), elemu(5), elem(4), sat.x(), sat.xd())

 END SELECT
 END SUB
```

REFERENCES

1. Private communication, Mr. Fred Lipp, Code 20, Naval Space Surveillance Center, Dahlgren, VA 22448.
2. Brouwer, D., "Solution of the Problem of Artificial Satellite Theory without Drag," *The Astronomical Journal*, Vol. 64, No. 1274, Nov. 1959, pp. 378-397.
3. Escobal, P. R., *Methods of Orbit Determination*, John Wiley & Sons, Inc., New York, 1965.
4. Grissom, W., & Guy, R., "Satellite Tracking (SATRAK) Operator's Guide," Version 2.6, Report MG87-SATRAK-HV-6-0001, September 1987, Headquarters Air Force Space Command/DOA, Peterson AFB, CO 80914-5001.
5. Hoots, F. R. and Roehrich, R. L., "Models for Propagation of NORAD Element Sets," Project Spacetrack Report No. 3, December 1980, Aerospace Defense Command, United States Air Force, Peterson AFB, CO.
6. Mikkola, S., "A Cubic Approximation for Kepler's Equation," *Celestial Mechanics*, Vol. 40, Nos. 3–4, 1987, pp. 329-334.
7. Shudde, R. H., "An Analysis of the Satellite Kinematics Model in the Naval Warfare Gaming System," Technical Report NPS55-83-022, September 1983, Naval Postgraduate School, Monterey, CA 93943-5000.
8. Shudde, R. H., "Some Tactical Algorithms for Spherical Geometry," Technical Report NPS55-86-008, March 1986, Naval Postgraduate School, Monterey, CA 93943-5000.
9. Shudde, R. H., "Some Navigation and Almanac Algorithms," Technical Report NPS55-85-023, September 1985, Naval Postgraduate School, Monterey, CA 93943-5000.
10. *The Astronomical Almanac for the Year 1984*, U.S. Government Printing Office, Washington, D.C., 1983.