APPENDIX B: USING BROUWER-MEAN ELEMENTS FROM TBUS PART IV

The Brouwer-mean elements in part IV of the APT predict bulletin (TBUS) can be used in a stand-alone Brouwer-Lyddane orbit prediction package to determine orbit position information at any time ($t-t_o$) where t_o represents the time of the Brouwer mean elements in part IV and t represents the user request time. The Brouwer-Lyddane algorithm is an analytical solution of satellite motion for a simplified disturbing potential field limited to zonal harmonic coefficients for J_2 through J_5. Lyddane modified Brouwer's formulation to obtain algorithms applicable for zero eccentricity and zero inclination.

The Brouwer-Lyddane orbit prediction package contains seven subroutines and one block data subprogram which can be called from a user supplied driver to obtain orbit information in the form desired by the user.

The first subroutine to be called is BROLYD. This subroutine takes as input the Brouwer mean or osculating elements at time t_o, and outputs the osculating Keplerian and Brouwer mean elements at the time ($t-t_o$) given in common block BLCNST. The calling sequence for subroutine BROLYD is described below.

If users require output in the form of inertial position and velocity vectors then a second subroutine CELEM can be called. This subroutine takes as input the osculating Keplerian elements for BROLYD and outputs the inertial position and velocity vectors. The calling sequences for subroutine CELEM is also described below.

A third subroutine, BFIXED, transforms the position and velocity vectors from CELEM to earth fixed coordinates. The user must supply the Greenwich hour angle to this subroutine.

Subroutine XYZPLH converts the position vector in the earth fixed coordinates to geodetic latitude, east longitude, and height.

Three other subroutines are included in this prediction package. These are DKEPLR, MA3331, and DATAN0. DKEPLR is a subroutine to solve Kepler's equation. MA3331 computes the product of a 3X3 matrix and a 3X1 matrix. DATAN0 computes a value for the arc-tangent between 0 and 2π.

A block data subprogram for the common block BLCNST includes several constants needed by the stand-alone orbit prediction package. These constants are described below and are presently used in NESDIS's polar navigation system.

CALLING SEQUENCE FOR SUBROUTINE BROLYD:

CALL BROLYD (OSCELE, DPELE, IPERT, IPASS, IDMEAN, ORBEL)

ARGUMENTS:

OSCELE - OUTPUT OSCULATING ELEMENTS AT TIME TTO

OSCELE (1) = SEMI-MAJOR AXIS
OSCELE (2) = ECCENTRICITY
OSCELE (3) = INCLINATION
OSCELE (4) = NODE
OSCELE (5) = ARGUMENT OF PERIGEE
OSCELE (6) = MEAN ANOMALY

DPELE - INPUT IS OSCULATING ELEMENTS AT EPOCH IF IDMEAN = 0
INPUT IS BROUWER MEAN AT EPOCH IF IDMEAN ≠ 0
OUTPUT ELEMENTS ARE BROUWER MEAN AT TIME TTO

DPELE (1) = SEMI-MAJOR AXIS
DPELE (2) = ECCENTRICITY
DPELE (3) = INCLINATION
DPELE (4) = NODE
DPELE (5) = ARGUMENT OF PERIGEE
DPELE (6) = MEAN ANOMALY

IDMEAN - DETERMINES WHICH ELEMENTS ARE INPUT IN DPELE
= 0, OSCULATING
≠ 0, BROUWER MEAN

IPASS =1, COMPUTE CONSTANTS NEEDED IN COMPUTATION OF OSCULATING ELEMENTS
=2, UPDATE OSCULATING ELEMENT TO OBSERVATION TIME WITHOUT UPDATING CONSTANTS

IPERT =0, NO PERTURBATIONS DUE TO OBLATENESS COMPUTED
=1, SECULAR TERMS COMPUTED
=2, SECULAR + LONG PERIODIC + SHORT PERIODIC TERMS

ORBEL - OUTPUT AUXILIARY ORBITAL ELEMENTS

CALLING SEQUENCE FOR SUBROUTINE CELEM:

CALL CELEM (ORBEL, GMC, PV, VV)

ARGUMENTS:

ORBEL - INPUT OSCULATING ELEMENTS

ORBEL (1) = SEMI-MAJOR AXIS

ORBEL (2) = ECCENTRICITY

ORBEL (3) = INCLINATION

ORBEL (4) = NODE

ORBEL (5) = ARGUMENT OF PERIGEE

ORBEL (6) = MEAN ANOMALY

GMC - INPUT GRAVITATIONAL CONSTANT

PV - OUTPUT CARTESIAN POSITION VECTOR

PV (1) = X

PV (2) = Y

PV (3) = Z

VV - OUTPUT CARTESIAN VELOCITY VECTOR

VV (1) = XDOT

VV (2) = YDOT

VV (3) = ZDOT

COMMON BLOCK BLCNST

COMMON/BLCNST/TTO, R, AE, GM, BJ2, BJ3, BJ4, BJ5, FLTINV, XKE, ESQ

VARIABLES USED IN COMMON/BLCNST/:

TTO - INPUT REQUEST TIME IN SECONDS FROM EPOCH

R - OUTPUT MAGNITUDE OF SATELLITE RADIUS VECTOR

AE - INPUT MEAN EQUATORIAL RADIUS OF THE EARTH (KM)

GM - INPUT GRAVITATIONAL CONSTANT OF THE EARTH (KM^3/SEC^2)

BJ2 - INPUT $C_{2,0}$ ZONAL HARMONIC COEFFICIENT

BJ3 - INPUT $C_{3,0}$ ZONAL HARMONIC COEFFICIENT

BJ4 - INPUT $C_{4,0}$ ZONAL HARMONIC COEFFICIENT

BJ5 - INPUT $C_{5,0}$ ZONAL HARMONIC COEFFICIENT

FLTINV - INPUT INVERSE FLATTENING COEFFICIENT (1/F)

XKE - GRAVITATIONAL CONSTANT ($(EARTH\ RADII)^{3/2}/MIN$)

ESQ - THE SQUARE OF THE MAJOR ECCENTRICITY CALCULATED FROM
$e^2=(2f-f^2)$

```
C**********************************************************************
C
C                              NAME - AMMSMA
C        LANGUAGE - FORTRAN              TYPE - SUBROUTINE
C     VERSIONS - 1.0        DATE - 07/01/81      PROGRAMMER - T.LIU
C
C                              FUNCTIONS:
C          TO CALCULATE THE AVERAGE MEAN MOTION AND THE SEMIMAJOR
C                              AXIES.
C                              INPUT PARAMETERS:
C                              COMMON/DATA2/........
C                              OUTPUT PARAMETERS:
C                              DMEAN - AVERAGE MEAN MOTION
C                              BMELMT(1) - SEMIMAJOR AXIES
C
C                              SUBROUTINES CALLED: NONE
C**********************************************************************
      SUBROUTINE AMMSMA(DMEAN,BMELMT)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/DATA2/ DESIGI,EPTIME,DMMDT,D2MDT,
     1 DRAGT,IETYPE,NELSET,DINCL,RASC,ECC,ARGP,DMEANA,
     2 DMMOT,IREVNO,ISATNO
      REAL*8 BMELMT(6)
      COMMON/BLCNST/ TTO,R,AE,GM,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ
      DATA TOTHRD,RE,DEGRAD/0.66666667,1.,0.01745329252D0/
      DATA TWOPI/6.2831853/
      DATA XMNPDA/1440./
      XJ2=-BJ2
      CK2=.5*XJ2*RE**2
      TEMP=TWOPI/XMNPDA/XMNPDA
      DMEN=DMMOT*TEMP*XMNPDA
      RINCL=DINCL*DEGRAD
      A1=(XKE/DMEN)**TOTHRD
      COSIO=DCOS(RINCL)
      THETA2=COSIO*COSIO
      X3THM1=3.*THETA2-1.
      EOSQ=ECC*ECC
      BETAO2=1.-EOSQ
      BETAO=DSQRT(BETAO2)
      DEL1=1.5*CK2*X3THM1/(A1*A1*BETAO*BETAO2)
      AO=A1*(1.-DEL1*(.5*TOTHRD+DEL1*(1.+134./81.*DEL1)))
      DELO=1.5*CK2*X3THM1/(AO*AO*BETAO*BETAO2)
      DMEAN=DMEN/(1.+DELO)
      DSEMI=AO/(1.-DELO)
      BMELMT(1)=DSEMI*AE
      RETURN
      END
      SUBROUTINE BFIXED(KEY,GHA,PV,VV,POSOUT,VELOUT,B)
      IMPLICIT REAL*8 (A-H, O-Z)
C****************************************************************C
C     NAME - BFIXED
C
C     LANGUAGE- FORTHXP TYPE- SUBROUTINE
C
C     VERSION- 1.0      DATE- 10/14/77          PROGRAMMER- SACHS, A.
C
C     PURPOSE -TRANSFORM THE POSITION AND VELOCITY FROM TOD TO
C     PSUEDO BODY FIXED.
C
C     INPUT PARAMETERS - KEY= 3 FOR RETURN OF BODY FIXED POSITION
C     ONLY, GHA= GREENWICH HOUR ANGLE IN RADIANS, PV= POSITION VECTOR,
C     VV= VELOCITY VECTOR (KM/SEC).
C
C     OUTPUT PARAMETERS - POSOUT =POSITION VECTOR, VELOUT =VELOCITY
C     VECTOR, B =ROTATION MATRIX.
C
C     SUBROUTINES CALLED - MA3331
C
C     COMMENT- B MATRIX COMPUTATION IS FROM SUBROUTINE EVAL OF GTDS.
C
C****************************************************************C
C
      DIMENSION B(3,3), BDOT(3,3), PV(3), VV(3), VELOUT(3), VOUT(3)
      DIMENSION POSOUT(3)
      DATA OMEGAE, BDOT/7.29211585494D-5, 9*0.D0/
C COMPUTE MATRIX TO ROTATE POSITION FROM TOD TO PSUEDO BODY FIXED.
C SPIN FACTOR IS ZERO.
      XP=0.0D0
      YP=0.0D0
      B(1,1) = DCOS(GHA)
      B(1,2) = DSIN(GHA)
      B(1,3) = XP
      B(2,1) = -B(1,2)
      B(2,2) = B(1,1)
      B(2,3) = -YP
      B(3,1) = -XP*B(1,1)-YP*B(1,2)
      B(3,2) = -XP*B(1,2)+YP*B(1,1)
      B(3,3) = 1.0D0
C ROTATE THE INPUT POSITION VECTOR.
      CALL MA3331 (B,PV.POSOUT)
      IF (KEY.EQ.3) GO TO 30
C COMPUTE MATRIX TO ROTATE VELOCITY FROM TOD TO PSUEDO BODY FIXED.
      BDOT(1,1) = -B(1,2)
      BDOT(1,2) = B(1,1)
      BDOT(2,1) = -B(1,1)
      BDOT(2,2) = -B(1,2)
C ROTATE THE INPUT VELOCITY VECTOR.
      CALL MA3331(B,VV,VOUT)
C ROTATE THE INPUT POSITION VECTOR.
      CALL MA3331(BDOT,PV,VELOUT)
```

```
C OBTAIN THE BODY FIXED VELOCITY.
      DO 20 I=1,3
   20 VELOUT(I) = VELOUT(I)*OMEGAE + VOUT(I)
   30 CONTINUE
      RETURN
      END
      BLOCK DATA
      IMPLICIT REAL*8 (A-H,O-Z)
C****************************************************************C
C     NAME- BLCNST
C
C     LANGUAGE- FORTHXP       TYPE- PROGRAM
C
C     THIS COMMON BLOCK WAS UPDATED MARCH 28, 1984 TO INCLUDE XKE
C     AND ESQ BY E. HARROD S/SP12
C     THIS BLOCK DATA IS COMPILED WITH THE ROUTINE PSCEAR, ANY
C     PROGRAM USING PSCEAR DOES NOT NEED TO RECOMPILE THIS BLOCK
C     DATA
C
C****************************************************************C
      COMMON/BLCNST/ TTO,R,AE,GM,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ
      DATA TTO,R,GM,AE,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ/2*0.D0,
     1 398600.8D0,6378.135D0,-0.10826158D-02,0.25388100D-05,
     2 0.16559700D-05,0.21848266D-06,298.25D0,0.743669161D-01,
     3 0.6994317778266721D-02/
      END
      SUBROUTINE BROLYD(OSCELE,DPELE,IPERT,IPASS,IDMEAN,ORBEL)
C*****************************************************************
C*    REF. "BROUWER-LYDDANE ORBIT GENERATOR ROUTINE"            *
C*                (X-553-70-223)                                *
C*          BY E.A. GALBREATH 1970                              *
C*--------------------------------------------------------------*
C*    MODIFIED 7/31/74 VIONA BROWN AND R.A. GORDON TO INTERFACE *
C*    WITH GTDS                                                 *
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 PI2/6.283185307179586D0/
      DIMENSION OSCELE(6), DPELE(6), ORBEL(5)
      COMMON /BLCNST/ TTO,R,AE,GM,BJ2,BJ3,BJ4,BJ5,FLTINV,XKE,ESQ
      DATA BMU,RE/1.0D0,1.0D0/,BKSUBC/0.01D0/
      EK = DSQRT(GM/AE**3)
      DELT = EK*TTO
      GO TO (10,111), IPASS
C
C EPOCH ELEMENTS AT EPOCH TIME
C
   10 ADP = DPELE(1)/AE
      EDP = DPELE(2)
      BIDP = DPELE(3)
      HDP = DPELE(4)
      GDP = DPELE(5)
      BLDP = DPELE(6)
      A0 = ADP
      E0 = EDP
      BI0 = BIDP
      H0 = HDP
      G0 = GDP
      BL0 = BLDP
      IFLG = 0
C
C COMPUTE MEAN MOTION
C
      ANU=DSQRT(BMU/A0**3)
C
C COMPUTE FRACTIONS
C
      F3D8=3.0D0/8.0D0
      F1D2=1.0D0/2.0D0
      F3D2=3.0D0/2.0D0
      F1D4=1.0D0/4.0D0
      F5D4=5.0D0/4.0D0
      F1D8=1.0D0/8.0D0
      F5D12=5.0D0/12.0D0
      F1D16=1.0D0/16.0D0
      F15D16=15.0D0/16.0D0
      F5D24=5.0D0/24.0D0
      F3D32=3.0D0/32.0D0
      F15D32=15.0D0/32.0D0
      F5D64=5.0D0/64.0D0
      F35384=35.0D0/384.0D0
      F35576=35.ODO/576.0D0
      F35D52=35.0D0/1152.0D0
      F1D3=1.0D0/3.0D0
      F5D16=5.0D0/16.0D0
      BK2 = -F1D2*(BJ2*RE*RE)
      BK3 = BJ3*RE**3
      BK4 = F3D8*(BJ4*RE**4)
      BK5=BJ5*RE**5
      GO TO 153
  111 IF(IPERT.EQ.0)GO TO 7
      IF(IDMEAN.NE.0)GO TO 202
      ADP = DPELE(1)/AE
      EDP = DPELE(2)
      BIDP = DPELE(3)
      HDP = DPELE(4)
      GDP = DPELE(5)
      BLDP = DPELE(6)
  153 EDP2=EDP*EDP
      CN2=1.0-EDP2
      CN=DSQRT(CN2)
      GM2=BK2/ADP**2
      GMP2=GM2/(CN2*CN2)
      GM4=BK4/ADP**4
      GMP4=GM4/CN**8
      THETA=DCOS(BIDP)
      THETA2=THETA*THETA
      THETA4=THETA2*THETA2
  202 IF(IDMEAN.EQ.0)GO TO 155
      IF(IPASS.EQ.2) GO TO 150
C
C COMPUTE LDOT,GDOT,HDOT
C
  157 BLDOT=CN*ANU*(GMP2*(F3D2*(3.0*THETA2-1)+GMP2*F3D32*(THETA2
     1*(-96.0*CN+30.0-90.0*CN2)+(16.0*CN+25.0*CN2-15.0)+THETA4
     2*(144.0*CN+25.0*CN2+105.0)))+EDP2*GMP4*F15D16*(3.0+35.0*THETA4
     3-30.0*THETA2))
      GDOT=ANU*(F5D16*GMP4*((THETA2*(126.0*CN2-270.0)+THETA4*(385.0
     1-189.0*CN2))-9.0*CN2+21.0)+GMP2*(F3D32*GMP2*(THETA4*(45.0*CN2
     2+360.0*CN+385.0)+THETA2*(90.0-192.0*CN-126.0*CN2)+(24.0*CN
     3+25.0*CN2-35))+F3D2*(5*THETA2-1)))
      HDOT=ANU*(GMP4*F5D4*THETA*(3.0-7.0*THETA2)*(5.0-3.0*CN2)+GMP2
     1*(GMP2*F3D8*(THETA*(12.0*CN+9.0*CN2-5.0)-THETA*THETA2*(5.0*CN2
     2+36.0*CN+35.0))-3*THETA))
  155 IF(IFLG.EQ.1)GO TO 19
C
C  COMPUTE ISUBC TO TEST CRITICAL INCLINATION
C
      BISUBC=((1.0-5.0*THETA2)**(-2))*((25.0*THETA4*THETA)*(GMP2*EDP2))
      IFLG=1
C
C  FIRST CHECK FOR CRITICAL INCLINATION
C
```

```
      IF(BISUBC.GT.BKSUBC)GO TO 158
      ASSIGN 163 TO ID8
      GO TO 159
C
C  IS THERE CRITICAL INCLINATION?
C
   19 IF(BISUBC.GT.BKSUBC)GO TO 150
  159 IF(IPERT.EQ.1)GO TO 150
      GM3=BK3/ADP**3
      GMP3=GM3/(CN2*CN2*CN2)
      GM5=BK5/ADP**5
      GMP5=GM5/CN**10
      G3DG2=GMP3/GMP2
      G4DG2=GMP4/GMP2
      G5DG2=GMP5/GMP2
C
C COMPUTE A1-A8
C
      A1=(F1D8*GMP2*CN2)*(1.0-11.0*THETA2-((40.0*THETA4)/(1.0-5.0*THETA2)))
      A2=(F5D12*G4DG2*CN2)*(1.0-((8.0*THETA4)/(1.0-5.0*THETA2))-3.0*THETA2)
      A3=G5DG2*((3.0*EDP2)+4.0)
      A4=G5DG2*(1.0-(24.0*THETA4)/(1.0-5.0*THETA2)-9.0*THETA2)
      A5=(G5DG2*(3.0*EDP2+4.0))*(1.0-(24.0*THETA4)/(1.0-5.0*THETA2)-
9.0*THETA2)
      A6=G3DG2*F1D4
      SINI=DSIN(BIDP)
      A10=CN2*SINI
      A7=A6*A10
      A8P=G5DG2*EDP*(1.0-(16.0*THETA4)/(1.0-5.0*THETA2)-5.0*THETA2)
      A8=A8P*EDP
C
C   COMPUTE B13-B15
C
      B13=EDP*(A1-A2)
      B14=A7+F5D64*A5*A10
      B15=A8*A10*F35384
C
C   COMPUTE A11-A27
C
      A11=2.0+EDP2
      A12=3.0*EDP2+2.0
      A13=THETA2*A12
      A14=(5.0*EDP2+2.0)*(THETA4/(1.0-5.0*THETA2))
      A17=THETA4/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
      A15=(EDP2*THETA4*THETA2)/((1.0-5.0*THETA2)*(1.0-5.0*THETA2))
      A16=THETA2/(1.0-5.0*THETA2)
      A18=EDP*SINI
      A19=A18/(1.0+CN)
      A21=EDP*THETA
      A22=EDP2*THETA
      SINI2=DSIN(BIDP/2.0)
      COSI2=DCOS(BIDP/2.0)
      TANI2=DTAN(BIDP/2.0)
      A26=16.0*A16+40.0*A17+3.0
      A27=A22*F1D8*(11.0+200.0*A17+80.0*A16)
C
C  COMPUTE B1-B12
C
      B1=CN*(A1-A2)-((A11-400.0*A15-40.0*A14-11.0*A13)*F1D16+(11.0+200.0
     1*A17+80.0*A16)*A22*F1D8)*GMP2+((-80.0*A15-8.0*A14-3.0*A13+A11)
     2*F5D24+F5D12*A26*A22)*G4DG2
      B2=A6*A19*(2.0+CN-EDP2)+F5D64*A5*A19*CN2-F15D32*A4*A18*CN*CN2
1+(F5D64*A5+A6)*A21*TANI2+(9.0*EDP2+26.0)*F5D64*A4*A18+F15D32*A3*
2A21*A26*SINI*(1.0-THETA)
      B3=((80.0*A17+5.0+32.0*A16)*A22*SINI*(THETA-1.0)*F35576 *G5DG2*EDP)
     1-((A22*TANI2+(2.0*EDP2+3.0*(1.0-CN2*CN))*SINI)*F35D52*A8P)
      B4=CN*EDP*(A1-A2)
      B5=((9.0*EDP2+4.0)*A10*A4*F5D64+A7)*CN
      B6=F35384*A8*CN2*CN*SINI
      B7=((CN2*A18)/(1.0-5.0*THETA2))*(F1D8*GMP2*(1.0-15.0*THETA2)+(1.0
     1-7.0*THETA2)*G4DG2*(-F5D12))
      B8=F5D64*(A3*CN2*(1.0-9.0*THETA2-(24.0*THETA4/(1.0-5.0*THETA2))))
     1+A6*CN2
      B9=A8*F35384*CN2
      B10=SINI*(A22*A26*G4DG2*F5D12-A27*GMP2)
      B11=A21*(A5*F5D64+A6+A3*A26*F15D32*SINI*SINI)
      B12=-((80.0*A17+32.0*A16+5.0)*(A22*EDP*SINI*SINI*F35576*G5DG2)+(A8
     1*A21*F35D52))
  150 IF (IPERT.EQ.0)GO TO 7
      IF (IDMEAN.EQ.0)GO TO 4
C
C COMPUTE SECULAR TERMS
C  "MEAN" MEAN ANOMALY
C
      BLDP = ANU*DELT + BLDOT*DELT+BL0
      BLDP = DMOD(BLDP,PI2)
      IF(BLDP.LT.0.0D0)BLDP = BLDP + PI2
C
C MEAN ARGUMENT OF PERIGEE
C
      GDP = GDOT*DELT + G0
      GDP = DMOD(GDP,PI2)
      IF(GDP.LT.0.0D0)GDP = GDP + PI2
C
C  MEAN LONGITUDE OF ASCENDING NODE
C
      HDP = HDOT*DELT + H0
      HDP = DMOD(HDP,PI2)
      IF(HDP.LT.0.0D0)HDP = HDP + PI2
    4 DO 33 NN=1,6
   33 OSCELE(NN) = DPELE(NN)
      A = ADP
      E = EDP
      BI = BIDP
      H = HDP
      G = GDP
      BL = BLDP
C
C COMPUTE TRUE ANOMALY (DOUBLE PRIMED)
C
      EADP = DKEPLR(BLDP,EDP)
      SINDE = DSIN(EADP)
      COSDE = DCOS(EADP)
      SINFD = CN*SINDE
      COSFD = COSDE - EDP
      FDP = DATAN0(SINFD,COSFD)
      IF(IPERT.EQ.1)GO TO 7
      DADR=(1.0-EDP*COSDE)**(-1)
      SINFD=SINFD*DADR
      COSFD=COSFD*DADR
      CS2GFD=DCOS(2.0*GDP+2.0*FDP)
      DADR2=DADR*DADR
      DADR3=DADR2*DADR
      COSFD2=COSFD*COSFD
C
C COMPUTE A (SEMI-MAJOR AXIS)
C
      A=ADP*(1.0+GM2*((3.0*THETA2-1.0)*(EDP2/(CN2*CN2*CN2))*(CN+(1.0/(1.
     1+CN)))+((3.0*THETA2-1.0)/(CN2*CN2*CN2))*(EDP*COSFD)*(3.0+3.0*EDP
     2*COSFD+EDP2*COSFD2)+3.0*(1.0-THETA2)*DADR3*CS2GFD))
      SN2GFD=DSIN(2.0*GDP+2.0*FDP)
      SNF2GD=DSIN(2.0*GDP+FDP)
      CSF2GD=DCOS(2.0*GDP+FDP)
      SN2GD=DSIN(2.0*GDP)
      CS2GD=DCOS(2.0*GDP)
      SN3GD=DSIN(3.0*GDP)
      CS3GD=DCOS(3.0*GDP)
      SN3FGD=DSIN(3.0*FDP+2.0*GDP)
      CS3FGD=DCOS(3.0*FDP+2.0*GDP)
      SINGD=DSIN(GDP)
      COSGD=DCOS(GDP)
      GO TO ID8, (163,164)
  163 DLT1E=B14*SINGD+B13*CS2GD-B15*SN3GD
C
C COMPUTE (L+G+H) PRIMED
C
      BLGHP=HDP+GDP+BLDP+B3*CS3GD+B1*SN2GD+B2*COSGD
      BLGHP=DMOD(BLGHP,PI2)
      IF(BLGHP.LT.0.0D0)BLGHP=BLGHP+PI2
      EDPDL=B4*SN2GD-B5*COSGD+B6*CS3GD-F1D4*CN2*CN*GMP2*(2.0*(3.0*THETA2
     1-1.0)*(DADR2*CN2+DADR+1.0)*SINFD+3.0*(1.0-THETA2)*((-DADR2*CN2
     2-DADR+1.0)*SNF2GD+(DADR2*CN2+DADR+F1D3)*SN3FGD))
      DLTI=F1D2*THETA*GMP2*SINI*(EDP*CS3FGD+3.0*(EDP*CSF2GD+CS2GFD))
     1-(A21/CN2)*(B8*SINGD+B7*CS2GD-B9*SN3GD)
      SINDH=(1.0/COSI2)*(F1D2*(B12*CS3GD+B11*COSGD+B10*SN2GD-(F1D2*GMP2
     1*THETA*SINI*(6.0*(EDP*SINFD-BLDP+FDP)-(3.0*(SN2GFD+EDP*SNF2GD)+EDP
     2*SN3FGD)))))
C
C COMPUTE (L+G+H)
```

```
C
  164 BLGH=BLGHP+((1.0/(CN+1.0))*F1D4*EDP*GMP2*CN2*(3.0*(1.0-THETA2)*
     1(SN3FGD*(F1D3+DADR2*CN2+DADR)+SNF2GD*(1.0-(DADR2*CN2+DADR)))+2.0*
     2SINFD*(3.0*THETA2-1.0)*(DADR2*CN2+DADR+1.0)))+GMP2*F3D2*((-2.0*
     3THETA-1.0+5.0*THETA2)*(EDP*SINFD+FDP-BLDP))+(3.0+2.0*THETA-5.0*
     4THETA2)*(GMP2*F1D4*(EDP*SN3FGD+3.0*(SN2GFD+EDP*SNF2GD)))
      BLGH=DMOD(BLGH,PI2)
      IF(BLGH.LT.0.0D0)BLGH=BLGH+PI2
      DLTE=DLT1E+(F1D2*CN2*((3.0*(1.0/(CN2*CN2*CN2))*GM2*(1.0-THETA2)
     1*CS2GFD*(3.0*EDP*COSFD2+3.0*COSFD+EDP2*COSFD*COSFD2+EDP))-(GMP2
     2*(1.0-THETA2)*(3.0*CSF2GD+CS3FGD))+(3.0*THETA2-1.0)*GM2*(1.0/
     3(CN2*CN2*CN2))*(EDP*CN+(EDP/(1.0+CN))+3.0*EDP*COSFD2+3.0*COSFD+
     4EDP2*COSFD*COSFD2)))
      EDPDL2=EDPDL*EDPDL
      EDPDE2=(EDP+DLTE)*(EDP+DLTE)
C
C COMPUTE E (ECCENTRICITY)
C
      E=DSQRT(EDPDL2+EDPDE2)
      SINDH2=SINDH*SINDH
      SQUAR=(DLTI*COSI2*F1D2+SINI2)*(DLTI*COSI2*F1D2+SINI2)
      SQRI=DSQRT(SINDH2+SQUAR)
C
C COMPUTE BI (INCLINATION)
C
      BI=DARSIN(SQRI)
      BI=2.0*BI
      BI=DMOD(BI,PI2)
      IF(BI.LT.0.0D0)BI=BI+PI2
C
C CHECK FOR E (ECCENTRICITY)=0
C
      IF(E.NE.0.0) GO TO 168
      BL=0.0
C
C CHECK FOR BI (INCLINATION)=0
C
  145 IF(BI.NE.0.0) GO TO 169
      H=0.0
C
C COMPUTE G (ARGUMENT OF PERIGEE)
C
  146 G=BLGH-BL-H
      G=DMOD(G,PI2)
      IF(G.LT.0.0D0)G=G+PI2
C
C COMPUTE TRUE ANOMALY
C
      EA = DKEPLR(BL,E)
      ARG1 = DSIN(EA) * DSQRT(1.0-E**2)
      ARG2 = DCOS(EA) - E
      IF = DATAN0(ARG1,ARG2)
      OSCELE(1) = A*AE
      OSCELE(2) = E
      OSCELE(3) = BI
      OSCELE(4) = H
      OSCELE(5) = G
      OSCELE(6) = BL
    7 CONTINUE
      DPELE(1) = ADP*AE
      DPELE(2) = EDP
      DPELE(3) = BIDP
      DPELE(4) = HDP
      DPELE(5) = GDP
      DPELE(6) =  BLDP
      IF(IPERT.EQ.0)BL = DMOD(ANU*DELT,PI2)
      ORBEL(1) = EADP
      ORBEL(2) = GDP+FDP
      ORBEL(3) = GDP
      ORBEL(4) = EK*(ANU + BLDOT)
      ORBEL(5) = FDP
      R = A*AE*(1.0D0 - E*DCOS(EA))
      GO TO 45
C
C MODIFICATIONS FOR CRITICAL INCLINATION
C
  158 DLT1E=0.0
      BLGHP=0.0
      EDPDL=0.0
      DLTI=0.0
      SINDH=0.0
      ASSIGN 164 TO ID8
      GO TO 150
  168 SINLDP=DSIN(BLDP)
      COSLDP=DCOS(BLDP)
      SINHDP=DSIN(HDP)
      COSHDP=DCOS(HDP)
C
C COMPUTE L (MEAN ANOMALY)
C
       ARG1=EDPDL*COSLDP+(EDP+DLTE)*SINLDP
       ARG2=(EDP+DLTE)*COSLDP-(EDPDL*SINLDP)
       BL=DATAN2(ARG1,ARG2)
       BL=DMOD(BL,PI2)
       IF(BL.LT.0.0D0)BL=BL+PI2
       GO TO 145
C
C COMPUTE H (LONGITUDE OF ASCENDING NODE)
   169 ARG1=SINDH*COSHDP+SINHDP*(F1D2*DLTI*COSI2+SINI2)
       ARG2=COSHDP*(F1D2*DLTI*COSI2+SINI2)-(SINDH*SINHDP)
       H=DATAN2(ARG1,ARG2)
       H=DMOD(H,PI2)
       IF(H.LT.0.0D0)H=H+PI2
       GO TO 146
    45 CONTINUE
       RETURN
```

```
END
```

```
        SUBROUTINE CELEM (ORBEL,GMC,PV,VV)
C     ORIGINAL VERSION...1/22/71...CHARLES K. CAPPS
C     PURPOSE:
C                 THIS ROUTINE CONVERTS CLASSICAL OSCULATING ORBITAL ELEMENTS
            TO
C           CARTESIAN ELEMENTS.
C     CALLING SEQUENCE:
C           CALL CELEM(ORBEL,GMC,PV,VV)
C     INPUT THROUGH ARGUMENT LIST:
C                 ORBEL(1) = SEMI-MAJOR AXIS, A (OSCULATING ELEMENTS)
C                 ORBEL(2) = ECCENTRICITY, E
C                 ORBEL(3) = INCLINATION, I
C                 ORBEL(4) = LONGITUDE OF ASCENDING NODE, CAP OMEGA
C                 ORBEL(5) = ARGUMENT OF PERIFOCUS, OMEGA
C                 ORBEL(6) = MEAN ANOMALY, M
C                 GMC = GRAVITATIONAL CONSTANT
C     OUTPUT THROUGH ARGUMENT LIST:
C                 PV = CARTESIAN POSITION VECTOR
C                 VV = CARTESIAN VELOCITY VECTOR
C     METHOD:
C                 USES  MILES  STANDISH  ITERATIVE  SCHEME  FOR  SOLUTION  TO
            KEPLERS EQN.
C     REFERENCES:
C                 GTDS TASK SPEC FOR CELEM, C.E. VELEZ, 13 JANUARY 1971
C                 DODS SYSTEM DESCRIPTION, SUBROUTINE KEPLR1
C                 P. EXCOBAL- "METHODS OF ORBIT DETERMINATION"
C                 X-552-67-421,"COMPARISON  OF  ITERATIVE  TECHNIQUES  FOR  THE
            SOLUTION OF
C           KEPLERS EQUATION", I.COLE AND R.BORCHERS
C           PROGRAMMER:
C                 CHARLES K. CAPPS, CODE 553.2, GSFC
C
        IMPLICIT REAL*8(A-H,O-Z)
        DATA MAX /10/
        DIMENSION PV(3),VV(3),ORBEL(6)
        DATA TOL /+0.5D-16/
        ITER = 0
C       FIND IF THIS IS ELLIPTIC OR HYPERBOLIC ORBIT
        IF (ORBEL (1).LE.0.0D0.AND.ORBEL(2).GT.1.0D0) GO TO 50
C       ELLIPTIC ORBIT TAKES THIS ROUTE.
C       FIRST FIND ECCENTRIC ANOMALY VIA NEWTONS (MILES STANDISH VERSION)
        E1 = ORBEL(6)
  10    F = E1 - (ORBEL(2) * DSIN(E1)) - ORBEL (6)
        D = 1.0D0 - (ORBEL (2) * DCOS (E1 - 0.5D0 *F))
        E2 = E1 - (F / D)
        IF (DABS (E1-E2)-TOL)40,40,20
  20    ITER =ITER + 1
        E1 = E2
        IF(ITER - MAX) 10,10,30
C       SET UP ERROR CODE TO RETURN FROM SUBROUTINE
  30    NERR = 13
C       ECCENTRIC ANOMALY CONVERGED, NOW GET XO, YO, R
  40    COSE = DCOS(E2)
```

```
         SINE = DSIN (E2)
         TEMP = 1.0D0 - ORBEL(2) * ORBEL (2)
         XO = ORBEL(1) * (COSE - ORBEL(2))
         YO = ORBEL(1) * (DSQRT(TEMP)* SINE)
         R = ORBEL(1) * (1.0D0 - ORBEL (2) * COSE)
         XOD = (-DSQRT(GMC* ORBEL(1))* SINE)/R
         YOD = (DSQRT(GMC*ORBEL(1)*(TEMP))*COSE) /R
         GO TO 100
C        HYPERBOLIC ORBITS TAKE THIS ROUTE
      50 E1 = ORBEL(6) /2.0D0
      60 F = ORBEL(2) * DSINH(E1) - E1 - ORBEL(6)
         D = ORBEL(2) * DCOSH(E1 - 0.5D0 * F ) - 1.0D0
```

```
        E2=E1-(F/D)
        IF (DABS (E1-E2)-TOL)90,90,70
     70 ITER = ITER + 1
       E1 = E2
       IF (ITER - MAX) 60,60,80
C      SET UP ERROR CODE FOR NON-CONVERGENCE PRIOR TO EXIT.
    80 NERR = 14
C      ECCENTRIC ANOMALY COMPUTED, NOW GET XO,YO,R
    90 COSE = DCOSH (E2)
       SINE = DSINH(E2)
       TEMP = ORBEL(2) * ORBEL (2) - 1.0D0
       XO = ORBEL(1)*(COSE- ORBEL(2))
       YO = -ORBEL (1)*DSQRT (TEMP) * SINE
       R = ORBEL (1)*(1.0D0 - ORBEL(2) * COSE)
       XOD = (-DSQRT(-GMC*ORBEL(1))*SINE)/R
       YOD = (DSQRT(-GMC*ORBEL(1)*TEMP)*COSE)/R
   100 COSO = DCOS(ORBEL(5))
       SINO = DSIN (ORBEL(5))
       COSOM = DCOS (ORBEL(4))
       SINOM = DSIN (ORBEL(4))
       COSI = DCOS(ORBEL(3))
       SINI = DSIN (ORBEL(3))
       B11 = COSO * COSOM - SINO * SINOM * COSI
       B21 = COSO * SINOM + SINO * COSOM * COSI
       B31 = SINO * SINI
       B12 = -SINO * COSOM - COSO * SINOM * COSI
       B22 = -SINO * SINOM + COSO * COSOM * COSI
       B32 = COSO * SINI
C      NOW MULTIPLY 3 X 2 MATRIX BY 2 X 1 VECTORS FOR POSITION, VELOCITY.
       PV(1) = B11 * XO + B12 * YO
       PV(2) = B21 * XO + B22 * YO
       PV(3) = B31 * XO + B32 * YO
       VV(1) = B11*XOD + B12 * YOD
       VV(2) = B21 * XOD + B22 * YOD
       VV(3) = B31 * XOD + B32 * YOD
   999 RETURN
       END
      DOUBLE PRECISION FUNCTION DATAN0(ARG1,ARG2)
C                 VERSION OF 03/10/71
C
C                 FORTRAN IV FUNCTION SUBROUTINE FOR THE IBM-360
C
C                 PURPOSE
C                             COMPUTE A VALUE FOR THE ARCTAN BETWEEN 0 AND 2
                  PI WHERE THE
C                 TANGENT IS DEFINED BY THE TWO INPUT ARGUMENTS AS ARG1/ARG2
C
C                 CALLING SEQUENCE
C                             NONE
C                 INPUT
C                             ARG1 - FIST ARGUMENT OF THE ARC TANGENT
C                             ARG2 - SECOND ARGUMENT OF THE ARC TANGENT
C
```

```
C                 OUTPUT
C                             A DOUBLE PRECISION ARC TANGENT (+ VALUE BETWEEN
                  0 AND 2PI)
C
C                 METHOD
C
C                             USES  FORTRAN  MATH  SUBROUTINE  DATAN2  WHICH
                  RETURNS A VALUE
C                 BETWEEN -PI AND PI, GIVEN TWO ARGUMENTS
C
C                 REQUIRED SUBROUTINES
C                             1- FUNCTION SUBROUTINE DATAN2
C
C                 PROGRAMMER
C                             R. E. GILLIAN - COMPUTING AND SOFTWARE
C
C*********START PROGRAM********************************************
C
C                 COMPUTE ARCTAN BETWEEN -PI AND PI
C
      IMPLICIT REAL*8 (A-H,P-Z)
   50 DATAN0=DATAN2(ARG1,ARG2)
C
C                 IF ARCTAN IS NEGATIVE, ADD 2PI TO THE RESULT
C
  100 IF(DATAN0.GE.0) GO TO 999
      DATAN0 = DATAN0 + 6.283185307179586D0
      ARG = DATAN0
  999 RETURN
      END
      FUNCTION DKEPLR(M,E)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M,PI2/6.283185307179586D0/,TOL/0.5D-15/
C
C  SUBROUTINE TO SOLVE KEPLER'S EQUATION
C  KEPLER'S EQUATION RELATES GEOMETRY OR POSITION IN ORBIT PLANE TO TIME.
C
C  M - MEAN ANOMALY (0<M<2PI)
C  E - ECCENTRICITY
C  EA - ECCENTRIC ANOMALY
C
      EA=0
      IF(M)1,2,1
    1 EA=M + E*DSIN(M)
      DO 22 I=1,12
      OLDEA=EA
      FE=EA-E*DSIN(EA)-M
      EA=EA-FE/(1-E*DCOS(EA-0.5D0*FE))
C TEST FOR CONVERGENCE
      DELEA=DABS(EA-OLDEA)
      IF(DELEA.LE.TOL)GO TO 2
   22 CONTINUE
    2 EA=DMOD(EA,PI2)
```

```
      DKEPLR=EA
      RETURN
      END
      SUBROUTINE MA3331(/A/,/B/,/C/)
C
C     PURPOSE
C         TO COMPUTE THE PRODUCT OF A 3X3 MATRIX AND A 3X1 MATRIX
C
C     VERSION OF  JULY 23, 1971
C
C     METHOD
C         WRITE THE EXPLICIT CODE FOR THE MULTIPLICATION OF A 3X3 MATRIX AND
C     A 3X1 MATRIX AND RETURN THE RESULT IN THE 'C' MATRIX
C
C     CALLING SEQUENCE
C         CALL MAT31(A,B,C)
C         A     = INPUT 3X3 MATRIX
C         B     = INPUT 3X1 MATRIX
C         C     = OUTPUT 3X3 MATRIX
C
C     PROGRAMMER
C         N.R. BURTON COMPUTER SCIENCES CORPORATION
C
C     *********START PROGRAM*******************************************
      IMPLICIT       REAL*8(A-H,O-Z)
      DIMENSION  A(9),B(3),C(3)
      C(1)=A(1)*B(1)+A(4)*B(2)+A(7)*B(3)
      C(2)=A(2)*B(1)+A(5)*B(2)+A(8)*B(3)
      C(3)=A(3)*B(1)+A(6)*B(2)+A(9)*B(3)
      RETURN
      END
      SUBROUTINE XYZPLH(EQS,XSTA,YSTA,ZSTA,RLAT,RLON,AE,HE,IERR)
C
C                 FORTRAN IV SUBROUTINE FOR THE IBM-360, 3/20/74 VERSION
C                 PURPOSE
C                             TO CONVERT STATION COORDINATES FROM THE EARTH-
                  FIXED CARTESIAN
C                 COORDINATES  TO  GEODETIC  LATITUDE,  EAST  LONGITUDE,  AND
SPHEROID C              HEIGHT
C                 CALLING SEQUENCE
C                             CALL
                  XYZPLH(EQS,XSTA,YSTA,ZSTA,RLAT,RLON,HE,IERR,AE)
C                 INPUT
C                             EQS - ECCENTRICITY OF THE BODY SQUARED
C                             AE  - SEMI-MAJOR AXIS
C                             XSTA - EARTH-FIXED CARTESIAN COORDINATE X
C                             YSTA - EARTH-FIXED CARTESIAN COORDINATE Y
C                             ZSTA - EARTH-FIXED CARTESIAN COORDINATE Z
C                 OUTPUT
C                             RLAT - GEODETIC LATITUDE
C                             RLON - EAST LONGITUDE
C                             HE - SPHEROID HEIGHT
C                             IERR - ERROR FLAG
C                                0=HEIGHT CONVERGED
C                                1=HEIGHT DID NOT CONVERGE
C                                2=LONGITUDE IS UNDEFINED
C                 REQUIRED SUBPROGRAMS
C                             DATAN0
C                 PROGRAMMER
C                             R.E. GILLIAN, COMPUTING AND SOFTWARE
C*************************************************************
      IMPLICIT REAL*8(A-H,P-Z)
      IERR=0
      T=EQS*ZSTA
      XYSQ=XSTA**2+YSTA**2
      IF (DABS(ZSTA).GE.1.0D-15) GO TO 5
      HE = DSQRT(XYSQ) - AE
      RLAT = 0.0 D0
      GO TO 21
    5 DO 10 J = 1, 25
      ZT=ZSTA+T
      H1=DSQRT(XYSQ+ZT**2)
      SINPHI=ZT/H1
      ESQSP=EQS*SINPHI
      H2=AE/DSQRT(1.0D0-ESQSP*SINPHI)
      T1=H2*ESQSP
      IF(DABS((T1-T)/T1).LT..1D-14) GO TO 20
   10 T=T1
      IERR=1
      GO TO 30
   20 HE=H1-H2
      RLAT=DARSIN(SINPHI)
   21 IF(XSTA.EQ.0.0D0) GO TO 40
      GO TO 25
```

```
40 IF(YSTA.EQ.0.0D0) IERR=IERR+2
   IF(IERR.GT.0) GO TO 30
   IF(YSTA.LT.0.0D0) GO TO 50
   RLON=3.14159265358793/2.0D0
   GO TO 30
50 RLON=3.14159265358793*1.5D0
   GO TO 30
25 RLON=DATAN0(YSTA,XSTA)
30 CONTINUE
   RETURN
   End
```