CR-184242

Dynetics, Inc.

P.O. Drawer B
Huntsville, Alabama
35814-5050

TR-91-NASA-37850-012

FINAL REPORT

USERS MANUAL FOR THE IMA PROGRAM

APPENDIX C. PROFILE DESIGN PROGRAM LISTING

CONTRACT NAS8-37850

JANUARY 1991

PREPARED FOR:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, AL 35812

(NASA-CR-184242) USERS MANUAL FOR THE IMA PROGRAM. APPENDIX C: PROFILE DESIGN PROGRAM LISTING Final Report (Dynetics) 200 p
N92-13680
CSCL 09B
Unclas
G3/61 0046418

TR-91-NASA-37850-012

FINAL REPORT

USERS MANUAL FOR THE IMA PROGRAM

APPENDIX C. PROFILE DESIGN PROGRAM LISTING

CONTRACT NAS8-37850

JANUARY 1991

PREPARED FOR:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, AL 35812

APPENDIX C. PROFILE DESIGN PROGRAM LISTING

The source code for the Profile Design Program (PDP) is divided into several files. In a similar manner, the Fortran listings of the PDP's subroutines and function routines are organized into several groups in this appendix. Within each group, the Fortran listings are ordered alphabetically by routine name. Names and brief descriptions of each routine are listed below, in the same order as the Fortran listings.

IMA_PDP	"Main Program" for the Profile Design Program (PDP)

ALW	computes argument of latitude of the intersection of two orbits
ANG	converts angle to value between 0 and 2p
ARC	computes change in argument of latitude, given the change in time
ASCTIM	computes change in right ascension and time for an N-burn transfer
BURN	computes burn time and propellants remaining after a single burn, given DV
CB180C	computes propellants remaining after transfer between inclined co-elliptic orbits
CONREF	computes unit-vector triad aligned with perigee radius and angular momentum vector
CSTBRN	computes coast time, burn time, and propellants remaining for single leg
DJUL	computes Julian date, given the year, month, day, hour, minute, and second of GMT
DV2BRC	computes the DV's and transfer conic between coplanar circular orbits, given the orbit radii, transfer angle, and transfer time
ECITRAN	transforms ECI coordinates to ECID coordinates, or vice versa
FSTRAN	transforms ECF to ECS (EC Spherical) coordinates, or vice versa
GMT	computes GMT year, month, day, hour, minute, and second, given the Julian date
ORBECF	computes ECF position coordinates of a satellite, given the orbit I.D. and Julian date
TLNCH	determines the near-optimum launch GMT for an LI-type initial orbit
TRKWIN	defines the tracking windows, given a time interval and orbit I.D.

EQSAM	converts from a,M set of orbit elements to p,f set
GAUSS	computes v_1, g_1, v_2, g_2, given r_1, r_2, transfer angle between r_1 and r_2, and transfer time
GETOEM	computes mean orbital elements and Julian date reference for orbits in mission profile
GHANG	computes Greenwich hour angle, given the Julian date
IFTRAN	transforms ECID coordinates to ECF coordinates, or vice versa
INIT	initializes mission-dependent constants
KEPLE	computes eccentric anomaly and true anomaly, given eccentricity and mean anomaly
LITRAN	transforms launch-inertial (LI) coordinates to ECID coordinates, or vice versa
MOTRAN	transforms mean orbital elements to osculating orbital elements, or vice versa
ORBMOVE	computes secular rates of W, w, and M due to the J_2 and J_4 harmonics
OSCMN	computes short-period perturbations of equinoctial elements
RVAT	computes radii, arrival velocities, and departure velocities at multi-burn impulse points
SAMEQ	converts from p,f set of orbit elements to a,M set
TGAUSS	computes transfer time, p, and e of transfer conic, given r_1, r_2, Df, and f_1
TRAN	converts from a,M set of orbital elements to equinoctial elements, or vice versa
XMANOM	computes mean anomaly (M), given eccentricity (e) and true anomaly (f)

Appendix C (continued)

FINISH	extends the flight profile from the joint time being considered to the final time
ORBCID	computes the ECID position coordinates, given the orbit I.D. and Julian date
SIMPLX	Simplex algorithm of linear programming (used by segment worker SW01)
SIMQ	solves n x n set of simultaneous equations
SUN	computes unit vector from earth to sun in ECID coordinates, given the Julian date
SUNCOV	determines sunlight/shadow condition, given ECID position coordinates

SUNWIN	defines the sunlight windows, given a time interval and orbit I.D.
TANOM	computes the true anomaly, given the orbit I.D. and Julian date
TRKCOV	determines LOS conditions to tracking stations, given ECF position coordinates
TRNSFR	computes transfer conic between two known orbits, given the starting and ending times
VCROSS	vector cross-product utility
VDOT	vector dot produce utility
VMAG	vector magnitude utility
VUNIT	vector normalization utility
WEDGE	computes total angle (wedge) between two orbital planes
WINDINT	used by TLOPT to create window array information
XCOAST	projects the orbital state over a specified coast arc

MSG — controls reaction to messages and writes messages to screen

---

OITRAN — transforms osculating elements to ECID coordinates, or vice versa

---

OUTPUT — creates output files for access by the user-interface program

---

SW01 — segment worker for the Double Co-elliptic Rendezvous mission segment

---

SW02 — segment worker for the Payload Delivery mission segment

---

SW03 — segment worker for the Payload De-orbit/Spacecraft Recovery mission segment

---

TLOPT — Top-Level Optimizer subroutine

```
      subroutine    IMA_pdp(filenam)

      IMPLICIT REAL*8 (A-H,O-Z)

      logical exists,opened
      character*(*) filenam
      character*2 extension

      real*4 xGMTSEC(12),xORBELE(12,8),xPLMASS(12),xACCLIM(12),
     *       xTRKLAT(15),xTRKLONG(15),xTRKALT(15),xVEHMASS,xPCAP(6),
     *       xPMTHRST(6),xTHMISP(6),xTH0ISP(6),xFILL(6),xRESERV(6),
     *       xGMTMNS,xGEOM(15,14),xTFRAC(15,6),xACFLOW(15,6,2),
     *       xPNTDOCK(15,6,2),xSBTLIM(15),xTMAX,xWFACT(6),xALTMIN,xPBURN,
     *       xGMTLIS,xPLAT,xPLONG,xTSHIFTS


      LOGICAL LUNIT,LGMT(12),LORBELE(12,8),LFILL(6),
     1        LGEOM(15,14),LOBJ,LTMAX,LIREF,lref,LOWACC(15)

      CHARACTER*32 ORBNAM,PLNAME,TRKNAME,VEHNAME,
     1             TNAME,REFNAME
      CHARACTER*22 PNAME

      DIMENSION KTYPE(12),NGMT(12,5),GMTSEC(12),ORBELE(12,8),PLMASS(12),
     1          ACCLIM(12),TRKLAT(15),TRKLONG(15),TRKALT(15),PCAP(6),
     2          PMTHRST(6),THMISP(6),TH0ISP(6),FILL(6),RESERV(6),
     3          KTRAN(15),NTORB(15),GEOM(15,14),TFRAC(15,6),
     4          ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15),NPLD(15),
     5          IPLD(15,12),WFACT(6),NGMTLI(5),NTSHIFT(3),NGMTMN(5)

      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA08/XLPSUN,ECCSUN,SOBL,COBL,ASUN0,ASUND
      COMMON/IMA10/ECCS0,ECCSD,XLPS0,XLPSD
      COMMON/IMA11/NGMTMN,GMTMNS
      COMMON/IMA12/NORBIT,KTYPE,NGMT,GMTSEC,LGMT,
     &             ORBELE,LORBELE
      COMMON/IMA14/NPL,PLMASS,ACCLIM
      COMMON/IMA16/NTRK,TRKLAT,TRKLONG,TRKALT
      COMMON/IMA18/VEHMASS,NPROP,PCAP,PMTHRST,THMISP,TH0ISP
      COMMON/IMA20/NORB0,FILL,LFILL,RESERV
      COMMON/IMA22/NTRAN,KTRAN,NTORB,GEOM,LGEOM
      COMMON/IMA24/TFRAC,ACFLOW,PNTDOCK,SBTLIM,
     &             LOWACC
      COMMON/IMA26/NPLD,IPLD
      COMMON/IMA28/LOBJ,TMAX,LTMAX,WFACT
c      COMMON/IMA30/PCOAST,PBURN,LIREF,NGMTLI,GMTLIS,PLAT,PLONG
      COMMON/IMA30/ALTMIN
      COMMON/IMA32/LREF,NTSHIFT,TSHIFTS
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA38/ACCMAX(15),EMASS(15),MPSYS(15),THRNOM(15,6),
     &             FLWNOM(15,6),BCOEF(15,4)
```

```
      COMMON/IMA40/RPMIN
      COMMON/IMA42/MPRINT
      COMMON/IMA50/LUNIT
      COMMON/IMA52/TNAME(15),ORBNAM(12),PLNAME(12),TRKNAME(15)
      COMMON/IMA54/VEHNAME,PNAME(6),REFNAME
      COMMON/IMA60/ORBMIN(15),ORBMAX(15),TSTAY

      DATA   MPRINT  /1/
      DATA   DJUL0   /2433282.5/
      DATA   OBL0    /.40920621/
      DATA   SOBL0   /.39788120/
      DATA   COBL0   /.91743695/
      DATA   OBLD    /-.6218E-8/
      DATA   PEQD    /.6675E-6/
      DATA   GHA0    /1.74664770/
      DATA   GHADI   /.0172027918/
      DATA   GHADF   /6.3003881/

      DATA   ASUN0   /6.2482947/
      DATA   ASUND   /.01720197/
      DATA   ECCS0   /.016730108/
      DATA   ECCSD   /-.1148E-8/
      DATA   XLPS0   /4.9232341/
      DATA   XLPSD   /.8217E-6/

      DATA   XMU     /.3986032E15/
      DATA   XJ2     /.0010827/
      DATA   XJ3     /-.256E-5/
      DATA   XJ4     /-.158E-5/
      DATA   REQ     /6378160./
      DATA   RPL     /6356778./
      DATA   OMEGA   /.72921151E-4/

      DATA   PI      /3.14159265/
      DATA   TWOPI   /6.28318531/
      DATA   PIO2    /1.57079633/
      DATA   RPMIN   /6563360.0/

c      NAMELIST/SCREEN/NORBIT,KTYPE,ORBNAM,NGMT,GMTSEC,LGMT,ORBELE,
c     1               LORBELE,NPL,PLNAME,PLMASS,ACCLIM,LACCLIM,NTRK,
c     2               TRKNAME,TRKLAT,TRKLONG,TRKALT,VEHNAME,VEHMASS,
c     3               NPROP,PNAME,PCAP,PMTHRST,THMISP,TH0ISP,NORB0,
c     4               LFILL,RESERV,NTRAN,KTRAN,TNAME,NTORB,GEOM,LGEOM,
c     5               TFRAC,ACFLOW,PNTDOCK,SBTLIM,NPLD,IPLD,LOBJ,
c     6               TMAX,LTMAX,WFACT,PCOAST,PBURN,LIREF,NGMTLI,
c     7               GMTLIS,PLAT,PLONG,LREF,REFNAME,NTSHIFT,TSHIFTS,
c     8               LUNIT,FILL,LOWACC,NGMTMN,GMTMNS
c
c      OPEN(UNIT=10,FILE='SCREEN.DAT',STATUS='OLD')
      OPEN(UNIT=11,FILE='CHECKOUT.DAT',STATUS='unknown')
c      OPEN(UNIT=20,FILE='LOG.DAT',STATUS='unknown')
c      OPEN(UNIT=12,FILE='SUMMARY.DAT',STATUS='NEW')
c      OPEN(UNIT=17,FILE='SAMBO.DAT',STATUS='NEW')
```

```
      n=index(filenam,'.')
      extension=filenam(n+1:n+2)
        open(unit=12,file='internal.'//extension//'b',status='unknown')
        open(unit=13,file='internal.'//extension//'d',status='unknown')
        open(unit=14,file='internal.'//extension//'e',status='unknown')
        open(unit=15,file='internal.'//extension//'f',status='unknown')
        open(unit=16,file='internal.'//extension//'g',status='unknown')
        open(unit=17,file='internal.'//extension//'h',status='unknown')
        open(unit=18,file='internal.'//extension//'c',status='unknown')

      inquire(file=filenam,EXIST=exists,OPENED=opened,err=999)
      if(.not.exists)go to 999
c Open file for reading
      open(UNIT=1,File=filenam,ACCESS='sequential',
     & Form='unformatted',STATUS='unknown',err=999)
      read(1,err=999)
C Screen 2 Variables
     * LUNIT,
c Screen 3 Variables
     * NORBIT,
     * KTYPE,
     * NGMT,
     *xGMTSEC,
     * LGMT,
     *xORBELE,
     * LORBELE,
c Screen 4 Variables
     * NPL,
     *xPLMASS,
     *xACCLIM,
c Screen 5 variables
     * NTRK,
     *xTRKLAT,
     *xTRKLONG,
     *xTRKALT
      read(1,err=999)

c Screen 6 variables
     *xVEHMASS,
     * NPROP,
     *xPCAP,
     *xPMTHRST,
     *xTHMISP,
     *xTHOISP,
c Screen 7 Variables
     *xFILL,
     * LFILL,
     *xRESERV,
     * NGMTMN,
     *xGMTMNS,
c Screen 8 variables
     * NTRAN,
     * KTRAN,
```

```
c Screen 9 variables
     * NORB0,
     * NTORB,
c Screen 10 and 15 variables
     *xGEOM,
     * LGEOM
      read(1,err=999)

c Screen 16 variables
     *xTFRAC,
     *xACFLOW,
     *xPNTDOCK,
     *xSBTLIM,
     * LOWACC,
c Screen 17 Variables
     * NPLD,
     * IPLD,
c Screen 18 Variables
     * LOBJ,
     *xTMAX,
     * LTMAX,
     *xWFACT
      read(1,err=999)

c Screen 19 Variables
     *xALTMIN,
     *xPBURN,
     * LIREF,
     * NGMTLI,
     *xGMTLIS,
     *xPLAT,
     *xPLONG,
     * LREF,
     * NTSHIFT,
     *xTSHIFTS,
     * ORBNAM,
     * PLNAME,
     * TRKNAME,
     * PNAME,
     * TNAME,
     * VEHNAME,
     * REFNAME
C End of Read of data
      close(1)
c Convert to double precision (REAL*8)
      do ii=1,12
        GMTSEC(ii)=DBLE(xGMTSEC(ii))
        do jj=1,8
          ORBELE(ii,jj)=DBLE(xORBELE(ii,jj))
        end do
        PLMASS(ii)=DBLE(xPLMASS(ii))
        ACCLIM(ii)=DBLE(xACCLIM(ii))
      end do
      do ii=1,15
```

```
      TRKLAT(ii)=DBLE(xTRKLAT(ii))
      TRKLONG(ii)=DBLE(xTRKLONG(ii))
      TRKALT(ii)=DBLE(xTRKALT(ii))
      do jj=1,14
        GEOM(ii,jj)=DBLE(xGEOM(ii,jj))
      end do
      do jj=1,6
        TFRAC(ii,jj)=DBLE(xTFRAC(ii,jj))
        do kk=1,2
          ACFLOW(ii,jj,kk)=DBLE(xACFLOW(ii,jj,kk))
          PNTDOCK(ii,jj,kk)=DBLE(xPNTDOCK(ii,jj,kk))
        end do
      end do
      SBTLIM(ii)=DBLE(xSBTLIM(ii))
    end do
    do ii=1,6
      PCAP(ii)=DBLE(xPCAP(ii))
      PMTHRST(ii)=DBLE(xPMTHRST(ii))
      THMISP(ii)=DBLE(xTHMISP(ii))
      TH0ISP(ii)=DBLE(xTH0ISP(ii))
      FILL(ii)=DBLE(xFILL(ii))
      RESERV(ii)=DBLE(xRESERV(ii))
      WFACT(ii)=DBLE(xWFACT(ii))
    end do
    VEHMASS=DBLE(xVEHMASS)
    GMTMNS=DBLE(xGMTMNS)
    TMAX=DBLE(xTMAX)
c   PCOAST=DBLE(xPCOAST)
c   PBURN=DBLE(xPBURN)
c   GMTLIS=DBLE(xGMTLIS)
c   PLAT=DBLE(xPLAT)
c   PLONG=DBLE(xPLONG)
C (The lines commented out above represent variables no longer used.)
    ALTMIN=DBLE(xALTMIN)
    TSHIFTS=DBLE(xTSHIFTS)

    do ii=1,12
     if(lgmt(ii)) then
      if(ngmt(ii,1).le.60) then
       ngmt(ii,1)=ngmt(ii,1)+2000
      elseif(ngmt(ii,1).gt.60.and.ngmt(ii,1).le.99) then
       ngmt(ii,1)=ngmt(ii,1)+1900
      endif
     endif
    end do

    if(ngmtmn(1).le.60) then
     ngmtmn(1)=ngmtmn(1)+2000
    elseif(ngmtmn(1).gt.60.and.ngmtmn(1).le.99) then
     ngmtmn(1)=ngmtmn(1)+1900
    endif

c   if(liref)then
c    if(ngmtli(1).le.60) then
```

```
c        ngmtli(1)=ngmtli(1)+2000
c       elseif(ngmtli(1).gt.60.and.ngmtli(1).le.99) then
c        ngmtli(1)=ngmtli(1)+1900
c       endif
c      endif

      do ii=1,15
        SBTLIM(ii)=SBTLIM(ii)/86400.
      end do

c        READ(10,SCREEN)
C
C CALL INITIALIZATION ROUTINE
C
      DO I = 1,15
         ORBMIN(I)  = GEOM(I,1)
         ORBMAX(I)  = GEOM(I,2)
      END DO
      RPMIN=ALTMIN+REQ

      CALL INIT
C
      CALL TLOPT
C

      return


999   close(1)
      print*,'File Not Found!!'
      return

      END
```

```
      FUNCTION    ALW(SI,CI,RADIF)

      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES THE ARGUMENT OF LATITUDE OF THE ASCENDING NODE
C     OF ORBIT NO.1  W.R.T. ORBIT NO.2, EXPRESSED IN THE PLANE
C     OF ORBIT NO.1

C     SI(J)     = SINES OF INCLINATIONS OF ORBIT NOS .1 AND 2
C     CI(J)     = COSINES OF INCLINATIONS OF ORBIT NOS .1 AND 2

C     RADIF   = DIFFERENCE IN RIGHT ASCENSION OF ORBIT NO.2 AND
C               ORBIT NO.1 (NO.2 ANGLE - NO.1 ANGLE)
C     ---------------------------------------------
      DIMENSION SI(2), CI(2)
      DATA KERR/0/

      IF(ABS(RADIF).LT.1.E-6)THEN
        IF(ABS(SI(1)-SI(2)).LT.1.E-6.AND.KERR.EQ.0)THEN
          CALL MSG('WARNING: ORBIT INTERSECTION MAY CHANGE RAPIDLY',1)
          KERR=1
        END IF
      END IF

      CRADIF=COS(RADIF)
      SRADIF=SIN(RADIF)

      W1=-SI(2)*CI(1)*CRADIF +SI(1)*CI(2)
      W2=-SI(2)*CI(1)*SRADIF
      W3=-SI(1)*SI(2)*SRADIF

      ALW=ATAN2(W2*CI(1)+W3*SI(1), W1)

      RETURN
      END


      FUNCTION   ANG(X)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/IMA06/PI,TWOPI,PIO2
      ANG=X-TWOPI*FLOAT(INT(X/TWOPI))
      IF(ANG) 1,2,2
    1 ANG=ANG+TWOPI
    2 RETURN
      END


      SUBROUTINE       ARC(AL0,AP0,ECC,XMD,APD,DT,DAL,MODE)
      IMPLICIT REAL*8 (A-H,O-Z)

C     MODE >0    COMPUTES CHANGE IN ARGUMENT OF LATITUDE (DAL), GIVEN
C                THE CHANGE IN TIME (DT).
```

```
C     MODE <0    COMPUTES CHANGE IN TIME (DT), GIVEN THE CHANGE IN
C                ARGUMENT OF LATITUDE (DAL).

C     ***************************************************************
C     NOTE: THE DT AND DAL VALUES CAN BE EITHER POSITIVE OR NEGATIVE
C     ***************************************************************

C     AL0, AP0 = STARTING VALUES OF ARGUMENTS OF LATITUDE AND PERIGEE
C     ECC      = ORBITAL ECCENTRICITY
C     XMD,APD  = RATES OF CHANGE OF MEAN ANOMALY AND ARGUMENT OF PERIGEE
C     -----------------------------------------------------------------
      COMMON/IMA06/PI,TWOPI,PIO2

      DATA TOL/1.E-7/, MAXIT/30/, KERR/0/

      IF(MODE.GT.0)THEN
C     COMPUTE DAL, GIVEN DT
C     ---------------------
        F0=AL0-AP0
        XMF=XMANOM(ECC,F0) +XMD*DT
        CALL KEPLE(ECC,XMF,DUM,FF)
        DAL= FF +AP0 +APD*DT -AL0

        NPER=DT*(XMD+APD)/TWOPI
        DALLO=NPER*TWOPI
        IF(DT.LT.0.)DALLO=DALLO-TWOPI
        DALHI=DALLO+TWOPI

    1   IF(DAL.GT.DALHI)THEN
          DAL=DAL-TWOPI
          GO TO 1
        END IF

    2   IF(DAL.LT.DALLO)THEN
          DAL=DAL+TWOPI
          GO TO 2
        END IF

      ELSE
C     COMPUTE DT, GIVEN DAL
C     ---------------------
        ITER=0
        ALD=XMD+APD
        F0=AL0-AP0
        XM0=XMANOM(ECC,F0)
        FFCON=AL0+DAL-AP0
        DT=DAL/ALD

    3   ITER=ITER+1
        XMF=XM0+XMD*DT
        FF=FFCON-APD*DT
        XMFERR=XMANOM(ECC,FF) -XMF
```

```
    4   IF(XMFERR.GT.PI)THEN
          XMFERR=XMFERR-TWOPI
          GO TO 4
        END IF

    5   IF(XMFERR.LT.-PI)THEN
          XMFERR=XMFERR+TWOPI
          GO TO 5
        END IF

        IF(ABS(XMFERR).LE.TOL)RETURN

        IF(ITER.EQ.MAXIT)THEN
          IF(KERR.EQ.0)THEN
            CALL MSG('***WARNING: ARC TIME NOT IN TOLERENCE',1)
            WRITE(20,*) 'MEAN-ANOMALY ERROR (RADIANS) = ', XMFERR
            PRINT *, 'MEAN-ANOMALY ERROR (RADIANS) = ', XMFERR
            KERR=1
          END IF
          RETURN
        END IF

        DT=DT +XMFERR/ALD
        GO TO 3
      END IF

      END


      SUBROUTINE     ASCTIM(NB,SI,CI,R,TI,AS,TT)
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES CHANGE IS RIGHT ASCENSION AND TOTAL TIME FOR AN
C     N-BURN TRANSFER.

C     INPUTS:---------------------------------------------------
C     NB         NUMBER OF IMPULSES IN THE TRANSFER
C     SI(J)      SINES OF INCLINATIONS OF STARTING AND ENDING ORBITS
C     CI(J)      COSINES OF INCLINATIONS OF STARTING AND ENDING ORBITS
C     R(I)       RADIUS AT POINT OF EACH IMPULSE IN THE TRANSFER

C     OUTPUTS:--------------------------------------------------
C     TI(I)      TIME BETWEEN IMPULSE I AND IMPULSE I+1
C     AS         CHANGE IN RIGHT ASCENSION DURING TRANSFER
C     TT         TOTAL TIME OF TRANSFER (*** DAYS ***)
C     ----------------------------------------------------------

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2

      DIMENSION R(12), TI(11), SI(2), CI(2)

      AS=0.
```

```
      TT=0.
      DSI=(SI(2)-SI(1))/NB
      DCI=(CI(2)-CI(1))/NB
      NBM1=NB-1

      DO 10 I=1,NBM1
      RPR=R(I) +R(I+1)
      SLR=2.*R(I)*R(I+1)/RPR
      ECC=ABS(R(I)-R(I+1))/RPR

      IF(ECC.GT..99)THEN
        CALL MSG('HALT: ECCENTRICITY .GT. .99 IN IMA_ASCTIM',3)
        STOP
      END IF

C     APPROXIMATIONS FOR SINE, COSINE OF INTERMEDIATE CONICS,
C     ASSUMING AN EQUAL SPLIT OF A REASONABLY SMALL PLANE CHANGE
C     ----------------------------------------------------------
      SINC=SI(1) +DSI*I
      CINC=CI(1) +DCI*I

      CALL ORBMOVE(SLR,ECC,SINC,CINC,RAD,APD,XMD)

      TI(I)=PI/(APD+XMD)
      TT=TT +TI(I)
   10 AS=AS +RAD*TI(I)

      RETURN
      END


      SUBROUTINE   BURN   (ITRAN,DV,P1,P2,BT)
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES TOTAL BURN TIME AND PROPELLANTS REMAINING AFTER BURN
C     FOR EACH PROPULSION SUBSYSTEM.  CAN HANDLE BURNS THAT START ON
C     AN ACCELERATION LIMIT OR THAT HIT THE LIMIT DURING THE BURN.
C     THE THRUST OF THE PRIMARY SUBSYSTEM (MPSYS) IS THROTTLED, IF
C     NECESSARY,TO MAINTAIN ACCELERATION AT THE LIMIT.  THRUSTS OF
C     THE OTHER SUBSYSTEMS REMAIN CONSTANT.
C     -------------------------------------------------------------
C     INPUTS:
C     ITRAN      I.D. NUMBER IF THE ASSOCIATED TRANSFER
C     DV         TOTAL DELTA-VELOCITY TO BE GAINED BY THE BURN
C     P1(M)      PROPELLANT LOAD FOR SUBSYSTEM M AT BEGINNING OF BURN

C     OUTPUTS:
C     P2(M)      PROPELLANT LOAD FOR SUBSYSTEM M AT END OF BURN
C     BT         TOTAL BURN TIME (*** OUTPUT IN UNITS OF DAYS ***)
C     **************************************************************
C     NOTE: INTERNAL CALCULATIONS USE BURN TIMES IN UNITS OF SECONDS.
C     THE TOTAL BURN TIME IS CONVERTED TO UNITS OF DAYS BEFORE THE
```

```
C     RETURN TO THE CALLING PROGRAM.
C     **************************************************************
C     --------------------------------------------------------------
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA38/ACCMAX(15),EMASS(15),MPSYS(15),THRNOM(15,6),
     &             FLWNOM(15,6),BCOEF(15,4)

      DIMENSION  P1(6), P2(6)

      DATA FTOL/.001/, KERR1/0/, MAXIT/10/

      DVP=0.
      BTP=0.

      THR=0.
      FLW=0.
      XM0=EMASS(ITRAN)

      DO 10 M=1,NPROP
      XM0=XM0+P1(M)
      THR=THR+THRNOM(ITRAN,M)
      FLW=FLW+FLWNOM(ITRAN,M)
   10 P2(M)=P1(M)

      IF(THR/XM0.GE.ACCMAX(ITRAN)) GO TO 50
C     BURN BEGINS WITH ACCELERATION LESS THAN THE LIMIT
      C=THR/FLW
      XMF=XM0*EXP(-DV/C)

      IF(THR/XMF.GT.ACCMAX(ITRAN)) GO TO 30
C     BURN DOES NOT REQUIRE THROTTLING
      BT=(XM0-XMF)/FLW
      DO 20 M=1,NPROP
   20 P2(M)=P2(M)-FLWNOM(ITRAN,M)*BT
      BT=BT/86400.
      RETURN

C     ACCELERATION LIMIT IS REACHED DURING THE BURN
   30 XMP=THR/ACCMAX(ITRAN)
      DVP=C*DLOG(XM0/XMP)
      BTP=(XM0-XMP)/FLW
      XM0=XMP
      DO 40 M=1,NPROP
   40 P2(M)=P2(M)-FLWNOM(ITRAN,M)*BTP

C     THROTTLE PRIMARY SUBSYSTEM TO MAINTAIN ACCELERATION LIMIT
   50 DVL=DV-DVP
      BTL=DVL/ACCMAX(ITRAN)
      X=BCOEF(ITRAN,3)+BCOEF(ITRAN,4)*XM0
      D=-BCOEF(ITRAN,1)*DLOG(X)-BCOEF(ITRAN,2)*X +BTL

      ITER=0

   60 IF(X.LE.0.)THEN
```

```
      CALL MSG('***HALT: THROTTLING OF PRIMARY SUBSYSTEM CANNOT MAINTA
     &IN ACCELERATION LIMIT',3)
      STOP
      END IF

      F=BCOEF(ITRAN,1)*DLOG(X)  +BCOEF(ITRAN,2)*X +D

      IF(ABS(F).LE.FTOL.OR.ITER.EQ.MAXIT)THEN

        IF(ITER.EQ.MAXIT)THEN
          IF(KERR1.EQ.0)THEN
            CALL MSG('***WARNING: THROTTLING COMPUTATIONS MAY BE FAULTY'
     &,1)
            WRITE(20,*) 'EQUIVALENT BURN-TIME ERROR (SEC) = ', F
            PRINT *, 'EQUIVALENT BURN-TIME ERROR (SEC) = ', F
            KERR1=1
          END IF
        END IF

        XMF=(X-BCOEF(ITRAN,3))/BCOEF(ITRAN,4)
        PSUM=0.
        DO 70 M=1,NPROP
        IF(M.EQ.MPSYS(ITRAN)) GO TO 70
        P2(M)=P2(M)-FLWNOM(ITRAN,M)*BTL
        PSUM=PSUM+P2(M)
70      CONTINUE
        P2(MPSYS(ITRAN))=XMF-EMASS(ITRAN)-PSUM
        BT=(BTP+BTL)/86400.
        RETURN
      END IF

      ITER=ITER+1

      DFDX=BCOEF(ITRAN,1)/X +BCOEF(ITRAN,2)
      X=X-F/DFDX
      GO TO 60

      END


      SUBROUTINE       CB180C(ITRAN,CTI1,VA,VD,TBI,WEDGE,P1,P2,BT)
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LOWACC

C     COMPUTES DELTA-V'S (INTERNAL USE), BURN TIMES, AND PROPELLANTS
C     USED FOR A TRANSFER BETWEEN CIRCULAR ORBITS, OR ORBITS WITH
C     ALIGNED APSIDES.  THE TRANSFER CONSISTS OF NBIT(ITRAN) BURNS.
C     CB180C ESTIMATES THE NUMBER OF BURNS, NBITS(ITRAN), REQUIRED TO
C     SATISFY THE SINGLE-BURN-TIME LIMIT (SBTLIM).

C     INPUTS:
C     ITRAN        I.D. NUMBER OF TRANSFER
C     CTI1         COAST TIME TO THE FIRST IMPULSE POINT
```

```
C     VA(I)        ARRIVAL VELOCITY MAGNITUDES AT THE IMPULSE POINTS
C     VD(I)        DEPARTURE VELOCITY MAGNITUDES AT THE IMPULSE POINTS
C     TBI(I)       TIMES BETWEEN IMPULSES
C     WEDGE        WEDGE ANGLE (PLANE CHANGE) TO BE ACCOMPLISHED
C     P1(M)        BEGINNING PROPELLANT MASSES FOR PROPULSION SUBSYSTEMS

C     OUTPUTS:
C     P2(M)        ENDING PROPELLANT MASSES FOR PROPULSION SUBSYSTEMS
C     BT           BURN TIME AT THE FINAL IMPULSE POINT

C     ****************************************************
C     ALL BURN TIMES ARE CENTERED ON THE TIME OF THE IMPULSE.
C     EQUAL PLANE CHANGE IS ACCOMPLISHED BY ALL BURNS.
C     ALL TIMES ARE EXPRESSED IN DAYS.
C     ****************************************************
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA24/TFRAC(15,6),ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15)
     &              ,LOWACC(15)
      COMMON/IMA42/MPRINT
      COMMON/IMA58/NBIT(15),NBITS(15)

      DIMENSION VA(12), VD(12), TBI(11)
      DIMENSION P1(6), P2(6), PI(6), PC(6)
C     ----------------------------------------------------------------

      NB=NBIT(ITRAN)
      BTMAX=0.

      IF(WEDGE.GE.1.E-6) CWPI=COS(WEDGE/NB)

      DO 1 M=1,NPROP
    1 PI(M)=P1(M)
      I=1
      CTI=CTI1

    2 IF(WEDGE.LT.1.E-6)THEN
        DV=ABS(VD(I)-VA(I))
      ELSE
        DV=SQRT(VA(I)**2 +VD(I)**2 -2.*VA(I)*VD(I)*CWPI)
      END IF

C     ---------------------------------------------
      CALL CSTBRN(ITRAN,I,CTI,0.d0,DV,PI,PC,P2,CT,BT)
C     ---------------------------------------------

C     ********* CHECKOUT ONLY *************************
      IF(MPRINT.NE.0)THEN
        WRITE(11,701)CT,DV,BT,(PC(M),M=1,3),(P2(M),M=1,3)
  701   FORMAT(1X,9E13.5/)
      END IF
C     ***********************************************

      IF(BT.GT.BTMAX) BTMAX=BT
```

```
      IF(I.LT.NB)THEN
        DO 3 M=1,NPROP
3       PI(M)=P2(M)

        CTI=TBI(I)-.5*BT
        I=I+1
        GO TO 2
      END IF

C     ESTIMATE THE SMALLEST EVEN NUMBER OF BURNS REQUIRED
C     TO SATISFY THE SINGLE-BURN-TIME LIMIT
C     --------------------------------------------------
      NBITS(ITRAN)=BTMAX*NB/SBTLIM(ITRAN) +1
      IF(MOD(NBITS(ITRAN),2).NE.0) NBITS(ITRAN)=NBITS(ITRAN)+1


      RETURN
      END



      SUBROUTINE   CONREF   (A,UP,UN,UH)

      IMPLICIT REAL*8 (A-H,O-Z)
C     -------------------------------------------------------------
C     COMPUTES A UNIT-VECTOR TRIAD: UP(I) ALONG RADIUS TO PERIGEE,
C     UH(I) ALONG ANGULAR MOMENTUM VECTOR, AND UN(I) OBTAINED BY
C     CROSSING UH INTO UP.

C     A(1)= ARGUMENT OF PERIGEE
C     A(2)= INCLINATION
C     A(3)= RIGHT ASCENSION
C     ----------------------------
      DIMENSION A(3), UP(3), UN(3), UH(3)

      SA=SIN(A(1))
      CA=COS(A(1))
      SI=SIN(A(2))
      CI=COS(A(2))
      SR=SIN(A(3))
      CR=COS(A(3))
      UP(1)=CA*CR-SA*CI*SR
      UP(2)=CA*SR+SA*CI*CR
      UP(3)=SA*SI
      UH(1)=SI*SR
      UH(2)=-SI*CR
      UH(3)=CI
      CALL VCROSS (UH,UP,UN)
      RETURN

      END
```

```
      SUBROUTINE    CSTBRN    (ITRAN,IC,CTI,BPF,DV,P1,PC,P2,CT,BT)
      IMPLICIT REAL*8 (A-H,O-Z)

      LOGICAL*4 LOWACC

C     COMPUTES PROPELLANTS REMAINING AFTER COAST/BURN LEGS, AS WELL
C     AS THE COAST AND BURN TIMES, FROM THE INPUT DELTAV AND PROPELLANTS

C     INPUTS:----------------------------------------------
C     ITRAN    I.D. NUMBER OF TRANSFER
C     IC       COAST-ARC I.D.
C     CTI      COAST TIME TO THE IMPULSE POINT (*** DAYS ***)

C     BPF      BURN PLACEMENT FACTOR (0.=CENTERED ON IMPULSE POINT,
C              +1.=BEGINS AT IMPULSE POINT, -1.=ENDS AT IMPULSE POINT)

C     DV       DELTA VELOCITY TO BE GAINED BY BURN
C     P1(M)    REMAINING PROPELLANT FOR SUBSYSTEM M AT START OF LEG

C     OUTPUTS:---------------------------------------------
C     PC(M)    REMAINING PROPELLANT FOR SUBSYSTEM M AT END OF COAST
C     P2(M)    REMAINING PROPELLANT FOR SUBSYSTEM M AT END OF LEG
C     CT       COAST TIME (IE., CTI-.5*(1.-BPF)*BT) (*** DAYS ***)
C     BT       TOTAL BURN TIME (*** DAYS ***)
C     -----------------------------------------------------

      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA24/TFRAC(15,6),ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15)
     &              ,LOWACC(15)
C     *****************************************************************
C     NOTE: PROPELLANT FLOW RATES AND SPECIFIC IMPULSES ARE BASED ON
C     THE TIME UNIT OF SECONDS.  THE INPUT COAST TIME IS IN UNITS OF
C     DAYS, AND THE OUTPUT COAST AND BURN TIMES MUST BE IN UNITS OF DAYS
C     *****************************************************************

      DIMENSION P1(6), PC(6), P2(6)

      DATA MAXIT/10/, KERR1/0/

      BT=0.
      BTS=1.E20
      CT=CTI
      ITER=0
C     *****************************************************************
C     NOTE: FOR TRANSFERS CONSISTING OF MORE THAN 2 IMPULSES, SUCH AS
C     CAN HAPPEN WHEN ADDITIONAL LEGS ARE ADDED TO PREVENT VIOLATION OF
C     THE SINGLE-BURN-TIME LIMIT, THE PROPELLANT REQUIRED FOR POINTING
C     AND DOCKING IS USED DURING THE FIRST TWO COAST ARCS.
C     *****************************************************************
    1 DO 2 M=1,NPROP
      PDCK=0.
      IF(IC.LE.2)PDCK=PNTDOCK(ITRAN,M,IC)
    2 PC(M)=P1(M) -86400.* ACFLOW(ITRAN,M,1)*CT -PDCK
```

```
      CALL BURN(ITRAN,DV,PC,P2,BT)

      CT=CTI-.5*(1.-BPF)*BT

      ITER=ITER+1
      IF(ITER.EQ.MAXIT)THEN

        IF(KERR1.EQ.0)THEN
          CALL MSG('*** WARNING: BURN TIMES MAY NOT BE PRECISELY COMPUTE
     &D',1)
          BTERR=86400.*(BT-BTS)
          WRITE(20,*) 'BURN TIME ERROR (SEC) = ', BTERR
          PRINT *, 'BURN TIME ERROR (SEC) = ', BTERR
          KERR1=1
        END IF

        RETURN
      END IF

      IF(ABS(BT-BTS).GT.1.E-8)THEN
        BTS=BT
        GO TO 1
      END IF

      RETURN
      END
```

```
      FUNCTION   DJUL(IYM,S)

      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IYM(5), NDCUM(12)
C     ------------------------------------------------------------
C     PROVIDES THE JULIAN DATE BASED ON THE IYM ARRAY (YEAR, MONTH,
C     DAY, HOUR, MINUTE) AND S (SECOND) OF GMT.  THE REFERENCE EPOCH
C     IS 0000 HRS GMT, JANUARY 0, 1950 (J.D.=2433281.5).  THE INPUT
C     GMT MUST BE NO EARLIER THAN 0000 HRS GMT, JANUARY 1,1950, AND
C     NO LATER THAN THE END OF THE YEAR 2099.

      DATA EPOCH/2433281.5/
      DATA NDCUM/0,31,59,90,120,151,181,212,243,273,304,334/
C     ------------------------------------------------------------

      IF(IYM(1).LT.1950.OR.IYM(1).GT.2099) THEN
        CALL MSG('*** HALT: GMT OUTSIDE THE 1950-2099 INTERVAL',3)
        STOP
      END IF

      IF(IYM(2).LT.1.OR.IYM(2).GT.12)THEN
        CALL MSG('***HALT: INVALID GMT MONTH',3)
      END IF

      NYEAR=IYM(1) -1950
```

```
      NDLEAP=(NYEAR+1)/4

      IF(MOD(NYEAR+2,4).EQ.0) THEN
        IF(IYM(2).GT.2) NDLEAP=NDLEAP+1
      ELSE
        IF(IYM(2).EQ.2.AND.IYM(3).GT.28) THEN
          CALL MSG('*** HALT: FEB 29 SPECIFIED FOR NON-LEAP YEAR',3)
          STOP
        END IF
      END IF

      DJUL=EPOCH +NYEAR*365. +NDCUM(IYM(2)) +IYM(3) +NDLEAP
     &     +IYM(4)/24. +IYM(5)/1440. +S/86400.

      RETURN
      END


      SUBROUTINE      DV2BRC(R1,R2,ANG,TME,DV1,DV2,F1,SLR,ECC)
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES THE DELTAV'S (DV1,DV2) OF A TWO-IMPULSE PLANAR TRANSFER
C     FROM A CIRCULAR ORBIT WITH RADIUS R1 TO A CIRCULAR ORBIT WITH
C     RADIUS R2, COVERING A PRESCRIBED CENTRAL ANGLE (ANG) IN A
C     PRESCRIBED TIME (TME).  SUBROUTINE GAUSS IS CALLED TO OBTAIN
C     THE STARTING AND ENDING VELOCITY COMPONENTS AND THE PARAMETERS
C     OF THE REQUIRED TRANSFER CONIC.
C     ***********************************************************
C     NOTE: DELTAV'S ARE GIVEN IN METERS/SEC, BUT TME IS GIVEN IN
C     UNITS OF DAYS.
C     ***********************************************************

C     VTC(I) = CIRCUMFERENTIAL VELOCITY COMPONENTS ON THE TRANSFER
C              CONIC AT R1(I=1) AND R2(I=2)

C     VTR(I) = RADIAL VELOCITY COMPONENTS ON THE TRANSFER CONIC
C              AT R1(I=1) AND R2(I=2)

C     F1     = TRUE ANOMALY OF R1 ON THE TRANSFER CONIC

C     SLR,ECC= PARAMETERS OF THE TRANSFER CONIC
C     ---------------------------------------------------------------

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL

      DIMENSION VTC(2), VTR(2)


      CALL GAUSS(R1,R2,ANG,TME,VTC,VTR,F1,SLR,ECC,KGERR)

      IF(KGERR.EQ.-1)THEN
        CALL MSG('***HALT: ROUTINE DV2BRC REQUIRES PERIGEE THAT IS TOO L
     &OW',3)
```

```
        STOP
      END IF

      IF(KGERR.NE.0)THEN
        CALL MSG('***HALT: INVALID CALCULATIONS IN ROUTINE GAUSS, CALLED
     & BY ROUTINE DV2BRC',3)
        STOP
      END IF


      VC1=SQRT(XMU/R1)
      VC2=SQRT(XMU/R2)

      DV1=SQRT((VTC(1)-VC1)**2 +VTR(1)**2)
      DV2=SQRT((VTC(2)-VC2)**2 +VTR(2)**2)

      RETURN
      END
```

SUBROUTINE ECITRAN (X0, X, DJUL, MODE)

```
      IMPLICIT REAL*8 (A-H,O-Z)
C     ---------------------------------------------------------------
C     TRANSFORMS ECI COORDINATES (X0) TO ECID COORDINATES (X)
C     OR VICE VERSA, DEPENDING ON WHETHER MODE>0 OR MODE<0.

C     X0(1)...X0(3), X(1)...X(3) = X,Y,Z VELOCITY COMPONENTS
C     X0(4)...X0(6), X(4)...X(6) = X,Y,Z POSITION COORDINATES

C     DJUL = ECID REFERENCE JULIAN DATE
C     DJUL0= ECI REFERENCE JULIAN DATE (NOMINALLY, DJUL0=2433282.5,
C            CORRESPONDING TO 0.0 HR GMT, 1 JANUARY 1950)

C     SOBL0,COBL0 = SINE, COSINE OF OBLIQUITY ON DJUL0 (FIXED CONSTANTS)
C     SOBL,COBL   = SINE, COSINE OF OBLIQUITY ON DJUL (PRECOMPUTED)
C     PEQ =  PRECESSION OF THE EQUINOX BETWEEN DJUL0 AND DJUL
C     ---------------------------------------------------------------
      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
      COMMON/IMA08/XLPSUN,ECCSUN,SOBL,COBL,ASUN0,ASUND
C     ---------------------------------------------------------------
      DIMENSION X0(6), X(6), A(3,3)

      DAYS= DJUL -DJUL0
      PEQ= PEQD*DAYS

      SPEQ=SIN(PEQ)
      CPEQ=COS(PEQ)

      A(1,1)= CPEQ
      A(1,2)=-SPEQ*COBL0
      A(1,3)=-SPEQ*SOBL0
      A(2,1)= SPEQ*COBL
```

```
      A(2,2)= CPEQ*COBL*COBL0 +SOBL*SOBL0
      A(2,3)= CPEQ*COBL*SOBL0 -SOBL*COBL0
      A(3,1)= SPEQ*SOBL
      A(3,2)= CPEQ*SOBL*COBL0 -COBL*SOBL0
      A(3,3)= CPEQ*SOBL*SOBL0 +COBL*COBL0

      IF(MODE.GT.0) THEN
        DO 10 I=1,3
        X(I)   =A(I,1)*X0(1)  +A(I,2)*X0(2)  +A(I,3)*X0(3)
   10   X(I+3)=A(I,1)*X0(4)  +A(I,2)*X0(5)  +A(I,3)*X0(6)

      ELSE
        DO 20 I=1,3
        X0(I)   =A(1,I)*X(1)  +A(2,I)*X(2)  +A(3,I)*X(3)
   20   X0(I+3)=A(1,I)*X(4)  +A(2,I)*X(5)  +A(3,I)*X(6)

      END IF

      RETURN
      END
```

```
      SUBROUTINE  FSTRAN  (XF,  XS,  MODE)

      IMPLICIT REAL*8 (A-H,O-Z)
C     -------------------------------------------------------------
C     TRANSFORMS ECF COORDINATES (XF) TO EC SPHERICAL (ECS)
C     COORDINATES (XS) OR VICE VERSA, DEPENDING ON WHETHER
C     MODE>0 OR MODE<0.

C     XF(1)...XF(3)= X,Y,Z VELOCITY COMPONENTS
C     XF(4)...XF(6)= X,Y,Z POSITION COORDINATES

C     XS(1)  = VELOCITY MAGNITUDE
C     XS(2)  = FLIGHT PATH ANGLE (ABOVE HORIZONTAL PLANE)
C     XS(3)  = HEADING ANGLE W.R.T. TRUE NORTH
C     XS(4)  = RADIUS MAGNITUDE
C     XS(5)  = GEOCENTRIC LATITUDE
C     XS(6)  = EAST LONGITUDE
C     -------------------------------------------------------------
      DIMENSION XF(6),XS(6)

      IF(MODE.GT.0) THEN
        XS(1)=VMAG(XF(1))
        XS(4)=VMAG(XF(4))
        XS(2)=ASIN(VDOT(XF(1),XF(4))/(XS(1)*XS(4)))
        RPEQ2=XF(4)**2 +XF(5)**2
        VERPEQ=XF(2)*XF(4)  -XF(1)*XF(5)
        VNRPEQ=(XF(3)*RPEQ2  -XF(6)*(XF(1)*XF(4)  +XF(2)*XF(5)))/XS(4)
        XS(3)=ATAN2(VERPEQ,VNRPEQ)
        XS(5)=ASIN(XF(6)/XS(4))
        XS(6)=ATAN2(XF(5),XF(4))
      ELSE
```

```
      VH=XS(1)*COS(XS(2))
      VN=VH*COS(XS(3))
      VE=VH*SIN(XS(3))
      VV=XS(1)*SIN(XS(2))
      S5=SIN(XS(5))
      C5=COS(XS(5))
      S6=SIN(XS(6))
      C6=COS(XS(6))
      C5C6=C5*C6
      C5S6=C5*S6
      XF(1)=VV*C5C6   -VN*S5*C6 -VE*S6
      XF(2)=VV*C5S6   -VN*S5*S6 +VE*C6
      XF(3)=VV*S5     +VN*C5
      XF(4)=XS(4)*C5C6
      XF(5)=XS(4)*C5S6
      XF(6)=XS(4)*S5
   END IF

   RETURN
   END


   SUBROUTINE     GMT(DJUL,IYMDHM,SECND)
   IMPLICIT REAL*8 (A-H,O-Z)

   DIMENSION IYMDHM(5), NDIM(12)
C  ------------------------------------------------------------
C  THE EPOCH JULAIN DAY 2433281.5 CORRESPONDS TO 0 HOURS JAN 0, 1950
C  ------------------------------------------------------------
   DATA EPOCH/2433281.5/
   DATA NDIM/31,28,31,30,31,30,31,31,30,31,30,31/


   DAYS=DJUL-EPOCH

   IF(DAYS.LT.0.)THEN
     CALL MSG('HALT: GMT MUST BE LATER THAN 0 JAN 1950',3)
     STOP
   END IF

   NDAYS=DAYS


C  COMPUTE HOURS, MINUTES, AND SECONDS
C  ---------------------------------
   FDAY=DAYS-NDAYS
   HRS=24.*FDAY
   IYMDHM(4)=HRS
   FHR=HRS-IYMDHM(4)
   XMN=60.*FHR
   IYMDHM(5)=XMN
   FMN=XMN-IYMDHM(5)
```

```
      SECND=60.*FMN


C     COMPUTE YEAR, MONTH, AND DAY
C     ----------------------------
      IYR=1950
      IMO=1
      IDY=0
      NDIM(2)=28

      DO 100 I=1,NDAYS
      IDY=IDY+1
      IF(IDY.GT.NDIM(IMO))THEN
        IDY=1
        IMO=IMO+1
        IF(IMO.GT.12)THEN
          IMO=1
          IYR=IYR+1
          IF(MOD(IYR,4).EQ.0) NDIM(2)=29
          IF(MOD(IYR,4).NE.0) NDIM(2)=28
        END IF
      END IF
  100 CONTINUE

      IYMDHM(1)=IYR
      IYMDHM(2)=IMO
      IYMDHM(3)=IDY


C     CLEANUP
C     -------
      ISEC= 10000.*SECND +.5
      SECND= DFLOAT(ISEC)/10000.

      IF(SECND.GE.60.)THEN
        SECND=0.
        IYMDHM(5)=IYMDHM(5) +1
        IF(IYMDHM(5).EQ.60)THEN
          IYMDHM(5)=0
          IYMDHM(4)=IYMDHM(4) +1
          IF(IYMDHM(4).EQ.24)THEN
            IYMDHM(4)=0
            IYMDHM(3)=IYMDHM(3) +1
            IF(IYMDHM(3).GT.NDIM(IYMDHM(2)))THEN
              IYMDHM(3)=1
              IYMDHM(2)=IYMDHM(2) +1
              IF(IYMDHM(2).GT.12)THEN
                IYMDHM(2)=1
                IYMDHM(1)=IYMDHM(1) +1
              END IF
            END IF
          END IF
```

```
      END IF
    END IF

    RETURN
    END


    SUBROUTINE    ORBECF(IORB,T,XFPOS)

    IMPLICIT REAL*8 (A-H,O-Z)

C   COMPUTES THE ECS POSITION COORDINATES (XFPOS) OF A SATELLITE
C   AT JULIAN DATE T IN ORBIT NO. IORB
C   -----------------------------------------------------------
    DIMENSION XFPOS(3), XCID(3)

    GHA=GHANG(T)
    SGHA=SIN(GHA)
    CGHA=COS(GHA)

    CALL ORBCID(IORB,T,XCID)

    XFPOS(1)= XCID(1)*CGHA +XCID(2)*SGHA
    XFPOS(2)=-XCID(1)*SGHA +XCID(2)*CGHA
    XFPOS(3)= XCID(3)

    RETURN
    END


    SUBROUTINE   TLNCH
    IMPLICIT REAL*8 (A-H,O-Z)
    LOGICAL*4 LREF, LGEOM, LGMT, LORBELE, LFILL

C   THIS SUBROUTINE COMPUTES GUESSES FOR ODJ(NORB0) AND OEM(NORB0,6)
C   THAT ARE NEEDED WHEN THE INITIAL ORBIT IS SPECIFIED IN LI COORDI-
C   NATES WITH A FREE LAUNCH GMT.  SUBROUTINE GETOEM HAS ALREADY
C   COMPUTED VALUES FOR THESE QUANTITIES THAT CORRESPOND TO THE
C   MINIMUM ALLOWABLE LAUNCH TIME.
C   -----------------------------------
    COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
    COMMON/IMA06/PI,TWOPI,PIO2
    COMMON/IMA12/NORBIT,KTYPE(12),NGMT(12,5),GMTSEC(12),LGMT(12),
   &             ORBELE(12,8),LORBELE(12,8)
    COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
    COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
    COMMON/IMA32/LREF,NTSHIFT(3),TSHIFTS
    COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
    COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)

    DIMENSION IYMDHM(5)
    DATA RADEG/57.2957795/
```

```
      IF(LGMT(NORB0))THEN
        CALL MSG('INAPPROPRIATE CALL TO SUBROUTINE TLNCH',1)
        RETURN
      END IF


C     COMPUTE TIME DELAY (DAYS) UNTIL MISSION START
C     ---------------------------------------------
      TSHIFT=0.

      IF(LREF)THEN
        TSHIFT=NTSHIFT(1) +FLOAT(NTSHIFT(2))/24. +FLOAT(NTSHIFT(3))/1440.
     &         +TSHIFTS/86400.
      END IF


C     IDENTIFY, UP TO A MAXIMUM OF TWO ORBITS, THE TARGET ORBITS
C     HAVING FIXED RIGHT ASCENSIONS
C     ----------------------------
      N1=0
      N2=0

      DO 10 I=1,NTRAN
      IF(KTRAN(I).GT.2)THEN
        N=NTORB(I)
        IF(KIND(N).LT.4)THEN

          IF(N1.EQ.0)THEN
            N1=N
          ELSE
            N2=N
            GO TO 20
          END IF

        END IF
      END IF

   10 CONTINUE

   20 CONTINUE


C     COMPUTE MINIMUM STAY TIME IN INITIAL ORBIT
C     ------------------------------------------
      DTMIN0 =GEOM(1,1)*TWOPI/(DMADT(NORB0) +DAPDT(NORB0))


      IF(N2.NE.0)THEN
C     THERE ARE AT LEAST TWO TARGET ORBITS WITH FIXED RIGHT ASCENSIONS.
C     GUESS LAUNCH DAY WHEN THE RIGHT ASCENSIONS ARE ALMOST ALIGNED.
C     ----------------------------------------------------------------
        DELANG=OEM(N2,6) -OEM(N1,6) +(DRADT(N2) -DRADT(N1))*
```

```
     &        (TSHIFT +DTMIN0 +1.5) +DRADT(N2)*(ODJ(NORB0) -
     &        ODJ(N2)) -DRADT(N1)*(ODJ(NORB0) -ODJ(N1))

        DELANG=DMOD(DELANG,TWOPI)
        DELRAD=DRADT(N1) -DRADT(N2)
        IF(DELANG.LT.0..AND.DELRAD.GT.0.) DELANG=DELANG+TWOPI
        IF(DELANG.GT.0..AND.DELRAD.LT.0.) DELANG=DELANG-TWOPI

        DELODJ=DELANG/DELRAD
        DAYSI=INT(DELODJ)
        DAYSF=DELODJ -DAYSI
        ODJ(NORB0) =ODJ(NORB0) +DELODJ
        OEM(NORB0,6) =ANG(OEM(NORB0,6) +DAYSI*GHADI +DAYSF*GHADF)
      END IF


      IF(N1.NE.0)THEN
C     GUESS BEST LAUNCH INSTANT (APPROX.) WITHIN ONE-DAY WINDOW.
C     ---------------------------------------------------------
        DELANG=OEM(N1,6) -OEM(NORB0,6) +(DRADT(N1)-DRADT(NORB0))*
     &        (TSHIFT +DTMIN0 +.0333) +DRADT(N1)*(ODJ(NORB0) -ODJ(N1))

        DELANG=DMOD(DELANG,TWOPI)
        IF(DELANG.LT.0.) DELANG=DELANG +TWOPI

        OPRATE =GHADF -DRADT(N1)
        DELODJ =DELANG/OPRATE
        ODJ(NORB0) =ODJ(NORB0) +DELODJ
        OEM(NORB0,6) =ANG(OEM(NORB0,6) +GHADF*DELODJ)
      END IF

      PRINT *, ' '
      PRINT *,
     &'THE LAUNCH TIME FOR THIS MISSION WILL BE SELECTED TO AVOID SIGNIF
     &ICANT'
      PRINT *,
     &'PLANE CHANGES.  THE RESULTING MISSION DESIGN WILL PROVIDE A REFER
     &ENCE FOR'
      PRINT *,
     &'FURTHER ANALYSIS BY THE USER IN WHICH HE CAN SPECIFY THE LAUNCH T
     &IME.'
      PRINT *,
     &'-----------------------------------------------------------------
     &----'


      WRITE(20,*) ' '
      WRITE(20,*)
     &'THE LAUNCH TIME FOR THIS MISSION WILL BE SELECTED TO AVOID SIGNIF
     &ICANT'
      WRITE(20,*)
     &'PLANE CHANGES.  THE RESULTING MISSION DESIGN WILL PROVIDE A REFER
     &ENCE FOR'
```

```
      WRITE(20,*)
     &'FURTHER ANALYSIS BY THE USER IN WHICH HE CAN SPECIFY THE LAUNCH T
     &IME.'
      WRITE(20,*)
     &'-----------------------------------------------------------------
     &----'


      DJLNCH=ODJ(NORB0) -ORBELE(NORB0,7)/86400.
      CALL GMT(DJLNCH,IYMDHM,SECND)


      PRINT 100,
     &'LAUNCH GMT YEAR,  MONTH,    DAY:',IYMDHM(1),IYMDHM(2),IYMDHM(3)
      PRINT 110,
     &'            HOUR, MINUTE, SECOND:',IYMDHM(4),IYMDHM(5),SECND
  100 FORMAT(1X,A,2I8,I11)
  110 FORMAT(1X,A,2I8,F11.3)
      PRINT *, ' '

      WRITE(20,100)
     &'LAUNCH GMT YEAR,  MONTH,    DAY:',IYMDHM(1),IYMDHM(2),IYMDHM(3)
      WRITE(20,110)
     &'            HOUR, MINUTE, SECOND:',IYMDHM(4),IYMDHM(5),SECND
      WRITE(20,*) ' '


      IF(N1.EQ.0)THEN
        PRINT *, '*** NO TARGET ORBIT WITH FIXED RIGHT ASCENSION ***'
        WRITE(20,*) '*** NO TARGET ORBIT WITH FIXED RIGHT ASCENSION ***'
      ELSE

      OPH=24.*TWOPI/OPRATE
      IOPH=INT(OPH)
      OPM=(OPH-IOPH)*60.
      IOPM=INT(OPM)
      OPS=(OPM-IOPM)*60.

      PRINT *,
     &'INTERVAL BETWEEN DAILY LAUNCH OPPORTUNITIES:'
      PRINT 110,
     &'         HOURS, MINUTES, SECONDS:', IOPH, IOPM, OPS
      PRINT *, ' '

      WRITE(20,*)
     &'INTERVAL BETWEEN DAILY LAUNCH OPPORTUNITIES:'
      WRITE(20,110)
     &'         HOURS, MINUTES, SECONDS:', IOPH, IOPM, OPS
      WRITE(20,*) ' '

      OPRM=OPRATE*RADEG/1440.

      PRINT *,
```

```
     &'CHANGE IN NODAL ALIGNMENT PER MINUTE OF LAUNCH DELAY:'
      PRINT 120,
     &'DEGREES/MINUTE:', OPRM
  120 FORMAT(1X,A,F9.3)

      WRITE(20,*)
     &'CHANGE IN NODAL ALIGNMENT PER MINUTE OF LAUNCH DELAY:'
      WRITE(20,120)
     &'DEGREES/MINUTE:', OPRM

      END IF

C     RESET KIND(NORB0) TO INDICATE ALL ELEMENTS ARE NOW SPECIFIED

      KIND(NORB0) = 1

      RETURN
      END


      SUBROUTINE    TRKWIN(IORB,TB,TE,TBE)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LCOV(15)

C     COMPUTES AN ARRAY OF TIMES, TBE(I), THAT DEFINE THE BEGINNINGS
C     (ODD I) AND ENDS (EVEN I) OF TRACKING WINDOWS WITHIN THE INPUT
C     INTERVAL (TB TO TE) ON ORBIT NO. IORB.  ALL TIMES ARE EXPRESSED
C     AS JULIAN DATES.  A VERY LARGE VALUE OF TBE(1) (=9.E9) INDICATES
C     THAT THERE IS NO TRACKING COVERAGE ANYWHERE WITHIN THE INTERVAL.
C     IF THERE IS COVERAGE AT THE END OF THE INTERVAL, TBE(I)=TE FOR
C     I=2 OR 4 OR... DEPENDING OF THE NUMBER OF TRACKING WINDOWS WITHIN
C     THE INTERVAL.  TRACKING COVERAGE EXISTS IF AT LEAST ONE OF THE
C     NTRK TRACKING STATIONS HAS AN UNOBSTRUCTED LOS TO THE SATELLITE
C     IN ORBIT NO. IORB.
C     ---------------------------------------------------------------

      COMMON/IMA16/NTRK,TRKLAT(15),TRKLONG(15),TRKALT(15)

      DIMENSION TBE(100), XFPOS(3)

      DATA DT/2.E-4/


      T=TB
      CALL ORBECF(IORB,T,XFPOS)
      CALL TRKCOV(XFPOS,-NTRK,LCOV)
      T1=T
      T=T1+DT

      IF(LCOV(1))THEN
        IW=1
        TBE(IW)=TB
```

```
          GO TO 20
        ELSE
          IW=0
        END IF

C       FIND BEGINNING OF TRACKING WINDOW
C       ---------------------------------
   10   IW=IW+1

        IF(IW.GT.99)THEN
          CALL MSG('***HALT: TRACK INTERVAL HAS TOO MANY WINDOWS',3)
          STOP
        END IF

   12   IF(T.GT.TE) T=TE
        CALL ORBECF(IORB,T,XFPOS)
        CALL TRKCOV(XFPOS,-NTRK,LCOV)

        IF(.NOT.LCOV(1))THEN
          IF(T.EQ.TE)THEN
            TBE(IW)=9.E9
            RETURN
          ELSE
            T1=T
            T=T1+DT
            GO TO 12
          END IF
        END IF

        T2=T

   14   IF(ABS(T2-T1).GT.1.E-5)THEN
          T=.5*(T1+T2)
          CALL ORBECF(IORB,T,XFPOS)
          CALL TRKCOV(XFPOS,-NTRK,LCOV)
          IF(LCOV(1))T2=T
          IF(.NOT.LCOV(1))T1=T
          GO TO 14
        END IF

        TBE(IW)=.5*(T1+T2)
        T1=TBE(IW)
        T=T1+DT


C       FIND END OF TRACKING WINDOW
C       ---------------------------
   20   IW=IW+1

   22   IF(T.GT.TE) T=TE
        CALL ORBECF(IORB,T,XFPOS)
        CALL TRKCOV(XFPOS,-NTRK,LCOV)
```

```
      IF(LCOV(1))THEN
        IF(T.EQ.TE)THEN
          TBE(IW)=TE
          RETURN
        ELSE
          T1=T
          T=T1+DT
          GO TO 22
        END IF
      END IF

      T2=T

   24 IF(ABS(T2-T1).GT.1.E-5)THEN
        T=.5*(T1+T2)
        CALL ORBECF(IORB,T,XFPOS)
        CALL TRKCOV(XFPOS,-NTRK,LCOV)
        IF(LCOV(1))T1=T
        IF(.NOT.LCOV(1))T2=T
        GO TO 24
      END IF

      TBE(IW)=.5*(T1+T2)
      T1=TBE(IW)
      T=T1+DT

      GO TO 10

      END
```

```
      SUBROUTINE  EQSAM  (EQ,EL)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ---------------------------------------------------------
C     CONVERTS EQ ELEMENT SET TO EL ELEMENT SET (SEE COMMENT IN
C     SAMEQ SUBROUTINE ABOVE FOR DEFINITIONS OF EQ AND EL)
C     ---------------------------------------------------------
      DIMENSION EQ(6), EL(6)
      EL(1)=EQ(1)*(1.-EQ(2)*EQ(2))
      EL(2)=EQ(2)
      CALL KEPLE (EQ(2),EQ(6),XE,FF)
      EL(3)=FF
      EL(4)=EQ(5)
      EL(5)=EQ(3)
      EL(6)=EQ(4)
      RETURN
      END


      SUBROUTINE      GAUSS(R1,R2,ANG,TME,VTC,VTR,F1,SLR,ECC,KGERR)
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES THE VELOCITY COMPONENTS AT BOTH ENDS OF A TRANSFER CONIC
C     BETWEEN RADII R1 AND R2, COVERING A PRESCRIBED ANGLE (ANG) IN A
C     PRESCRIBED TIME (TME).  THE METHOD IS TO DETERMINE THE
C     TRUE ANOMALY (F1) OF R1 ON THE TRANSFER CONIC AND TO DERIVE THE
C     SEMILATUS RECTUM AND ECCENTRICITY (SLR,ECC) OF THE TRANSFER CONIC
C     FROM F1.  IF THE TRANSFER REQUIRES A PERIGEE RADIUS LOWER THAN
C     RPMIN OR AN ECCENTRICITY GREATER THAN .99, THE FLAG KGERR WILL BE
C     SET TO EITHER -1 OR -2, RESPECTIVELY.  IF THE MATH FORMULAE
C     PRODUCE A VALUE OF SLR LESS THAN RPMIN, KGERR IS SET TO -3.
C     ***********************************************************
C     NOTE: VELOCITIES ARE GIVEN IN METERS/SEC, BUT TME IS GIVEN IN
C     UNITS OF DAYS.
C     ***********************************************************

C     VTC(I) = CIRCUMFERENTIAL VELOCITY COMPONENTS ON THE TRANSFER
C              CONIC AT R1(I=1) AND R2(I=2)

C     VTR(I) = RADIAL VELOCITY COMPONENTS ON THE TRANSFER
C              CONIC AT R1(I=1) AND R2(I=2)
C     ------------------------------------------------------

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA40/RPMIN

      DIMENSION VTC(2), VTR(2)

      DATA MAXIT/20/, KERR1/0/, TOL/1.E-7/, ECCMAX/.99/, DF1TOL/.0002/

      KGERR=0

C     COMPUTE F1 LIMITS (F1A, F1B) BASED ON MINIMUM ALLOWABLE
```

```
C      PERIGEE (RPMIN)
C      ---------------------------------------------------
       CANG=COS(ANG)
       SANG=SIN(ANG)
       XMS=R1*(R2-RPMIN)-R2*(R1-RPMIN)*CANG
       XMC=R2*(R1-RPMIN)*SANG
       DEL=ATAN2(XMS,XMC)
       XM=SQRT(XMS*XMS+XMC*XMC)
       SINF1A=RPMIN*(R2-R1)/XM
       IF(ABS(SINF1A).GT.1.)THEN
         KGERR=-4
         RETURN
       END IF
       F1A=ASIN(SINF1A)-DEL
       F1B=PI-F1A-2.*DEL

       IF(R1.GT.R2)THEN
         F1T=F1A+TWOPI
         F1A=F1B
         F1B=F1T
       END IF


C      ADJUST F1 LIMITS IF NECESSARY TO KEEP
C      TRANSFER-ARC ECCENTRICITY LESS THAN ECCMAX
C      -----------------------------------------
       ET=(R2-R1)/(R1*COS(F1A) -R2*COS(F1A +ANG))
       IF(ET.GE.ECCMAX)THEN
         KGERR=-2
         F1T=F1A +(F1B-F1A)/2.
         DF1=(F1B-F1A)/4.
   10    ET=(R2-R1)/(R1*COS(F1T) -R2*COS(F1T +ANG))
         IF(ET.GE.ECCMAX)THEN
           F1T=F1T+DF1
         ELSE
           KGERR=0
           F1A=F1T
           F1T=F1T-DF1
         END IF
         IF(DF1.GT.DF1TOL)THEN
           DF1=DF1/2.
           GO TO 10
         END IF
       END IF
       IF(KGERR.NE.0)RETURN

       ET=(R2-R1)/(R1*COS(F1B) -R2*COS(F1B +ANG))
       IF(ET.GE.ECCMAX)THEN
         KGERR=-2
         F1T=F1B +(F1A-F1B)/2.
         DF1=(F1A-F1B)/4.
   20    ET=(R2-R1)/(R1*COS(F1T) -R2*COS(F1T +ANG))
         IF(ET.GE.ECCMAX)THEN
           F1T=F1T+DF1
```

```
        ELSE
          KGERR=0
          F1B=F1T
          F1T=F1T-DF1
        END IF
        IF(ABS(DF1).GT.DF1TOL)THEN
          DF1=DF1/2.
          GO TO 20
        END IF
      END IF
      IF(KGERR.NE.0)RETURN

C     COMPUTE LIMITS ON FEASIBLE TRANSFER TIMES
C     -----------------------------------------------------------------
      CALL TGAUSS(R1,R2,SANG,CANG,F1A,TA,SLR,ECC,CF1,CF2,SF1,SF2,KGERR)
      IF(KGERR.NE.0)RETURN
      CALL TGAUSS(R1,R2,SANG,CANG,F1B,TB,SLR,ECC,CF1,CF2,SF1,SF2,KGERR)
      IF(KGERR.NE.0)RETURN

      IF((TME.GT.TA.AND.TME.GT.TB).OR.(TME.LT.TA.AND.TME.LT.TB))THEN
C     A PERIGEE RADIUS LOWER THAN RPMIN IS REQUIRED
        KGERR=-1
        RETURN
      END IF

      ITER=0
   30 ITER=ITER+1

      IF(MOD(ITER,2).EQ.0)THEN
        F1=F1A +(TME-TA)*(F1B-F1A)/(TB-TA)
      ELSE
        F1=.5*(F1A +F1B)
      END IF

      CALL TGAUSS(R1,R2,SANG,CANG,F1,TG,SLR,ECC,CF1,CF2,SF1,SF2,KGERR)
      IF(KGERR.NE.0)RETURN

      IF(ABS(TG-TME).GT.TOL.AND.ITER.LT.MAXIT)THEN
        IF((TG-TME)*(TA-TME).LT.0.)THEN
          TB=TG
          F1B=F1
        ELSE
          TA=TG
          F1A=F1
        END IF
        GO TO 30
      END IF

      IF(ITER.EQ.MAXIT)THEN

        IF(KERR1.EQ.0)THEN
          CALL MSG('*** WARNING: 2BTOR DELTAVS MAY NOT BE ACCURATELY COM
     &PUTED',1)
```

```
         TERR=TG-TME
         PRINT *, 'TIME ERROR = ',TERR
         WRITE(20,*) 'TIME ERROR = ',TERR
       END IF

       KERR1=1
     END IF


     FAC=SQRT(XMU/SLR)

     VTC(1)=FAC*(1.+ECC*CF1)
     VTC(2)=FAC*(1.+ECC*CF2)
     VTR(1)=FAC*ECC*SF1
     VTR(2)=FAC*ECC*SF2


     RETURN
     END


     SUBROUTINE  GETOEM  (IORB)

     IMPLICIT REAL*8 (A-H,O-Z)
     LOGICAL*4 LGMT, LORBELE

C    COMPUTES MEAN ORBITAL ELEMENTS FOR ORBIT IORB, DETERMINES THE
C    KIND OF ORBIT (IE., THE COMBINATION OF FREE PARAMETERS),
C    AND SENDS A FATAL ERROR MESSAGE FOR UNACCEPTABLE COMBINATION
C    OF FREE ELEMENTS.  ALSO COMPUTES SECULAR ANGULAR RATES FOR
C    ARGUMENT OF PERIGEE, TRUE ANOMALY, AND MEAN ANOMALY FOR ALL
C    ORBITS EXCEPT THOSE OF KIND 6 (FREE INCLINATION), AND COMPUTES
C    SINES AND COSINES OF THE MEAN ORBITAL INCLINATIONS.  THIS
C    SUBROUTINE IS CALLED ONLY ONCE, AT THE BEGINNING OF IMA PROGRAM
C    EXECUTION.


C    -------
C    INPUTS: *********************************************
C    -------

C    KTYPE(IORB)        INPUT ORBIT TYPE

C      1                MEAN ELEMENTS
C      2                OSCULATING ELEMENTS
C      3                ECI CARTESIAN
C      4                ECID CARTESIAN
C      5                ECF CARTESIAN
C      6                ECF SPHERICAL
C      7                LI CARTESIAN

C    ORBELE(IORB,J) = INPUT ORBITAL PARAMETER VALUES.  PARAMETER
C                     DEFINITIONS DEPEND ON KTYPE(IORB)
```

```
C      NGMT(IORB,K)     = INTEGER ARRAY INDICATING YEAR, MONTH, DAY, HOUR,
C                         AND MINUTE OF GMT

C      GMTSEC(IORB)     = SECONDS OF GMT (ALLOWS FRACTIONS OF SECONDS)

C      LGMT(IORB)       = .TRUE. WHEN GMT IS SPECIFIED
C                        .FALSE. WHEN GMT IS FREE

C      LORBELE(IORB,J)= .TRUE. WHEN ORBELE(IORB,J) IS SPECIFIED
C                        .FALSE. WHEN ORBELE(IORB,J) IS FREE



C      --------
C      OUTPUTS: ************************************
C      --------

C      ODJ(IORB) =      REFERENCE JULIAN DATE.  FOR ALL ORBITS EXCEPT
C                       THOSE OF TYPE 7, THIS REFERENCE CORRESPONDS TO
C                       THE INPUT GMT.  FOR TYPE 7 ORBITS, THE REFERENCE
C                       CORRESPONDS TO THE INPUT GMT PLUS THE INPUT TIME
C                       AFTER LAUNCH (CONVERTED FROM SECONDS TO DAYS).

C      MEAN ORBITAL ELEMENTS AT EPOCH (AT ODJ(IORB))
C      ************************************************************
C      THE THIRD ELEMENT USED TO BE TRUE ANOMALY.  HOWEVER, FOR
C      THE EPOCHAL ELEMENT VALUES, MEAN ANOMALY IS COMPUTED INSTEAD
C      OF TRUE ANOMALY BECAUSE IT IS MEAN ANOMALY THAT HAS A
C      SECULAR RATE CAUSED BY EARTH OBLATENESS.  THEN, ANY FUTURE VALUE
C      OF MEAN ANOMALY CAN BE QUICKLY COMPUTED.  IF TRUE ANOMALIES WERE
C      COMPUTED HERE, A CORRESPONDING ARRAY OF MEAN ANOMALIES WOULD HAVE TO
C      BE COMPUTED OR ELSE A CONVERSION FROM TRUE TO MEAN ANOMALY WOULD
C      ALWAYS BE REQUIRED BEFORE THE FUTURE POSITION OF A SATELLITE
C      COULD BE COMPUTED.
C      *************************************************************

C      OEM(IORB,1) =  SEMILATUS RECTUM
C      OEM(IORB,2) =  ECCENTRICITY
C      OEM(IORB,3) =  MEAN ANOMALY
C      OEM(IORB,4) =  ARGUMENT OF PERIGEE
C      OEM(IORB,5) =  INCLINATION
C      OEM(IORB,6) =  RIGHT ASCENSION


C      KIND(IORB)=1 FOR ALL TYPES OF ORBITS WITHOUT ANY FREE PARAMETERS.
C                   (GMT AND ALL OEM VALUES ARE FIXED)

C      KIND(IORB)=7 FOR LI CARTESIAN ORBIT WITH FREE GMT.
C                  (ALL OEM VALUES ARE FIXED EXCEPT RIGHT ASCENSION)

C      KIND(IORB)=2,3,.....6 APPLIES TO MEAN-ELEMENT-TYPE ORBITS,
C      INDICATING THE COMBINATION OF FREE PARAMETERS AS SHOWN
C      IN THE FOLLOWING TABLE (* INDICATES A FREE PARAMETER).
```

```
C      KIND          GMT    SLR    ECC    ARP    TRA    INC    RAC
C       2            ---    ---    ---    ---     *     ---    ---
C       3            ---    ---    ---     *      *     ---    ---
C       4            ---    ---    ---    ---     *     ---     *
C       5             *     ---    ---     *      *     ---     *
C       6            ---    ---    ---    ---     *      *      *


C      SININC(IORB)  SINE OF MEAN INCLINATION
C      COSINC(IORB)  COSINE OF MEAN INCLINATION


C      SECULAR ANGULAR RATES CAUSED BY EARTH OBLATENESS
C      -----------------------------------------------
C      DRADT(IORB)  RATE OF RIGHT ASCENSION, RADIANS/DAY
C      DAPDT(IORB)  RATE OF ARGUMENT OF PERIGEE, RADIANS/DAY
C      DMADT(IORB)  RATE OF MEAN ANOMALY, RADIANS/DAY

C      NOTE: THESE RATES ARE COMPUTED BY CALLS TO SUBROUTINE ORBMOVE
C      ***********************************************************

C      -----------------------------------------------------------
       COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
       COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
       COMMON/IMA11/NGMTMN(5),GMTMNS
       COMMON/IMA12/NORBIT,KTYPE(12),NGMT(12,5),GMTSEC(12),LGMT(12),
     &              ORBELE(12,8),LORBELE(12,8)
       COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
       COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
C      ------------------------------------------------------------
       DIMENSION IYM(5), YY(6), ZZ(6)



C      FOR ALL ORBIT-PARAMETER TYPES EXCEPT MEAN ELEMENTS, CHECK FOR
C      UNACCEPTABLE FREE VARIABLES, AND COMPUTE JULIAN DATE OF EPOCH
C      WHEN APPROPRIATE.
C      ---------------------------------------------
       IF(KTYPE(IORB).NE.1) THEN

         IF(KTYPE(IORB).NE.7) JEND=6
         IF(KTYPE(IORB).EQ.7) JEND=8

         DO 10 J=1,JEND
         IF(LORBELE(IORB,J)) GO TO 10
         CALL MSG('HALT: ALL ORBIT PARAMETERS MUST BE SPECIFIED UNLESS TH
     &E ORBIT IS DEFINED BY MEAN ELEMENTS',3)
         STOP
  10     CONTINUE

         IF(KTYPE(IORB).NE.7.AND..NOT.LGMT(IORB)) THEN
           CALL MSG('HALT: GMT MUST BE SPECIFIED UNLESS THE ORBIT IS DEFI
     &NED BY MEAN ELEMENTS OR LI COORDINATES',3)
           STOP
```

```
         END IF

         IF(KTYPE(IORB).EQ.7.AND..NOT.LGMT(IORB))THEN
           KIND(IORB)=7
C     --------- TEMPORARY ODJ COMPUTED FOR THE KIND=7 ORBIT------

           IF(NGMTMN(1).LT.1950.OR.NGMTMN(1).GT.2099.or.ngmtmn(2).
     &        lt.1.or.ngmtmn(2).gt.12)then
             CALL MSG('***HALT: MINIMUM GMT FOR LI ORBIT IS OUTSIDE ACCEP
     &TABLE RANGE.  MAKE CORRECTION ON INITIAL-CONDITIONS SCREEN.',3)
             STOP
           END IF

           ODJ(IORB)=DJUL(NGMTMN,GMTMNS) +ORBELE(IORB,7)/86400.
         END IF

         IF(KTYPE(IORB).NE.7.OR.KTYPE(IORB).EQ.7.AND.LGMT(IORB)) THEN
           KIND(IORB)=1
           SEC=GMTSEC(IORB)
           DO 12 J=1,5
   12      IYM(J)=NGMT(IORB,J)
           ODJ(IORB)=DJUL(IYM,SEC)

           IF(KTYPE(IORB).EQ.7) THEN
             ODJ(IORB)= ODJ(IORB) +ORBELE(IORB,7)/86400.
           END IF

         END IF

       END IF
C     ---------------------------------------------------------------


C     FOR MEAN-ELEMENT TYPE ORBIT, DETERMINE KIND (FREE COMBINATION),
C     CHECK FOR UNACCEPTABLE KIND, AND LOAD ORBELE ARRAY INTO OEM ARRAY.
C     --------------------------------------------------------------
       IF(KTYPE(IORB).EQ.1) THEN
         IF(.NOT.LORBELE(IORB,1).OR..NOT.LORBELE(IORB,2)) THEN
           CALL MSG('HALT: MEAN APOGEE AND PERIGEE ALTITUDES MUST BE SPEC
     &IFIED',3)
           STOP
         END IF

         KIND(IORB)=0

         IF(      LGMT(IORB).AND.
     &            LORBELE(IORB,3).AND.
     &            LORBELE(IORB,4).AND.
     &            LORBELE(IORB,5).AND.
     &            LORBELE(IORB,6))       KIND(IORB)=1

         IF(      LGMT(IORB).AND.
     &            LORBELE(IORB,3).AND.
     &            LORBELE(IORB,4).AND.
```

```
     &          LORBELE(IORB,5).AND.
     &     .NOT.LORBELE(IORB,6))        KIND(IORB)=2

        IF(     LGMT(IORB).AND.
     &          LORBELE(IORB,3).AND.
     &          LORBELE(IORB,4).AND.
     &     .NOT.LORBELE(IORB,5).AND.
     &     .NOT.LORBELE(IORB,6))        KIND(IORB)=3

        IF(     LGMT(IORB).AND.
     &          LORBELE(IORB,3).AND.
     &     .NOT.LORBELE(IORB,4).AND.
     &          LORBELE(IORB,5).AND.
     &     .NOT.LORBELE(IORB,6))        KIND(IORB)=4

        IF(.NOT.LGMT(IORB).AND.
     &          LORBELE(IORB,3).AND.
     &     .NOT.LORBELE(IORB,4).AND.
     &     .NOT.LORBELE(IORB,5).AND.
     &     .NOT.LORBELE(IORB,6))        KIND(IORB)=5

        IF(     LGMT(IORB).AND.
     &     .NOT.LORBELE(IORB,3).AND.
     &     .NOT.LORBELE(IORB,4).AND.
     &          LORBELE(IORB,5).AND.
     &     .NOT.LORBELE(IORB,6))        KIND(IORB)=6

        IF(KIND(IORB).EQ.0) THEN
          CALL MSG('HALT: UNACCEPTABLE COMBINATION OF FREE MEAN ORBITAL
     &ELEMENTS',3)
          STOP
        END IF


        IF(LGMT(IORB)) THEN
          SEC=GMTSEC(IORB)
          DO 15 J=1,5
   15     IYM(J)=NGMT(IORB,J)
          ODJ(IORB)=DJUL(IYM,SEC)
        END IF

        RA=REQ +ORBELE(IORB,1)
        RP=REQ +ORBELE(IORB,2)

        OEM(IORB,1)=2.*RA*RP/(RA +RP)
        OEM(IORB,2)=(RA-RP)/(RA+RP)
        OEM(IORB,3)=XMANOM(OEM(IORB,2),ORBELE(IORB,6))
        OEM(IORB,4)=ORBELE(IORB,5)
        OEM(IORB,5)=ORBELE(IORB,3)
        OEM(IORB,6)=ORBELE(IORB,4)
C     *****************************************************************
C     NOTE: ONE OR MORE OF THE OEM VALUES WILL BE BOGUS UNLESS
C     KIND(IORB)=1, BUT NO HARM IS DONE BECAUSE GOOD OEM VALUES
C     WILL BE COMPUTED WHERE APPROPRIATE IN OTHER PROGRAM SUBROUTINES.
```

```
C     ******************************************************************
      END IF
C     ----------------------------------------------------------



C     FOR OSCULATING TYPE ORBIT, COMPUTE SLR AND ECC AND LOAD ELEMENT
C     VALUES INTO YY ARRAY
C     ----------------------------------
      IF(KTYPE(IORB).EQ.2) THEN

        RA=REQ +ORBELE(IORB,1)
        RP=REQ +ORBELE(IORB,2)

        YY(1)=2.*RA*RP/(RA +RP)
        YY(2)=(RA-RP)/(RA+RP)
        YY(3)=ORBELE(IORB,6)
        YY(4)=ORBELE(IORB,5)
        YY(5)=ORBELE(IORB,3)
        YY(6)=ORBELE(IORB,4)
      END IF
C     ------------------------

C     FOR ECI TYPE ORBIT, TRANSFORM ECI VALUES TO ECID VALUES
C     -------------------------------------------------------
      IF(KTYPE(IORB).EQ.3) THEN
        DJ=ODJ(IORB)
        YY(1)=ORBELE(IORB,4)
        YY(2)=ORBELE(IORB,5)
        YY(3)=ORBELE(IORB,6)
        YY(4)=ORBELE(IORB,1)
        YY(5)=ORBELE(IORB,2)
        YY(6)=ORBELE(IORB,3)
        CALL ECITRAN(YY,ZZ,DJ,1)
      END IF

C     FOR ECID TYPE ORBIT, LOAD ELEMENT VALUES INTO ZZ ARRAY
C     ----------------------------------
      IF(KTYPE(IORB).EQ.4) THEN
        ZZ(1)=ORBELE(IORB,4)
        ZZ(2)=ORBELE(IORB,5)
        ZZ(3)=ORBELE(IORB,6)
        ZZ(4)=ORBELE(IORB,1)
        ZZ(5)=ORBELE(IORB,2)
        ZZ(6)=ORBELE(IORB,3)
      END IF

C     FOR ECF CARTESIAN TYPE ORBIT, LOAD VALUES INTO YY ARRAY
C     -------------------------------------------------------
      IF(KTYPE(IORB).EQ.5) THEN
```

```
          YY(1)=ORBELE(IORB,4)
          YY(2)=ORBELE(IORB,5)
          YY(3)=ORBELE(IORB,6)
          YY(4)=ORBELE(IORB,1)
          YY(5)=ORBELE(IORB,2)
          YY(6)=ORBELE(IORB,3)
        END IF

C       FOR ECF SPHERICAL TYPE ORBIT, TRANSFORM TO ECF CARTESIAN VALUES
C       ---------------------------------------------------------------
        IF(KTYPE(IORB).EQ.6) THEN
          ZZ(1)=ORBELE(IORB,4)
          ZZ(2)=ORBELE(IORB,5)
          ZZ(3)=ORBELE(IORB,6)
          ZZ(4)=ORBELE(IORB,1) +REQ
          ZZ(5)=ORBELE(IORB,2)
          ZZ(6)=ORBELE(IORB,3)
          CALL FSTRAN(YY,ZZ,-1)
        END IF

C       FOR LI TYPE ORBIT, TRANSFORM TO ECID VALUES
C       ------------------------------------------
        IF(KTYPE(IORB).EQ.7) THEN
C         ***************************************************************
C         NOTE: IF GMT IS FREE (KIND=7), A TEMPORARY ODJ(IORB) HAS BEEN
C         COMPUTED BASED ON THE MINIMUM ACCEPTABLE LAUNCH GMT (NGMTMN(I),
C         GMTMNS).  ALL OF THE MEAN ELEMENTS FOR THIS TYPE OF ORBIT,
C         EXCEPT RIGHT ASCENSION, ARE INDEPENDENT OF THE INITIAL TIME.
C         THE RIGHT ASCENSION COMPUTED HERE MUST BE ADJUSTED BY THE TOP-
C         LEVEL OPTIMIAZTION PROGRAM WHENEVER THE INITIAL TIME, ODJ(IORB),
C         IS VARIED FROM THE TEMPORARY VALUE COMPUTED IN THIS SUBROUTINE.
C         ***************************************************************
          DJ=ODJ(IORB)
          DJLNCH=DJ - ORBELE(IORB,7)/86400.
          XLONG=ORBELE(IORB,8)
          YY(1)=ORBELE(IORB,4)
          YY(2)=ORBELE(IORB,5)
          YY(3)=ORBELE(IORB,6)
          YY(4)=ORBELE(IORB,1)
          YY(5)=ORBELE(IORB,2)
          YY(6)=ORBELE(IORB,3)
          CALL LITRAN(YY,ZZ,DJ,DJLNCH,XLONG,1)
        END IF

C       TRANSFORM ECF CARTESIAN VALUES TO ECID VALUES
C       ---------------------------------------------
        IF(KTYPE(IORB).EQ.5.OR. KTYPE(IORB).EQ.6) THEN
          DJ=ODJ(IORB)
          CALL IFTRAN(ZZ,YY,DJ,-1)
        END IF
```

```
C     TRANSFORM ECID VALUES TO OSCULATING ELEMENTS
C     ------------------------------------------
      IF(KTYPE(IORB).NE.1.AND.KTYPE(IORB).NE.2) THEN
        CALL OITRAN(YY,ZZ,-1)
      END IF


C     TRANSFORM OSCULATING ELEMENTS TO MEAN ELEMENTS AND LOAD THEM
C     INTO OEM(IORB,J) ARRAY
C     -----------------------
      IF(KTYPE(IORB).NE.1) THEN
        CALL MOTRAN(ZZ,YY,-1)
        DO 80 J=1,6
   80   OEM(IORB,J)=ZZ(J)
        OEM(IORB,3)=XMANOM(ZZ(2),ZZ(3))
      END IF



C     COMPUTE THE ANGULAR RATES (RADIANS/DAY) OF RIGHT ASCENSION
C     (DRADT(IORB)), ARGUMENT OF PERIGEE (DAPDT(IORB)), AND MEAN
C     ANOMALY (DMADT(IORB)), AS WELL AS THE SINE AND COSINE OF THE
C     MEAN INCLINATION, WHEN ORBITAL INCLINATION IS FIXED.
C     --------------------------------------------------
      IF(KIND(IORB).NE.6) THEN
        SININC(IORB)=SIN(OEM(IORB,5))
        COSINC(IORB)=COS(OEM(IORB,5))
        CALL ORBMOVE(OEM(IORB,1),OEM(IORB,2),SININC(IORB),
     &       COSINC(IORB),DRADT(IORB),DAPDT(IORB),DMADT(IORB))
      END IF


      RETURN
      END
```

FUNCTION GHANG (DJUL)

```
      IMPLICIT REAL*8 (A-H,O-Z)
C     --------------------------------------------------------------
C     COMPUTES THE GREENWICH HOUR ANGLE, GIVEN THE JULIAN DATE (DJUL)
C
C     NOMINALLY, DJUL0 =2433282.5 (JULIAN DATE AT REFERENCE GMT,
C     0.HR, 1 JAN 1950)
C     ---------------------------------------------------------------
      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0

      DAYS=DJUL-DJUL0
      DAYSI=INT(DAYS)
      DAYSF=DAYS-DAYSI

      GHANG =GHA0 +GHADI*DAYSI +GHADF*DAYSF

      RETURN
```

```
      END


      SUBROUTINE  IFTRAN  (X,  XF,  DJUL,  MODE)

      IMPLICIT REAL*8 (A-H,O-Z)
C     ----------------------------------------------------------------
C     TRANSFORMS ECID COORDINATES (X) TO ECF COORDINATES (XF)
C     OR VICE VERSA, DEPENDING ON WHETHER MODE>0 OR MODE<0.

C     X(1)...X(3), XF(1)...XF(3) = X,Y,Z VELOCITY COMPONENTS
C     X(4)...X(6), XF(4)...XF(6) = X,Y,Z POSITION COORDINATES

C     DJUL = JULIAN DATE FOR THE TRANSFORMATION
C     GHA  = GREENWICH HOUR ANGLE
C     ----------------------------------------------------------------
      DIMENSION X(6), XF(6)

      GHA=GHANG(DJUL)

      SGHA=SIN(GHA)
      CGHA=COS(GHA)

      IF(MODE.GT.0) THEN
        XF(1)= X(1)*CGHA +X(2)*SGHA
        XF(2)=-X(1)*SGHA +X(2)*CGHA
        XF(3)= X(3)
        XF(4)= X(4)*CGHA +X(5)*SGHA
        XF(5)=-X(4)*SGHA +X(5)*CGHA
        XF(6)= X(6)
      ELSE
        X(1)= XF(1)*CGHA -XF(2)*SGHA
        X(2)= XF(1)*SGHA +XF(2)*CGHA
        X(3)= XF(3)
        X(4)= XF(4)*CGHA -XF(5)*SGHA
        X(5)= XF(4)*SGHA +XF(5)*CGHA
        X(6)= XF(6)
      END IF

      RETURN
      END


      SUBROUTINE   INIT

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LOWACC, LMORB, LGMT, LORBELE, LFILL, LGEOM
C     ----------------------------------------------------------------
C     INITIALIZES ALL VALUES THAT ARE A FUNCTION OF THE MISSION BUT
C     THAT REMAIN CONSTANT OVER THE MISSION TIME. THIS ROUTINE IS THE
C     FIRST ONE CALLED WHEN THE USER OPTS TO EXECUTE.
C     ----------------------------------------------------------------
      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
```

```
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA08/XLPSUN,ECCSUN,SOBL,COBL,ASUN0,ASUND
      COMMON/IMA10/ECCS0,ECCSD,XLPS0,XLPSD
      COMMON/IMA12/NORBIT,KTYPE(12),NGMT(12,5),GMTSEC(12),LGMT(12),
     &             ORBELE(12,8),LORBELE(12,8)
      COMMON/IMA14/NPL,PLMASS(12),ACCLIM(12)
      COMMON/IMA16/NTRK,TRKLAT(15),TRKLONG(15),TRKALT(15)
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA24/TFRAC(15,6),ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15)
     &            ,LOWACC(15)
      COMMON/IMA26/NPLD(15),IPLD(15,12)
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA38/ACCMAX(15),EMASS(15),MPSYS(15),THRNOM(15,6),
     &             FLWNOM(15,6),BCOEF(15,4)
      COMMON/IMA56/XFTRK(15,3),RADTRK(15)
C     ------------------------------------------------------------
      DIMENSION LMORB(12)
      DATA GZERO/9.80665/

      NSUN=0
      DO 10 I=1,NORBIT
   10 LMORB(I)=.FALSE.


C     COMPUTE MEAN ELEMENTS, KIND, AND REFERENCE JULIAN DATE FOR
C     ALL MISSION ORBITS ----------------------------------------

      CALL GETOEM(NORB0)
      IF(KIND(NORB0).NE.1.AND.KIND(NORB0).NE.7)THEN
        CALL MSG('***HALT: ALL ELEMENTS MUST BE SPECIFIED FOR INITIAL OR
     &BIT',3)
        STOP
      END IF
      LMORB(NORB0)=.TRUE.
      IF(LGMT(NORB0)) NSUN=NORB0

      DO 20 I=1,NTRAN
      K=KTRAN(I)

      IF(K.EQ.3.OR.K.EQ.4.OR.K.EQ.5) THEN
C     PERFORM COMPUTATIONS RELATED TO TARGET ORBITS
C     ---------------------------------------------
        IORB=NTORB(I)
        IF(.NOT.LMORB(IORB)) THEN
          CALL GETOEM(IORB)
          LMORB(IORB)=.TRUE.
          IF(LGMT(IORB)) NSUN=IORB
          IF(.NOT.LGMT(IORB))THEN
            CALL MSG('***HALT: ONLY INITIAL ORBIT CAN HAVE FREE GMT',3)
            STOP
          END IF
```

```
        END IF
      END IF
   20 CONTINUE

C     COMPUTE SUN'S ORBITAL PARAMETERS BASED ON THE GMT OF ONE (NSUN)
C     OF THE MISSION ORBITS

      IF(NSUN.GT.0) THEN
        DAYS=ODJ(NSUN) -DJUL0
        OBL=OBL0 +OBLD*DAYS
        SOBL=SIN(OBL)
        COBL=COS(OBL)
        ECCSUN=ECCS0 +ECCSD*DAYS
        XLPSUN=XLPS0 +XLPSD*DAYS
      ELSE
        CALL MSG('HALT: AT LEAST ONE ORBIT MUST HAVE A SPECIFIED GMT',3)
        STOP
      END IF


C     COMPUTE MAXIMUM ACCELERATION (ACCMAX) AND EMPTY MASS (EMASS) FOR
C     EACH TRANSFER.  IDENTIFY THE PRIMARY (THROTTLEABLE) PROPULSION
C     SUBSYSTEM (MPSYS) FOR EACH TRANSFER.  COMPUTE THE NOMINAL THRUST
C     AND PROPELLANT FLOWRATE (THRNOM, FLWNOM) DURING BURNS FOR EACH
C     TRANSFER AND SUBSYSTEM ----------------------------------------

      DO 60 I=1,NTRAN
      ACCMAX(I)=1.E6
      EMASS(I)=VEHMASS
      TPRIME=0.

      IF(NPLD(I).GT.0)THEN
        DO 30 J=1,NPLD(I)
        K=IPLD(I,J)
        IF(ACCLIM(K).LT.ACCMAX(I)) ACCMAX(I)=ACCLIM(K)
   30   EMASS(I)=EMASS(I) +PLMASS(K)
      END IF

      THRTOT=0.
      FLWTOT=0.

      DO 40 M=1,NPROP
      THRNOM(I,M)=TFRAC(I,M)*PMTHRST(M)
      EV0=GZERO*TH0ISP(M)

      IF(PMTHRST(M).GT.0.)THEN
        DEV=GZERO*(THMISP(M)-TH0ISP(M))/PMTHRST(M)
      ELSE
        DEV=0.
      END IF

      EXV=EV0 +DEV*THRNOM(I,M)
```

```
      IF(EXV.GT.0.)THEN
        FLWNOM(I,M) =THRNOM(I,M)/EXV +ACFLOW(I,M,2)
      ELSE
        CALL MSG('***HALT: INVALID EXHAUST VELOCITY',3)
        STOP
      END IF

      THRTOT=THRTOT+THRNOM(I,M)
      FLWTOT=FLWTOT+FLWNOM(I,M)
      IF(THRNOM(I,M).GT.TPRIME)THEN
        TPRIME=THRNOM(I,M)
        MPSYS(I)=M
        PEV0=EV0
        PDEV=DEV
      END IF
   40 CONTINUE

C     COMPUTE BURN THROTTLING COEFFICIENTS FOR EACH TRANSFER
C     (CK1=TOTAL FLOW RATE MINUS DELTAV FLOWRATE OF PRIMARY SUBSYSTEM)
C     (CK2=TOTAL THRUST MINUS THRUST OF PRIMARY SUBSYSTEM)
C     ************************************************************
C     NOTE: PROPELLANT FLOWRATES AND SPECIFIC IMPULSES ARE BASED ON
C     THE TIME UNIT OF SECONDS INSTEAD OF DAYS
C     ************************************************************

      CK1=FLWTOT-FLWNOM(I,MPSYS(I)) +ACFLOW(I,MPSYS(I),2)
      CK2=THRTOT-THRNOM(I,MPSYS(I))
      AA=PEV0-PDEV*CK2
      BB=PDEV*ACCMAX(I)
      BCOEF(I,3)=CK1*AA -CK2
      BCOEF(I,4)=ACCMAX(I)*(1. +PDEV*CK1)
      BCOEF(I,2)=BB/BCOEF(I,4)**2
      BCOEF(I,1)=AA/BCOEF(I,4) -BCOEF(I,2)*BCOEF(I,3)

   60 CONTINUE


C     COMPUTE ECF POSITION COORDINATES AND RADIUS MAGNITUDE FOR
C     EACH OF THE VARIOUS TRACKING STATIONS.
C     -------------------------------------------------------
      IF(NTRK.GT.0)THEN
        DO 80 I=1,NTRK
        RADTRK(I)=REQ +TRKALT(I)
        CLAT=COS(TRKLAT(I))
        SLAT=SIN(TRKLAT(I))
        CLNG=COS(TRKLONG(I))
        SLNG=SIN(TRKLONG(I))
        XFTRK(I,1)=RADTRK(I)*CLAT*CLNG
        XFTRK(I,2)=RADTRK(I)*CLAT*SLNG
        XFTRK(I,3)=RADTRK(I)*SLAT
   80   CONTINUE
      END IF

C     IF INITIAL RENDEZVOUS PHASING IS LEFT BLANK, WRITE MESSAGE
```

```
C       ---------------------------------------------------------
        DO 90 I=1, NTRAN
        IF(KTRAN(I).NE.5) GO TO 90
          IF(.NOT.LGEOM(I,5))THEN
           CALL MSG('INITIAL PHASING OF 2BTOR HAS NOT BEEN INPUT. A TYPICA
     &L VALUE WILL BE USED',1)
          END IF
          IF(ABS(GEOM(I,14)-PI).LT..034906585)THEN
            CALL MSG('**WARNING: RENDEZVOUS TRANSFER ANGLE NEAR 180 DEGREE
     &S.  RESULTS MAY BE INVALID',2)
          END IF
   90 CONTINUE

        RETURN
        END



        SUBROUTINE    KEPLE(E,XM,XE,F)
        IMPLICIT REAL*8 (A-H,O-Z)
C       ---------------------------------------------------------
C       SOLVES KEPLERS EQUATION FOR ECCENTRIC AND TRUE ANOMALIES
C       (XE,F) GIVEN ECCENTRICITY (E) AND MEAN ANOMALY (XM).
C       ---------------------------------------------------------
        DATA KERR/0/

        K=0
        XXE=XM
  1     SE=SIN(XXE)
        XMO=XXE-E*SE
        CE=COS(XXE)
        TEMP=1.-E*CE
        DELTA=(XM-XMO)/TEMP
        IF(ABS(XM-XMO)-1.E-9) 3,3,2
  2     XXE=XXE+DELTA
        K=K+1
        IF(K.EQ.30) GO TO 5
        GO TO 1

    5 IF(KERR.EQ.0) THEN
          CALL MSG('*** WARNING: KEPLE SOLUTION NOT IN TOLERENCE',1)
          XMERR=XM-XMO
          WRITE(20,*) 'MEAN-ANOMALY ERROR (RADIANS) = ', XMERR
          PRINT *, 'MEAN-ANOMALY ERROR (RADIANS) = ', XMERR
          PRINT *, 'E,XM=', E,XM
          KERR=1
      END IF

    3 XE=XXE
      ROASF=SQRT(1.-E*E)*SE
      ROACF=CE-E
      F=ANG(ATAN2(ROASF,ROACF))
      RETURN
      END
```

```
      SUBROUTINE  LITRAN  (XL,  X,  DJUL,  DJLNCH,  XLNG,  MODE)

      IMPLICIT REAL*8 (A-H,O-Z)
C     ----------------------------------------------------------
C     TRANSFORMS LI COORDINATES (XL) TO ECID COORDINATES (X) OR
C     VICE VERSA, DEPENDING ON WHETHER MODE>0 OR MODE<0.

C     XL(1)...XL(3), X(1)...X(3) = X,Y,Z VELOCITY COMPONENTS
C     XL(4)...XL(6), X(4)...X(6) = X,Y,Z POSITION COORDINATES
C
C     DJUL   =JULIAN DATE OF TRANSFORMATION
C     DJLNCH =JULIAN DATE AT LAUNCH REFERENCE TIME
C     XLNG   =LAUNCH-SITE LONGITUDE (EAST)
C     PHIREF =EQUATORIAL ANGLE BETWEEN XL(1) AND EQUINOX AT DJLNCH
C     PHI    =  "        "      "      "     "     "     "  DJUL
C
C     NOTE: THE DIFFERENCE BETWEEN PHI AND PHIREF WILL BE VERY SMALL


C     --------------------------------------------------------------
      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
C     --------------------------------------------------------------
      DIMENSION XL(6), X(6)

      DATA XLNGSV/1.E9/,  DJLNSV/1.E9/


      IF(DJLNCH .NE. DJLNSV  .OR.  XLNG .NE. XLNGSV) THEN
        PHIREF=GHANG(DJLNCH) +XLNG
        DJLNSV=DJLNCH
        XLNGSV=XLNG
      END IF

      PHI=PHIREF +PEQD*(DJUL-DJLNCH)
      SPHI=SIN(PHI)
      CPHI=COS(PHI)

      IF(MODE.GT.0) THEN
        X(1)= XL(1)*CPHI -XL(2)*SPHI
        X(2)= XL(1)*SPHI +XL(2)*CPHI
        X(3)= XL(3)
        X(4)= XL(4)*CPHI -XL(5)*SPHI
        X(5)= XL(4)*SPHI +XL(5)*CPHI
        X(6)= XL(6)
      ELSE
        XL(1)= X(1)*CPHI +X(2)*SPHI
        XL(2)=-X(1)*SPHI +X(2)*CPHI
        XL(3)= X(3)
        XL(4)= X(4)*CPHI +X(5)*SPHI
        XL(5)=-X(4)*SPHI +X(5)*CPHI
        XL(6)= X(6)
      END IF

      RETURN
```

```
      END


      SUBROUTINE MOTRAN (EM, EO, MODE)

      IMPLICIT REAL*8 (A-H,O-Z)
C     -----------------------------------------------------------
C     TRANSFORMS MEAN ELEMENTS (EM) TO OSCULATING ELEMENTS (EO),
C     OR VICE VERSA, DEPENDING ON WHETHER MODE>0 OR MODE<0.

C     ******* SEVERAL SUBROUTINES ARE CONTAINED IN THIS FILE *****

C     EM(1), EO(1) = SEMILATUS RECTUM
C              (2) = ECCENTRICITY
C              (3) = TRUE ANOMALY
C              (4) = ARGUMENT OF PERIGEE
C              (5) = INCLINATION
C              (6) = RIGHT ASCENSION

C     EQ(I) - ELEMENT SET CONTAINING SEMIMAJOR AXIS AND MEAN ANOMALY
C             INSTEAD OF SEMILATUS RECTUM AND TRUE ANOMALY

C     Y(I)    = MEAN EQUINOCTIAL ELEMENT SET
C     SP(I)   = SHORT-PERIOD EQUINOCTIAL PERTURBATIONS
C     OSCY(I)= OSCULATING EQUINOCTIAL ELEMENT SET

C     -------------------------------------------
      COMMON/IMA06/PI,TWOPI,PIO2
C     -------------------------------------------
      DIMENSION EM(6), EO(6), EQ(6), Y(6), SP(6), OSCY(6)
      DATA KERR/0/


      IF(MODE.GT.0) THEN

        IF(EM(2).GT..97) THEN
          DO 2 I=1,6
    2     EO(I)=EM(I)

          IF(KERR.EQ.0) THEN
            CALL MSG('*** WARNING: HYPERBOLIC CONIC',1)
            KERR=1
          END IF

          RETURN
        END IF

        IFLAG=3
        IF(EM(5).GE.PIO2) IFLAG=4
        DO 4 I=3,6
    4   EM(I)=ANG(EM(I))
        CALL SAMEQ (EM,EQ)
        CALL TRAN (EQ,Y,1,IFLAG)
```

```
      CALL KEPLE(EQ(2),EQ(6),XE,F)
      CALL OSCMN(EQ,SP,XE,F,IFLAG)
      DO 5 I=1,6
    5 OSCY(I)=Y(I)+SP(I)
      CALL TRAN(EQ,OSCY,0,IFLAG)
      CALL EQSAM (EQ,EO)

    ELSE

      IF(EO(2).GT..97) THEN
        DO 8 I=1,6
    8   EM(I)=EO(I)

        IF(KERR.EQ.0) THEN
          CALL MSG('*** WARNING: HYPERBOLIC CONIC',1)
          KERR=1
        END IF

        RETURN
      END IF

      IFLAG=3
      IF(EO(5).GE.PIO2) IFLAG=4
      DO 10 I=3,6
   10 EO(I)=ANG(EO(I))
      CALL SAMEQ (EO,EQ)
      CALL TRAN(EQ,Y,1,IFLAG)
      DO 70 K=1,4
      CALL KEPLE(EQ(2),EQ(6),XE,F)
      CALL OSCMN(EQ,SP,XE,F,IFLAG)
      DO 60 I=1,6
   60 OSCY(I)=Y(I)-SP(I)
   70 CALL TRAN(EQ,OSCY,0,IFLAG)
      CALL EQSAM (EQ,EM)

    END IF

    RETURN
    END



    SUBROUTINE      ORBMOVE(SLR,ECC,SINI,COSI,RAD,APD,XMD)
    IMPLICIT REAL*8 (A-H,O-Z)

C   THIS VERSION COMPUTES SECULAR RATES DUE TO THE J2 AND J4 SPHERICAL
C   HARMONIC COEFFICIENTS.

C   INPUTS:
C   SLR            MEAN SEMILATUS RECTUM
C   ECC            MEAN ECCENTRICITY
C   SINI, COSI     SINE, COSINE OF MEAN INCLINATION
```

```
C     OUTPUTS:
C     RAD        RIGHT ASCENSION MEAN RATE (RAD/DAY)
C     APD        ARGUMENT OF PERIGEE MEAN RATE (RAD/DAY)
C     XMD        MEAN ANOMALY MEAN RATE (RAD/DAY)
C     -------------------------------------------------------
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL

      IF(ECC.GT..99)THEN
        CALL MSG('HALT: ECCENTRICITY .GT. .99 IN IMA_ORBMOVE',3)
        STOP
      END IF


      A=SLR/(1.-ECC**2)
      XJ2SQ=XJ2**2

      XN = (XMU / A) ** 0.5 / A
      RDP = REQ / SLR
      RDPSQ = RDP * RDP
      RDPCU = RDPSQ * RDP
      RDPQU = RDPCU * RDP
      SINSQI = SINI * SINI
      SINQUI = SINSQI * SINSQI
      ESQ = ECC * ECC
      SQR1MESQ = (1.0 - ESQ) ** 0.5
      XNJ4T = XN * XJ4 * RDPQU
      XNJ2SQT = XN * XJ2SQ * RDPQU
      XNJ2T = XN * XJ2 * RDPSQ


      RAD = -1.5 * XNJ2T * COSI - 1.5 * XNJ2SQT
     1         * COSI * (2.25 + 1.5 * SQR1MESQ - SINSQI *
     2         (2.5 + 2.25 * SQR1MESQ) + 0.25 * ESQ * (1.0 + 1.25 *
     3         SINSQI)) + .9375 * XNJ4T * COSI *
     4         (4. - 7. * SINSQI)  * (1. + 1.5 * ESQ)


      APD = 0.75 * XNJ2T * (4.0 - 5.0 * SINSQI) + 0.1875 *
     1     XNJ2SQT * (48.0 - 103.0 * SINSQI + 53.75 *
     2     SINQUI + (7.0 - 4.5 * SINSQI - 5.625 * SINQUI) * ESQ + 6.0 *
     3     (1.0 - 1.5 * SINSQI) * (4.0 - 5.0 * SINSQI) * SQR1MESQ) -
     4     .46875 * XNJ4T * (16. -62. * SINSQI + 49. *
     5     SINQUI + .75 * (24. - 84. * SINSQI + 63. * SINQUI ) * ESQ )


      XMD = XN * (1.0 + 1.5 * XJ2 * RDPSQ * (1.0 - 1.5 * SINSQI) *
     1     SQR1MESQ) + 1.5 * XNJ2SQT * ((1.0 - 1.5 *
     2     SINSQI) ** 2.0 * (1.0 - ESQ) + (1.25 * (1.0 - 2.5 *
     3     SINSQI + 1.625 * SINQUI) + 0.625 * (1.0 - SINSQI - 0.625 *
     4     SINQUI) * ESQ) * SQR1MESQ) + 1.125 *
     5     XNJ2SQT / SQR1MESQ * (3.0 - 7.5 * SINSQI +
     6     5.875 * SINQUI + (1.5 - 5.0 * SINSQI +7.3125 *SINQUI)*ESQ -
     7     0.125 * (1.0 + 5.0 * SINSQI - 12.625 * SINQUI) * ESQ * ESQ)
     8     -.3515625 * XNJ4T * (8. -40.*SINSQI +35.* SINQUI)
```

```
      9     * ESQ * SQR1MESQ

       APD=APD*86400.
       RAD=RAD*86400.
       XMD=XMD*86400.

       RETURN

       END


       SUBROUTINE     OSCMN(Y,YSP,XE,F,IFLAG)
       IMPLICIT REAL*8 (A-H,O-Z)
C      ---------------------------------------------------------
C      COMPUTES SHORT-PERIOD PERTURBATIONS (YSP) OF EQUINOCTIAL
C      ELEMENTS (Y)
C
C      XE=ECCENTRIC ANOMALY
C      F =TRUE ANOMALY
C      -------------------------------------------
       COMMON /IMA04/ XMU,XJ2,XJ3,XJ4,REQ,RPL
C      -------------------------------------------
       DIMENSION X(6),SP(6),Y(6),YSP(6)
       EQUIVALENCE (A,X(1)),(E,X(2)),(XI,X(3)),(XZ,X(4)),(XW,X(5)),
      1            (XM,X(6))

       DO 10 I=1,6
    10 X(I)=Y(I)
       SINF=SIN(F)
       COSE=COS(XE)
       SINI=SIN(XI)
       COSI=COS(XI)
       SINI2=SINI*SINI
       SI32=1.0-1.5*SINI2
       E2=E*E
       EPM=(1.0+E)*(1.0-E)
       SQE1=SQRT(EPM)
       IF(E.GT.0.01) GO TO 20
       FESP=3.0*E-0.375*E*E2
       FZSP=0.5*E
       GO TO 22
    20 FESP=(1.+1.5*E2-SQE1**3)/E
       FZSP=(1.-SQE1)/E
    22 CONTINUE
       PZ=A*EPM
       TEMP1=1.0-E*COSE
       R=A*TEMP1
       C2W2F=COS(2.*(XW+F))
       ASP=XJ2*REQ*REQ*((A/R)**3*(SI32+1.5*SINI2*C2W2F)-(SI32/SQE1**3))/A
       COEF=XJ2*(REQ/PZ)**2
       SINW=SIN(XW)
       COSW=COS(XW)
```

```
      C2WF=COS(2.*XW+F)
      C2W3F=COS(2.*XW+3.*F)
      ESP=0.5*COEF*SI32*(FESP+(3.0+0.75*E2)*COS(F)
     *     +1.5*E*COS(2.*F)+0.25*E2*COS(3.*F))
     *     +0.375*COEF*SINI2*((1.0+2.75*E2)*C2WF+E2*COS(2.*XW-F)*0.25
     *     +5.0*E*C2W2F+(2.3333333+1.4166667*E2)*C2W3F+1.5*E*COS(2.*XW
     *     +4.*F)+0.25*E2*COS(2.*XW+5.*F)+1.5*E*COS(2.*XW))
      XISP=0.75*COEF*SINI*COSI*(E*C2WF+C2W2F+E*C2W3F/3.0)
      TEMP2=F-ANG(XM)+E*SINF
      S2WF=SIN(2.*XW+F)
      S2W2F=SIN(2.*(XW+F))
      S2W3F=SIN(2.*XW+3.*F)
      S2WMF=SIN(2.*XW-F)
      XOSP=-1.5*COEF*COSI*(TEMP2-0.5*(E*S2WF+S2W2F)-E*S2W3F/6.)
      S2W4F=SIN(2.*XW+4.*F)
      S2W5F=SIN(2.*XW+5.*F)
      S2W=SIN(2.*XW)
      SIN2F=SIN(2.*F)
      D1=4.-5.*SINI2
      D2=1.-SQE1
      D3=FZSP*(1.-.25*E2)*SINF+D2*(.5*SIN2F+E*SIN(3.*F)/12.)
      ETEMP=SI32*((1.-.25*E2)*SINF+.5*SIN2F*E+E2*SIN(3.*F)/12.)
      EWSP=COEF*((3.-3.75*SINI2)*TEMP2*E+1.5*ETEMP-1.5*((.25*SINI2
     1   +E2*(.5-.9375*SINI2))*S2WF+E2*SINI2*S2WMF/16.
     2   +(.5-1.25*SINI2)*S2W2F*E-(.58333333*SINI2-E2*(.16666667
     3   -.39583333*SINI2))*S2W3F-.375*SINI2*S2W4F*E
     4   -.0625*E2*SINI2*S2W5F)-.5625*SINI2*S2W*E)
      SINXO=SIN(XZ)
      COSXO=COS(XZ)
      T1=1.0+COSI
      IF(IFLAG.EQ.3) GO TO 7
      WMO=XW-XZ
      SINWMO=SIN(WMO)
      COSWMO=COS(WMO)
      T2=T1/SINI
      EWOSP=EWSP-E*XOSP
      XSP=ESP*COSWMO-EWOSP*SINWMO
      YYSP=ESP*SINWMO+EWOSP*COSWMO
      T3=-0.5/(SIN(0.5*XI)**2)
      GO TO 8
    7 CONTINUE
      OPW=XZ+XW
      SINOPW=SIN(OPW)
      COSOPW=COS(OPW)
      T2=SINI/T1
      EOWSP=E*XOSP+EWSP
      XSP=ESP*COSOPW-EOWSP*SINOPW
      YYSP=ESP*SINOPW+EOWSP*COSOPW
      T3=0.5/(COS(0.5*XI)**2)
    8 CONTINUE
      T3ISP=T3*XISP
      T2OSP=T2*XOSP
      SP(3)=T3ISP*COSXO-T2OSP*SINXO
      SP(4)=T3ISP*SINXO+T2OSP*COSXO
```

```
    4 CONTINUE
      ZSP=COEF*(0.75*D1*TEMP2+1.5*SI32*D3
     1     -1.5*FZSP*SINI2*(0.25*S2WF-.58333333*S2W3F)
     2     -0.75*E*(1.0-0.625*SINI2*(3.0+SQE1))*S2WF
     3     -0.1875*SINI2*D2*(0.5*E*S2WMF-3.0*S2W4F-0.5*E*S2W5F)
     4    -0.75*(1.0-2.5*SINI2) *S2W2F
     5     -0.25*E*(1.0-0.125*SINI2*(19.0+SQE1))*S2W3F
     6     -0.5625*D2*SINI2*S2W)
      IF (IFLAG.EQ.3) GO TO 9
      ZSP=ZSP-XOSP
      GO TO 5
    9 CONTINUE
      ZSP=ZSP+XOSP
    5 CONTINUE
      SP(1)=ASP
      SP(2)=XSP
      SP(5)=YYSP
      SP(6)=ZSP
    2 CONTINUE
      DO 11 I=1,6
   11 YSP(I)=SP(I)
      RETURN
      END
```

```
      SUBROUTINE      RVAT(R1,R2,NB,SI,CI,R,VA,VD,TI,AS,TT)
      IMPLICIT REAL*8 (A-H,O-Z)
C     FOR TRANSFERS CONSISTING OF MORE THAN TWO IMPULSES, THE OUTPUTS
C     ARE BASED ON A SEQUENCE OF IMPULSES FOR WHICH THE DIFFERENCES IN
C     ARRIVAL AND DEPARTURE VELOCITY MAGNITUDES AT THE ODD-NUMBERED
C     IMPULSE POINTS ARE THE SAME, AND THE DIFFERENCES AT THE EVEN-
C     NUMBERED IMPULSE POINTS ARE THE SAME (THE LAST TWO IMPULSES
C     WILL NOT FOLLOW THIS RULE EXACTLY).  THEN, IF EACH IMPULSE
C     ACCOMPLISHES THE SAME AMOUNT OF PLANE CHANGE, THE ODD-NUMBERED
C     DELTA-VELOCITIES WILL ALL BE NEARLY EQUAL, AND THE EVEN-NUMBERED
C     DELTA-VELOCITIES WILL ALL BE NEARLY EQUAL.
C     INPUTS:--------------------------------------------------
C     R1         RADIUS OF STARTING MEAN ORBIT
C     R2         RADIUS OF ENDING MEAN ORBIT
C     NB         NUMBER OF IMPULSES IN THE TRANSFER (.LE. 12)
C     SI(J)      SINES OF INCLINATIONS OF STARTING AND ENDING ORBITS
C     CI(J)      COSINES OF INCLINATIONS OF STARTING AND ENDING ORBITS
C     OUTPUTS:-------------------------------------------------
C     R(I)       RADIUS AT POINT OF EACH IMPULSE IN THE TRANSFER
C     VA(I)      ARRIVAL VELOCITY MAGNITUDE FOR EACH IMPULSE
C     VD(I)      DEPARTURE VELOCITY MAGNITUDE FOR EACH IMPULSE
C     TI(I)      TIME BETWEEN IMPULSE I AND IMPULSE I+1
C     AS         CHANGE IN RIGHT ASCENSION DURING THE TRANSFER
C     TT         TOTAL TIME REQUIRED FOR THE TRANSFER (*** DAYS ***)
C     ---------------------------------------------------------
```

```
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL

      DIMENSION R(12), VA(12), VD(12), TI(11), SI(2), CI(2)

      SMA12=.5*(R1 +R2)
      VA(1)=SQRT(XMU/R1)
      VD(1)=SQRT(XMU*(2./R1 -1./SMA12))
      VA(NB)=SQRT(XMU*(2./R2 -1./SMA12))
      VD(NB)=SQRT(XMU/R2)

      R(1)=R1
      R(NB)=R2

      IF(NB.EQ.2) GO TO 20

      NB1=(NB+1)/2
      NB2=NB/2

      DV1=(VD(1)-VA(1))/NB1
      DV2=(VD(NB)-VA(NB))/NB2
      NBM1=NB-1

      DO 10 I=2,NBM1
      DVC=DV2
      IF(MOD(I,2).EQ.0)DVC=DV1
      VD(I-1)=VA(I-1) +DVC
      R(I)=R(I-1)/(2.*XMU/(R(I-1)*VD(I-1)**2) -1.)
   10 VA(I)=SQRT(XMU*(2./R(I) -2./(R(I)+R(I-1))))

      VD(NBM1)=SQRT(XMU*(2./R(NBM1) -2./(R(NBM1)+R(NB))))
      VA(NB)   =SQRT(XMU*(2./R(NB) -2./(R(NBM1)+R(NB))))

   20 CALL ASCTIM(NB,SI,CI,R,TI,AS,TT)

      RETURN
      END


      SUBROUTINE   SAMEQ   (EL,EQ)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ----------------------------------------------------------------
C     CONVERTS EL ELEMENT SET TO EQ ELEMENT SET
C
C     EL(1)=SEMILATUS RECTUM               EQ(1)=SEMIMAJOR AXIS
C     EL(2)=ECCENTRICITY                   EQ(2)=ECCENTRICITY
C     EL(3)=TRUE ANOMALY                   EQ(3)=INCLINATION
C     EL(4)=ARGUMENT OF PERIGEE            EQ(4)=RIGHT ASCENSION
C     EL(5)=INCLINATION                    EQ(5)=ARGUMENT OF PERIGEE
C     EL(6)=RIGHT ASCENSION                EQ(6)=MEAN ANOMALY
C     ----------------------------------------------------------------
      DIMENSION EL(6), EQ(6)
      EQ(1)=EL(1)/(1.-EL(2)*EL(2))
```

```
      EQ(2)=EL(2)
      EQ(3)=EL(5)
      EQ(4)=EL(6)
      EQ(5)=EL(4)
      EQ(6)=XMANOM(EL(2),EL(3))
      RETURN
      END


      SUBROUTINE   TGAUSS(R1,R2,SANG,CANG,F1,TG,SLR,ECC,CF1,CF2,SF1,SF2,
     &KGERR)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ****************************************
C     NOTE: TG IS OUTPUT IN UNITS OF DAYS
C     ****************************************
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA40/RPMIN

      KGERR=0

      CF1=COS(F1)
      SF1=SIN(F1)
      CF2=CF1*CANG-SF1*SANG
      SF2=SF1*CANG+CF1*SANG

      D=R2*CF2-R1*CF1

      ECC=(R1-R2)/D

      IF(ECC.GT..99)THEN
        KGERR=-2
        RETURN
      END IF

      SLR=R1*R2*(CF2-CF1)/D

      IF(SLR.LT.RPMIN)THEN
        KGERR=-3
        RETURN
      END IF

      TEMP=1.-ECC*ECC
      SMA=SLR/TEMP
      SRE=SQRT(TEMP)
      SMASRE=SMA*SRE
      FAC=SQRT(SMA**3/XMU)

      SE1=SF1*R1/SMASRE
      SE2=SF2*R2/SMASRE
      CE1=(CF1 +ECC)/(1.+ECC*CF1)
      CE2=(CF2 +ECC)/(1.+ECC*CF2)
```

```
      E1=ATAN2(SE1,CE1)
      E2=ATAN2(SE2,CE2)

      DELE=DMOD(E2-E1,TWOPI)
      IF(DELE.LT.0.)DELE=DELE+TWOPI

      TG=FAC*(DELE -ECC*(SE2-SE1))/86400.

      RETURN
      END



      SUBROUTINE     TRAN(X,Y,KK,IFLAG)
      IMPLICIT REAL*8 (A-H,O-Z)
C     -------------------------------------------------------------
C     CONVERTS EQ ELEMENT SET (X) TO EQUINOCTIAL ELEMENT SET (Y) OR
C     VICE VERSA, DEPENDING ON WHETHER KK.GT.0 OR KK=0
C
C     Y(1)=SEMIMAJOR AXIS
C     Y(2)=ECCENTRICITY * COS(ANGSUM)
C     Y(3)=TANCOT(INCLINATION/2) * COS(RIGHT ASCENSION)
C     Y(4)=TANCOT(INCLINATION/2) * SIN(RIGHT ASCENSION)
C     Y(5)=ECCENTRICITY * SIN(ANGSUM)
C     Y(6)=ANGSUM + MEAN ANOMALY
C
C     IF INCLINATION .LT. PIO2 (IFLAG=3)
C         TANCOT=TANGENT FUNCTION
C         ANGSUM=ARGUMENT OF PERIGEE + RIGHT ASCENSION
C
C     IF INCLINATION .GE. PIO2 (IFLAG=4)
C         TANCOT=COTANGENT FUNCTION
C         ANGSUM=ARGUMENT OF PERIGEE - RIGHT ASCENSION
C     -------------------------------------------------------------
      DIMENSION X(6),Y(6)
      K1=IFLAG-2
      IF(KK.NE.0) GO TO 10
      X(1)=Y(1)
      X(2)=SQRT(Y(2)*Y(2)+Y(5)*Y(5))
    1 A=SQRT(Y(3)*Y(3)+Y(4)*Y(4))
      IF(IFLAG.NE.4) GO TO 11
      X(3)=2.0*ATAN(1.0/A)
      X(3)=ANG(X(3))
      GO TO 12
   11 CONTINUE
      X(3)=2.0*ATAN(A)
      X(3)=ANG(X(3))
   12 CONTINUE
      X(4)=0.0
      IF(ABS(Y(3)).LE..1E-05.AND.ABS(Y(4)).LE..1E-05) GO TO 3
      X(4)=ATAN2(Y(4),Y(3))
    3 CONTINUE
      X(4)=ANG(X(4))
```

```
      X(5)=0.0
      B=X(4)
      IF(ABS(Y(2)).LE..1E-05.AND.ABS(Y(5)).LE..1E-05) GO TO 4
      B=ATAN2(Y(5),Y(2))
      B=ANG(B)
      X(5)=ANG(B-X(4))
      IF(IFLAG.EQ.4) X(5)=ANG(B+X(4))
    4 CONTINUE
      X(6)=ANG(Y(6)-B)
      GO TO 100
   10 Y(1)=X(1)
    2 XHI=0.5*X(3)
      IF(IFLAG.NE.4) GO TO 14
      XWMO=X(5)-X(4)
      Y(2)=X(2)*COS(XWMO)
      C=COS(XHI)/SIN(XHI)
      Y(3)=C*COS(X(4))
      Y(4)=C*SIN(X(4))
      Y(5)=X(2)*SIN(XWMO)
      Y(6)=ANG(X(5)-X(4)+X(6))
      GO TO 100
   14 CONTINUE
      XOPW=X(4)+X(5)
      Y(2)=X(2)*COS(XOPW)
      C=SIN(XHI)/COS(XHI)
      Y(3)=C*COS(X(4))
      Y(4)=C*SIN(X(4))
      Y(5)=X(2)*SIN(XOPW)
      Y(6)=ANG(X(4)+X(5)+X(6))
  100 CONTINUE
      RETURN
      END


      FUNCTION   XMANOM   (ECC,TAN)

      IMPLICIT REAL*8 (A-H,O-Z)
C     ---------------------------------------------------
C     COMPUTES THE MEAN ANOMALY FROM ECCENTRICITY (ECC) AND
C     TRUE ANOMALY (TAN).
C     -------------------

      IF(ECC.GT..99)THEN
        CALL MSG('HALT: ECCENTRICITY .GT. .99 IN FUNCTION XMANOM',3)
        STOP
      END IF

      SF=SIN(TAN)
      CF=COS(TAN)
      COSE=(ECC+CF)/(ECC*CF +1.)
      SINE=(1.-ECC*COSE)*SF/SQRT(1.-ECC*ECC)
      EANOM=ANG(ATAN2(SINE,COSE))
      XMANOM=EANOM -ECC*SIN(EANOM)
```

```
RETURN
END
```

```
      SUBROUTINE    FINISH    (ITSTART,TSTART,BT0,PROP,BCOST,FTBL)
      IMPLICIT REAL*8 (A-H,O-Z)
      logical lorb,lobj
      COMMON/IMA13/NSEG,IESEG(15),IBSEG(15),IBLEG(7),KSEG(7)
      COMMON/IMA15/TMAXL,PMIN(7),LORB(12)
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA28/LOBJ,TMAX,LTMAX,WFACT(6)
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      DIMENSION BT(128),T(128),PMASS(128,6),PROP(6),PSMASS(6),PEMASS(6),
     1          XT(128),XBT(128),XPMASS(128,6),RA(128),XRA(128),
     2          IORG(128),IIORG(128),SAVE(3)
      BT(1) = BT0
      T(1) = TSTART
      XRA(1) = OEM(NTORB(IESEG(ITSTART-2)),6)
      DO I = 1,NPROP
         PMASS(1,I) = PROP(I)
      END DO
      NSTART = 1
      DO I = ITSTART,NSEG+1
         NSTOP = 1
         DO J = 1,NSTART
         IG = 0
            DO K = 1,NPROP
               PSMASS(K) = PMASS(J,K)
            END DO
            IF (KTRAN(IESEG(I-1)) .EQ. 5) THEN
               CALL SW01 (1,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                    XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
            ELSE IF (KTRAN(IESEG(I-1)) .EQ. 4) THEN
               IF (LORB(NTORB(IESEG(I-1)))) THEN
                  CALL SW02 (1,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                       XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
               ELSE
                  CALL SW02 (-1,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                       XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
               END IF
            ELSE
               CALL SW03 (1,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                    XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
            END IF
            WRITE(20,999) I,1,IBSEG(I-1),T(J),PSMASS(1),PSMASS(2),
     1                  XT(NSTOP),OEM(NTORB(IESEG(I-1)),6),
     2                  PEMASS(1),PEMASS(2),KERR
 999        FORMAT(' ',I1,1X,I2,1X,I1,1X,F15.7,1X,F11.5,1X,F11.5,1X,
     1             F15.7,1X,F15.7,1X,F11.5,1X,F11.5,1X,I1)
            IF (KERR .EQ. 0 .AND. XT(NSTOP) .LT. TMAXL) THEN
               DO K = 1,NPROP
                  XPMASS(NSTOP,K) = PEMASS(K)
               END DO
               XRA(NSTOP) = OEM(NTORB(IESEG(I-1)),6)
               IF (I .NE. ISTART) IIORG(NSTOP) = IORG(J)
               NSTOP = NSTOP + 1
```

```
            IG = IG + 1
         END IF
C
C GET MINIMUM TIME
C
         IF (KTRAN(IESEG(I-1)) .EQ. 5) THEN
            CALL SW01 (0,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                 XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
         ELSE IF (KTRAN(IESEG(I-1)) .EQ. 4) THEN
            IF (LORB(NTORB(IESEG(I-1)))) THEN
               CALL SW02 (0,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
     1                    XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
            ELSE
               GOTO 10
            END IF
         ELSE
            GOTO 10
         END IF
         WRITE(20,999) I,0,IBSEG(I-1),T(J),PSMASS(1),PSMASS(2),
     1                 XT(NSTOP),OEM(NTORB(IESEG(I-1)),6),
     2                 PEMASS(1),PEMASS(2),KERR
C         IJ = 1
C 99      IF (KTRAN(IESEG(I-1)) .EQ. 5) THEN
C            CALL SW01 (2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
C    1                  XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
C         ELSE
C             CALL SW02 (2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
C    1                   XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
C         END IF
C         WRITE(20,999) I,2,IBSEG(I-1),T(J),PSMASS(1),PSMASS(2),
C    1                  XT(NSTOP),OEM(NTORB(IESEG(I-1)),6),
C    2                  PEMASS(1),PEMASS(2),KERR
C
C         IF (KERR .NE. 0) THEN
C            IF (IJ .LE. 10) THEN
C               XT(NSTOP) = XT(NSTOP) + .01
C            ELSE
C               XT(NSTOP) = XT(NSTOP) + .1
C            END IF
C            IJ = IJ + 1
C            IF (IJ .EQ. 21) GOTO 10
C            GOTO 99
C         END IF
         IF (KERR .EQ. 0 .AND. XT(NSTOP) .LT. TMAXL) THEN
            DO K = 1,NPROP
               XPMASS(NSTOP,K) = PEMASS(K)
            END DO
            XRA(NSTOP) = OEM(NTORB(IESEG(I-1)),6)
            IF (I .NE. ISTART) IIORG(NSTOP) = IORG(J)
            NSTOP = NSTOP + 1
            IG = IG + 1
         END IF
 10      CONTINUE
C
```

```
         IF (IG .EQ. 2) THEN
            XT(NSTOP) = 0.5 * (XT(NSTOP-2) + XT(NSTOP-1))

         IF (KTRAN(IESEG(I-1)) .EQ. 5) THEN
            CALL SW01 (2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
   1                   XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
         ELSE IF (KTRAN(IESEG(I-1)) .EQ. 4) THEN
            IF (LORB(NTORB(IESEG(I-1)))) THEN
               CALL SW02 (2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
   1                      XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
            ELSE
               CALL SW02 (-2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
   1                      XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
            END IF
         ELSE
            CALL SW03 (2,IBSEG(I-1),IBLEG(I-1),BT(J),T(J),PSMASS,
   1                   XT(NSTOP),PEMASS,IELEG,XBT(NSTOP),KERR)
         END IF
         WRITE(20,999) I,2,IBSEG(I-1),T(J),PSMASS(1),PSMASS(2),
   1                  XT(NSTOP),OEM(NTORB(IESEG(I-1)),6),
   2                  PEMASS(1),PEMASS(2),KERR
         IF (KERR .EQ. 0) THEN
            DO K = 1,NPROP
               XPMASS(NSTOP,K) = PEMASS(K)
            END DO
            XRA(NSTOP) = OEM(NTORB(IESEG(I-1)),6)
            IF (I .NE. ISTART) IIORG(NSTOP) = IORG(J)
            NSTOP = NSTOP + 1
         END IF
         END IF
      END DO
      IBLEG(I) = IELEG + 1
      NSTART = NSTOP - 1
      DO J = 1,NSTART
         T(J) = XT(J)
         BT(J) = XBT(J)
         RA(J) = XRA(J)
         DO K = 1,NPROP
            PMASS(J,K) = XPMASS(J,K)
         END DO
      END DO
      IF (I .EQ. ITSTART) THEN
         DO J = 1,NSTART
            IORG(J) = J
            SAVE(J) = T(J)
         END DO
      ELSE
         DO J = 1,NSTART
            IORG(J) = IIORG(J)
         END DO
      END IF
   END DO
   BCOST = 9.99E9
   IF (NSTART .NE. 0) THEN
```

```
      IF (LOBJ) THEN
         DO I = 1,NSTART
            COST = 0.0
            DO J = 1,NPROP
               COST = COST + (PCAP(J) * FILL(J) - PMASS(I,J)) /
     1                  PCAP(J) * WFACT(J)
            END DO
            IF (COST .LT. BCOST) THEN
               BCOST = COST
               ISAVE = I
            END IF
         END DO
      ELSE
         DO I = 1,NSTART
            COST = T(I)
            DO J = 1,NPROP
               IF (PMASS(I,J) .LT. PMIN(J)) THEN
                  COST = 9.99E9
               END IF
            END DO
            IF (COST .LT. BCOST) THEN
               ISAVE = I
               BCOST = COST
            END IF
         END DO
      END IF
      END IF
      FTBL = SAVE(IORG(ISAVE))
      RETURN
      END
```

SUBROUTINE   ORBCID(IORB,T,XCID)

```
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES THE ECID POSITION COORDINATES (XCID) OF A SATELLITE
C     AT JULIAN DATE T IN ORBIT NO. IORB
C     -----------------------------------------------------------
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)

      DIMENSION E(6), XCID(3), UP(3), UN(3), UH(3)

      E(1)=OEM(IORB,1)
      E(2)=OEM(IORB,2)
      E(5)=OEM(IORB,5)

      DELDJ=T-ODJ(IORB)
      E(3)=OEM(IORB,3)  +DMADT(IORB)*DELDJ
      E(4)=OEM(IORB,4)  +DAPDT(IORB)*DELDJ
      E(6)=OEM(IORB,6)  +DRADT(IORB)*DELDJ
```

```
      CALL KEPLE(E(2),E(3),DUM,E(3))

      RM=E(1)/(E(2)*COS(E(3)) +1.)
      SF=SIN(E(3))
      CF=COS(E(3))
      CALL CONREF(E(4),UP,UN,UH)

      DO 10 I=1,3
   10 XCID(I)=RM*(CF*UP(I) +SF*UN(I))

      RETURN
      END
```

```
      SUBROUTINE      SIMPLX(A,KL,W,II,III,JJ,AMAX,KUB)
      implicit real*8 (a-h,o-z)

      DIMENSION A(20,26),KL(20),W(20)
      DATA FTOL/1.E-8/
C
      KUB=0
      KKK=0
      I=1
C
   10 I=I+1
      IF(I-III) 20,50,50
   20 IF(KL(I)) 10,30,10
C
   30 DO 40 J=1,JJ
      IF(A(I,J).EQ.0.) GO TO 40
      A(III,J)=A(III,J)-A(I,J)
   40 CONTINUE
C
      GO TO 10
C
   50 K=III
   60 J=0
      W(K)=0.
      KL(K)=0
C
   70 J=J+1
      IF(J-JJ) 80,90,90
   80 IF((A(K,J).GE.0.).OR.((K.EQ.III).AND.(A(K,J).GT.(-FTOL))))
     +  GO TO 70
      IF(W(K).LE.A(K,J)) GO TO 70
      W(K)=A(K,J)
      KL(K)=J
      GO TO 70
C
   90 IF(KL(K)) 100,220,100
  100 KJ=KL(K)
```

```
      DO 110 I=2,II
      IF(A(I,KJ).GT.0.) GO TO 120
  110 CONTINUE
      KUB=1
      RETURN
C
  120 I=1
      JK=0
  130 I=I+1
      IF(I-II) 140,140,170
  140 IF(A(I,KJ).LE.0.) GO TO 130
      X=A(I,JJ)/A(I,KJ)
      IF(JK) 150,160,150
  150 IF(X.GE.XMIN) GO TO 130
  160 XMIN=X
      JK=I
      GO TO 130
C
  170 X=A(JK,KJ)
      KL(JK)=KJ
      DO 180 I=1,III
  180 W(I)=A(I,KJ)
      IJ=JK-1
C
      DO 190 I=1,IJ
      DO 190 J=1,JJ
      IF(A(JK,J).EQ.0.) GO TO 190
      IF(W(I).EQ.0.) GO TO 190
      A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
  190 CONTINUE
      IJ=JK+1
C
      DO 200 I=IJ,III
      DO 200 J=1,JJ
      IF(A(JK,J).EQ.0.) GO TO 200
      IF(W(I).EQ.0.) GO TO 200
      A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
  200 CONTINUE
C
      DO 210 J=1,JJ
  210 A(JK,J)=A(JK,J)/X
      KKK=KKK+1
      GO TO 60
C
  220 IF(K-1) 260,260,230
  230 IJ=JJ-1
      AMAX=0.
C
      DO 240 J=1,IJ
      IF(A(K,J).LE.AMAX) GO TO 240
      AMAX=A(K,J)
  240 CONTINUE
C
      IF(AMAX.GT.FTOL) RETURN
```

```
      DO 250 J=1,JJ
  250 A(III,J)=0.
      K=1
      KKK=0
      GO TO 60
C
  260 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE    SIMQ(A,B,N,KS)
        implicit real*8 (a-h,o-z)

C       SOLVES A N*N SET OF SIMULTANEOUS EQUATIONS.  THE INPUT B(I)
C       ARRAY CONTAINS THE RIGHT HAND SIDES.  UPON OUTPUT, B(I) IS
C       CONVERTED TO THE REQUIRED CHANGES IN THE VARIABLES (B(J)).
C       THE INPUT A(I,J) ARRAY IS THE ARRAY OF PARTIAL DERIVITIVES
C       OF THE EQUATION VALUES W.R.T. THE VARIABLES.  IT IS DESTROYED
C       DURING THE MATRIX MANIPULATIONS.  A NON-ZERO VALUE OF KS
C       INDICATES AN ILL-CONDITIONED MATRIX.

      DIMENSION A(1),B(1)
      TOL=0.0
      KS=0
      JJ=-N
      DO 65 J=1,N
      JY=J+1
      JJ=JJ+N+1
      BIGA=0
      IT=JJ-J
      DO 30 I=J,N
      IJ=IT+I
      IF(ABS(BIGA)-ABS(A(IJ))) 20,30,30
20    BIGA=A(IJ)
      IMAX=I
30    CONTINUE
      IF(ABS(BIGA)-TOL) 35,35,40
35    KS=1
      RETURN
40    I1=J+N*(J-2)
      IT=IMAX-J
      DO 50 K=J,N
      I1=I1+N
      I2=I1+IT
      SAVE=A(I1)
      A(I1)=A(I2)
      A(I2)=SAVE
50    A(I1)=A(I1)/BIGA
      SAVE=B(IMAX)
      B(IMAX)=B(J)
      B(J)=SAVE/BIGA
      IF(J-N) 55,70,55
```

```
55    IQS=N*(J-1)
      DO 65 IX=JY,N
      IXJ=IQS+IX
      IT=J-IX
      DO 60 JX=JY,N
      IXJX=N*(JX-1)+IX
      JJX=IXJX+IT
60    A(IXJX)=A(IXJX)-(A(IXJ)*A(JJX))
65    B(IX)=B(IX)-(B(J)*A(IXJ))
70    NY=N-1
      IT=N*N
      DO 80 J=1,NY
      IA=IT-J
      IB=N-J
      IC=N
      DO 80 K=1,J
      B(IB)=B(IB)-A(IA)*B(IC)
      IA=IA-N
80    IC=IC-1
      RETURN
      END


      SUBROUTINE   SUN  (DJUL,XSUN)
      IMPLICIT REAL*8 (A-H,O-Z)
C     -------------------------------------------------------------
C     COMPUTES XSUN(I) UNIT VECTOR TOWARD SUN, EXPRESSED IN ECID
C     COORDINATES, ON THE INPUT JULIAN DATE (DJUL)
C     -------------------------------------------------------------
      COMMON/IMA02/OBL0,SOBL0,COBL0,OBLD,PEQD,GHA0,GHADI,GHADF,DJUL0
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA08/XLPSUN,ECCSUN,SOBL,COBL,ASUN0,ASUND
C     -------------------------------------------------------------
      DIMENSION XSUN(3)

      A =DMOD((ASUN0 +ASUND*(DJUL-DJUL0)),TWOPI)
      E=A
    1 B=E-ECCSUN*SIN(E)-A
      IF(ABS(B).LT.1.E-5)GO TO 5
      DBDE=1.-ECCSUN*COS(E)
      E=E-B/DBDE
      GO TO 1
    5 TN=SQRT(1.-ECCSUN**2)*SIN(E)
      TD=COS(E)-ECCSUN
      F=ATAN2(TN,TD)
      ANG=XLPSUN+F
      SANG =SIN(ANG)
      CANG =COS(ANG)
      XSUN(1) = CANG
      XSUN(2) = SANG*COBL
      XSUN(3) = SANG*SOBL
      RETURN
      END
```

```
      SUBROUTINE    SUNCOV(T,XCID,LSUN)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LSUN

C     DETERMINES IF A SATELLITE WITH ECID POSITION
C     COORDINATES = XCID(I) IS IN SUNLIGHT (LSUN=.TRUE.)
C     OR SHADOW (LSUN=.FALSE.) AT JULIAN DATE = T.
C     ***************************************************
C     NOTE: THE COMPUTATIONS ASSUME A SPHERICAL EARTH WITH
C     AN EQUATORIAL RADIUS AND A SUN AT INFINITE DISTANCE
C     ***************************************************
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL

      DIMENSION XCID(3), XSUN(3)

      CALL SUN(T,XSUN)
      XXSUN=VDOT(XCID,XSUN)

      IF(XXSUN.GE.0.)THEN
        LSUN=.TRUE.
      ELSE
        LSUN=.FALSE.
        RN2=VDOT(XCID,XCID) -XXSUN*XXSUN -REQ*REQ
        IF(RN2.GT.0.) LSUN=.TRUE.
      END IF

      RETURN
      END


      SUBROUTINE    SUNWIN(IORB,TB,TE,TBE)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LSUN

C     COMPUTES AN ARRAY OF TIMES, TBE(I), THAT DEFINE THE BEGINNINGS
C     (ODD I) AND ENDS (EVEN I) OF SUNLIGHT WINDOWS WITHIN THE INPUT
C     INTERVAL (TB TO TE) ON ORBIT NO. IORB.  ALL TIMES ARE EXPRESSED
C     AS JULIAN DATES.  A VERY LARGE VALUE OF TBE(1) (=9.E9) INDICATES
C     THAT THERE IS NO SUNLIGHT COVERAGE ANYWHERE WITHIN THE INTERVAL.
C     IF THERE IS COVERAGE AT THE END OF THE INTERVAL, TBE(I)=TE FOR
C     I=2 OR 4 OR... DEPENDING OF THE NUMBER OF SUNLIGHT WINDOWS WITHIN
C     THE INTERVAL.
C     *****************************************************************
C     NOTE: THE COMPUTATIONS ASSUME A SPHERICAL EARTH WITH RADIUS EQUAL
C     TO THE EQUATORIAL RADIUS AND A SUN AT INFINITE DISTANCE.  THESE
C     ARE ACCEPTABLE APPROXIMATIONS.
C     *****************************************************************
```

```
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)

      DIMENSION TBE(100), XCID(3)


      DTL=PI/(DMADT(IORB) +DAPDT(IORB))
      DTS=.005555*DTL

      T=TB
      CALL ORBCID(IORB,T,XCID)
      CALL SUNCOV(T,XCID,LSUN)
      T1=T

      IF(LSUN)THEN
        IW=1
        TBE(IW)=TB
        T=T1+DTS
        GO TO 20
      ELSE
        IW=0
        T=T1+DTL
      END IF


C     FIND BEGINNING OF SUNLIGHT WINDOW
C     --------------------------------
   10 IW=IW+1

      IF(IW.GT.99)THEN
        CALL MSG('***HALT: SUN INTERVAL HAS TOO MANY WINDOWS',3)
        STOP
      END IF

   12 IF(T.GT.TE) T=TE
      CALL ORBCID(IORB,T,XCID)
      CALL SUNCOV(T,XCID,LSUN)

      IF(.NOT.LSUN)THEN
        IF(T.EQ.TE)THEN
          TBE(IW)=9.E9
          RETURN
        ELSE
          T1=T
          T=T1+DTL
          GO TO 12
        END IF
      END IF

      T2=T

   14 IF(ABS(T2-T1).GT.1.E-5)THEN
        T=.5*(T1+T2)
        CALL ORBCID(IORB,T,XCID)
```

```
          CALL SUNCOV(T,XCID,LSUN)
          IF(LSUN)T2=T
          IF(.NOT.LSUN)T1=T
          GO TO 14
        END IF

        TBE(IW)=.5*(T1+T2)
        T1=TBE(IW)
        T=T1+DTL


C       FIND END OF SUNLIGHT WINDOW
C       ---------------------------
   20   IW=IW+1

   22   IF(T.GT.TE) T=TE
        CALL ORBCID(IORB,T,XCID)
        CALL SUNCOV(T,XCID,LSUN)

        IF(LSUN)THEN
          IF(T.EQ.TE)THEN
            TBE(IW)=TE
            RETURN
          ELSE
            T1=T
            T=T1+DTS
            GO TO 22
          END IF
        END IF

        T2=T

   24   IF(ABS(T2-T1).GT.1.E-5)THEN
          T=.5*(T1+T2)
          CALL ORBCID(IORB,T,XCID)
          CALL SUNCOV(T,XCID,LSUN)
          IF(LSUN)T1=T
          IF(.NOT.LSUN)T2=T
          GO TO 24
        END IF

        TBE(IW)=.5*(T1+T2)
        T1=TBE(IW)
        T=T1+DTL

        GO TO 10

        END
```

```
      FUNCTION   TANOM   (IORB,DJ)

      IMPLICIT REAL*8 (A-H,O-Z)

C     SPECIALIZED FUNCTION ROUTINE THAT COMPUTES THE TRUE ANOMALY
C     OF ORBIT NO. IORB, GIVEN THE JULIAN DATE (DJ), BASED ON THE
C     SECULAR RATE OF MEAN ANOMALY DUE TO EARTH OBALATENESS.
C     ----------------------------------------------------------
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
C     ----------------------------------------

      ECC=OEM(IORB,2)
      XMA=OEM(IORB,3) +DMADT(IORB)*(DJ-ODJ(IORB))
      CALL KEPLE(ECC,XMA,EA,TA)
      TANOM=TA

      RETURN
      END


      SUBROUTINE    TRKCOV(XF,NT,LCOV)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LCOV(15)

C     TRKCOV DETERMINES IF THERE IS LINE OF SIGHT BETWEEN A POINT
C     HAVING ECF POSTION COORDINATES, XF(J), AND EACH OF NT TRACKING
C     STATIONS (FIXED IN AN EARTH-REFERENCED SYSTEM).
C     **********************************************************
C     NOTE: A NEGATIVE INPUT VALUE FOR NT INDICATES THAT THE CALLING
C     PROGRAM ONLY WANTS TO KNOW IF THERE IS LOS TO ANY ONE OF THE
C     TRACKING STATIONS.  ON AVERAGE, THE SUBROUTINE IS NOT REQUIRED
C     TO CHECK ALL OF THE STATIONS BEFORE FINDING A LOS, AND THE
C     COMPUTER TIME IS REDUCED.  IN THIS CASE, IF A LOS IS DETERMINED
C     FOR ANY STATION, LCOV(1) IS SET TO TRUE.  OTHERWISE, LCOV(1) IS
C     SET TO FALSE.
C     **********************************************************

C     INPUTS:
C     ----------------------------------------------------------
C     XF(J)      X,Y,Z POSITION COORDINATES OF POINT IN ECF SYSTEM

C     NT         NUMBER OF TRACKING STATIONS TO BE CONSIDERED (MAY BE
C                A NEGATIVE VALUE, AS EXPLAINED IN THE NOTE ABOVE)


C     OUTPUTS:
C     ----------------------------------------------------------
C     LCOV(I)    .TRUE. MEANS THERE IS LINE OF SIGHT BETWEEN THE POINT
C                       AND STATION I

C                .FALSE. MEANS THAT THE EARTH BLOCKS THE LINE OF SIGHT
```

```
C     ************************************************************
C     NOTE: FOR THESE COMPUTATIONS, THE EARTH IS CONSIDERED TO BE
C     A SPHERE WITH A RADIUS EQUAL TO THAT AT THE EQUATOR.
C     ************************************************************

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA56/XFTRK(15,3),RADTRK(15)

      DIMENSION XF(3)

      A2=XF(1)**2 +XF(2)**2 +XF(3)**2
      A=SQRT(A2)
      TWOA=2.*A

      NTA=ABS(NT)

      DO 20 I=1,NTA

      B=RADTRK(I)
      B2=B*B
      C2=(XFTRK(I,1)-XF(1))**2 +(XFTRK(I,2)-XF(2))**2
     &  +(XFTRK(I,3)-XF(3))**2
      C=SQRT(C2)

      COSA=(B2 +C2 -A2)/(TWOA*B)
      COSB=(A2 +C2 -B2)/(TWOA*C)

      IF(COSA.LE.0..OR.COSB.LE.0.)THEN
        LCOV(I)=.TRUE.
      ELSE
        LCOV(I)=.FALSE.
        IF(A*SQRT(1.-COSB**2).GT.REQ) LCOV(I)=.TRUE.
      END IF

      IF(NT.LT.0.AND.LCOV(I))THEN
        LCOV(1)=.TRUE.
        RETURN
      END IF

   20 CONTINUE

      RETURN
      END


      SUBROUTINE        TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,KA,KRA,ET,DV1,DV2,KV)
      IMPLICIT REAL*8 (A-H,O-Z)

C     DETERMINES THE TRANSFER CONIC BETWEEN TWO ORBITAL STATES,
C     STARTING AT T1 AND ENDING AT T2, TAKING INTO ACCOUNT THE
C     TRANSFER CONIC'S SECULAR PERTURBATIONS DUE TO EARTH
C     OBLATENESS.  IT IS ASSUMED THAT THE RANGE ANGLE OF THE
C     TRANSFER IS LESS THAN TWO PI RADIANS
```

```
C      INPUTS:
C      ----------------------------------------------------------
C      EPOC1, EPOC2 = EPOCHS FOR STARTING AND ENDING ORBITS
C      E1(I)        = STARTING ORBITAL ELEMENTS AT T=EPOC1
C      E2(I)        = ENDING ORBITAL ELEMENTS AT T=EPOC2
C      T1, T2       = START AND END TIMES FOR THE TRANSFER

C      KA           = INDICATOR FOR ADJUSTMENT (KA=1) OR FOR
C                     NO ADJUSTMENT (KA=0) OF THE TRANSFER CONIC
C                     TO ACCOUNT FOR ITS SECULAR ROTATIONS

C      KRA          = INDICATOR FOR INCLUDING (KRA=1) OR
C                     EXCLUDING (KRA=0) THE SECULAR REGRESSION
C                     IN RIGHT ASCENSION


C      OUTPUTS:
C      ----------------------------------------------------------
C      ET(I)        = TRANSFER CONIC ELEMENTS AT T=T1
C         I=1  SEMILATUS RECTUM      I=4  ARGUMENT OF PERIGEE
C         I=2  ECCENTRICITY          I=5  INCLINATION
C         I=3  MEAN ANOMALY          I=6  RIGHT ASCENSION

C      DV1(I),DV2(I)= ECID COMPONENTS OF DELTA-VELOCITY FOR THE
C                     TRANSFER'S FIRST AND SECOND IMPULSES

C      KV           = 1: TRANSFER WAS SUCCESSFULLY COMPUTED.
C                     0: ADJUSTMENT OF THE TRANSFER WAS NOT SUCCESSFUL.
C                    -1: TRANSFER PERIGEE IS TOO LOW.  NO SOLUTION.
C                    -2: TRANSFER ECCENTRICITY IS TOO HIGH.  NO SOLUTION.
C      ----------------------------------------------------------------

       COMMON/IMA06/PI,TWOPI,PIO2

       DIMENSION E1(6), E2(6), ET(6), E(6)
       DIMENSION X1(6), X2(6), XT1(6), XT2(6)
       DIMENSION H1(3), H2(3), HT(3), HCR(3)
       DIMENSION RCORR(3), DREDV(3,3)
       DIMENSION VTC(2), VTR(2)
       DIMENSION DV1(3), DV2(3)

       DATA DV/.01/, ETOL/1./, MAXIT/15/, NEQ/3/

       KV=1

C      COMPUTE THE CARTESIAN STATE VECTORS OF THE STARTING AND ENDING
C      ORBITS AT T1 AND T2 RESPECTIVELY
C      ---------------------------------
       DO 1 I=1,6
     1 E(I)=E1(I)
       SI=SIN(E(5))
       CI=COS(E(5))
       CALL ORBMOVE(E(1),E(2),SI,CI,RAD,APD,XMD)
```

```
      IF(KRA.EQ.0) RAD=0.
      DT=T1-EPOC1
      E(3)=E(3)+XMD*DT
      CALL KEPLE(E(2),E(3),DUM,E(3))
      E(4)=E(4)+APD*DT
      E(6)=E(6)+RAD*DT
      CALL OITRAN(E,X1,1)

      DO 2 I=1,6
    2 E(I)=E2(I)
      SI=SIN(E(5))
      CI=COS(E(5))
      CALL ORBMOVE(E(1),E(2),SI,CI,RAD,APD,XMD)
      IF(KRA.EQ.0) RAD=0.
      DT=T2-EPOC2
      E(3)=E(3)+XMD*DT
      CALL KEPLE(E(2),E(3),DUM,E(3))
      E(4)=E(4)+APD*DT
      E(6)=E(6)+RAD*DT
      CALL OITRAN(E,X2,1)


C     COMPUTE THE STARTING AND ENDING RADIUS MAGNITUDES AND THE
C     RANGE ANGLE OF THE TRANSFER (R1, R2, DTHET)
C     -------------------------------------------
      R1=VMAG(X1(4))
      R2=VMAG(X2(4))
      CALL VCROSS(X1(4),X2(4),HT)

      IF(VMAG(HT)/(R1*R2).LT.1.E-7)THEN
        CALL VCROSS(X1(4),X1,H1)
        CALL VCROSS(X2(4),X2,H2)
        CALL VUNIT(H1,H1)
        CALL VUNIT(H2,H2)
        DO 3 J=1,3
    3   HT(J)=H1(J)+H2(J)
        CALL VUNIT(HT,HT)
        CALL VCROSS(HT,X1(4),HCR)
C       -------------------------------------------
C       NOTE THAT THE MAGNITUDE OF HCR IS EQUAL TO R1
C       -------------------------------------------
        DTHET=PI
      ELSE
        CALL VCROSS(X1(4),X1,H1)
        IF(VDOT(HT,H1).LT.0.)THEN
          HT(1)=-HT(1)
          HT(2)=-HT(2)
          HT(3)=-HT(3)
        END IF
        CALL VUNIT(HT,HT)
        CALL VCROSS(HT,X1(4),HCR)
        DTHET=ATAN2(VDOT(X2(4),HCR),VDOT(X2(4),X1(4)))
        IF(DTHET.LT.0.) DTHET=DTHET +TWOPI
      END IF
```

```
C     COMPUTE FIRST GUESSES FOR THE STARTING VELOCITY REQUIRED TO
C     PRODUCE A TRANSFER THAT INTERSECTS THE R2 VECTOR AT T=T2
C     ---------------------------------------------------------
      DT=T2-T1
      CALL GAUSS(R1,R2,DTHET,DT,VTC,VTR,DUM1,DUM2,DUM3,KGERR)
      IF(KGERR.NE.0)THEN
        KV=KGERR
        RETURN
      END IF
      TEMP1=VTC(1)/R1
      TEMP2=VTR(1)/R1
      DO 4 J=1,3
      XT1(J)=TEMP1*HCR(J) +TEMP2*X1(J+3)
    4 XT1(J+3)=X1(J+3)

C     COMPUTE REQUIRED CORRECTION IN TRANSFER RADIUS VECTOR AT T=T2
C     (THIS IS THE NEGATIVE OF THE RADIUS-VECTOR ERROR)
C     ---------------------------------------------------
      ITER=-1
    5 ITER=ITER+1
      CALL XCOAST(XT1,DT,KRA,XT2,KEHI)
      IF(KEHI.EQ.1)THEN
        KV=-2
        RETURN
      END IF


      IF(KA.EQ.0) GO TO 15

      DO 6 I=1,3
    6 RCORR(I)=X2(I+3) -XT2(I+3)

C     IF REQUIRED CORRECTION IS WITHIN TOLERENCE, STOP ITERATING
C     -----------------------------------------------------------
      ERR=(ABS(RCORR(1)) +ABS(RCORR(2)) +ABS(RCORR(3)))/3.
      IF(ERR.LT.ETOL)GO TO 15

      IF(ITER.GE.MAXIT)THEN
        KV=0
        RETURN
      END IF

C     COMPUTE PARTIAL DERIVATIVES OF RADIUS VECTOR ERROR W.R.T.
C     CHANGES IN VELOCITY COMPONENTS AT T=T1
C     ---------------------------------------
      DO 8 J=1,3
      XT1(J)=XT1(J) +DV
      CALL XCOAST(XT1,DT,KRA,XT2,KEHI)
      IF(KEHI.EQ.1)THEN
        KV=-2
        RETURN
```

```
      END IF
      XT1(J)=XT1(J) -DV
      DO 9 I=1,3
    9 DREDV(I,J)=(XT2(I+3) -X2(I+3) +RCORR(I))/DV
    8 CONTINUE

C     COMPUTE AND APPLY CORRECTIONS TO VELOCITY COMPONENTS AT T=T1.
C     *****************************************************************
C     DREDV(I,J) IS DESTROYED AND RCORR(I) BECOMES VELOCITY CORRECTION.
C     *****************************************************************
      CALL SIMQ(DREDV,RCORR,NEQ,KFLAG)
      IF(KFLAG.NE.0)THEN
        CALL MSG('***HALT: ILL-CONDITIONED MATRIX IN SUBROUTINE TRNSFR',
     &            3)
        STOP
      END IF
      DO 10 J=1,3
   10 XT1(J)=XT1(J) +RCORR(J)

      GO TO 5

   15 CALL OITRAN(ET,XT1,-1)
      IF(ET(2).GT..99)THEN
        KV=-2
        RETURN
      END IF
      ET(3)=XMANOM(ET(2),ET(3))
      DO 20 J=1,3
      DV1(J)=XT1(J)-X1(J)
   20 DV2(J)=X2(J)-XT2(J)
      RETURN

      END

      SUBROUTINE   VCROSS(A,B,C)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ------------------------
C     C= A CROSS B
C     -------------
      DIMENSION A(3),B(3),C(3)

      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)

      RETURN
      END
```

```
      FUNCTION    VDOT(A,B)
      IMPLICIT REAL*8 (A-H,O-Z)
C     ------------------------------------------
C     COMPUTES THE DOT PRODUCT OF VECTORS A AND B
C     ------------------------------------------
      DIMENSION A(3),B(3)

      VDOT=A(1)*B(1) +A(2)*B(2) +A(3)*B(3)
      RETURN
      END


      FUNCTION    VMAG(A)
      IMPLICIT REAL*8(A-H,O-Z)
C     ---------------------------------
C     COMPUTES THE MAGNITUDE OF VECTOR A
C     ---------------------------------
      DIMENSION A(3)

      VMAG=SQRT(A(1)*A(1) +A(2)*A(2) +A(3)*A(3))
      RETURN
      END


      SUBROUTINE    VUNIT(A,U)

      IMPLICIT REAL*8 (A-H,O-Z)
C     ------------------------------------------------------------
C     COMPUTES UNIT-VECTOR COMPONENTS, U(I), FROM INPUT VECTOR A(I)
C     ------------------------------------------------------------
      DIMENSION A(3),U(3)

      AM=SQRT(A(1)*A(1) +A(2)*A(2) +A(3)*A(3))

      IF(AM.NE.0.)THEN
        U(1)=A(1)/AM
        U(2)=A(2)/AM
        U(3)=A(3)/AM
      ELSE
        U(1)=0.
        U(2)=0.
        U(3)=0.
      END IF

      RETURN
      END


      FUNCTION    WEDGE(SI,CI,RADIF)
      IMPLICIT REAL*8 (A-H,O-Z)

C     COMPUTES THE TOTAL ANGLE (WEDGE ANGLE) BETWEEN TWO ORBITAL
C     PLANES GIVEN:
```

```
C     SI(J)          SINES OF INCLINATIONS OF ORBIT NOS. 1 AND 2
C     CI(J)          COSINES OF INCLINATIONS OF ORBIT NOS. 1 AND 2

C     RADIF          DIFFERENCE IN RIGHT ASCENSIONS (ORBIT 2 VALUE
C                    MINUS ORBIT 1 VALUE)
C     ----------------------------------
      DIMENSION SI(2), CI(2)

      CR=COS(RADIF)
      SR=SIN(RADIF)

      W1=-SI(2)*CI(1)*CR +SI(1)*CI(2)
      W2=-SI(2)*CI(1)*SR
      W3=-SI(1)*SI(2)*SR

      WEDGE=ASIN(SQRT(W1*W1+W2*W2+W3*W3))

      RETURN
      END



      SUBROUTINE    WINDINT(A,B,C,INDX)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(100),B(100),C(100)
      IA = 1
      IB = 1
      IC = 1
 1    IF (A(IA) .EQ. 9.0E9 .OR. B(IB) .EQ. 9.0E9) THEN
         C(IC) = 9.0E9
         INDX = IC
         RETURN
      END IF
      IF (A(IA) .LT. B(IB)) THEN
         IF (A(IA+1) .LT. B(IB)) THEN
            IA = IA + 2
         ELSE
            C(IC) = B(IB)
            IF (A(IA+1) .LT. B(IB+1)) THEN
               C(IC+1) = A(IA+1)
               IA = IA + 2
            ELSE
               C(IC+1) = B(IB+1)
               IB = IB + 2
            END IF
            IC = IC + 2
         END IF
      ELSE
         IF (B(IB+1) .LT. A(IA)) THEN
            IB = IB + 2
         ELSE
            C(IC) = A(IA)
            IF (B(IB+1) .LT. A(IA+1)) THEN
```

```
            C(IC+1) = B(IB+1)
            IB = IB + 2
         ELSE
            C(IC+1) = A(IA+1)
            IA = IA + 2
         END IF
         IC = IC + 2
      END IF
   END IF
   GOTO 1
   END
```

```
      SUBROUTINE     XCOAST(XT1,DT,KRA,XT2,KEHI)
      IMPLICIT REAL*8 (A-H,O-Z)

C     GIVEN THE ECID STATE XT1(I), COMPUTES THE ECID STATE XT2(I) OVER
C     A COAST ARC OF DT DAYS

      DIMENSION XT1(6),XT2(6), E(6)

      CALL OITRAN(E,XT1,-1)

      KEHI=0
      IF(E(2).GT..99)THEN
        KEHI=1
        RETURN
      END IF

      SI=SIN(E(5))
      CI=COS(E(5))
      CALL ORBMOVE(E(1),E(2),SI,CI,RAD,APD,XMD)
      IF(KRA.EQ.0) RAD=0.
      E(3)=XMANOM(E(2),E(3)) +XMD*DT
      CALL KEPLE(E(2),E(3),DUM,E(3))
      E(4)=E(4) +APD*DT
      E(6)=E(6) +RAD*DT
      CALL OITRAN(E,XT2,1)

      RETURN
      END
```

```
      SUBROUTINE   MSG   (MESSAGE,KFLAG)
      CHARACTER *(*) MESSAGE
      character*2 filenum
c     character*1 response
      integer*2 key,inkey$
      integer faulty_mission
      integer mission_number,numexec,flag
      common/ima_numbers/mission_number,numexec,flag
      flag=kflag
C     ---------------------------------------------------------------
C     CAUSES INTERFACE PROGRAM TO DISPLAY MESSAGE ON SCREEN AND TO
C     WRITE MESSAGE INTO LOG FILE.  ALSO, IF KFLAG IS GREATER THAN
C     ONE, INTERFACE PROGRAM WILL TAKE APPROPRIATE ACTION:
C
C     KFLAG=1   WARNING. INFORMATION ONLY. PROGRAM IS NOT INTERRUPTED.
C     KFLAG=2   WARNING. PROGRAM PAUSES FOR USER RESTART COMMAND.
C     KFLAG=3   HALT. PROGRAM STOPS.  USER MUST RESTART.
C     KFLAG=4   SIGNAL FOR NORMAL PROGRAM COMPLETION.
C     ----------------------------------------------------

      PRINT *, MESSAGE
      write(20,*)message
      WRITE (11,*) MESSAGE
C     -----------------------------------------------------------
      if(flag.eq.1)return

      if(flag.eq.2)then
        print*, ' '
        print*, 'DO YOU WISH TO CONTINUE? (Y/N):'
        write(20,*) 'DO YOU WISH TO CONTINUE? (Y/N):'
        do i=1,32
          call rt_cursor
        end do
c       read(*,5)response
c5      format(a1)
c       if(response.eq.'Y'.or.response.eq.'y')then
c         return
c       else
c         go to 999
c       endif
        do i=1,3675
          key=inkey$()
          if(key.eq.ichar('y').or.key.eq.ichar('Y'))then
c           print*, char(key)
            call write_string(char(key))
            write(20,*)char(key)
            write(20,*)'** EXECUTION RESUMED'
            print*, '** EXECUTION RESUMED'
            print*, ' '
            return
          elseif(key.eq.ichar('n').or.key.eq.ichar('N'))then
c           print*, char(key)
            call write_string(char(key))
```

```
          write(20,*)char(key)
          go to 999
        endif
        call delay
      end do
      write(20,*)'** TIMEOUT EXPIRED; EXECUTION RESUMED'
      print*, '** TIMEOUT EXPIRED; EXECUTION RESUMED'
      print*, ' '
      return
    endif

    if(flag.eq.3)then
      go to 999
    endif

    if(flag.eq.4)return
C   ----------------------------------------------------

999 endfile(12)
    endfile(13)
    endfile(14)
    endfile(15)
    endfile(16)
    endfile(17)
    endfile(18)
    close(12)
    close(13)
    close(14)
    close(15)
    close(16)
    close(17)
    close(18)
    faulty_mission=numexec+1
    write(filenum,'(i2.2)')faulty_mission
    print *, ' '
    print *,
   *'*** HIT ENTER TO RETURN TO MAIN MENU, WHERE YOU MAY MODIFY'
    print 10, filenum
10  format('     MISSION ',a,
   *       ' (THE INPUT DATA FOR THAT MISSION WILL ALREADY')
    print *,
   *'    BE IN THE INTERFACE ROUTINES, READY TO BE MODIFIED.)'
    print *, ' '
    call pause

c  Jump back to Main Menu by calling "main" subroutine (recursive call)
    call ima

    RETURN
    END
```

```
      SUBROUTINE OITRAN (EO, X, MODE)

      IMPLICIT REAL*8 (A-H,O-Z)
C     ------------------------------------------------------------
C     TRANSFORMS OSCULATING ELEMENTS (EO) TO ECID COORDINATES (X)
C     OR VICE VERSA, DEPENDING ON WHETHER MODE>0 OR MODE<0.

C     EO(1)= SEMILATUS RECTUM
C     EO(2)= ECCENTRICITY
C     EO(3)= TRUE ANOMALY
C     EO(4)= ARGUMENT OF PERIGEE
C     EO(5)= INCLINATION
C     EO(6)= RIGHT ASCENSION

C     X(1)...X(3) = X,Y,Z VELOCITY COMPONENTS
C     X(4)...X(6) = X,Y,Z POSITION COORDINATES
C     ----------------------------------------
      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
C     ----------------------------------------

      DIMENSION EO(6), X(6)
      DIMENSION UP(3),UN(3),UH(3),H(3),EV(3),RU(3),TN(3),Q(3),P(3)

      IF(MODE.GT.0) THEN

        RM=EO(1)/(EO(2)*COS(EO(3))+1.)
        VS=SQRT(XMU/EO(1))
        SF=SIN(EO(3))
        CF=COS(EO(3))
        CALL CONREF (EO(4),UP,UN,UH)
        DO 10 I=1,3
        X(I)=VS*((EO(2)+CF)*UN(I)-SF*UP(I))
   10   X(I+3)=RM*CF*UP(I)+RM*SF*UN(I)

      ELSE

        CALL VCROSS (X(4),X,H)
        CALL VCROSS (X,H,EV)
        EO(1)=VDOT(H,H)/XMU
        CALL VUNIT (X(4),RU)
        CALL VUNIT (H,H)
        EV(1)=EV(1)/XMU-RU(1)
        EV(2)=EV(2)/XMU-RU(2)
        EV(3)=EV(3)/XMU-RU(3)
        EO(2)=VMAG(EV)
        HEQ=SQRT(H(1)**2 +H(2)**2)
        EO(5) =ATAN2(HEQ,H(3))

        IF(EO(5).LT.1.E-6) GO TO 15
        TN(1)=-H(2)
        TN(2)=H(1)
        TN(3)=0.
```

```
      CALL VCROSS (H,TN,Q)
      EO(6)=ATAN2(TN(2),TN(1))
      GO TO 20

15    TN(1)=1.
      TN(2)=0.
      TN(3)=0.
      Q(1)=0.
      Q(2)=1.
      Q(3)=0.
      EO(6)=0.

20    IF(EO(2).LT.1.E-6)GO TO 30
      EO(4)=ATAN2(VDOT(EV,Q),VDOT(EV,TN))
      GO TO 40

30    EO(4)=0.
      EV(1)=TN(1)
      EV(2)=TN(2)
      EV(3)=TN(3)

40    CALL VCROSS (H,EV,P)
      EO(3)=ATAN2(VDOT(RU,P),VDOT(RU,EV))

      DO 45 I=3,6
45    EO(I)=ANG(EO(I))

   END IF

   RETURN
   END
```

```
      SUBROUTINE   OUTPUT
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LFILL, LIREF, LREF, LGEOM, LUNIT, LCOV1(15), LCOV2(15)
      LOGICAL*4 LSUN, LOWACC
      CHARACTER*3 PRUNIT
      CHARACTER*4 COVER(15)
      CHARACTER*5 DVUNIT
      CHARACTER*8 ELUNIT
      CHARACTER*22 PNAME
      CHARACTER*32 REFNAME, ORBNAM, REFNOUT, TRKNAME
      CHARACTER*32 TNAME, PLNAME, VEHNAME
      CHARACTER*1  CSUN(74), CTRK(3,74)

C     ---------------------------------------------------------------
C     SUBROUTINE OUTPUT CREATES THE OUTPUT FILES THAT WILL BE
C     ACCESSED BY THE USER-INTERFACE PROGRAM IN ACCORD WITH THE USER'S
C     INSTRUCTIONS.  THE OUTPUT FILES CONSIST OF:

C     UNIT 12:  TRAJECTORY SUMMARY (BURN-ARCS, PROPELLANTS, ORBITS)
C     UNIT 13:  GROUND TRACK
C     UNIT 14:  ALTITUDE PROFILE
C     UNIT 15:  RELATIVE MOTION PLOTS FOR RENDEZVOUS SEGMENTS
C     UNIT 16:  ORBITAL FLIGHT PROFILES FOR MISSION SEGMENTS
C     UNIT 17:  SAMBO INPUT DATA
C     UNIT 18:  SUNLIGHT, COMMUNICATION TIMELINES
C     ---------------------------------------------------------------

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA13/NSEG,IESEG(15),IBSEG(15),IBLEG(7),KSEG(7)
      COMMON/IMA16/NTRK,TRKLAT(15),TRKLONG(15),TRKALT(15)
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),TH0ISP(6)
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA24/TFRAC(15,6),ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15)
     &             ,LOWACC(15)
      COMMON/IMA30/ALTMIN
      COMMON/IMA32/LREF,NTSHIFT(3),TSHIFTS
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA38/ACCMAX(15),EMASS(15),MPSYS(15),THRNOM(15,6),
     &             FLWNOM(15,6),BCOEF(15,4)
      COMMON/IMA40/RPMIN
      COMMON/IMA44/DJL(40),EML(40,6),DVCL(40,3),TCSTL(40),TBRNL(40)
      COMMON/IMA46/NLTRAN(15),PLEGC(40,6),PLEGB(40,6)
      COMMON/IMA48/RADOT(40),APDOT(40),XMDOT(40)
      COMMON/IMA50/LUNIT
      COMMON/IMA52/TNAME(15),ORBNAM(12),PLNAME(12),TRKNAME(15)
      COMMON/IMA54/VEHNAME,PNAME(6),REFNAME
      COMMON/IMA60/ORBMIN(15), ORBMAX(15), TSTAY

      DIMENSION DV(3), E(6), VX(6), VXF(6), H1(3), H2(3), WAV(3)
      DIMENSION RU(3), PU(3)
      DIMENSION PRUSED(6), PRLEFT(6)
```

```
      DIMENSION CTBSI(15), STBUF(15)

      DIMENSION THTM(0:40),THTP(40),ALTM(40),ALTP(40),CALT(20),
     1          IUSED(20),IALT(20),APOM(40),PERM(40),ARPP(0:40),
     1          PERP(40),APOP(40),TI(40),DUM1(2),DUM2(2),ODAT(40,5),
     1          IDAT(40,2)

      DATA NTRKOUT/4/, NPRPOUT/3/
      DATA RADEG/57.29577951/


C     COMPUTE AND WRITE TRAJECTORY SUMMARY DATA (UNIT=12, 4 PAGES)
C     ------------------------------------------------------------
C     ************************************************************
C     ------------------------------------------------------------

C     WRITE HEADER INFORMATION FOR BURN-ARC SUMMARY (PAGE 1)
C     -----------------------------------------------------
      WRITE(12,701)
  701 FORMAT(1X,'TRAJECTORY SUMMARY:   B U R N    A R C S')

      DJMETR=DJL(1) -.5*TBRNL(1) -TCSTL(1)

      IF(LREF)THEN
        SHIFT=NTSHIFT(1)+FLOAT(NTSHIFT(2))/24.+FLOAT(NTSHIFT(3))/1440.
     &        +TSHIFTS/86400.
        IF(ABS(SHIFT).LT.1.E-6)THEN
          REFNOUT=ORBNAM(NORB0)
        ELSE
          REFNOUT=REFNAME
        END IF
      ELSE
        REFNOUT=REFNAME
        SHIFT=DJMETR -ODJ(NORB0)
        IDAY=SHIFT
        XHR=24.*(SHIFT-IDAY)
        IHR=XHR
        XMN=60.*(XHR-IHR)
        IMN=XMN
        XSEC=60.*(XMN-IMN)
        PRINT 726, IDAY,IHR,IMN,XSEC
        WRITE(20,726) IDAY,IHR,IMN,XSEC
  726   FORMAT(1X,'M.E.T. SHIFT = ',3(I2,'-'),F5.2,' (DD-HH-MM-SS)')
      END IF

      WRITE(12,702)REFNOUT,DJMETR
  702 FORMAT(1X,'M.E.T. REF: ',A32,5X,'JULIAN DAY=',F15.7)

      IF(NTRK.GT.0)THEN
        WRITE(12,703)
  703   FORMAT(/1X,'TRACKING STATIONS')

        NT=NTRK
        IF(NT.GT.NTRKOUT) NT=NTRKOUT
```

```
        DO 10 I=1,NT
 10     WRITE(12,704)I,TRKNAME(I)
704     FORMAT(1X,I2,2X,A32)
      END IF

      IF(LUNIT)THEN
        DVUNIT='(M/S)'
      ELSE
        DVUNIT='(FPS)'
      END IF

      WRITE(12,705)DVUNIT
705   FORMAT(/1X,'BURN',4X,'START TIME',4X,'DURATION',3X,'DELTA-V',
     & 3X,'PLANE CHNG',6X,'TRACKING COVERAGE'/2X,'NO.',3X,
     & '(HR MIN SEC)',4X,'(SEC)',6X,A5,7X,'(DEG)',5X,
     & 'STA1  STA2  STA3  STA4')


C     COMPUTE AND WRITE BURN-ARC SUMMARY DATA (PAGE 1)
C     ------------------------------------------------
      LEG=0

      DO 50 ITRAN=1,NTRAN
      WRITE(12,706)ITRAN,TNAME(ITRAN)
706   FORMAT(/14X,'TRANSFER NO.',I2,': ',A32)

      DO 40 LIT=1,NLTRAN(ITRAN)
      LEG=LEG+1

      DJSTRT=DJL(LEG) -.5*TBRNL(LEG)
      DUR=86400.*TBRNL(LEG)
      FHR=24.*(DJSTRT-DJMETR)
      IHR=FHR
      FMIN=60.*(FHR-IHR)
      IMIN=FMIN
      FSEC=60.*(FMIN-IMIN)

      DO 15 J=1,3
 15   DV(J)=DVCL(LEG,J)
      DELV=VMAG(DV)
      IF(.NOT.LUNIT) DELV=DELV/.3048

      DO 20 J=1,6
 20   E(J)=EML(LEG,J)
      CALL KEPLE(E(2),E(3),DUM,E(3))
      CALL OITRAN(E,VX,1)
      CALL VCROSS(VX(4),VX(1),H1)

      DO 25 J=1,3
 25   VX(J)=VX(J) +DVCL(LEG,J)
      CALL VCROSS(VX(4),VX(1),H2)
      CALL VUNIT(H1,H1)
```

```
      CALL VUNIT(H2,H2)
      CALL VCROSS(H1,H2,WAV)
      WA= RADEG*ASIN(VMAG(WAV))

      IF(NTRK.GT.0)THEN
        DELDJ=DJL(LEG)-DJSTRT
        E(3)=EML(LEG,3)  -XMDOT(LEG)*DELDJ
        E(4)=EML(LEG,4)  -APDOT(LEG)*DELDJ
        E(6)=EML(LEG,6)  -RADOT(LEG)*DELDJ
        CALL KEPLE(E(2),E(3),DUM,E(3))
        CALL OITRAN(E,VX,1)
        CALL IFTRAN(VX,VXF,DJSTRT,1)
        CALL TRKCOV(VXF(4),NT,LCOV1)

        DJEND=DJSTRT +TBRNL(LEG)
        DELDJ=DJL(LEG) -DJEND
        E(3)=EML(LEG,3)  -XMDOT(LEG)*DELDJ
        E(4)=EML(LEG,4)  -APDOT(LEG)*DELDJ
        E(6)=EML(LEG,6)  -RADOT(LEG)*DELDJ
        CALL KEPLE(E(2),E(3),DUM,E(3))
        CALL OITRAN(E,VX,1)
        CALL IFTRAN(VX,VXF,DJEND,1)
        CALL TRKCOV(VXF(4),NT,LCOV2)

        DO 30 I=1,NT
        COVER(I)='NONE'
        IF(LCOV1(I).AND.LCOV2(I)) COVER(I)='FULL'
        IF(LCOV1(I).AND..NOT.LCOV2(I)) COVER(I)=' BEG'
   30   IF(.NOT.LCOV1(I).AND.LCOV2(I)) COVER(I)=' END'
      END IF

      IF(NTRK.GT.0)THEN
        WRITE(12,707)LEG,IHR,IMIN,FSEC,DUR,DELV,WA,(COVER(I),I=1,NT)
  707   FORMAT(1X,I3,I6,I4,F6.1,F11.3,F10.2,F12.5,2X,4(2X,A4))
      ELSE
        WRITE(12,708)LEG,IHR,IMIN,FSEC,DUR,DELV,WA
  708   FORMAT(1X,I3,I6,I4,F6.1,F11.3,F10.2,F12.5)
      END IF

   40 CONTINUE
   50 CONTINUE
C
C WRITE OUT ALTITUDE PROFILE NUMBER OF POINTS AND UNITS
C
      WRITE(14,*) LEG * 2 + 1,LUNIT
C
C
C     WRITE HEADER INFORMATION FOR ORBIT-ELEMENT SUMMARY (PAGE 2)
C     -------------------------------------------------------------
      IF(LUNIT)THEN
        ELUNIT='  (KM.) '
        FAC=.001
      ELSE
        ELUNIT='(N. MI.)'
```

```
            FAC=.539956803E-3
          END IF

          WRITE(12,709)ELUNIT, ELUNIT
      709 FORMAT(1H1,'TRAJECTORY SUMMARY:  MEAN ORBIT ELEMENTS'
         & //1X,'IMPULSE',3X,'M.E.T.',4X,'APOGEE',4X,'PERIGEE',4X,
         & 'TR.ANOM',3X,'ARG. P',4X,'INCLIN',4X,'R. ASCEN'/3X,'NO.',5X,
         & '(DAYS)',3X,A8,2X,A8,5X,'(DEG)',5X,'(DEG)',
         & 5X,'(DEG)',5X,'(DEG)'//14X,'INITIAL CONDITIONS')

    C     WRITE ORBIT-ELEMENT SUMMARY DATA (PAGE 2)
    C     ----------------------------------------
          LEG=0

    C     INITIAL CONDITIONS
          DELDJ=DJL(1) -DJMETR
          E(1)=FAC*(EML(1,1)/(1.-EML(1,2)) -REQ)
          E(2)=FAC*(EML(1,1)/(1.+EML(1,2)) -REQ)
          E(3)=EML(1,3) -XMDOT(1)*DELDJ
          CALL KEPLE(EML(1,2),E(3),DUM,E(3))
          E(3)=RADEG*ANG(E(3))
          E(4)=RADEG*ANG((EML(1,4) -APDOT(1)*DELDJ))
          E(5)=RADEG*ANG(EML(1,5))
          E(6)=RADEG*ANG((EML(1,6) -RADOT(1)*DELDJ))
          TLAPSE=0.
          ARPP(0) = E(4) / RADEG
          THTM(0) = (E(3) + E(4)) / RADEG
          WRITE(14,*) TLAPSE,(E(1) + E(2)) * 0.5 / FAC
          WRITE(12,719)TLAPSE,E
      719 FORMAT(5X,F12.6,2F11.4,1X,4F10.4)


          DO 60 ITRAN=1,NTRAN
          WRITE(12,706)ITRAN,TNAME(ITRAN)

          DO 55 LIT=1,NLTRAN(ITRAN)
          LEG=LEG+1

    C     BEFORE IMPULSE
          TLAPSE= DJL(LEG) -DJMETR
          TI(LEG) = TLAPSE
          E(1)=FAC*(EML(LEG,1)/(1.-EML(LEG,2)) -REQ)
          E(2)=FAC*(EML(LEG,1)/(1.+EML(LEG,2)) -REQ)
          CALL KEPLE(EML(LEG,2),EML(LEG,3),DUM,E(3))
          APOM(LEG) = E(1) / FAC
          PERM(LEG) = E(2) / FAC
          E(3)=RADEG*ANG(E(3))
          E(4)=RADEG*ANG(EML(LEG,4))
          E(5)=RADEG*ANG(EML(LEG,5))
          E(6)=RADEG*ANG(EML(LEG,6))
          THTM(LEG) = (E(3) + E(4)) / RADEG
          ALTM(LEG) = (E(1) + E(2)) * 0.5 / FAC
    C
    C ALT PROFILE INFO
```

```
C
      WRITE(14,*) (TLAPSE - 0.5 * TBRNL(LEG)) * 24.0,ALTM(LEG)
C
      WRITE(12,710)LEG,TLAPSE,E
  710 FORMAT(1X,I4,'-',F11.6,2F11.4,1X,4F10.4)

C     AFTER IMPULSE
      DO 52 J=1,6
   52 E(J)=EML(LEG,J)
      CALL KEPLE(E(2),E(3),DUM,E(3))
      CALL OITRAN(E,VX,1)
      VX(1)=VX(1) +DVCL(LEG,1)
      VX(2)=VX(2) +DVCL(LEG,2)
      VX(3)=VX(3) +DVCL(LEG,3)
      CALL OITRAN(E,VX,-1)
      SLR=E(1)
      ECC=E(2)
      E(1)=FAC*(SLR/(1.-ECC) -REQ)
      E(2)=FAC*(SLR/(1.+ECC) -REQ)
      E(3)=RADEG*ANG(E(3))
      E(4)=RADEG*ANG(E(4))
      E(5)=RADEG*ANG(E(5))
      E(6)=RADEG*ANG(E(6))
      THTP(LEG) = (E(3) + E(4)) / RADEG
      ALTP(LEG) = (E(1) + E(2)) * 0.5 / FAC
      ARPP(LEG) = E(4) / RADEG
      APOP(LEG) = E(1) / FAC
      PERP(LEG) = E(2) / FAC
      WRITE(14,*) (TLAPSE + 0.5 * TBRNL(LEG)) * 24.0,ALTP(LEG)
      WRITE(12,720)LEG,E
  720 FORMAT(1X,I4,'+',11X,2F11.4,1X,4F10.4)


   55 CONTINUE
   60 CONTINUE


C     WRITE HEADER INFORMATION FOR PROPELLANT SUMMARY (PAGE 3)
C     ---------------------------------------------------------
      IF(LUNIT)THEN
        PRUNIT='KG.'
        FAC=1.
      ELSE
        PRUNIT='LB.'
        FAC=2.20462228
      END IF

      WRITE(12,711)PRUNIT
  711 FORMAT(1H1,'TRAJECTORY SUMMARY: PROPELLANT USAGE',10X,'UNITS: ',A3
     &//1X,'SYSTEM',25X,'INITIAL LOADING')

      NP=NPROP
      IF(NP.GT.NPRPOUT) NP=NPRPOUT
```

```
      DO 70 M=1,NP
      PRP=FAC*FILL(M)*PCAP(M)
   70 WRITE(12,712)M,PNAME(M),PRP
  712 FORMAT(1X,I4,5X,A22,F14.2)

      WRITE(12,713)
  713 FORMAT(/1X,'BURN',4X,'END TIME',5X,'PROPELLANTS USED DURING BURN',
     &4X,'PROPELLANTS REMAINING'/2X,'NO.',5X,'(DAYS)',6X,'SYSTEM 1  SYST
     &EM 2  SYSTEM 3  SYSTEM 1  SYSTEM 2  SYSTEM 3')


C     WRITE PROPELLANT SUMMARY DATA (PAGE 3)
C     -------------------------------------
      LEG=0

      DO 90 ITRAN=1,NTRAN
      WRITE(12,706)ITRAN,TNAME(ITRAN)

      DO 85 LIT=1,NLTRAN(ITRAN)
      LEG=LEG+1
      TLAPSE=DJL(LEG) +.5*TBRNL(LEG) -DJMETR

      DO 75 M=1,NPRPOUT
      PRUSED(M)=0.
   75 PRLEFT(M)=0.

      DO 80 M=1,NP
      PRUSED(M)=FAC*(PLEGC(LEG,M) -PLEGB(LEG,M))
   80 PRLEFT(M)=FAC*PLEGB(LEG,M)

      WRITE(12,714)LEG,TLAPSE,(PRUSED(M),M=1,NPRPOUT),
     &(PRLEFT(M),M=1,NPRPOUT)
  714 FORMAT(1X,I3,F13.6,3X,6F10.2)

   85 CONTINUE
   90 CONTINUE


C     WRITE HEADER FOR ECS COORDINATES SUMMARY (PAGE 4)
C     -------------------------------------------------
      IF(LUNIT)THEN
        DVUNIT='(M/S)'
        ELUNIT='  (KM.) '
        FACV=1.
        FAC=.001
      ELSE
        DVUNIT='(FPS)'
        ELUNIT='(N. MI.)'
        FACV=3.28083989
        FAC=.539956803E-3
      END IF

      WRITE(12,715)DVUNIT,ELUNIT
  715 FORMAT(1H1,'TRAJECTORY SUMMARY: EC SPHERICAL COORDINATES'//
```

```
     &1X,'IMPULSE',4X,'M.E.T.',3X,'VELOCITY',2X,'F.P.ANGLE',2X,
     &'HEADING',2X,'ALTITUDE',2X,'LATITUDE',2X,'LONGITUDE'/
     &3X,'NO.',6X,'(DAYS)',4X,A5,6X,'(DEG)',5X,'(DEG)',3X,A8,4X,
     &'(DEG)',5X,'(DEG)'//14X,'INITIAL CONDITIONS')

C       WRITE EC SPHERICAL COORDINATE SUMMARY (PAGE 4)
C       ------------------------------------------------
        LEG=0

C       INITIAL CONDITIONS
        DO 93 J=1,6
   93   E(J)=EML(1,J)
        DELDJ=DJL(1)-DJMETR
        E(3)=E(3) -XMDOT(1)*DELDJ
        E(4)=E(4) -APDOT(1)*DELDJ
        E(6)=E(6) -RADOT(1)*DELDJ
        CALL KEPLE(E(2),E(3),DUM,E(3))
        CALL OITRAN(E,VX,1)
        CALL IFTRAN(VX,VXF,DJMETR,1)
        CALL FSTRAN(VXF,VX,1)
        E(1)=FACV*VX(1)
        E(2)=RADEG*VX(2)
        E(3)=RADEG*VX(3)
        E(4)=FAC*(VX(4) -REQ)
        E(5)=RADEG*VX(5)
        E(6)=RADEG*VX(6)
        TLAPSE=0.
        WRITE(12,721)TLAPSE,E
  721   FORMAT(5X,F12.6,F12.3,5F10.4)


        DO 120 ITRAN=1,NTRAN
        WRITE(12,706)ITRAN,TNAME(ITRAN)

        DO 115 LIT=1,NLTRAN(ITRAN)
        LEG=LEG+1

C       BEFORE IMPULSE
        TLAPSE=DJL(LEG) -DJMETR
        DO 95 J=1,6
   95   E(J)=EML(LEG,J)
        CALL KEPLE(E(2),E(3),DUM,E(3))
        CALL OITRAN(E,VX,1)
        CALL IFTRAN(VX,VXF,DJL(LEG),1)
        CALL FSTRAN(VXF,VX,1)
        E(1)=FACV*VX(1)
        E(2)=RADEG*VX(2)
        E(3)=RADEG*VX(3)
        E(4)=FAC*(VX(4) -REQ)
        E(5)=RADEG*VX(5)
        E(6)=RADEG*VX(6)
        WRITE(12,716)LEG,TLAPSE,E
  716   FORMAT(1X,I4,'-',F11.6,F12.3,5F10.4)
```

```
C     AFTER IMPULSE
      DO 97 J=1,6
   97 E(J)=EML(LEG,J)
      CALL KEPLE(E(2),E(3),DUM,E(3))
      CALL OITRAN(E,VX,1)
      VX(1)=VX(1) +DVCL(LEG,1)
      VX(2)=VX(2) +DVCL(LEG,2)
      VX(3)=VX(3) +DVCL(LEG,3)
      CALL IFTRAN(VX,VXF,DJL(LEG),1)
      CALL FSTRAN(VXF,VX,1)
      E(1)=FACV*VX(1)
      E(2)=RADEG*VX(2)
      E(3)=RADEG*VX(3)
      E(4)=FAC*(VX(4) -REQ)
      E(5)=RADEG*VX(5)
      E(6)=RADEG*VX(6)
      WRITE(12,722)LEG,E
  722 FORMAT(1X,I4,'+',11X,F12.3,5F10.4)

  115 CONTINUE
  120 CONTINUE
C     ********  END TRAJECTORY SUMMARY OUTPUT  ********************



C     COMPUTE AND WRITE SAMBO INPUT DATA (UNIT=17).  AN EXTENSIVE
C     AMOUNT OF DATA IN EASY-TO-READ FORMAT IS OUTPUT FOR USE BY
C     THE SAMBO_DRIVE PROGRAM, WHICH COMPUTES THE NAMELIST INPUT
C     FILES NEEDED BY THE SAMBO PROGRAM.
C     -----------------------------------------------------------
      NLEG=LEG

      WRITE(17,750)NSEG,(KSEG(I),I=1,NSEG)
  750 FORMAT(1X,'NUMBER OF MISSION SEGMENTS = ',I2/1X,'SEGMENT SEQUENCE
     &= ',15I3)

      WRITE(17,751)NTRAN,(NLTRAN(I),I=1,NTRAN)
  751 FORMAT(/1X,'TOTAL NUMBER OF TRANSFERS = ',I2/1X,'LEGS PER TRANSFER
     & = ',15I3)

      WRITE(17,752)
  752 FORMAT(/1X,'TRN',6X,'ACCMAX',7X,'EMASS',5X,'MPSYS'13X,'THRNOM FOR
     &SUBSYSTEMS',17X,'FLWNOM FOR SUBSYSTEMS'/)

      DO  I=1,NTRAN
      IF (ACCMAX(I) .GT. 9999.) ACCMAX(I) = 9999.
      WRITE(17,753)I,ACCMAX(I),EMASS(I),MPSYS(I),(THRNOM(I,M),M=1,3),
     &               (FLWNOM(I,M),M=1,3)
      END DO
  753 FORMAT(1X,I2,F13.5,F12.3,I8,4X,3F12.3,5X,3F12.6)


      WRITE(17,754)
```

```
 754 FORMAT(/1X,'TRN',16X,'SUBSYSTEM COAST ACFLOW',20X,'SUBSYSTEM BURN
    &ACFLOW',16X,'ORBMIN'4X,'ORBMAX'/)

      DO I=1,NTRAN
      IF (ORBMIN(I) .GT. 9999.) ORBMIN(I) = 9999.
      IF (ORBMAX(I) .GT. 9999.) ORBMAX(I) = 9999.
      IF (ORBMIN(I) .LT. 0.) ORBMIN(I) = 0.
      IF (ORBMAX(I) .LT. 0.) ORBMAX(I) = 0.
      WRITE(17,755)I,(ACFLOW(I,J,1),J=1,3),(ACFLOW(I,J,2),J=1,3),
     &              ORBMIN(I), ORBMAX(I)
      END DO

 755 FORMAT(1X,I2,6X,3F12.6,5X,3F12.6,9X,2F10.4)


      WRITE(17,756)
 756 FORMAT(/1X,'TRN',13X,'SUBSYS PNTDOCK:  COAST 1',17X,'SUBSYS PNTDOC
    &K:  COAST 2'/)

      DO I=1,NTRAN
      WRITE(17,757)I,(PNTDOCK(I,J,1),J=1,3),(PNTDOCK(I,J,2),J=1,3)
      END DO
 757 FORMAT(1X,I2,6X,3F12.3,5X,3F12.3)

      WRITE(17,758)NPROP
 758 FORMAT(/1X,'NPROP =',I2/1X,' M',5X,'PMTHRST',6X,'THMISP',6X,'THOIS
    &P',8X,'PCAP',8X,'FILL'/)

      DO M=1,NPROP
      WRITE(17,759) M, PMTHRST(M),THMISP(M),THOISP(M),PCAP(M),FILL(M)
      END DO
 759 FORMAT(1X,I2,5F12.3)

      WRITE(17,760) NLEG
 760 FORMAT(/1X,'NLEG = ',I2/1X,' L',6X,'IMPULSE JULIAN DAY',28X,'M E A
    & N   O R B I T A L   E L E M E N T S'/)

      DO L=1,NLEG
      WRITE(17,761) L, DJL(L), (EML(L,J),J=1,6)
      END DO
 761 FORMAT(1X,I2,5X,F18.7,8X,F14.1,5F14.7)

      WRITE(17,762)
 762 FORMAT(/1X,' L',17X,'DELTAV COMPONENTS',13X,'  COAST TIME','   BUR
    &N TIME',14X,'XMDOT',8X,'APDOT',8X,'RADOT'/)

      DO L=1,NLEG
      WRITE(17,763) L, (DVCL(L,J),J=1,3), TCSTL(L), TBRNL(L),
     &              XMDOT(L), APDOT(L), RADOT(L)
      END DO
 763 FORMAT(1X,I2,5X,3F12.3,6X,2F12.7,6X,3F13.8)


      WRITE(17,764)
```

```
  764 FORMAT(/1X,' L',13X,'AFTER-COAST PROPELLANTS',18X,'AFTER-BURN PROP
     &ELLANTS'/)

      DO L=1,NLEG
      WRITE(17,765) L, (PLEGC(L,J),J=1,3), (PLEGB(L,J),J=1,3)
      END DO
  765 FORMAT(1X,I2,4X,3F12.3,4X,3F12.3)

C     SPECIAL SAMBO OUTPUT FOR DE-ORBIT SEGMENTS
      ITRAN1=1
      KWRITE=0
      DO I=1,NSEG
       IF(KSEG(I).EQ.3)THEN
         IF(KWRITE.EQ.0)THEN
           WRITE(17,766)RPMIN
  766      FORMAT(//1X,'SPECIAL OUTPUT FOR DE-ORBIT SEGMENTS'/
     &            1X,'RPMIN = ',F8.0//1X,'TRN',8X,'CTBSI',6X,'STBUF'/)
           KWRITE=1
         END IF
         ITR1P1=ITRAN1+1
         CTBSI(ITR1P1) =GEOM(ITR1P1,5)/86400.
         STBUF(ITR1P1) =GEOM(ITR1P1,6)/86400.
         WRITE(17,767) ITR1P1, CTBSI(ITR1P1), STBUF(ITR1P1)
  767    FORMAT(1X,I3,5X,F8.6,3X,F8.6)
         ITRAN1=ITRAN1+2
       ELSE
         IF(KSEG(I).EQ.1) ITRAN1=ITRAN1+3
         IF(KSEG(I).EQ.2) ITRAN1=ITRAN1+1
       END IF
      END DO




C     CREATE TEMPORARY PRINTER GRAPHIC FOR SUNLIGHT/COMMUNICATION
C     COVERAGE FOR FINAL IMPULSE OF 2BTOR TRANSFERS
C     -------------------------------------------
      WRITE(18,723)
  723 FORMAT(1X,'SUNLIGHT/COMMUNICATION COVERAGE'/1X,'(MARKS ON THE TIME
     &LINES ARE SPACED ONE-MINUTE APART)'/)

      LEG=0
      DO 200 ITRAN=1,NTRAN
      DO 190 LIT=1,NLTRAN(ITRAN)
      LEG=LEG+1

      IF(KTRAN(ITRAN).EQ.5.AND.LIT.EQ.NLTRAN(ITRAN))THEN
        DDJ=1./1440.
        DJ=DJL(LEG) -26.*DDJ

        DO 180 J=1,74
        DJ=DJ +DDJ
```

```
      DELDJ=DJ-DJL(LEG)
      E(1)=EML(LEG,1)
      E(2)=EML(LEG,2)
      E(3)=EML(LEG,3)  +XMDOT(LEG)*DELDJ
      CALL KEPLE(E(2),E(3),DUM,E(3))
      E(4)=EML(LEG,4)  +APDOT(LEG)*DELDJ
      E(5)=EML(LEG,5)
      E(6)=EML(LEG,6)  +RADOT(LEG)*DELDJ

      CALL OITRAN(E,VX,1)
      CALL SUNCOV(DJ,VX(4),LSUN)
      CSUN(J)='#'
      IF(LSUN) CSUN(J)='.'

      CALL IFTRAN(VX,VXF,DJ,1)
      NTOT=NTRK
      IF(NTOT.GT.3) NTOT=3
      CALL TRKCOV(VXF(4),NTOT,LCOV1)

      DO 170 N=1,NTOT
      CTRK(N,J)='#'
      IF(LCOV1(N)) CTRK(N,J)='.'
170   CONTINUE

180   CONTINUE

      CSUN(26)='V'

      WRITE(18,724)LEG,CSUN
724   FORMAT(/1X,'LEG',I3/1X,'SUN  ',74A1)

      DO 185 N=1,NTOT
      WRITE(18,725) N,(CTRK(N,J),J=1,74)
725   FORMAT(1X,'TRK',I1,1X,74A1)
185   CONTINUE

    END IF

190 CONTINUE
200 CONTINUE

C-------------------------ORBITAL PROFILE CALCULATIONS------------------

    ISTRT = 0
    RPSAV=RPMIN
    RPMIN=REQ
    WRITE(16,*) NSEG,LUNIT
    DO I = 1,NSEG
       IF (KTRAN(IESEG(I)) .GE. 4) THEN
          NT1 = NLTRAN(IBSEG(I))
          IF (KTRAN(IESEG(I)) .EQ. 4) THEN
             NT2 = NT1
             NT3 = NT1
             NLS = NT1
```

```
            ELSE
               NT2 = NT1 + NLTRAN(IBSEG(I) + 1)
               NT3 = NT2+ NLTRAN(IBSEG(I) + 2)
               NLS = NT3
            END IF
            NC = NLS / 2 + 1
            DO J = 1,NC-1
               CALT(J) = ALTM(IBLEG(I) + 2 * (J-1))
            END DO
            CALT(NC) = ALTP(IBLEG(I) + 2 * (NC-1) - 1)
            DO J = 1,NC
               IUSED(J) = 0
            END DO
            DO J = 1,NC
               ALTMAX = 0.0
               DO K = 1,NC
                  IF (CALT(K) .GT. ALTMAX .AND. IUSED(K) .EQ. 0) THEN
                     ALTMAX = CALT(K)
                     KMAX = K
                  END IF
               END DO
               IALT(J) = KMAX
               IUSED(KMAX) = J
            END DO
            IF (KTRAN(IESEG(I)) .EQ. 5) THEN
            XK = (CALT(IALT(1)) - CALT(IALT(NC))) / 140.0
            J = 2
1000        IF (CALT(IALT(J-1)) - CALT(IALT(J)) .LT. 10.0 * XK) THEN
               CALT(IALT(J)) = CALT(IALT(J-1)) - 10.0 * XK
            END IF
            IF (CALT(IALT(J)) .GT. CALT(IALT(NC)) + 10.0 * XK) THEN
               J = J + 1
               IF (J .EQ. NC) THEN
                  GOTO 1001
               ELSE
                  GOTO 1000
               END IF
            ELSE
               J = NC - 1
1002           IF (CALT(IALT(J)) - CALT(IALT(J+1)) .LT. 10.0 * XK) THEN
                  CALT(IALT(J)) = CALT(IALT(J+1)) + 10.0 * XK
               END IF
               J = J - 1
               IF (J .EQ. 1) THEN
                  GOTO 1001
               ELSE
                  GOTO 1002
               END IF
            END IF
            END IF
1001        XMIN = CALT(IALT(NC)) + REQ
            XMAX = CALT(IALT(1)) + REQ
            M = 1
            DO J = 1,NLS
```

```
               K = IBLEG(I) + J - 1
               IF (J .NE. 2 * INT(J/2)) THEN
                  R1 = CALT(J/2 + 1) + REQ
                  R2 = CALT(J/2 + 2) + REQ
                  CALL GAUSS(R1,R2,ANG(THTP(K+1)-THTM(K)),TI(K+1)-TI(K),
     1                        DUM1,DUM2,F1,P,ECC,KGERR)
                  W = THTM(K) - F1
                  ODAT(M,1) = P/(1.0-ECC)
                  ODAT(M,2) = P/(1.0+ECC)
                  ODAT(M,3) = ANG(W)
                  ODAT(M,4) = ANG(THTM(K))
                  IDAT(M,1) = J
                  ODAT(M,5) = ANG(THTP(K+1))
                  IDAT(M,2) = J+1
                  DTHT = ANG(THTP(K+1)-THTM(K))
                  IF (ODAT(M,1) .GT. XMAX) THEN
                     APOGEE = ODAT(M,3) + PI
                     DAP = ANG(APOGEE-THTM(K))
                     IF (DAP .LT. DTHT) THEN
                        XMAX = ODAT(M,1)
                     END IF
                  END IF
                  IF (ODAT(M,2) .LT. XMIN) THEN
                     DPE = ANG(ODAT(M,3)-THTM(K))
                     IF (DPE .LT. DTHT) THEN
                        XMIN = ODAT(M,2)
                     END IF
                  END IF
                  M=M+1
 2                FORMAT(1X,F18.10,1X,F18.10,1X,F18.16,1X,F18.16,1X,I2,
     1                   1X,F18.16,1X,I2)
               ELSE
                  IF (J .NE. NT1 .AND. J .NE. NT2 .AND. J .NE. NT3) THEN
                     ODAT(M,1) = CALT(J/2+1) + REQ
                     ODAT(M,2) = CALT(J/2+1) + REQ
                     ODAT(M,3) = 0.0
                     ODAT(M,4) = ANG(THTM(K))
                     IDAT(M,1) = 0
                     ODAT(M,5) = ANG(THTP(K+1))
                     IDAT(M,2) = 0
                     M=M+1
                  END IF
               END IF
            END DO
         ELSE
            NT3=3
            CALT(1) = ALTM(IBLEG(I))
            CALT(2) = ALTP(IBLEG(I)+2)
            IF (CALT(1) .GT. CALT(2)) THEN
               XMAX = CALT(1) + REQ
            ELSE
               XMAX = CALT(2) + REQ
            END IF
            R1 = CALT(1) + REQ
```

```
                K = IBLEG(I) + 1
                RA = APOP(K) + REQ
                RP = PERP(K) + REQ
                R2 = 2.0 * RA * RP / (RA + RP + (RA-RP) * COS(THTP(K)-
     1               ARPP(K)))
                DO M = 1,2
                K = IBLEG(I) + M - 1
                CALL GAUSS(R1,R2,ANG(THTP(K+1)-THTM(K)),TI(K+1)-TI(K),
     1                             DUM1,DUM2,F1,P,ECC,KGERR)
                R1 = R2
                R2 = CALT(2)+REQ
                W = THTM(K) - F1
                ODAT(M,1) = P/(1.0-ECC)
                ODAT(M,2) = P/(1.0+ECC)
                ODAT(M,3) = ANG(W)
                ODAT(M,4) = ANG(THTM(K))
                ODAT(M,5) = ANG(THTP(K+1))
                IDAT(M,1) = M
                END DO
                ODAT(1,5) = PI / 18.0 + ODAT(1,5)
                IDAT(1,2) = 0
                IDAT(2,2) = 3
                NLS=3
                XMIN = ODAT(1,2)
                DTHT = ANG(ODAT(2,5) - ODAT(2,4))
                IF (ODAT(2,1) .GT. XMAX) THEN
                   DAP = ANG(ODAT(2,3)+PI-ODAT(2,4))
                   IF (DAP .LT. DTHT) THEN
                      XMAX = ODAT(2,1)
                   END IF
                END IF
             END IF
                WRITE(16,*) XMIN,XMAX
                IF (KTRAN(IESEG(I)) .EQ. 5) THEN
                   WRITE(16,*) 4
                   II = 3
                ELSE
                   WRITE(16,*) 2
                   II = 1
                END IF
                WRITE(16,1) CALT(1)+REQ,ANG(ARPP(IBLEG(I)-1)),
     1                      APOM(IBLEG(I)),PERM(IBLEG(I))
 1              FORMAT(1X,F18.10,1X,F18.16,1X,F18.10,1X,F18.10)
                IF (KTRAN(IESEG(I)) .EQ. 5) THEN
                WRITE(16,1) CALT(1+NT1/2)+REQ,ANG(ARPP(IBLEG(I)+NT1-1)),
     1                      APOM(IBLEG(I) + NT1),PERM(IBLEG(I) + NT1)
                WRITE(16,1) CALT(1+NT2/2)+REQ,ANG(ARPP(IBLEG(I)+NT2-1)),
     1                      APOM(IBLEG(I) + NT2),PERM(IBLEG(I) + NT2)
                END IF
                WRITE(16,1) CALT(1+NT3/2)+REQ,ANG(ARPP(IBLEG(I)+NT3-1)),
     1                      APOP(IBLEG(I)+NT3-1),PERP(IBLEG(I)+NT3-1)
                WRITE(16,*) NLS-II
                DO J = 1,NLS-II
                   IDAT1 = IDAT(J,1)
```

```
            IDAT2 = IDAT(J,2)
            IF (IDAT1 .NE. 0) IDAT1 = IDAT1 + ISTRT
            IF (IDAT2 .NE. 0) IDAT2 = IDAT2 + ISTRT
            WRITE(16,2) ODAT(J,1),ODAT(J,2),ODAT(J,3),ODAT(J,4),
     1                  IDAT1,ODAT(J,5),IDAT2
         END DO
         ISTRT = ISTRT + IDAT(NLS-II,2)
      END DO
      RPMIN=RPSAV
C-------------------------GROUND TRACK STUFF----------------------
C
      DJLAST = DJMETR
      THR = 1.0
      DT = .0014
      IFLG=0
      DO LEG = 1,NLEG
         DO J = 1,6
            E(J) = EML(LEG,J)
         END DO
         TIME = DJLAST
         IF (LEG .EQ. 1) THEN
            TBSTOP = 0.0
            IBRN = 0
         ELSE
            TBSTOP = DJLAST + 0.5 * TBRNL(LEG-1)
            IBRN = 1
         END IF
         TBSTART = DJL(LEG) - 0.5 * TBRNL(LEG)
         IF(TBSTART.LT.TBSTOP)THEN
           TBSTART=TBSTOP+DT
         END IF
 1500    E(3) = EML(LEG,3) - XMDOT(LEG) * (DJL(LEG) - TIME)
         E(4) = EML(LEG,4) - APDOT(LEG) * (DJL(LEG) - TIME)
         E(6) = EML(LEG,6) - RADOT(LEG) * (DJL(LEG) - TIME)
         CALL KEPLE (E(2),E(3),DUM,E(3))
         CALL OITRAN (E,VX,1)
         CALL IFTRAN (VX,VXF,TIME,1)
         CALL FSTRAN (VXF,VX,1)
         IF (TIME .LE. TBSTOP .OR. TIME .GE. TBSTART) THEN
            IBR = 1
         ELSE
            IBR = 0
         END IF
         IF (IFLG.EQ.1) THEN
            IFLG=0
            IHR = THR
            THR = THR + 1.0
         ELSE
            IHR = 0
         END IF
         IF (TIME .EQ. DJLAST) THEN
            IMP = LEG-1
         ELSE
            IMP = 0
```

```
            END IF

            WRITE (13,3) VX(5),VX(6),IMP,IHR,IBR
  3         FORMAT(1X,F19.16,1X,F19.16,1X,I2,1X,I3,1X,I1)
            TIME = TIME + DT
            IF (TIME .GT. THR / 24.0 + DJMETR) THEN
               IFLG=1
               TIME = THR / 24.0 + DJMETR
            END IF
            IF (IBRN .EQ. 1 .AND. TIME .GT. TBSTOP) THEN
               TIME = TBSTOP
               IBRN = 0
            ELSE IF (IBRN .EQ. 0 .AND. TIME .GT. TBSTART) THEN
               TIME = TBSTART
               IBRN = 2
            END IF
            IF (TIME .LT. DJL(LEG)) GOTO 1500
            DJLAST = DJL(LEG)
         END DO
         WRITE (13,3) VX(5),VX(6),NLEG,IHR,IBR
C------------------------RELATIVE MOTION PLOT-------------------------
C
C        COMPUTE RELATIVE MOTION POINT PAIRS FOR EACH 2BTOR TRANSFER,
C        INCLUDING UP TO TWO REVOLUTIONS IN THE NEAR PHASING ORBIT.
C        ---------------------------------------------------------
         NRMPLT=0
         LEG=0
         THR = 1.0
         DDJREF=1./1440.

         DO ITRAN = 1,NTRAN
            IF (KTRAN(ITRAN) .EQ. 5) THEN
               NRMPLT = NRMPLT + 1
            END IF
         END DO
         WRITE(15,*) NRMPLT,LUNIT

         DO 500 ITRAN=1,NTRAN
         LEG=LEG+NLTRAN(ITRAN)
         IF(KTRAN(ITRAN).NE.5) GO TO 500

         IF(NRMPLT.GT.5) GO TO 500

         LM1=LEG-1
         DELDJ=DJL(LM1) -DJL(LM1-1)
         DELDJM=2.*TWOPI/(APDOT(LM1) +XMDOT(LM1))
         IF(DELDJ.GT.DELDJM) DELDJ=DELDJM
         DJMIN=DJL(LM1)-DELDJ

         NTO=NTORB(ITRAN)
         NPT=0
         LG=LEG
         DJ=DJL(LEG)
         DDJ=DDJREF
```

```
  450 NPT=NPT +1

      XMTGT=OEM(NTO,3) +DMADT(NTO)*(DJ -ODJ(NTO))
      CALL KEPLE(OEM(NTO,2),XMTGT,DUM,TATGT)
      ALTGT=OEM(NTO,4) +DAPDT(NTO)*(DJ-ODJ(NTO)) +TATGT
      RTGT =OEM(NTO,1)/(1. +OEM(NTO,2)*COS(TATGT))

      XMVEH=EML(LG,3) +XMDOT(LG)*(DJ -DJL(LG))
      CALL KEPLE(EML(LG,2),XMVEH,DUM,TAVEH)
      ALVEH=EML(LG,4) +APDOT(LG)*(DJ-DJL(LG)) +TAVEH
      RVEH =EML(LG,1)/(1. +EML(LG,2)*COS(TAVEH))

      DELAL=ANG(ALVEH -ALTGT)
      IF(DELAL.GT.PI) DELAL=DELAL -TWOPI

      IF (DJL(LG) - DJ .GE. THR / 24.0) THEN
         IHR = THR
         THR = THR + 1.0
      ELSE
         IHR = 0
      END IF
      WRITE(15,*) RTGT*DELAL,RVEH -RTGT,IHR
C      HCOORD(NRMPLT,NPT) =RTGT*DELAL
C      VCOORD(NRMPLT,NPT) =RVEH -RTGT

      IF(DJ.GT.DJMIN.AND.NPT.LT.250)THEN
        DJ=DJ-DDJ
        IF(DJ.LT.DJL(LM1).AND.LG.EQ.LEG)THEN
          LG=LM1
          DDJ=DJL(LM1) -DJ
          DJ=DJL(LM1)
        ELSE
          DDJ=DDJREF
        END IF
        GO TO 450
      ELSE
        WRITE(15,*) 9.0E10,1,1
        thr=1.0
      END IF

  500 CONTINUE

      endfile(12)
      endfile(13)
      endfile(14)
      endfile(15)
      endfile(16)
      endfile(17)
      endfile(18)
      close(12)
      close(13)
      close(14)
      close(15)
```

```
      close(16)
      close(17)
      close(18)
      RETURN
      END
```

```
      SUBROUTINE        SW01(MD,ITRAN1,LG1,BT0,DJ1,PROP1,DJ2,PROP2,LG2,BTF,KFB)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LFILL, LGEOM, LASCEND, LOWACC

C     SW01 IS THE SEGMENT WORKER THAT SOLVES A MISSION SEGMENT
C     COMPOSED OF A 2BCDH-2BCDH-2BTOR TRANSFER SEQUENCE.  THE DELTAV
C     REQUIRED FOR THE TRANSFER BETWEEN THE INITIAL ORBIT AND FAR-
C     PHASING ORBIT IS MINIMIZED FOR ALL MODES EXCEPT MD=0, WHICH
C     PRODUCES A MINIMUM-TIME MISSION SEGMENT.  THE NUMBER OF BURNS
C     IN THE 2BCDH TRANSFERS ARE GIVEN BY NB1F=NBIT(ITRAN1) AND
C     NBFN=NBIT(ITRAN1+1) RESPECTIVELY.  THE NUMBER OF BURNS IN THE
C     2BTOR TRANSFER IS ALWAYS EQUAL TO 2.
C     ****************************************************************
C     *** NOTE THAT THIS VERSION DOES NOT COMPUTE CONTINUOUS BURNS ***
C     ALSO NOTE THAT THE MAXIMUM AND MINIMUM ALLOWABLE COAST TIMES
C     IN THE INITIAL, FAR-PHASING, AND NEAR-PHASING ORBITS ARE DEFINED
C     FOR THE IMPULSIVE SOLUTION.  BURN TIMES ARE CENTERED ON THE
C     IMPULSE POINTS AND WILL THEREBY REDUCE THE COAST TIMES BETWEEN
C     BURNS.
C     ****************************************************************

C     INPUTS: ---------------------------------------------------------
C     MD        = 0   INSTRUCTS SW01 TO DETERMINE THE MINIMUM-TIME
C                     MISSION SEGMENT (WITHOUT CONSTRAINING IMPULSES TO
C                     INTERSECTIONS) USING MEAN-ORBIT APPROXIMATIONS

C     ******* MD .GT. 0 PRODUCES MINIMUM-DELTAV MISSION SEGMENT *******

C     MD        = 1   INSTRUCTS SW01 TO SOLVE FOR THE MISSION SEGMENT
C                     WITHOUT BEING CONSTRAINED BY DJ2, USING MEAN-ORBIT
C                     APPROXIMATIONS.

C     MD        = 2   INSTRUCTS SW01 TO SOLVE FOR THE MISSION SEGMENT
C                     BETWEEN DJ1 AND DJ2, USING MEAN-ORBIT APPROX.
C                     ---------------------------------------------
C                     NOTE: IN MODE 2, THIS SEGMENT WORKER SETS THE STAY
C                     TIME IN THE INITIAL ORBIT EQUAL TO "TSTAY" AND
C                     PASSES IT TO THE TOP-LEVEL OPTIMIZER THRU COMMON
C                     BLOCK IMA60.  THIS TIME MAY BE USED TO RESTRICT
C                     THE DJ2 VARIATION IN A PRECEDING 2BTO SEGMENT.
C                     ---------------------------------------------

C     MD        = 3   SAME AS MD=2, EXCEPT THAT THE ECCENTRICITIES OF THE
C                     ORBITS ARE TO BE INCLUDED IN THE COMPUTATIONS. THIS
C                     MODE IS REQUIRED FOR GENERATION OF OUTPUT DATA.

C     ITRAN1    = NUMBER OF THE FIRST TRANSFER IN THE SEGMENT
C     LG1       = NUMBER OF FIRST LEG IN THE SEGMENT
C     BT0       = BURN TIME JUST PRIOR TO START OF THIS SEGMENT
C     DJ1       = JULIAN DATE AT START OF SEGMENT
C     PROP1(M)  = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ1
C     DJ2       = JULIAN DATE AT END OF SEGMENT (INPUT WHEN MD .GE. 2)
```

```
C     OUTPUTS:------------------------------------------------------------
C     DJ2       = JULIAN DATE AT END OF SEGMENT (OUTPUT WHEN MD=0 OR 1)

C     PROP2(M) = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ2
C     LG2       = NUMBER OF LAST LEG IN THE SEGMENT
C     BTF       = FINAL BURN TIME IN THIS SEGMENT

C     KFB   =1
C     NO FEASIBLE SOLUTION WITHIN PLUS AND MINUS TWELVE-PI PHASING, EVEN
C     WITHOUT CONSTRAINING THE FIRST IMPULSE TO OCCUR AT ORBIT-PLANE
C     INTERSECTION

C     KFB   =2
C     NO FEASIBLE SOLUTION WITHIN PLUS AND MINUS TWELVE-PI PHASING
C     FOR NINE NEIGHBORING CROSSINGS OF ORBIT-PLANE INTERSECTION,
C     CENTERED ON THE CROSSING NEAREST THE APPROXIMATE SOLUTION THAT
C     IGNORES THE INTERSECTION CONSTRAINT ON THE FIRST IMPULSE.

C     ------------------------------------------------------------

C     ********************************************************
C     NOTE: ALL TIMES IN THIS SUBROUTINE ARE EXPRESSED IN DAYS
C     ********************************************************

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA24/TFRAC(15,6),ACFLOW(15,6,2),PNTDOCK(15,6,2),SBTLIM(15)
     &             ,LOWACC(15)
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA40/RPMIN
      COMMON/IMA42/MPRINT
      COMMON/IMA44/DJL(40),EML(40,6),DVCL(40,3),TCSTL(40),TBRNL(40)
      COMMON/IMA46/NLTRAN(15),PLEGC(40,6),PLEGB(40,6)
      COMMON/IMA48/RADOT(40),APDOT(40),XMDOT(40)
      COMMON/IMA58/NBIT(15),NBITS(15)
      COMMON/IMA60/ORBMIN(15), ORBMAX(15), TSTAY


      DIMENSION A(20,26), KL(20), W(20)
      DIMENSION DTMIN(3), DTMAX(3), DT(3)
      DIMENSION PROP1(6), PROP2(6)

      DIMENSION R1F(12), RFN(12), VA1F(12), VD1F(12), VAFN(12), VDFN(12)
      DIMENSION TI1F(11), TIFN(11)
      DIMENSION SI(2), CI(2)

      DIMENSION PC5(6),PC6(6)
      DIMENSION PB1F(6), PBFN(6), PB5(6)

      DIMENSION E1(6), E2(6), ET(6), DV1(3), DV2(3)
```

```
      DIMENSION ACR(3), UPER(3), UTAN(3), UMOM(3)
      DIMENSION UP1(3), UT1(3), UM1(3), UPF(3), UTF(3), UMF(3)
      DIMENSION SMAIO(5), ARMLIO(5)
      DIMENSION ZETA(3), UNM1(3), VECC1(3), VECCF(3), VECCIO(3)
      DIMENSION UMIO(3), UNIO(3), ULIO(3)
      DIMENSION IMUL(16), JMUL(16)
      DIMENSION PIN(6), PCOUT(6)


      DATA FTOL/1.E-8/
      DATA KERR1/0/
      DATA IMUL/-2,-2,-1,-1,-1,-1,-1, 0, 0, 1, 1, 1, 1, 1, 2, 2/
      DATA JMUL/-1, 1,-2,-1, 0, 1, 2,-1, 1,-2,-1, 0, 1, 2,-1, 1/
      DATA DTMTOL/1.E-4/

C     *************************** ERROR CHECKS *************************
      IF(.NOT.LGEOM(ITRAN1,5)) THEN
        CALL MSG('***HALT: DELTA HEIGHT OF FAR-PHASING ORBIT IS REQUIRED
     & BY THIS PROGRAM VERSION',3)
        STOP
      END IF

      IF(.NOT.LGEOM(ITRAN1+1,5)) THEN
        CALL MSG('***HALT: DELTA HEIGHT OF NEAR-PHASING ORBIT IS ALWAYS
     &REQUIRED',3)
        STOP
      END IF

      IF(.NOT.LGEOM(ITRAN1+2,14)) THEN
        CALL MSG('***HALT: TOTAL RANGE ANGLE OF 2BTOR IS REQUIRED BY THI
     &S PROGRAM VERSION',3)
        STOP
      END IF
C     *****************************************************************

      KFB=0

C     COMPUTE I.D. NUMBERS OF INITIAL AND FINAL ORBITS AND GIVE LOCAL
C     NAMES TO THE SINES AND COSINES OF THE MEAN ORBITAL INCLINATIONS
C     ---------------------------------------------------------------
      IF(ITRAN1.EQ.1) IORB1=NORB0
      IF(ITRAN1.GT.1) IORB1=NTORB(ITRAN1-1)
      IORB2=NTORB(ITRAN1+2)
      SI(1)=SININC(IORB1)
      CI(1)=COSINC(IORB1)
      SI(2)=SININC(IORB2)
      CI(2)=COSINC(IORB2)

C     TARGET ORBIT FOR RENDEZVOUS SEGMENT MUST BE COMPLETELY SPECIFIED
C     ----------------------------------------------------------------
      IF(KIND(IORB2).NE.1)THEN
        CALL MSG('***HALT: 2BTOR TARGET ORBIT MUST BE COMPLETELY DEFINED
     &',3)
```

```
         STOP
      END IF

C     COMPUTE SEMILATUS RECTUM, SEMIMAJOR AXIS, ECCENTRICITY,
C     AND SECULAR RATES OF RIGHT ASCENSION, ARGUMENT OF PERIGEE,
C     AND MEAN ANOMALY FOR THE FAR- AND NEAR-PHASING ORBITS.
C     -----------------------------------------------------
      APO2 =OEM(IORB2,1)/(1.-OEM(IORB2,2))
      PER2 =OEM(IORB2,1)/(1.+OEM(IORB2,2))
      SMA2=.5*(APO2+PER2)

      APO=APO2 +GEOM(ITRAN1,5)
      PER=PER2 +GEOM(ITRAN1,5)
      SMAF=.5*(APO+PER)
      SLRF=APO*PER/SMAF
      ECCF=(APO-PER)/(APO+PER)
      CALL ORBMOVE(SLRF,ECCF,SI(2),CI(2),DRAFDT,DAPFDT,DMAFDT)

      APO=APO2 +GEOM(ITRAN1+1,5)
      PER=PER2 +GEOM(ITRAN1+1,5)
      SMAN=.5*(APO+PER)
      SLRN=APO*PER/SMAN
      ECCN=(APO-PER)/(APO+PER)
      CALL ORBMOVE(SLRN,ECCN,SI(2),CI(2),DRANDT,DAPNDT,DMANDT)


C     COMPUTE THE 'AVERAGE' SECULAR RATE OF ARGUMENT OF LATITUDE
C     FOR ALL ORBITS IN THE SEGMENT
C     --------------------------------------
      DAL1DT =DAPDT(IORB1) +DMADT(IORB1)
      DALFDT =DAPFDT +DMAFDT
      DALNDT =DAPNDT +DMANDT
      DAL2DT =DAPDT(IORB2) +DMADT(IORB2)


C     COMPUTE THE "ARGUMENT-OF-LATITUDE" PERIODS AND THE MINIMUM
C     AND MAXIMUM ALLOWABLE STAY TIMES FOR THE INITIAL ORBIT AND
C     THE PHASING ORBITS
C     ------------------------------------------------------------
C     NOTE: THE MINIMUM AND MAXIMUM ALLOWABLE STAY TIMES ARE BASED
C     ON IMPULSIVE DELTAV ASSUMPTIONS.  THE ACTUAL COASTING STAY
C     TIMES WILL USUALLY BE LESS TO PROVIDE ROOM FOR THE FINITE
C     BURN TIMES.
C     ------------------------------------------------------------
      PERIOD1=TWOPI/DAL1DT
      PERIODF=TWOPI/DALFDT
      PERIODN=TWOPI/DALNDT
      DTMIN(1)=PERIOD1*ORBMIN(ITRAN1)
      DTMIN(2)=PERIODF*ORBMIN(ITRAN1+1)
      DTMIN(3)=PERIODN*ORBMIN(ITRAN1+2)
      DTMAX(1)=PERIOD1*ORBMAX(ITRAN1)
      DTMAX(2)=PERIODF*ORBMAX(ITRAN1+1)
      DTMAX(3)=PERIODN*ORBMAX(ITRAN1+2)
```

```
C     COMPUTE THE RENDEZVOUS ANGLE OFFSET
C     ----------------------------------
      ANGOFF=GEOM(ITRAN1+2,12)/SMA2


C     IF LEFT FREE, COMPUTE INITIAL PHASING FOR 2BTOR
C     ----------------------------------------------
      IF(.NOT.LGEOM(ITRAN1+2,5))THEN
        PHI0=(DAL2DT/DALNDT -1.)*GEOM(ITRAN1+2,14) +ANGOFF
      ELSE
        PHI0=GEOM(ITRAN1+2,5)
      END IF


C     COMPUTE THE SEMIMAJOR AXIS OF THE INITIAL ORBIT, THE TRANSFER
C     TIME FOR THE FINAL RENDEZVOUS TRANSFER (ALWAYS TWO IMPUSLES),
C     THE TWO DELTAV'S FOR THE FINAL TRANSFER, AND THE CHANGE IN
C     RIGHT ASCENSION THAT OCCURS DURING THE FINAL TRANSFER.
C     THE DELTHT VALUE FED TO DV2BRC IS ADJUSTED TO ACCOUNT FOR
C     THE FACT THAT DV2BRC ASSUMES KEPLERIAN MOTION WHICH DOES NOT
C     INCLUDE THE EFFECTS OF OBLATENESS ON THE MEAN MOTION.
C     ------------------------------------------------------------
      SMA1=OEM(IORB1,1)/(1. -OEM(IORB1,2)**2)
      DTN2=(PHI0 +GEOM(ITRAN1+2,14) -ANGOFF)/DAL2DT
      DELTHT=86400.*SQRT(XMU/SMA2**3)*GEOM(ITRAN1+2,14)/DAL2DT
      CALL DV2BRC(SMAN,SMA2,DELTHT,DTN2,DV1R,DV2R,FN2,SN2,EN2)
      CALL ORBMOVE(SN2,EN2,SI(2),CI(2),DRAN2DT,DUM1,DUM2)
      ASN2=DRAN2DT*DTN2


C     SET THE NUMBER OF BURNS FOR THE
C     TRANSFERS TO THE PHASING ORBITS
C     -------------------------------
      NB1F=NBIT(ITRAN1)
      NBFN=NBIT(ITRAN1+1)


C     DETERMINE TRANSFER TIMES AND CHANGES IN RIGHT ASCENSION FOR
C     TRANSFERS TO THE PHASING ORBITS. DETERMINE MAGNITUDES OF
C     ARRIVAL VELOCITIES, DEPARTURE VELOCITIES, AND RADII AT EACH
C     IMPULSE POINT.  DETERMINE TRANSIT TIMES BETWEEN IMPULSES.
C     ------------------------------------------------------------
      CALL RVAT(SMA1,SMAF,NB1F,SI,CI,R1F,VA1F,VD1F,TI1F,AS1F,DT1F)
        SI1=SI(1)
        CI1=CI(1)
        SI(1)=SI(2)
        CI(1)=CI(2)
      CALL RVAT(SMAF,SMAN,NBFN,SI,CI,RFN,VAFN,VDFN,TIFN,ASFN,DTFN)
        SI(1)=SI1
        CI(1)=CI1
```

```
C     COMPUTE THE TOTAL STAY TIME IN THE INITIAL AND PHASING ORBITS
C     (NEEDED WHEN MD .GE. 2).  COMPUTE SEGMENT'S FINAL LEG NUMBER
C     -------------------------------------------------------------
      DTSTAY =DJ2 -DJ1 -DT1F -DTFN -DTN2
      LG2=LG1 +NB1F +NBFN +1

C     ************* CHECKOUT ONLY *******************
      IF(MPRINT.NE.0) WRITE(11,333)DT1F,DTFN,DTN2
  333 FORMAT(1X,'DT1F,DTFN,DTN2 = ',3F15.7)
C     ********************************************


C     COMPUTE CONSTANTS THAT APPROXIMATE THE DIFFERENCE IN RIGHT
C     ASCENSION OF THE FAR-PHASING AND INITIAL ORBITS (FAR-PHASING
C     ANGLE - INITIAL ANGLE), AT THE MID-TIME OF THE TRANSFER,
C     EXPRESSED AS A LINEAR FUNCTION OF THE STAY TIMES IN THE
C     INITIAL, FAR-PHASING, AND NEAR-PHASING ORBITS:
C     (RADIF = RADIF0 +RADIF1*DT1 +RADIFF*DTF +RADIFN*DTN)
C     ---------------------------------------------------------------
      RA11=OEM(IORB1,6) +DRADT(IORB1)*(DJ1-ODJ(IORB1))
      RA21=OEM(IORB2,6) +DRADT(IORB2)*(DJ1-ODJ(IORB2))

      RADIF0= RA21 -RA11 +DRADT(IORB2)*(DT1F +DTFN +DTN2)
     &         -AS1F -ASFN -ASN2

      RADIF0=ANG(RADIF0)
      IF(RADIF0.GT.PI) RADIF0=RADIF0-TWOPI

      RADIF1=DRADT(IORB2) -DRADT(IORB1)
      RADIFF=DRADT(IORB2) -DRAFDT
      RADIFN=DRADT(IORB2) -DRANDT



C     COMPUTE CONSTANTS THAT APPROXIMATE THE ARGUMENT-OF-LATITUDE
C     PHASING ERROR (TARGET ANGLE - VEHICLE ANGLE) AS A LINEAR
C     FUNCTION OF THE STAY TIMES IN THE INITIAL, FAR-PHASING, AND
C     NEAR-PHASING ORBITS:
C     (ALERR = ALERR0 +ALERR1*DT1 +ALERRF*DTF +ALERRN*DTN)
C     THE SHIFT IN RIGHT ASCENSION THAT OCCURS IN THE PLANE CHANGE
C     FROM THE INITIAL ORBIT TO THE FAR PHASING ORBIT CAUSES A
C     SHIFT IN THE ARGUMENT-OF-LATITUDE PHASING WHICH MUST BE INCLUDED.
C     -----------------------------------------------------------------
C     NOTE: REMEMBER THAT THE TARGET POINT MAY BE AHEAD OF OR BEHIND
C     THE TARGET SATELLITE AS INDICATED BY THE ANGLE "ANGOFF".
C     -------------------------------------------------------
      AL11=OEM(IORB1,3) +OEM(IORB1,4) +DAL1DT*(DJ1-ODJ(IORB1))
      AL21=OEM(IORB2,3) +OEM(IORB2,4) +DAL2DT*(DJ1-ODJ(IORB2))

      FAC=COS(.5*(OEM(IORB1,5) +OEM(IORB2,5)))

      ALERR0 =ANG(AL21-AL11) -GEOM(ITRAN1+2,14) -PI*MOD((NB1F+NBFN),2)
     &    +DAL2DT*(DT1F+DTFN+DTN2) +FAC*RADIF0 +ANGOFF
```

```
      ALERR1=DAL2DT-DAL1DT +FAC*RADIF1

      ALERRF=DAL2DT-DALFDT +FAC*RADIFF
      ALERRN=DAL2DT-DALNDT +FAC*RADIFN


C     COMPUTE ARG-OF-LAT PHASING ERROR WITH ORBIT STAY TIMES AT
C     THEIR MINIMUM VALUES
C     ---------------------------------------------------------------
      ALERR=ALERR0 +ALERR1*DTMIN(1) +ALERRF*DTMIN(2) +ALERRN*DTMIN(3)


C     ADJUST ALERR0 SO THAT AN INCREASE FROM MINIMUM IN THE MOST EFFECTIVE
C     DT WILL MOVE ALERR TOWARD ZERO (THIS CONDITION MAY NOT BE
C     APPROPRIATE FOR ALL CIRCUMSTANCES)
C     ----------------------------------
      AEFF=ALERR1
      IF(ABS(ALERRF).GT.ABS(AEFF)) AEFF=ALERRF
      IF(ABS(ALERRN).GT.ABS(AEFF)) AEFF=ALERRN

      NLAP=ALERR/TWOPI
      ALERR=ALERR-NLAP*TWOPI
      ALERR0=ALERR0-NLAP*TWOPI
      IF(AEFF.GT.0..AND.ALERR.GE.0.) ALERR0=ALERR0-TWOPI
      IF(AEFF.LT.0..AND.ALERR.LE.0.) ALERR0=ALERR0+TWOPI


C     SOLVE THE MINIMUM-TIME PROBLEM WITH PROPER ARGUMENT-OF-LATITUDE
C     PHASING USING MEAN-MOTION APPROXIMATIONS AND IGNORING THE
C     REQUIREMENT THAT THE FIRST IMPULSE MUST OCCUR ON THE ORBIT-
C     PLANE INTERSECTION LINE.  WHEN MD .GT. 0, THIS SOLUTION PROVIDES
C     A STARTING POINT FOR THE MINIMUM-WEDGE-ANGLE SOLUTION.
C     ---------------------------------------------------------------
C     WHEN MD=0, THIS SOLUTION IS USED TO COMPUTE BURN AND COAST TIMES,
C     PROPELLANTS USED, AND ADDITIONAL LEGS (IF NECESSARY).
C     -----------------------------------------------------

      NTRY=0

    4 II=8
      III=9
      JJ=10

      DO 10 I=1,20
      W(I)=0.
      KL(I)=0
      DO 5 J=1,26
    5 A(I,J)=0.
   10 CONTINUE

      A(1,1)=1.
      A(1,2)=1.
      A(1,3)=1.
```

```
      A(2,1)=ALERR1
      A(2,2)=ALERRF
      A(2,3)=ALERRN
      A(2,JJ)=-ALERR0
      IF(A(2,JJ).LT.0.)THEN
        A(2,JJ)=-A(2,JJ)
        A(2,1)=-A(2,1)
        A(2,2)=-A(2,2)
        A(2,3)=-A(2,3)
      END IF

      DO 15 I=1,3
      A(I+2,I)=1.
      A(I+2,I+3)=-1.
      DTMN=DTMIN(I)
      IF(MD.GE.2) DTMN=DTMN -DTMTOL
   15 A(I+2,JJ)=DTMN

      DO 20 I=1,3
      A(I+5,I)=1.
      A(I+5,I+6)=1.
      DTMX=DTMAX(I)
      IF(MD.GE.2) DTMX=DTMX +DTMTOL
      A(I+5,JJ)=DTMX
   20 KL(I+5)=I+6

C     ************************************
      CALL SIMPLX(A,KL,W,II,III,JJ,AMAX,KUB)
C     ************************************

      IF(KUB.NE.0)THEN
        CALL MSG('*** HALT: UNBOUNDED SOLUTION IN SUBROUTINE SW01',3)
        STOP
      END IF

      IF(AMAX.GT.FTOL)THEN
C     THERE IS NO FEASIBLE SOLUTION FOR THE GIVEN PHASING REQUIREMENT.
C     TRY INCREASING AND DECREASING THE PHASING REQUIREMENT IN TWO-PI
C     STEPS UP TO PLUS OR MINUS TWELVE PI.
C     ------------------------------------
        NTRY=NTRY+1
        IF(NTRY.EQ.7)THEN
          KFB=1
          RETURN
        END IF

        IF(NTRY.EQ.1)ALERR0=ALERR0+TWOPI
        IF(NTRY.EQ.2)ALERR0=ALERR0-2*TWOPI
        IF(NTRY.EQ.3)ALERR0=ALERR0+3*TWOPI
        IF(NTRY.EQ.4)ALERR0=ALERR0-4*TWOPI
        IF(NTRY.EQ.5)ALERR0=ALERR0 +5*TWOPI
        IF(NTRY.EQ.6)ALERR0=ALERR0 -6*TWOPI
        GO TO 4
```

```
      END IF

C     EXTRACT THE CONSTRAINED OPTIMUM ORBIT STAY TIMES (DT(I)) FROM
C     THE SIMPLEX R.H.S. COLUMN
C     ------------------------
      DO 100 J=1,3
      DT(J)=0.
      DO 90 I=2,II
      IF(KL(I).NE.J) GO TO 90
      DT(J)=A(I,JJ)
      GO TO 100
   90 CONTINUE
  100 CONTINUE


C     COMPUTE DIFFERENCE IN RIGHT ASCENSION (FAR-PHASING-ORBIT ANGLE
C     MINUS INITIAL-ORBIT ANGLE) AT TIME MIDWAY ALONG THE TRANSFER
C     BETWEEN THE INITIAL AND FAR-PHASING ORBITS
C     ------------------------------------------------------
      RADIF=RADIF0 +RADIF1*DT(1) +RADIFF*DT(2) +RADIFN*DT(3)


C     PLACE RADIF BETWEEN +PI AND -PI RADIANS
C     ----------------------------------------
      NREG=RADIF/TWOPI
      RADIF=RADIF-NREG*TWOPI
      IF(RADIF.GT. PI) RADIF=RADIF-TWOPI
      IF(RADIF.LT.-PI) RADIF=RADIF+TWOPI


C     WHEN MD=0, SKIP TO COMPUTATION OF BURN AND COAST ARCS
C     -----------------------------------------------------
      IF(MD.EQ.0)THEN
        DJ2=DJ1 +DT(1) +DT(2) +DT(3) +DT1F +DTFN +DTN2
        GO TO 500
      END IF


      RADIF0=RADIF -RADIF1*DT(1) -RADIFF*DT(2) -RADIFN*DT(3)

C     THE SIGN OF RADIF IS RECORDED IN THE VALUE OF INTEGER KSGNRD.
C     IF KSGNRD=1, RADIF MUST BE MINIMIZED.  IF KSGNRD=-1, RADIF
C     MUST BE MAXIMIZED.  IN EITHER CASE, A CONSTRAINT IS ADDED TO
C     BOUND THE SOLUTION FOR RADIF SO THAT IT DOES NOT CROSS ZERO.
C     ------------------------------------------------------------
      KSGNRD=1
      IF(RADIF.LT.0.) KSGNRD=-1


C     SOLVE THE MINIMUM-WEDGE-ANGLE PROBLEM  WITH PROPER
C     ARGUMENT-OF-LATITUDE PHASING.  AN APPROXIMATE SOLUTION IS
C     ACHIEVED BY IGNORING THE REQUIREMENT THAT THE FIRST IMPULSE
```

```
C     MUST OCCUR AT THE INTERSECTION OF THE INITIAL AND FAR-PHASING
C     ORBIT PLANES.
C     -----------------------------------------------------------------
C     -----------------------------------------------------------------

      RDMIN=1.E20
      NSGN=0
      NOPPOP=0
      NTRY=0

  104 II=9
      III=10
      JJ=11

      DO 110 I=1,20
      W(I)=0.
      KL(I)=0
      DO 105 J=1,26
  105 A(I,J)=0.
  110 CONTINUE

      A(1,1)=RADIF1*KSGNRD
      A(1,2)=RADIFF*KSGNRD
      A(1,3)=RADIFN*KSGNRD

      A(2,1)=ALERR1
      A(2,2)=ALERRF
      A(2,3)=ALERRN
      A(2,JJ)=-ALERR0
      IF(A(2,JJ).LT.0.)THEN
        A(2,JJ)=-A(2,JJ)
        A(2,1)=-A(2,1)
        A(2,2)=-A(2,2)
        A(2,3)=-A(2,3)
      END IF

      DO 115 I=1,3
      DTMN=DTMIN(I)
      IF(MD.GE.2) DTMN=DTMN -DTMTOL
      A(I+2,I)=1.
      A(I+2,I+3)=-1.
  115 A(I+2,JJ)=DTMN

      DO 120 I=1,3
      DTMX=DTMAX(I)
      IF(MD.GE.2) DTMX=DTMX +DTMTOL
      A(I+5,I)=1.
      A(I+5,I+6)=1.
      A(I+5,JJ)=DTMX
  120 KL(I+5)=I+6

      A(9,1)=RADIF1
      A(9,2)=RADIFF
      A(9,3)=RADIFN
```

```
      A(9,10)=-1.*KSGNRD
      A(9,JJ)=-RADIFO

      IF(A(9,JJ).LT.0.)THEN
        A(9,JJ)=-A(9,JJ)
        A(9,1)=-A(9,1)
        A(9,2)=-A(9,2)
        A(9,3)=-A(9,3)
        A(9,10)=-A(9,10)
      END IF

      IF(A(9,10).GT.0.) KL(9)=10


      IF(MD.GE.2)THEN
C     THE CONSTRAINT ON TOTAL ORBIT TIME MUST BE ADDED TO THE TABLEAU
C     -------------------------------------------------------------
        II=10
        III=11
        A(10,1)=1.
        A(10,2)=1.
        A(10,3)=1.
        A(10,JJ)=DTSTAY
      END IF


C     ************************************
      CALL SIMPLX(A,KL,W,II,III,JJ,AMAX,KUB)
C     ************************************

      IF(KUB.NE.0)THEN
        CALL MSG('*** HALT: UNBOUNDED SOLUTION IN SUBROUTINE SW01',3)
        STOP
      END IF

      IF(AMAX.GT.FTOL)THEN
C     THERE IS NO FEASIBLE SOLUTION FOR THE GIVEN PHASING REQUIREMENT.
C     TRY CHANGING THE SIGN OF THE ANTICIPATED DIFFERENCE IN THE RIGHT
C     ASCENSIONS OF THE INITIAL AND FAR-PHASING ORBITS.
C     ---------------------------------------------------
        IF(NSGN.EQ.0)THEN
          NSGN=1
          KSGNRD=-KSGNRD
          GO TO 104
        END IF
        KSGNRD=-KSGNRD
        NSGN=0
        NTRY=NTRY+1
        GO TO 450
      END IF

      NSGN=0

C     -----------
```

```
      NTRY=NTRY+1
C     -----------

C     EXTRACT THE CONSTRAINED OPTIMUM ORBIT STAY TIMES (DT(I)) FROM
C     THE SIMPLEX R.H.S. COLUMN
C     ------------------------
      DO 200 J=1,3
      DT(J)=0.
      DO 190 I=2,II
      IF(KL(I).NE.J) GO TO 190
      DT(J)=A(I,JJ)
      GO TO 200
  190 CONTINUE
  200 CONTINUE

      RADIF=RADIF0 +RADIF1*DT(1) +RADIFF*DT(2) +RADIFN*DT(3)


      IF(ABS(RADIF).LT.1.E-20.AND.MD.EQ.1)THEN
C     OBTAIN THE MINIMUM-TIME SOLUTION WITH RADIF CONSTRAINED TO
C     BE ZERO (EQUIVALENT TO MINIMUM-WEDGE ANGLE SOLUTION)
C     ----------------------------------------------------
        II=9
        III=10
        JJ=10

        DO 210 I=1,20
        W(I)=0.
        KL(I)=0
        DO 205 J=1,26
  205   A(I,J)=0.
  210   CONTINUE

        A(1,1)=1.
        A(1,2)=1.
        A(1,3)=1.

        A(2,1)=ALERR1
        A(2,2)=ALERRF
        A(2,3)=ALERRN
        A(2,JJ)=-ALERR0
        IF(A(2,JJ).LT.0.)THEN
          A(2,JJ)=-A(2,JJ)
          A(2,1)=-A(2,1)
          A(2,2)=-A(2,2)
          A(2,3)=-A(2,3)
        END IF

        DO 215 I=1,3
        DTMN=DTMIN(I)
        IF(MD.GE.2) DTMN=DTMN -DTMTOL
        A(I+2,I)=1.
        A(I+2,I+3)=-1.
```

```
  215   A(I+2,JJ)=DTMN

        DO 220 I=1,3
        DTMX=DTMAX(I)
        IF(MD.GE.2) DTMX=DTMX +DTMTOL
        A(I+5,I)=1.
        A(I+5,I+6)=1.
        A(I+5,JJ)=DTMX
  220   KL(I+5)=I+6

        A(9,1)=RADIF1
        A(9,2)=RADIFF
        A(9,3)=RADIFN
        A(9,JJ)=-RADIF0

        IF(A(9,JJ).LT.0.)THEN
          A(9,JJ)=-A(9,JJ)
          A(9,1)=-A(9,1)
          A(9,2)=-A(9,2)
          A(9,3)=-A(9,3)
        END IF

C       ************************************
        CALL SIMPLX(A,KL,W,II,III,JJ,AMAX,KUB)
C       ************************************

        IF(KUB.NE.0)THEN
        CALL MSG('*** HALT: UNBOUNDED SOLUTION IN SUBROUTINE SW01',3)
          STOP
        END IF

C     EXTRACT THE CONSTRAINED OPTIMUM ORBIT STAY TIMES (DT(I)) FROM
C     THE SIMPLEX R.H.S. COLUMN
C     ------------------------
        DO 300 J=1,3
        DT(J)=0.
        DO 290 I=2,II
        IF(KL(I).NE.J) GO TO 290
        DT(J)=A(I,JJ)
        GO TO 300
  290   CONTINUE
  300   CONTINUE

      END IF


C     COMPUTE THE COEFFICIENTS THAT ARE NEEDED TO EXPRESS THE
C     DIFFERENCE IN THE ARGUMENTS OF LATITUDE OF THE VEHICLE AND
C     ORBIT-PLANE LINE OF INTERSECTION IN TERMS OF THE ORBIT STAY
C     TIMES.  THE INITIAL VALUE OF THE COEFFICIENT ALW0 IS COMPUTED
C     BASED ON THE ASCENDING NODE OF THE INITIAL ORBIT W.R.T. THE
```

```
C     FAR PHASING ORBIT AND ON THE PASSAGE THAT IS CLOSEST TO THE
C     APPROXIMATE SOLUTION OBTAINED BY IGNORING THE INTERSECTION-LINE
C     CONSTRAINT.  THE ARGUMENT-OF-LATITUDE DIFFERENCE (VEHICLE ANGLE
C     MINUS INTERSECTION-LINE ANGLE) IS EXPRESSED AS:

C               ALW =ALW0 +ALW1*DT1 +ALWF*DTF +ALWN*DTN
C     -----------------------------------------------------------------
      ALW0=ALW(SI,CI,RADIF)
      RADIF=RADIF+RADIF1*PERIOD1
      ALWP=ALW(SI,CI,RADIF)
      RADIF=RADIF-2.*RADIF1*PERIOD1
      ALWM=ALW(SI,CI,RADIF)
      DELAL=ALWP-ALWM
      IF(DELAL.GT.PI) DELAL=DELAL-TWOPI
      IF(DELAL.LT.-PI)DELAL=DELAL+TWOPI
      RADIF=RADIF+RADIF1*PERIOD1

      ALWR=DELAL/(2.*RADIF1*PERIOD1)
      ALW1=ALWR*RADIF1
      ALWF=ALWR*RADIFF
      ALWN=ALWR*RADIFN
      ALW0=ALW0-ALW1*DT(1)-ALWF*DT(2)-ALWN*DT(3)

      ALW0=AL11-ALW0
      ALW1=DAL1DT-ALW1
      ALWF=-ALWF
      ALWN=-ALWN

C     ADD OR SUBTRACT MULTIPLES OF PI RADIANS TO MINIMIZE THE ABSOLUTE
C     VALUE OF THE DIFFERENCE IN ARGUMENTS OF LATITUDE OF THE VEHICLE
C     AND INTERSECTION LINE AT THE INSTANT OF THE FIRST IMPULSE OF THE
C     APPROXIMATE SOLUTION (THE SOLUTION THAT IGNORES THE
C     INTERSECTION-LINE CONSTRAINT).
C     --------------------------------------------------
      AL1ERR=ALW0 +ALW1*DT(1) +ALWF*DT(2) +ALWN*DT(3)

      KPI=AL1ERR/PI
      AL1ERR=AL1ERR-KPI*PI
      ALW0=ALW0 -KPI*PI

      IF(AL1ERR.GT.PIO2)THEN
        ALW0=ALW0-PI
        KPI=KPI +1
      END IF

      IF(AL1ERR.LT.-PIO2)THEN
        ALW0=ALW0+PI
        KPI=KPI-1
      END IF

      NOPP=0

  304 II=9
      III=10
```

```
      JJ=11

      DO 310 I=1,20
      W(I)=0.
      KL(I)=0
      DO 305 J=1,26
  305 A(I,J)=0.
  310 CONTINUE

      A(1,1)=RADIF1*KSGNRD
      A(1,2)=RADIFF*KSGNRD
      A(1,3)=RADIFN*KSGNRD

      A(2,1)=ALERR1
      A(2,2)=ALERRF
      A(2,3)=ALERRN
      A(2,JJ)=-ALERR0
      IF(A(2,JJ).LT.0.)THEN
        A(2,JJ)=-A(2,JJ)
        A(2,1)=-A(2,1)
        A(2,2)=-A(2,2)
        A(2,3)=-A(2,3)
      END IF

      DO 315 I=1,3
      DTMN=DTMIN(I)
      IF(MD.GE.2) DTMN=DTMN -DTMTOL
      A(I+2,I)=1.
      A(I+2,I+3)=-1.
  315 A(I+2,JJ)=DTMN

      DO 320 I=1,3
      DTMX=DTMAX(I)
      IF(MD.GE.2) DTMX=DTMX +DTMTOL
      A(I+5,I)=1.
      A(I+5,I+6)=1.
      A(I+5,JJ)=DTMX
  320 KL(I+5)=I+6

      A(9,1)=RADIF1
      A(9,2)=RADIFF
      A(9,3)=RADIFN
      A(9,JJ)=-RADIF0

      A(9,10)=-1.*KSGNRD

      IF(A(9,JJ).LT.0.)THEN
        A(9,JJ)=-A(9,JJ)
        A(9,1)=-A(9,1)
        A(9,2)=-A(9,2)
        A(9,3)=-A(9,3)
        A(9,10)=-A(9,10)
      END IF
```

```
      IF(A(9,10).GT.0.) KL(9)=10


      IF(MD.GE.2)THEN
C     THE CONSTRAINT ON TOTAL ORBIT TIME MUST BE ADDED TO THE TABLEAU
C     ----------------------------------------------------------------
        II=10
        III=11
        A(10,1)=1.
        A(10,2)=1.
        A(10,3)=1.
        A(10,JJ)=DTSTAY
      END IF


C     INCLUDE THE CONSTRAINT THAT FORCES THE FIRST IMPULSE TO
C     OCCUR AT THE LINE OF INTERSECTION OF THE INITAL AND FAR-PHASING
C     ORBITAL PLANES.  THE DIFFERENCE IN THE ARGUMENTS OF LATITUDE OF
C     THE VEHICLE AND ORBIT-INTERSECTION LINE (IN THE INITIAL ORBIT)
C     IS COMPUTED AS:
C                 ALW=ALW0 +ALW1*DT1 +ALWF*DTF +ALWN*DTN
C     -------------------------------------------------------
      II=II+1
      III=III+1
      A(II,1)=ALW1
      A(II,2)=ALWF
      A(II,3)=ALWN
      A(II,JJ)=-ALW0
      IF(A(II,JJ).LT.0.)THEN
        A(II,JJ)=-A(II,JJ)
        A(II,1)=-A(II,1)
        A(II,2)=-A(II,2)
        A(II,3)=-A(II,3)
      END IF


C     ************************************
      CALL SIMPLX(A,KL,W,II,III,JJ,AMAX,KUB)
C     ************************************

      IF(KUB.NE.0)THEN
        CALL MSG('*** HALT: UNBOUNDED SOLUTION IN SUBROUTINE SW01',3)
        STOP
      END IF


      IF(AMAX.GT.FTOL.AND.NSGN.EQ.0)THEN
        NSGN=1
        KSGNRD=-KSGNRD
        GO TO 304
      END IF

      IF(NSGN.EQ.1)THEN
        NSGN=0
```

```
         KSGNRD=-KSGNRD
      END IF

C     -----------
      NOPP=NOPP+1
C     -----------

      IF(AMAX.LE.FTOL)THEN
C     EXTRACT THE CONSTRAINED OPTIMUM ORBIT STAY TIMES (DT(I)) FROM
C     THE SIMPLEX R.H.S. COLUMN
C     ------------------------
        DO 400 J=1,3
        DT(J)=0.
        DO 390 I=2,II
        IF(KL(I).NE.J) GO TO 390
        DT(J)=A(I,JJ)
        GO TO 400
  390   CONTINUE
  400   CONTINUE

        RADIF=RADIF0 +RADIF1*DT(1) +RADIFF*DT(2) +RADIFN*DT(3)
        IF(ABS(RADIF).LT.RDMIN)THEN
          RDMIN=ABS(RADIF)
          DT1OPT=DT(1)
          DT2OPT=DT(2)
          DT3OPT=DT(3)
          RDOPT=RADIF
          NTRYOP=NTRY
          NOPPOP=NOPP
          KPIOP=KPI
        END IF
      END IF


      IF(NOPP.LT.9)THEN
        IF(NOPP.EQ.1)ALW0=ALW0 +PI
        IF(NOPP.EQ.2)ALW0=ALW0 -2*PI
        IF(NOPP.EQ.3)ALW0=ALW0 +3*PI
        IF(NOPP.EQ.4)ALW0=ALW0 -4*PI
        IF(NOPP.EQ.5)ALW0=ALW0 +5*PI
        IF(NOPP.EQ.6)ALW0=ALW0 -6*PI
        IF(NOPP.EQ.7)ALW0=ALW0 +7*PI
        IF(NOPP.EQ.8)ALW0=ALW0 -8*PI
        GO TO 304
      END IF

  450 IF(NTRY.LT.7)THEN
        IF(NTRY.EQ.1)ALERR0=ALERR0 +TWOPI
        IF(NTRY.EQ.2)ALERR0=ALERR0 -2*TWOPI
        IF(NTRY.EQ.3)ALERR0=ALERR0 +3*TWOPI
        IF(NTRY.EQ.4)ALERR0=ALERR0 -4*TWOPI
        IF(NTRY.EQ.5)ALERR0=ALERR0 +5*TWOPI
        IF(NTRY.EQ.6)ALERR0=ALERR0 -6*TWOPI
        GO TO 104
```

```
      END IF

      IF(NOPPOP.EQ.0)THEN
C     THERE IS NO FEASIBLE SOLUTION TO THE MINIMUM-WEDGE-ANGLE PROBLEM
C     ------------------------------------------------------------------
        KFB=2
        RETURN
      END IF

      RADIF=RDOPT
      DT(1)=DT1OPT
      DT(2)=DT2OPT
      DT(3)=DT3OPT

C     COMPUTE OPTIMUM DJ2 FOR MINIMUM-PROPELLANT SEGMENT (MD=1 ONLY)
C     -----------------------------------------------------------------
      IF(MD.EQ.1) DJ2=DJ1 +DT(1) +DT(2) +DT(3) +DT1F +DTFN +DTN2


C     DETERMINE WHETHER THE FIRST IMPULSE IS MADE ON AN ASCENDING
C     OR DESCENDING NODE (INITIAL ORBIT W.R.T. FAR-PHASING ORBIT)
C     ---------------------------------------------------------------
      LASCEND=.TRUE.
      IF(NOPPOP.EQ.2.OR.NOPPOP.EQ.3.OR.NOPPOP.EQ.6.OR.NOPPOP.EQ.7)THEN
        IF(MOD(KPIOP,2).EQ.0) LASCEND=.FALSE.
      ELSE
        IF(MOD(KPIOP,2).NE.0) LASCEND=.FALSE.
      END IF


C     ******************************************
C     PROPELLANT REQUIREMENTS ARE COMPUTED BELOW
C     ******************************************

C     COMPUTE PROPELLANTS REQUIRED TO COAST IN THE INITIAL ORBIT
C     AND TO TRANSFER FROM THE INITIAL ORBIT TO THE FAR PHASING
C     ORBIT (MEAN-MOTION APPROXIMATIONS)
C     ---------------------------------
  500 WA=WEDGE(SI,CI,RADIF)

C     ********* OUTPUT FOR CHECKOUT ONLY ****************
      IF(MPRINT.NE.0)THEN
        AL1B1=DMOD(AL11 +DAL1DT*DT(1),TWOPI)
        WRITE(11,700)ITRAN1,MD,DJ1,DJ2
        WRITE(11,701)DT,WA

        IF(MD.GT.0)THEN
          IF(LASCEND) WRITE(11,702)AL1B1
          IF(.NOT.LASCEND) WRITE(11,703)AL1B1
        ELSE
          WRITE(11,706)AL1B1
        END IF
```

```
        WRITE(11,704)
       END IF
C      **************************************************

       CALL CB180C(ITRAN1,DT(1)-.5*BT0,VA1F,VD1F,TI1F,
     &            WA,PROP1,PB1F,BTE)


C      COMPUTE PROPELLANTS REQUIRED TO COAST IN THE FAR PHASING
C      ORBIT AND TO TRANSFER FROM THE FAR PHASING ORBIT TO THE NEAR
C      PHASING ORBIT (MEAN-MOTION APPROXIMATIONS)
C      -----------------------------------------------------------
       WA=0.
       CALL CB180C(ITRAN1+1,DT(2)-.5*BTE,VAFN,VDFN,TIFN,
     &            WA,PB1F,PBFN,BTE)



C      COMPUTE PROPELLANTS AND BURN TIMES TO COAST IN THE NEAR PHASING
C      ORBIT AND TO TRANSFER FROM THE NEAR PHASING ORBIT TO TARGET-
C      ORBIT RENDEZVOUS (MEAN-MOTION APPROXIMATIONS).  INCLUDE FINAL BURN
C      TIME IN THE TIME INTERVAL ALLOTTED TO THE TRANSFER.
C      -----------------------------------------------------------
       CALL CSTBRN(ITRAN1+2,1,DT(3)-.5*BTE,0d0,DV1R,PBFN,PC5,PB5,CT5,BT5)
       CALL CSTBRN(ITRAN1+2,2,DTN2-.5*BT5,0d0,DV2R,PB5,PC6,PROP2,CT6,BTF)

C      ********* OUTPUT FOR CHECKOUT ONLY ****************
       IF(MPRINT.NE.0)THEN
         WRITE(11,705)CT5,DV1R,BT5,(PC5(M),M=1,3),(PB5(M),M=1,3)
         WRITE(11,705)CT6,DV2R,BTF,(PC6(M),M=1,3),(PROP2(M),M=1,3)
       END IF

C      -----------------------------------------------------------------
  700 FORMAT(1X,'ITRAN1, MODE = ',2I5/1X,'DJ1, DJ2 = ',
     &       2F18.7/)
  701 FORMAT(1X,'DT1, DTF, DTN = ',3F12.7/1X,'WEDGE ANGLE = ',F9.5/)
  702 FORMAT(1X,'ARG OF LAT OF IMPULSE 1 = ',F9.5,'(ASCENDING NODE)'//)
  703 FORMAT(1X,'ARG OF LAT OF IMPULSE 1 = ',F9.5,'(DESCENDING NODE)'//)
  704 FORMAT(1X,'   COAST TIME    DELTA VEL    BURN TIME             PROP
     &ELLANTS AT END OF COAST            PROPELLANTS AT END OF BURN'/)
  705 FORMAT(1X,9E13.5)
  706 FORMAT(1X,'ARG OF LAT OF IMPULSE 1 = ',F9.5//)
C      -----------------------------------------------------------------


       IF(MD.EQ.2)THEN
C      SET TSTAY TIME FOR POSSIBLE USE IN PRECEDING 2BTO SEGMENT
C      ---------------------------------------------------------
       TSTAY=DT(1)
       END IF
```

```
      IF(MD.NE.3)RETURN
C     *************************************************************
C     COMPUTATIONS PAST THIS POINT ARE ONLY FOR MD = 3  AND TAKE
C     INTO ACCOUNT THE ECCENTRICITIES OF THE ORBITS AND OTHER ACTUAL
C     ORBITAL PARAMETERS.  THESE COMPUTATIONS MUST BE MADE BEFORE
C     PROPER OUTPUT CAN BE GENERATED.
C     *************************************************************


C     IF SW01 HAS COMPUTED THE INITIAL PHASING FOR THE 2BTOR TRANSFER,
C     WRITE THE INITIAL PHASE ANGLE TO THE SCREEN AND LOG.DAT FILE
C     ----------------------------------------------------------
      IF(.NOT.LGEOM(ITRAN1+2,5))THEN
        ITROUT=ITRAN1+2
        PHIOUT=PHI0*180./PI
        PRINT 555, ITROUT, PHIOUT
        WRITE(20,555)ITROUT, PHIOUT
  555   FORMAT(1X,'INITIAL PHASING FOR TRANSFER',I3,' = ',F10.6,' DEG')
      END IF



C     COMPUTE TARGET ARGUMENTS OF PERIGEE AND LATITUDE AT DJ2
C     ------------------------------------------------------------
C     NOTE: REMEMBER THAT THE TARGET POINT MAY BE AHEAD OF OR BEHIND
C     THE TARGET SATELLITE AS SPECIFIED BY THE ANGLE "ANGOFF".
C     --------------------------------------------------------
      DELDJ=DJ2-ODJ(IORB2)
      XMA2S=ANG(OEM(IORB2,3) +DMADT(IORB2)*DELDJ)
      XMA2=XMA2S +ANGOFF
      CALL KEPLE(OEM(IORB2,2),XMA2,DUM,TRA2)
      ARP2=OEM(IORB2,4) +DAPDT(IORB2)*DELDJ
      ARL2=ARP2 +TRA2

C     COMPUTE TIME REQUIRED (DTR) FOR THE 2BTOR TRANSFER BY WORKING
C     BACKWARD FROM THE SATELLITE POSITION AT DJ2.
C     --------------------------------------------
      CALL KEPLE(OEM(IORB2,2),XMA2S,DUM,TRA2S)
      ARL2S=ARP2 +TRA2S
      DELARL=ARL2-ARL2S
      IF(DELARL.GT. PI) DELARL=DELARL -TWOPI
      IF(DELARL.LT.-PI) DELARL=DELARL +TWOPI
      DARL=-PHI0 -GEOM(ITRAN1+2,14) +DELARL
      CALL ARC(ARL2S,ARP2,OEM(IORB2,2),DMADT(IORB2),DAPDT(IORB2),
     &         DTR,DARL,-1)
      DTR=-DTR


C     COMPUTE STATE (EXCEPT FOR RIGHT ASCENSION) AT THE END OF THE
C     NEAR PHASING ORBIT.  THIS STATE IS THAT FOR LEG=LG2-1
C     ---------------------------------------------------
      LEG=LG2-1
      DJL(LEG)=DJ2-DTR
      EML(LEG,1)=SLRN
```

```
      EML(LEG,2)=ECCN
      EML(LEG,4)=ARP2 -.5*DTR*(DAPDT(IORB2) +DAPNDT)
      TEMP=ARL2-GEOM(ITRAN1+2,14) -EML(LEG,4)
      EML(LEG,3)=XMANOM(ECCN,TEMP)
      EML(LEG,5)=OEM(IORB2,5)

      RADOT(LEG)=DRANDT
      APDOT(LEG)=DAPNDT
      XMDOT(LEG)=DMANDT

C     TEMPORARILY SET RIGHT ASCENSION AT END OF NEAR PHASING
C     ORBIT TO THAT OF TARGET ORBIT AT DJ2, AS IF THERE WERE
C     TO BE NO REGRESSION OF THE NODES
C     ---------------------------------------
      RAC2=OEM(IORB2,6) +DRADT(IORB2)*DELDJ
      EML(LEG,6)=RAC2


C     COMPUTE PARAMETERS (ET) OF 2BTOR TRANSFER CONIC, FIRST IGNORING
C     REGRESSION OF THE NODES AND THEN ACCOUNTING FOR THEM.  FOR THESE
C     COMPUTATIONS, THE TARGET ORBIT MUST BE REFERENCED TO DJ2.
C     ------------------------------------------------------------
      EPOC1=DJL(LEG)
      EPOC2=DJ2
      DO 600 I=1,6
      E1(I)=EML(LEG,I)
  600 E2(I)=OEM(IORB2,I)
      E2(3)=XMA2
      E2(4)=ARP2
      E2(6)=RAC2
      T1=DJL(LEG)
      T2=DJ2
      KRA=0

C     RELAX RPMIN FOR TRNSFR MODE 3 CALCULATIONS
C     ------------------------------------------
  601 RPMIN=RPMIN-18520.

      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,KRA,ET,DV1,DV2,KV)
      RPMIN=RPMIN +18520.
      IF(KV.EQ.0) CALL MSG('WARNING: 2BTOR RADIUS ERROR',1)
      IF(KV.EQ.-1)CALL MSG('***HALT: 2BTOR TAKES RP .LT. RPMIN',3)
      IF(KV.EQ.-2)CALL MSG('***HALT: 2BTOR TAKES ECC .GT. .99',3)
      IF(KRA.EQ.0)THEN
        KRA=1
        SIT=SIN(ET(5))
        CIT=COS(ET(5))
        CALL ORBMOVE(ET(1),ET(2),SIT,CIT,RAD,APD,XMD)
        E1(6)=RAC2-RAD*DTR
        GO TO 601
      END IF
```

```
C     SAVE DELTA-VELOCITY COMPONENTS FOR LEG = LG2-1 AND LG2
C     ------------------------------------------------------
      DO 605 J=1,3
      DVCL(LEG,J)=DV1(J)
  605 DVCL(LG2,J)=DV2(J)

C     UPDATE RIGHT ASCENSION OF NEAR PHASING ORBIT TO ACCOUNT FOR
C     NODAL REGRESSION DURING THE 2BTOR
C     --------------------------------
      EML(LEG,6)=E1(6)


C     PROJECT THE 2BTOR ELEMENTS (ET, VALID AT T1) FORWARD TO T2
C     TO CREATE THE ELEMENTS FOR LEG=LG2, EML(LG2,I).  NOTE THAT THE
C     APPROPRIATE VALUES FOR RAD, APD, AND XMD ARE ALREADY COMPUTED.
C     -------------------------------------------------------------
      LEG=LG2
      DJL(LEG)=DJ2
      EML(LEG,1)=ET(1)
      EML(LEG,2)=ET(2)
      EML(LEG,3)=ET(3)  +XMD*DTR
      EML(LEG,4)=ET(4)  +APD*DTR
      EML(LEG,5)=ET(5)
      EML(LEG,6)=RAC2

      RADOT(LEG)=RAD
      APDOT(LEG)=APD
      XMDOT(LEG)=XMD


C     WORK BACKWARD FROM THE FIRST IMPULSE OF THE 2BTOR.
C     GUESS DJL(LG2-2) BASED ON MEAN-ORBIT APPROXIMATIONS AND THEN
C     ADJUST IT SO THAT THE VEHICLE IS AT THE NEAREST APSIDAL POINT.
C     HOWEVER, IF THE ORBIT ECCENTRICITY IS VIRTUALLY ZERO, DO NOT
C     ADJUST THE VALUE OF DJL(LG2-2).
C     ---------------------------------------------------------
      LEG=LG2-2
      DJL(LEG)=DJ2-DTN2-DT(3)
      XMA=ANG(EML(LEG+1,3)-DMANDT*(DJL(LEG+1)-DJL(LEG)))

      IF(ECCN.GT.1.E-7)THEN
        IF(XMA.LE.PIO2) DJL(LEG)=DJL(LEG) -XMA/DMANDT
        IF(XMA.GT.PIO2.AND.XMA.LE.3.*PIO2) DJL(LEG)=DJL(LEG)
     &                                          +(PI-XMA)/DMANDT
        IF(XMA.GT.3.*PIO2) DJL(LEG)=DJL(LEG) +(TWOPI-XMA)/DMANDT
      END IF

C     COMPUTE CERTAIN PARAMETER VALUES JUST AFTER THE IMPULSE AT LG2-2
C     ---------------------------------------------------------------
      DELDJ=DJL(LEG+1) -DJL(LEG)
      XMA=ANG(EML(LEG+1,3)-DMANDT*DELDJ)
      ARP=EML(LEG+1,4)-DAPNDT*DELDJ
      RAC=EML(LEG+1,6)-DRANDT*DELDJ
```

```
C     COMPUTE CERTAIN CONSTANTS USED IN COMPUTING THE LEG VALUES FOR
C     THE TRANSFER BETWEEN PHASING ORBITS
C     ------------------------------------------------------------
      IF(ABS(XMA-PI).LT..1)THEN
        REND=SLRN/(1.-ECCN)
        RBEG=SLRF/(1.+ECCF)
      ELSE
        REND=SLRN/(1.+ECCN)
        RBEG=SLRF/(1.-ECCF)
      END IF

      SMABE=.5*(RBEG+REND)
      VDEND=SQRT(XMU*(2./REND -1./SMAN))
      VAEND=SQRT(XMU*(2./REND -1./SMABE))
      VDBEG=SQRT(XMU*(2./RBEG -1./SMABE))
      VABEG=SQRT(XMU*(2./RBEG -1./SMAF))
      DVEVN=2.*(VDEND-VAEND)/NBFN
      DVODD=2.*(VDBEG-VABEG)/NBFN

C     DEFINE THE LEG NUMBER FOR THE FIRST IMPULSE OF THE TRANSFER
C     BETWEEN THE PHASING ORBITS AND INITIALIZE THE CURRENT RADIUS
C     AND VELOCITY MAGNITUDES
C     -----------------------
      LEGBEG=LG2-NBFN-1
      RNOW=REND
      DVNOW=DVEVN
      VNOW=VDEND-DVNOW

C     WORK BACKWARD TO COMPUTE THE LEG VALUES FOR THE TRANSFER
C     BETWEEN PHASING ORBITS
C     ---------------------------------------------

  610 RBEF=RNOW/(2.*XMU/(RNOW*VNOW**2)-1.)
      VBEF=RNOW*VNOW/RBEF
      EML(LEG,1)=2.*RNOW*RBEF/(RNOW+RBEF)
      EML(LEG,2)=ABS(RNOW-RBEF)/(RNOW+RBEF)
      EML(LEG,3)=0.
      IF(RNOW.GT.RBEF)EML(LEG,3)=PI
      EML(LEG,4)=ARP +XMA -EML(LEG,3)
      EML(LEG,5)=EML(LEG+1,5)
      EML(LEG,6)=RAC

      DO 615 I=1,3
  615 ACR(I)=EML(LEG,I+3)

      CALL CONREF(ACR,UPER,UTAN,UMOM)

      DO 620 J=1,3
      DVCL(LEG,J)=DVNOW*UTAN(J)
  620 IF(ABS(EML(LEG,3)-PI).LT..1) DVCL(LEG,J)=-DVCL(LEG,J)


      IF(LEG.GT.LEGBEG)THEN
```

```
          CALL ORBMOVE(EML(LEG,1),EML(LEG,2),SI(2),CI(2),RAD,APD,XMD)

          RADOT(LEG)=RAD
          APDOT(LEG)=APD
          XMDOT(LEG)=XMD

          LEG=LEG-1
          DJL(LEG)=DJL(LEG+1)  -PI/XMD
          DELDJ=DJL(LEG+1)  -DJL(LEG)
          XMA=ANG(EML(LEG+1,3)-PI)
          ARP=EML(LEG+1,4)-APD*DELDJ
          RAC=EML(LEG+1,6)-RAD*DELDJ
          RNOW=RBEF

          IF(LEG.GT.LEGBEG+1)THEN
            IF(DVNOW.EQ.DVEVN)THEN
              DVNOW=DVODD
            ELSE
              DVNOW=DVEVN
            END IF
            VNOW=VBEF-DVNOW
          END IF

          IF(LEG.EQ.LEGBEG+1)THEN
            VNOW=SQRT(2.*XMU*RBEG/(RNOW*(RNOW+RBEG)))
            DVNOW=VBEF-VNOW
          END IF

          IF(LEG.EQ.LEGBEG)THEN
            VNOW=SQRT(XMU*(2./RNOW -1./SMAF))
            DVNOW=VBEF-VNOW
          END IF

          GO TO 610

        END IF

        RADOT(LEGBEG)=DRAFDT
        APDOT(LEGBEG)=DAPFDT
        XMDOT(LEGBEG)=DMAFDT


C       COMPUTE THE STATE FOR LEG = LG1, INCLUDING THE VALUE FOR
C       THE SPECIAL ARGUMENT OF LATITUDE (SUM OF MEAN ANOMALY AND
C       ARGUMENT OF PERIGEE).  NOTE THAT THE DJL(LG1) VALUE COMPUTED
C       HERE IS THE RESULT OF MEAN-ORBIT APPROXIMATIONS AND WILL BE
C       ADJUSTED LATER IF NECESSARY.
C       ----------------------------
        DJL(LG1)=DJ1 +DT(1)
        DO 625 I=1,6
  625   EML(LG1,I)=OEM(IORB1,I)
        DELDJ=DJL(LG1)  -ODJ(IORB1)
        EML(LG1,3)=ANG(EML(LG1,3)  +DMADT(IORB1)*DELDJ)
        EML(LG1,4)=EML(LG1,4)  +DAPDT(IORB1)*DELDJ
```

```
      EML(LG1,6)=EML(LG1,6) +DRADT(IORB1)*DELDJ
      ARML1=EML(LG1,3) +EML(LG1,4)


      IF(NB1F.GT.2)THEN
C     COMPUTE INTERMEDIATE ORBIT STATES (AT DJL(LG1)) (*** NOTE THAT
C     NB1F MUST BE AN EVEN NUMBER ***).  FIRST, PROJECT THE FAR PHASING
C     ORBIT BACKWARD TO DJL(LG1) AND COMPUTE UNIT ECCENTRICITY AND
C     ANGULAR MOMENTUM VECTORS FOR THE INITIAL AND FAR PHASING ORBITS
C     AT THE SAME TIME REFERENCE.
C     -------------------------------------------------------------
C     -------------------------------------------------------------
        LEG =LEGBEG
        DELDJ=DJL(LEGBEG) -DJL(LG1)
        XMA=ANG(EML(LEG,3)-DMAFDT*DELDJ)
        ARP=EML(LEG,4) -DAPFDT*DELDJ
        RAC=EML(LEG,6) -DRAFDT*DELDJ
        ACR(1)=ARP
        ACR(2)=EML(LEG,5)
        ACR(3)=RAC
        CALL CONREF(ACR,UPF,UTF,UMF)

        ACR(1)=EML(LG1,4)
        ACR(2)=EML(LG1,5)
        ACR(3)=EML(LG1,6)
        CALL CONREF(ACR,UP1,UT1,UM1)


C       COMPUTE THE NUMBER OF INTERMEDIATE PHASING ORBITS AND THE
C       MEAN-ORBIT APPROXIMATIONS FOR THEIR SEMIMAJOR AXES AND FOR
C       THEIR SPECIAL ARGUMENTS OF LATITUDE AT DJL(LG1).  ALSO COMPUTE
C       THE MEAN-ORBIT APPROXIMATION FOR THE FAR PHASING ORBIT'S
C       SPECIAL ARGUMENT OF LATITUDE AT DJL(LG1)
C       ----------------------------------------
        NIO=(NB1F-2)/2
        ISTOP=NIO+1

        DO 635 I=1,ISTOP
        JTOT=2*I-1
        XNUM=0.
        DO 630 J=1,JTOT
  630   XNUM=XNUM+TI1F(J)
        IF(I.LT.ISTOP)THEN
          DENOM=TI1F(JTOT+1)
          ARMLIO(I)=ARML1 +PI*(JTOT -XNUM/DENOM)
          SMAIO(I) =.5*(R1F(2*I) +R1F(2*I+1))
        ELSE
          ARMLF= ARML1 +PI*JTOT -XNUM*DALFDT
        END IF
  635   CONTINUE

C       ADJUST THE INTERMEDIATE ORBITS' SPECIAL ARGUMENTS OF LATITUDE
C       BASED ON A COMPARISON OF THE FAR PHASING ORBIT'S ACTUAL PHASING
C       AND MEAN-APPROXIMATION PHASING.
```

```
C        -------------------------------
         ARLADJ=ANG(ARP+XMA-ARMLF)
         IF(ARLADJ.GT.PI) ARLADJ=ARLADJ-TWOPI

C      ******** OUTPUT FOR CHECKOUT ONLY ************
         IF(MPRINT.NE.0) WRITE(11,707)ARLADJ
  707    FORMAT(1X,'ARLADJ = ',F10.6)
C      *********************************************

         DO 640 I=1,NIO
  640    ARMLIO(I)=ARMLIO(I) +(I*ARLADJ)/ISTOP


C        COMPUTE THE ORBITAL PARAMETER VALUES OF THE INTERMEDIATE ORBITS,
C        VALID AT DJL(LG1).  THESE VALUES CAN BE STORED IN THE EML ARRAY
C        AS THE LEG-PARAMETER VALUES FOR ALTERNATE LEGS IN THE TRANSFER
C        BETWEEN THE INITIAL AND FAR PHASING ORBITS.  THE MEAN ANOMALY,
C        RIGHT ASCENSION, AND ARGUMENT OF PERIGEE VALUES WILL BE ADJUSTED
C        LATER TO CORRESPOND TO THE ACTUAL DJL VALUES FOR THOSE LEGS.
C        ------------------------------------------------------------
         CALL VCROSS(UM1,UMF,ZETA)
         SIGTOT=ASIN(VMAG(ZETA))
         CALL VUNIT(ZETA,ZETA)
         CALL VCROSS(ZETA,UM1,UNM1)

         DO 645 I=1,3
         VECC1(I)=EML(LG1,2)*UP1(I)
  645    VECCF(I)=EML(LEGBEG,2)*UPF(I)

         FAC=SIGTOT/(1. +NIO)
         TEMP=1./(1. +NIO)

         DO 650 I=1,NIO
         LEG=LG1 +2*I
         SIG=FAC*I
         CSIG=COS(SIG)
         SSIG=SIN(SIG)
         UMIO(1)=CSIG*UM1(1) +SSIG*UNM1(1)
         UMIO(2)=CSIG*UM1(2) +SSIG*UNM1(2)
         UMIO(3)=CSIG*UM1(3) +SSIG*UNM1(3)

         SINIO=SQRT(UMIO(1)**2 +UMIO(2)**2)
         EML(LEG,5)=ATAN2(SINIO,UMIO(3))

         IF(EML(LEG,5).NE.0.)THEN
           EML(LEG,6)=ATAN2(UMIO(1),-UMIO(2))
         ELSE
           EML(LEG,6)=0.
         END IF

         UNIO(1)=COS(EML(LEG,6))
         UNIO(2)=SIN(EML(LEG,6))
         UNIO(3)=0.
```

```
      CALL VCROSS(UMIO,UNIO,ULIO)

      TEMP1=TEMP*I
      VECCIO(1)=VECC1(1)  +(VECCF(1)-VECC1(1))*TEMP1
      VECCIO(2)=VECC1(2)  +(VECCF(2)-VECC1(2))*TEMP1
      VECCIO(3)=VECC1(3)  +(VECCF(3)-VECC1(3))*TEMP1

      EML(LEG,2)=VMAG(VECCIO)

      CALL VUNIT(VECCIO,VECCIO)
      SINEPS=VDOT(VECCIO,UMIO)
      COSEPS=SQRT(1.-SINEPS**2)

C     ADJUST ECCENTRICITY VECTOR SO THAT IT IS PERPENDICULAR TO
C     ANGULAR MOMENTUM VECTOR
      VECCIO(1)=(VECCIO(1)  -SINEPS*UMIO(1))/COSEPS
      VECCIO(2)=(VECCIO(2)  -SINEPS*UMIO(2))/COSEPS
      VECCIO(3)=(VECCIO(3)  -SINEPS*UMIO(3))/COSEPS

      EML(LEG,1)=SMAIO(I)*(1.-EML(LEG,2)**2)

      IF(EML(LEG,2).NE.0.)THEN
        EML(LEG,4)=ATAN2(VDOT(VECCIO,ULIO),VDOT(VECCIO,UNIO))
      ELSE
        EML(LEG,4)=0.
      END IF

      EML(LEG,3)=ARMLIO(I)  -EML(LEG,4)

  650 CONTINUE

      END IF
C     ------------------------------------------------------------
C     ------------------------------------------------------------


C     DETERMINE THE ORBITAL-PARAMETER VALUES FOR THE TRANSFER SEQUENCE
C     FROM THE INITIAL TO THE FAR PHASING ORBIT.  ADJUST THE DJL VALUES
C     AND THE MEAN ANOMALY, ARGUMENT OF PERIGEE, AND RIGHT ASCENSION
C     FOR THE INITIAL ORBIT AND INTERMEDIATE ORBITS (IF ANY) IN THE
C     TRANSFER SEQUENCE.
C     ***************************************************************
      IF(NB1F.EQ.2) ISTOP=1
      DJLREF=DJL(LG1)
      EPOC1=DJLREF
      EPOC2=EPOC1

C     <<<<<<<<<<<<<<<<<<    L O N G    L O O P    <<<<<<<<<<<<<<<<<<<<<<
      DO 720 I=1,ISTOP
      IK=2*I-2

      IF(I.EQ.1)THEN
        LEG1=LG1
        T1MIN=DJ1 +DT(1)  -.5*TI1F(1)
```

```
          T1MAX=T1MIN +TI1F(1)
          IF(T1MIN.LT.DJ1) T1MIN=DJ1
        ELSE
          LEG1=LEG1+2
          T1MIN=T2MAX +1.E-5
          T1MAX=T2MAX +.5*(TI1F(IK) +TI1F(IK+1))
        END IF

        IF(I.EQ.ISTOP)THEN
          T2MAX=T1MAX +TI1F(IK+1)
          EPOC2=DJL(LEGBEG)
        ELSE
          T2MAX=T1MAX +.5*(TI1F(IK+1) +TI1F(IK+2))
        END IF

        LEG2=LEG1+2
        T2MIN=T1MAX +1.E-5

        DO 655 J=1,6
        E1(J)=EML(LEG1,J)
  655   E2(J)=EML(LEG2,J)

C       A GLOBAL SEARCH ACROSS T1 AND T2 FOR MINIMUM DELTA-V IS FIRST MADE
C       WITH T1 VARIED BETWEEN T1MIN AND T1MAX IN STEPS CORRESPONDING TO
C       APPROXIMATELY 10 DEGREES IN MEAN ANOMALY.  FOR EACH VALUE OF T1,
C       T2 IS VARIED BETWEEN VALUES CORRESPONDING TO MEAN ANOMALIES OF
C       APPROXIMATELY 135 AND 225 DEGREES W.R.T THE ANOMALY CORRESPONDING
C       TO T1.  EACH T2 STEP ALSO CORRESPONDS TO APPROXIMATELY 10 DEGREES.
C       ---------------------------------------------------------------
        DVMIN=1.E20
        RPHIGH=0.
        FAC=(T2MAX-T2MIN)/(T1MAX-T1MIN)
        DTGRD1=.05*(T1MAX-T1MIN)
        DTGRD2=FAC*DTGRD1
        T1=T1MIN-DTGRD1


        DO 665 IGRID=1,21
        T1=T1+DTGRD1
        T2=T2MIN +FAC*(T1-T1MIN) -6.*DTGRD2

        DO 660 JGRID=1,11
        T2=T2+DTGRD2
        IF(T2.LT.T2MIN.OR.T2.GT.T2MAX) GO TO 660

        RPMIN=RPMIN-185200.
        CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
        RPMIN=RPMIN+185200.

        IF(KV.NE.1) GO TO 660

        RPER=ET(1)/(1.+ET(2))
```

```
      DVTOT=VMAG(DV1) +VMAG(DV2)

      IF(DVTOT.LT.DVMIN.AND.RPER.GE.RPMIN)THEN
        DVMIN=DVTOT
        T1OPT=T1
        T2OPT=T2
      END IF

      IF(RPER.GT.RPHIGH)THEN
        RPHIGH=RPER
        T1HIGH=T1
        T2HIGH=T2
      END IF

  660 CONTINUE
  665 CONTINUE

      IF(DVMIN.GT..99E20)THEN

        IF(RPHIGH.LT..01)THEN
          WRITE(11,*)'EPOC1, EPOC2=',EPOC1,EPOC2
          WRITE(11,*)'E1=',E1
          WRITE(11,*)'E2=',E2
          WRITE(11,*)'KV=',KV
          CALL MSG('***HALT: SW01 GLOBAL SEARCH FAILED',3)
          STOP
        ELSE
          CALL MSG('GLOBAL-SEARCH PERIGEE TOO HIGH',1)
          T1OPT=T1HIGH
          T2OPT=T2HIGH
        END IF

      END IF


C     MAKE FIVE-LEVEL SEARCH, STARTING WITH THE T1OPT AND T2OPT
C     VALUES FROM THE GLOBAL SEARCH, TO REFINE THE OPTIMUM POINT.
C     -----------------------------------------------------------
      DO 675 NLEV=1,5
      DTGRD1=.5*DTGRD1
      DTGRD2=.5*DTGRD2
      T1REF=T1OPT
      T2REF=T2OPT

      DO 670 IJGRID=1,16
      T1=T1REF +IMUL(IJGRID)*DTGRD1
      T2=T2REF +JMUL(IJGRID)*DTGRD2
      RPMIN=RPMIN-185200.
      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
      RPMIN=RPMIN+185200.
      IF(KV.NE.1) GO TO 670
      DVTOT=VMAG(DV1) +VMAG(DV2)
      RPER=ET(1)/(1.+ET(2))
```

```
      IF(DVTOT.LT.DVMIN.AND.RPER.GE.RPMIN)THEN
        DVMIN=DVTOT
        T1OPT=T1
        T2OPT=T2
      END IF

      IF(RPER.GT.RPHIGH)THEN
        RPHIGH=RPER
        T1HIGH=T1
        T2HIGH=T2
      END IF

  670 CONTINUE

      IF(DVMIN.GT..99E20)THEN
        T1OPT=T1HIGH
        T2OPT=T2HIGH
      END IF

  675 CONTINUE

      T1=T1OPT
      T2=T2OPT

C     RELAX RPMIN FOR TRNSFR MODE 3 CALCULATIONS
C     ------------------------------------------
      RPMIN=RPMIN-18520.

      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
      RPMIN=RPMIN +18520.
      IF(KV.EQ.0) CALL MSG('WARNING: RADIUS ERROR ON TRANSFER TO FAR PHA
     &SING ORBIT',1)
      IF(KV.EQ.-1)CALL MSG('***HALT: TRANSFER TO FAR PHASING ORBIT HAS L
     &OW PERIGEE',3)
      IF(KV.EQ.-2)CALL MSG('***HALT: TRANSFER TO FAR PHASING ORBIT HAS E
     &CCENTRICITY .GT. .99',3)


C     STORE PARAMETER VALUES FOR THE TRANSFER LEG.  FIRST, PROJECT
C     THE ET VALUES TO T2.
C     --------------------
      LEG=LEG1 +1
      DJL(LEG)=T2
      SIT=SIN(ET(5))
      CIT=COS(ET(5))
      CALL ORBMOVE(ET(1),ET(2),SIT,CIT,RAD,APD,XMD)
      DELDJ=T2-T1
      ET(3)=ET(3) +XMD*DELDJ
      ET(4)=ET(4) +APD*DELDJ
      ET(6)=ET(6) +RAD*DELDJ

      DO 685 J=1,6
  685 EML(LEG,J)=ET(J)
```

```
      RADOT(LEG)=RAD
      APDOT(LEG)=APD
      XMDOT(LEG)=XMD

      DO 690 J=1,3
  690 DVCL(LEG,J)=DV2(J)


C     ADJUST PARAMETER VALUES FOR LEG PRECEDING THE TRANSFER LEG
C     AND STORE DELTA-V COMPONENTS.
C     -----------------------------
      LEG=LEG1
      DJL(LEG)=T1

      IF(LEG.EQ.LG1)THEN
        RAD=DRADT(IORB1)
        APD=DAPDT(IORB1)
        XMD=DMADT(IORB1)
      ELSE
        SIT=SIN(EML(LEG,5))
        CIT=COS(EML(LEG,5))
        CALL ORBMOVE(EML(LEG,1),EML(LEG,2),SIT,CIT,RAD,APD,XMD)
      END IF

      DELDJ=DJL(LEG) -DJLREF
      EML(LEG,3)=ANG(EML(LEG,3) +XMD*DELDJ)
      EML(LEG,4)=EML(LEG,4) +APD*DELDJ
      EML(LEG,6)=EML(LEG,6) +RAD*DELDJ

      RADOT(LEG)=RAD
      APDOT(LEG)=APD
      XMDOT(LEG)=XMD

      DO 695 J=1,3
  695 DVCL(LEG,J)=DV1(J)

  720 CONTINUE
C     <<<<<<<<<<<<<<<<<  END LONG LOOP  <<<<<<<<<<<<<<<<<<<<<<<<<<<<

C     ******* OUTPUT FOR CHECKOUT ONLY ********************
      IF(MPRINT.NE.0)THEN
        WRITE(11,708)
  708   FORMAT(1X,'LEG',3X,'DELTAV',4X,'TLAPSE',11X,'SLR',8X,'ECC',8X,
     &              'XMA',8X,'ARP',8X,'INC',8X,'RAC'/)

        DO 750 LEG=LG1,LG2
        TLAPSE=DJL(LEG) -DJ1
        DO 740 J=1,3
  740   DV1(J)=DVCL(LEG,J)
        DELTAV=VMAG(DV1)
        WRITE(11,709)LEG,DELTAV,TLAPSE,(EML(LEG,J),J=1,6)
  709   FORMAT(1X,I3,F9.2,F10.5,F14.0,F11.6,4F11.4)
  750   CONTINUE
```

```
      END IF
C     ******************************************************



C     COMPUTE THE PROPELLANT REQUIREMENTS FOR THE VARIOUS LEGS AND
C     STORE DATA INTO ARRAYS NEEDED BY THE CALLING PROGRAM TO PRODUCE
C     THE OUTPUT FILES.
C     --------------------------------------------------------------
      NLTRAN(ITRAN1)=NB1F
      NLTRAN(ITRAN1+1)=NBFN
      NLTRAN(ITRAN1+2)=2

      LEG=LG1-1
      BPF=0.
      TLAST=DJ1
      BTLAST=BT0

      DO 760 M=1,6
  760 PIN(M)=PROP1(M)

C     **********************************
C     CHECKOUT PRINTOUT
C     ----------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,900)
  900 FORMAT(/1X,'LEG',4X,'COAST TIME',5X,'BURN TIME',17X,'PROPELLANTS A
     &T END OF COAST',16X,'PROPELLANTS AT END OF BURN'/)
      END IF
C     ***********************************


      DO 800 ITRAN=ITRAN1, ITRAN1+2

C     ***********************************
C     CHECKOUT PRINTOUT
C     -----------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,901)ITRAN
  901 FORMAT(/1X,'BEGIN TRANSFER NO. ',I3/)
      END IF
C     ************************************

      DO 790 LIT=1,NLTRAN(ITRAN)
      LEG=LEG+1

      DO 765 J=1,3
  765 DV1(J)=DVCL(LEG,J)

      DELV=VMAG(DV1)
      DTIP=DJL(LEG) -TLAST -.5*BTLAST

      CALL CSTBRN(ITRAN,LIT,DTIP,BPF,DELV,PIN,PCOUT,PROP2,CTL,BTL)
      TCSTL(LEG)=CTL
```

```
      TBRNL(LEG)=BTL
      TLAST=DJL(LEG)
      BTLAST=BTL

      DO 780 M=1,6
      PLEGC(LEG,M)=PCOUT(M)
      PLEGB(LEG,M)=PROP2(M)
  780 PIN(M)=PROP2(M)

C     *******************************************
C     CHECKOUT PRINTOUT
C     -------------------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,902)LEG,TCSTL(LEG),TBRNL(LEG),(PLEGC(LEG,M),M=1,3),
     &             (PLEGB(LEG,M),M=1,3)
  902 FORMAT(1X,I3,2E14.5,2X,3E14.5,2X,3E14.5)
      END IF
C     ********************************************


  790 CONTINUE
  800 CONTINUE

      BTF=BTLAST

      RETURN
      END
```

```
      SUBROUTINE        SW02(MD,ITRAN1,LG1,BT0,DJ1,PROP1,DJ2,PROP2,LG2,BTF,KFB)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LFILL, LGEOM

C     SW02 IS THE SEGMENT WORKER THAT SOLVES THE MISSION SEGMENT
C     CONSISTING OF A SINGLE 2BTO TRANSFER.  THE NUMBER OF BURNS FOR
C     THE 2BTO TRANSFER IS SET TO NB12=NBIT(ITRAN1).
C     ****************************************************************
C     *** NOTE THAT THIS VERSION DOES NOT COMPUTE CONTINUOUS BURNS ***
C     ****************************************************************

C     SW02 IS RESPONSIBLE FOR COMPUTING THE ANOMALIC PHASING OF THE
C     TARGET ORBIT.  ALSO, IN CASES WHERE THE TARGET ORBIT'S RIGHT
C     ASCENSION IS FREE, SW02 MUST DETERMINE THE RIGHT ASCENSION.


C     INPUTS: ---------------------------------------------------------
C     MD    =     0   INSTRUCTS SW02 TO DETERMINE THE MINIMUM-TIME
C                     MISSION SEGMENT, BEGINNING AT DJ1, USING MEAN-
C                     ORBIT APPROXIMATIONS (NOTE: IN ALL MODES, SW02
C                     REQUIRES THAT IMPULSES OCCUR ON THE LINE OF
C                     INTERSECTION OF THE ORBIT PLANES... UNLESS THE
C                     ORBITS ARE COPLANER).  DJ2 IS AN OUTPUT.

C     MD    = +/- 1   INSTRUCTS SW02 TO DETERMINE THE MINIMUM-DELTA-V
C                     MISSION SEGMENT, BEGINNING AT DJ1, USING MEAN-
C                     ORBIT APPROXIMATIONS.  DJ2 IS AN OUTPUT.

C     MD    = +/- 2   INSTRUCTS SW02 TO DETERMINE THE MINIMUM-DELTA-V
C                     MISSION SEGMENT BETWEEN DJ1 AND DJ2, USING MEAN-
C                     ORBIT APPROXIMATIONS.  DJ2 IS AN INPUT, BUT SW02
C                     WILL ADJUST IT TO THE NEAREST OPPORTUNITY.
C                     ---------------------------------------------
C                     NOTE: IN MODE 2, THIS SEGMENT WORKER SETS THE STAY
C                     TIME IN THE INITIAL ORBIT EQUAL TO "TSTAY" AND
C                     PASSES IT TO THE TOP-LEVEL OPTIMIZER THRU COMMON
C                     BLOCK IMA60.  THIS TIME MAY BE USED TO RESTRICT
C                     THE DJ2 VARIATION IN A PRECEDING 2BTO SEGMENT.
C                     ---------------------------------------------

C     MD    = +/- 3   SAME AS MD=+/-2, EXCEPT THAT THE ECCENTRICITIES OF
C                     THE ORBITS ARE INCLUDED IN THE COMPUTATIONS (MEAN-
C                     ORBIT APPROXIMATIONS DO NOT APPLY).  ALSO, IN THIS
C                     MODE SW02 WILL ADJUST DJ2 TO ACHIEVE A MINIMUM
C                     DELTA-V SOLUTION IN THE PRESENCE OF SECULAR PERTUR-
C                     BATIONS OF THE TRANSFER ARCS AND IN CONSIDERATION
C                     OF ORBITAL ECCENTRICITIES.  IF THE ECCENTRICITIES
C                     ARE SIGNIFICANT, THE DJ2 ADJUSTMENT COULD APPROACH
C                     ONE-FOURTH OF AN ORBITAL PERIOD.  THIS MODE IS
C                     REQUIRED FOR GENERATION OF OUTPUT DATA.
C                     ---------------------------------------------
C                     NOTE: IN MODE 3, THIS SEGMENT WORKER USES TSTAY TO
C                     LIMIT ITS VARIATION OF DJ2.
```

```
C                 -------------------------

C     IT IS NOTED THAT SW02 USES THE SAME LOGIC FOR THE ACCURATE
C     SOLUTION (MD=+/-3) OF THE 2BTO TRANSFER AS THAT USED BY SW01
C     FOR THE TRANSFER BETWEEN THE INITIAL AND FAR PHASING ORBITS.

C     ******************  IMPORTANT NOTE *****************************
C     ***************************************************************
C     *  IF THE MD MODE INDICATOR HAS A NEGATIVE SIGN, SW02 WILL      *
C     *  COMPUTE THE TARGET ORBIT'S RIGHT ASCENSION THAT MINIMIZES    *
C     *  THE DELTA-V REQUIREMENT.  IF MD IS ZERO OR POSITIVE, RIGHT   *
C     *  ASCENSION MUST BE SPECIFIED BY THE CALLING ROUTINE.          *
C     ***************************************************************

C     ITRAN1   = NUMBER OF THE TRANSFER IN THE SEGMENT
C     LG1      = NUMBER OF FIRST LEG IN THE SEGMENT
C     BT0      = BURN TIME JUST PRIOR TO START OF THE SEGMENT
C     DJ1      = JULIAN DATE AT START OF THE SEGMENT
C     PROP1(M) = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ1
C     DJ2      = JULIAN DATE AT END OF SEGMENT (INPUT IF ABS(MD).GE.2)


C     OUTPUTS:-------------------------------------------------------
C     DJ2      = JULIAN DATE AT END OF SEGMENT (OUTPUT IF ABS(MD).LT.2)
C     PROP2(M) = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ2
C     LG2      = NUMBER OF LAST LEG IN THE SEGMENT
C     BTF      = FINAL BURN TIME IN THIS SEGMENT

C     KFB   =0
C     EVERYTHING IS O.K.

C     KFB   =1
C     THE INITIAL-ORBIT STAY TIME RESTRICTIONS DO NOT ALLOW A FEASIBLE
C     OPPORTUNITY (RARE OCCURRENCE).


C     *******************************************************
C     NOTE: ALL TIMES IN THIS SUBROUTINE ARE EXPRESSED IN DAYS
C     *******************************************************

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA40/RPMIN
      COMMON/IMA42/MPRINT
      COMMON/IMA44/DJL(40),EML(40,6),DVCL(40,3),TCSTL(40),TBRNL(40)
      COMMON/IMA46/NLTRAN(15),PLEGC(40,6),PLEGB(40,6)
      COMMON/IMA48/RADOT(40),APDOT(40),XMDOT(40)
      COMMON/IMA58/NBIT(15),NBITS(15)
      COMMON/IMA60/ORBMIN(15),ORBMAX(15), TSTAY
```

```
      DIMENSION PROP1(6), PROP2(6)

      DIMENSION R12(12), VA12(12), VD12(12)
      DIMENSION TI12(11)
      DIMENSION SI(2), CI(2)

      DIMENSION E1(6), E2(6), ET(6), DV1(3), DV2(3)
      DIMENSION ACR(3), UPER(3), UTAN(3), UMOM(3)
      DIMENSION UP1(3), UT1(3), UM1(3), UP2(3), UT2(3), UM2(3)
      DIMENSION SMAIO(5), ARMLIO(5)
      DIMENSION ZETA(3), UNM1(3), VECC1(3), VECC2(3), VECCIO(3)
      DIMENSION UMIO(3), UNIO(3), ULIO(3)
      DIMENSION IMUL(16), JMUL(16)
      DIMENSION PIN(6), PCOUT(6)


      DATA IMUL/-2,-2,-1,-1,-1,-1,-1, 0, 0, 1, 1, 1, 1, 1, 2, 2/
      DATA JMUL/-1, 1,-2,-1, 0, 1, 2,-1, 1,-2,-1, 0, 1, 2,-1, 1/
C     -------------------------------------------------------------

      KFB=0

C     COMPUTE I.D. NUMBERS OF INITIAL AND FINAL ORBITS AND GIVE LOCAL
C     NAMES TO THE SINES AND COSINES OF THE MEAN ORBITAL INCLINATIONS
C     ---------------------------------------------------------------
      IF(ITRAN1.EQ.1) IORB1=NORB0
      IF(ITRAN1.GT.1) IORB1=NTORB(ITRAN1-1)
      IORB2=NTORB(ITRAN1)

      IF(KIND(IORB2).NE.1.AND.KIND(IORB2).NE.2.AND.KIND(IORB2).NE.4)THEN
        CALL MSG('***HALT: TARGET ORBIT FOR 2BTO TRANSFER CAN HAVE NO FR
     &EE PARAMETERS OTHER THAN RIGHT ASCENSION AND TRUE ANOMALY',3)
        STOP
      END IF

      SI(1)=SININC(IORB1)
      CI(1)=COSINC(IORB1)
      SI(2)=SININC(IORB2)
      CI(2)=COSINC(IORB2)

C     COMPUTE THE AVERAGE SECULAR RATE OF ARGUMENT OF LATITUDE, THE
C     "ARGUMENT-OF-LATITUDE" PERIOD, AND THE MINIMUM AND MAXIMUM
C     ALLOWABLE STAY TIMES FOR THE INITIAL ORBIT
C     -----------------------------------------------------------
C     NOTE: THE MINIMUM AND MAXIMUM ALLOWABLE STAY TIMES ARE BASED
C     ON IMPULSIVE DELTA-V ASSUMPTIONS.  THE ACTUAL COASTING STAY
C     TIMES WILL BE LESS THAN THE TIMES BETWEEN IMPULSES SO AS TO
C     PROVIDE ROOM FOR THE FINITE BURN TIMES.
C     -----------------------------------------------------------
      DAL1DT =DAPDT(IORB1) +DMADT(IORB1)
      PERIOD1=TWOPI/DAL1DT
      DTMIN=PERIOD1*ORBMIN(ITRAN1)
      DTMAX=PERIOD1*ORBMAX(ITRAN1)
```

```
C     COMPUTE THE AVERAGE SECULAR RATE OF ARGUMENT OF LATITUDE FOR
C     THE TARGET ORBIT
C     --------------------------------
      DAL2DT=DAPDT(IORB2) +DMADT(IORB2)


C     COMPUTE THE SEMIMAJOR AXES OF THE INITIAL AND TARGET ORBITS
C     ------------------------------------------------------------
      SMA1=OEM(IORB1,1)/(1. -OEM(IORB1,2)**2)
      SMA2=OEM(IORB2,1)/(1. -OEM(IORB2,2)**2)


C     SET THE NUMBER OF BURNS FOR THE TRANSFER
C     ----------------------------------------
      NB12=NBIT(ITRAN1)


C     DETERMINE TRANSFER TIME AND CHANGE IN RIGHT ASCENSION FOR THE
C     2BTO TRANSFER. DETERMINE MAGNITUDES OF ARRIVAL VELOCITIES,
C     DEPARTURE VELOCITIES, AND RADII AT EACH IMPULSE POINT.
C     DETERMINE TRANSIT TIMES BETWEEN IMPULSES.
C     -------------------------------------------------------------
      CALL RVAT(SMA1,SMA2,NB12,SI,CI,R12,VA12,VD12,TI12,AS12,DT12)


C     COMPUTE THE 2BTO SEGMENT'S FINAL LEG NUMBER
C     -------------------------------------------
      LG2=LG1 +NB12 -1


C     COMPUTE SOME OF THE CONSTANTS NEEDED TO EXPRESS THE CHANGE
C     IN RIGHT ASCENSION THAT MUST BE PRODUCED BY THE IMPULSES IN
C     THE 2BTO TRANSFER.  THIS CHANGE IS EXPRESSED AS:
C                RADIF = RADIF0 +RADIF1*DT1
C     WHERE DT1 = STAY TIME IN THE INITIAL ORBIT FOR THIS SEGMENT
C     (RADIF0 IS NEEDED ONLY WHEN MD.GE.0 AND IS COMPUTED LATER)
C     -----------------------------------------------------------
      RA11=OEM(IORB1,6) +DRADT(IORB1)*(DJ1-ODJ(IORB1))
      RA11=ANG(RA11)
      IF(RA11.GT.PI) RA11=RA11-TWOPI
      RADIF1=DRADT(IORB2) -DRADT(IORB1)



C     COMPUTE INITIAL VALUE OF VEHICLE'S ARGUMENT OF LATITUDE
C     -------------------------------------------------------
      AL11=OEM(IORB1,3) +OEM(IORB1,4) +DAL1DT*(DJ1-ODJ(IORB1))


      IF(MD.GE.0)THEN
C     DT1 MUST BE COMPUTED SO THAT THE FIRST IMPULSE OCCURS AT THE
C     APPROPRIATE ORBIT-PLANE INTERSECTION
```

```
C       ------------------------------------
        RA21=OEM(IORB2,6) +DRADT(IORB2)*(DJ1-ODJ(IORB2))
        RA21=ANG(RA21)
        IF(RA21.GT.PI) RA21=RA21-TWOPI
        RADIF0= RA21 -RA11 +DRADT(IORB2)*DT12 -AS12
        RADIF0=ANG(RADIF0)
        IF(RADIF0.GT.PI) RADIF0=RADIF0-TWOPI

          IF(MD.EQ.0) DT1=DTMIN

          IF(MD.EQ.1)THEN
            DT1=-RADIF0/RADIF1 -.25*PERIOD1
            IF(DT1.LT.DTMIN) DT1=DTMIN
            IF(DT1.GT.DTMAX-.5*PERIOD1) DT1=DTMAX -.5*PERIOD1
          END IF

          IF(MD.GE.2) DT1=DJ2 -DJ1 -DT12 -.25*PERIOD1
          IF(DT1.LT.DTMIN) DT1=DTMIN


          ALW0=ALW(SI,CI,RADIF0+RADIF1*DT1)
          ALWP=ALW(SI,CI,RADIF0+RADIF1*(DT1+PERIOD1))
          ALWM=ALW(SI,CI,RADIF0+RADIF1*(DT1-PERIOD1))
          DELAL=ALWP-ALWM

C         COMPENSATE, IF NECESSARY, FOR LARGE DELAL VALUES
C         CAUSED BY ALWP AND ALWM VALUES NEAR +/-PI
C         ----------------------------------------
          IF(DELAL.GT. 1.5*PI) DELAL=DELAL-TWOPI
          IF(DELAL.LT.-1.5*PI) DELAL=DELAL+TWOPI

C         COMPENSATE, IF NECESSARY, FOR 180-DEGREE SWITCH
C         IN INTERSECTION NODES WHEN EQUALLY-INCLINED ORBITS
C         PASS THROUGH A CO-PLANAR CONDITION
C         -------------------------------------------------------
          IF(DELAL.GT. PIO2.AND.DELAL.LE. 1.5*PI) DELAL =DELAL-PI
          IF(DELAL.LT.-PIO2.AND.DELAL.GE.-1.5*PI) DELAL =DELAL+PI


          AL1WD=DAL1DT -DELAL/(2.*PERIOD1)
          AL1W0=ANG(AL11 +DAL1DT*DT1 -ALW0)
          IF(AL1W0.GT.PI) AL1W0=AL1W0-PI


          DT1=DT1 +(PI-AL1W0)/AL1WD


          IF(DT1.GT.DTMAX)THEN
            KFB=1
            RETURN
          END IF

          RADIF=RADIF0 +RADIF1*DT1
          DJ2=DJ1 +DT1 +DT12
```

```
      END IF

      IF(MD.LT.0)THEN
C       FOR ORBITS WITH UNEQUAL INCLINATIONS, DT1 MUST BE COMPUTED
C       SO THAT THE FIRST IMPULSE OCCURS IN THE EQUATORIAL PLANE
C       -------------------------------------------------------
        IF(MD.EQ.-1) DT1=DTMIN
        IF(MD.LE.-2) DT1=DJ2 -DJ1 -DT12 -.25*PERIOD1
        DAL=TWOPI-ANG(AL11+DAL1DT*DT1)
        IF(DAL.GT.PI) DAL=DAL-PI

C       IF ORBITS HAVE EQUAL INCLINATIONS, PLACE INTERSECTION
C       90 DEGREES FROM EQUATOR
C       -----------------------
        IF(CI(1).EQ.CI(2))THEN

          IF(DAL.LT.PIO2)THEN
            DAL=DAL+PIO2
          ELSE
            DAL=DAL-PIO2
          END IF

        END IF

        DT1=DT1 +DAL/DAL1DT

        IF(DT1.LT.DTMIN.OR.DT1.GT.DTMAX)THEN
          KFB=1
          RETURN
        END IF

        RADIF=0.
        DJ2=DJ1 +DT1 +DT12

C       COMPUTE RIGHT ASCENSION OF THE TARGET ORBIT THAT MINIMIZES THE
C       DELTA-V REQUIREMENT (IE., THAT MAKES RADIF=0.)

        RA21=RA11 +AS12 -DRADT(IORB2)*DT12 -RADIF1*DT1
        OEM(IORB2,6)=RA21 +DRADT(IORB2)*(ODJ(IORB2)-DJ1)

      END IF


C     COMPUTE THE MEAN ANOMALY OF THE TARGET ORBIT
C     --------------------------------------------
      FAC=COS(.5*(OEM(IORB1,5) +OEM(IORB2,5)))
      AL22=ANG(AL11 +DAL1DT*DT1 -FAC*RADIF +PI)
      AP22=OEM(IORB2,4) +DAPDT(IORB2)*(DJ2-ODJ(IORB2))
      XM22=AL22 -AP22
```

```
      OEM(IORB2,3)=XM22 +DMADT(IORB2)*(ODJ(IORB2) -DJ2)



C     COMPUTE PROPELLANTS REQUIRED TO COAST IN THE INITIAL ORBIT
C     AND TO TRANSFER TO THE TARGET ORBIT
C     ----------------------------------
      WA=WEDGE(SI,CI,RADIF)

C     ********* OUTPUT FOR CHECKOUT ONLY ****************
      IF(MPRINT.NE.0)THEN
        WRITE(11,700)ITRAN1,MD,DJ1,DJ2
        WRITE(11,701)DT1,WA

        IF(WA.GT.1.E-8)THEN
          ALVEH=ANG(AL11 +DAL1DT*DT1)
          ALAN=ALW(SI,CI,RADIF)
          DIFF=ANG(ALVEH-ALAN)
          IF(ABS(DIFF).LT.1.E-2.OR.ABS(DIFF-TWOPI).LT.1.E-2)
     &    WRITE(11,702)ALVEH
          IF(ABS(DIFF-PI).LT.1.E-2)WRITE(11,703)ALVEH
        END IF

        WRITE(11,704)

  700 FORMAT(1X,'ITRAN1, MODE = ',2I5/1X,'DJ1, DJ2 = ',
     &       2F18.7/)
  701 FORMAT(1X,'DT1 = ',F12.7/1X,'WEDGE ANGLE = ',F9.5/)
  702 FORMAT(1X,'ARG OF LAT OF IMPULSE 1 = ',F9.5,'(ASCENDING NODE)'//)
  703 FORMAT(1X,'ARG OF LAT OF IMPULSE 1 = ',F9.5,'(DESCENDING NODE)'//)
  704 FORMAT(1X,'   COAST TIME    DELTA VEL    BURN TIME             PROP
     &ELLANTS AT END OF COAST             PROPELLANTS AT END OF BURN'/)
      END IF
C     ************************************************


      CALL CB180C(ITRAN1,DT1-.5*BT0,VA12,VD12,TI12,
     &            WA,PROP1,PROP2,BTF)


      IF(MD.EQ.2)THEN
C     SET TSTAY TIME FOR POSSIBLE USE IN PRECEDING 2BTO SEGMENT
C     --------------------------------------------------------
      TSTAY=DT1
      END IF



      IF(ABS(MD).NE.3)RETURN
C     ***********************************************************
C     COMPUTATIONS PAST THIS POINT ARE ONLY FOR MD = +/- 3  AND TAKE
C     INTO ACCOUNT THE ECCENTRICITIES OF THE ORBITS AND OTHER ACTUAL
C     ORBITAL PARAMETERS.  THESE COMPUTATIONS MUST BE MADE BEFORE
```

```
C     PROPER OUTPUT CAN BE GENERATED.
C     ****************************************************************

C     COMPUTE THE STATE FOR LEG = LG1, INCLUDING THE VALUE FOR
C     THE SPECIAL ARGUMENT OF LATITUDE (SUM OF MEAN ANOMALY AND
C     ARGUMENT OF PERIGEE).  NOTE THAT THE DJL(LG1) VALUE COMPUTED
C     HERE IS THE RESULT OF MEAN-ORBIT APPROXIMATIONS AND WILL BE
C     ADJUSTED LATER IF NECESSARY.
C     ----------------------------
      DJL(LG1)=DJ1 +DT1
      DO 625 I=1,6
  625 EML(LG1,I)=OEM(IORB1,I)
      DELDJ=DJL(LG1) -ODJ(IORB1)
      EML(LG1,3)=ANG(EML(LG1,3) +DMADT(IORB1)*DELDJ)
      EML(LG1,4)=EML(LG1,4) +DAPDT(IORB1)*DELDJ
      EML(LG1,6)=EML(LG1,6) +DRADT(IORB1)*DELDJ
      ARML1=EML(LG1,3) +EML(LG1,4)


      IF(NB12.GT.2)THEN
C     COMPUTE INTERMEDIATE ORBIT STATES (AT DJL(LG1)) (*** NOTE THAT
C     NB1F MUST BE AN EVEN NUMBER ***).  FIRST, PROJECT THE TARGET
C     ORBIT TO DJL(LG1) AND COMPUTE UNIT ECCENTRICITY AND ANGULAR
C     MOMENTUM VECTORS FOR THE INITIAL AND TARGET ORBITS AT THE
C     SAME TIME REFERENCE.
C     -----------------------------------------------------------
C     -----------------------------------------------------------
        DELDJ=DJL(LG1) -ODJ(IORB2)
        XMA=ANG(OEM(IORB2,3) +DMADT(IORB2)*DELDJ)
        ARP=OEM(IORB2,4) +DAPDT(IORB2)*DELDJ
        RAC=OEM(IORB2,6) +DRADT(IORB2)*DELDJ
        ACR(1)=ARP
        ACR(2)=OEM(IORB2,5)
        ACR(3)=RAC
        CALL CONREF(ACR,UP2,UT2,UM2)

        ACR(1)=EML(LG1,4)
        ACR(2)=EML(LG1,5)
        ACR(3)=EML(LG1,6)
        CALL CONREF(ACR,UP1,UT1,UM1)


C       COMPUTE THE NUMBER OF INTERMEDIATE ORBITS AND THE
C       MEAN-ORBIT APPROXIMATIONS FOR THEIR SEMIMAJOR AXES AND FOR
C       THEIR SPECIAL ARGUMENTS OF LATITUDE AT DJL(LG1).  ALSO COMPUTE
C       THE MEAN-ORBIT APPROXIMATION FOR THE TARGET ORBIT'S SPECIAL
C       ARGUMENT OF LATITUDE AT DJL(LG1)
C       --------------------------------
        NIO=(NB12-2)/2
        ISTOP=NIO+1

        DO 635 I=1,ISTOP
        JTOT=2*I-1
        XNUM=0.
```

```
      DO 630 J=1,JTOT
  630 XNUM=XNUM+TI12(J)
      IF(I.LT.ISTOP)THEN
        DENOM=TI12(JTOT+1)
        ARMLIO(I)=ARML1 +PI*(JTOT -XNUM/DENOM)
        SMAIO(I) =.5*(R12(2*I) +R12(2*I+1))
      ELSE
        ARML2= ARML1 +PI*JTOT -XNUM*DAL2DT
      END IF
  635 CONTINUE

C     ADJUST THE INTERMEDIATE ORBITS' SPECIAL ARGUMENTS OF LATITUDE
C     BASED ON A COMPARISON OF THE TARGET ORBIT'S ACTUAL PHASING
C     AND MEAN-APPROXIMATION PHASING.
C     ------------------------------
      ARLADJ=ANG(ARP+XMA-ARML2)
      IF(ARLADJ.GT.PI) ARLADJ=ARLADJ-TWOPI

C    ******** OUTPUT FOR CHECKOUT ONLY ************
      IF(MPRINT.NE.0) WRITE(11,707)ARLADJ
  707 FORMAT(1X,'ARLADJ = ',F10.6)
C    *********************************************

      DO 640 I=1,NIO
  640 ARMLIO(I)=ARMLIO(I) +(I*ARLADJ)/ISTOP


C     COMPUTE THE ORBITAL PARAMETER VALUES OF THE INTERMEDIATE ORBITS,
C     VALID AT DJL(LG1).  THESE VALUES CAN BE STORED IN THE EML ARRAY
C     AS THE LEG-PARAMETER VALUES FOR ALTERNATE LEGS IN THE TRANSFER
C     BETWEEN THE INITIAL AND TARGET ORBITS.  THE MEAN ANOMALY,
C     RIGHT ASCENSION, AND ARGUMENT OF PERIGEE VALUES WILL BE ADJUSTED
C     LATER TO CORRESPOND TO THE ACTUAL DJL VALUES FOR THOSE LEGS.
C     ------------------------------------------------------------
      CALL VCROSS(UM1,UM2,ZETA)
      SIGTOT=ASIN(VMAG(ZETA))
      CALL VUNIT(ZETA,ZETA)
      CALL VCROSS(ZETA,UM1,UNM1)

      DO 645 I=1,3
      VECC1(I)=EML(LG1,2)*UP1(I)
  645 VECC2(I)=OEM(IORB2,2)*UP2(I)

      FAC=SIGTOT/(1. +NIO)
      TEMP=1./(1. +NIO)

      DO 650 I=1,NIO
      LEG=LG1 +2*I
      SIG=FAC*I
      CSIG=COS(SIG)
      SSIG=SIN(SIG)
      UMIO(1)=CSIG*UM1(1) +SSIG*UNM1(1)
      UMIO(2)=CSIG*UM1(2) +SSIG*UNM1(2)
      UMIO(3)=CSIG*UM1(3) +SSIG*UNM1(3)
```

```
      SINIO=SQRT(UMIO(1)**2 +UMIO(2)**2)
      EML(LEG,5)=ATAN2(SINIO,UMIO(3))

      IF(EML(LEG,5).NE.0.)THEN
        EML(LEG,6)=ATAN2(UMIO(1),-UMIO(2))
      ELSE
        EML(LEG,6)=0.
      END IF

      UNIO(1)=COS(EML(LEG,6))
      UNIO(2)=SIN(EML(LEG,6))
      UNIO(3)=0.

      CALL VCROSS(UMIO,UNIO,ULIO)

      TEMP1=TEMP*I
      VECCIO(1)=VECC1(1)  +(VECC2(1)-VECC1(1))*TEMP1
      VECCIO(2)=VECC1(2)  +(VECC2(2)-VECC1(2))*TEMP1
      VECCIO(3)=VECC1(3)  +(VECC2(3)-VECC1(3))*TEMP1

      EML(LEG,2)=VMAG(VECCIO)

      CALL VUNIT(VECCIO,VECCIO)
      SINEPS=VDOT(VECCIO,UMIO)
      COSEPS=SQRT(1.-SINEPS**2)


C     ADJUST ECCENTRICITY VECTOR SO THAT IT IS PERPENDICULAR TO
C     ANGULAR MOMENTUM VECTOR
C     ------------------------------------------
      VECCIO(1)=(VECCIO(1)  -SINEPS*UMIO(1))/COSEPS
      VECCIO(2)=(VECCIO(2)  -SINEPS*UMIO(2))/COSEPS
      VECCIO(3)=(VECCIO(3)  -SINEPS*UMIO(3))/COSEPS

      EML(LEG,1)=SMAIO(I)*(1.-EML(LEG,2)**2)

      IF(EML(LEG,2).NE.0.)THEN
        EML(LEG,4)=ATAN2(VDOT(VECCIO,ULIO),VDOT(VECCIO,UNIO))
      ELSE
        EML(LEG,4)=0.
      END IF

      EML(LEG,3)=ARMLIO(I)  -EML(LEG,4)

  650 CONTINUE

    END IF
C   --------------------------------------------------------------
C   --------------------------------------------------------------


C   DETERMINE THE ORBITAL-PARAMETER VALUES FOR THE TRANSFER SEQUENCE
C   FROM THE INITIAL TO THE TARGET ORBIT.  ADJUST THE DJL VALUES
```

```
C     AND THE MEAN ANOMALY, ARGUMENT OF PERIGEE, AND RIGHT ASCENSION
C     FOR THE INITIAL ORBIT AND INTERMEDIATE ORBITS (IF ANY) IN THE
C     TRANSFER SEQUENCE.
C     ***********************************************************
      IF(NB12.EQ.2) ISTOP=1
      DJLREF=DJL(LG1)
      EPOC1=DJLREF
      EPOC2=EPOC1

C     <<<<<<<<<<<<<<<<<<<   L O N G   L O O P   <<<<<<<<<<<<<<<<<<<<<<
C     IN WHICH THE BASIC DATA ARAYS ARE COMPUTED FOR ALL SEGMENT LEGS
C     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      DO 720 I=1,ISTOP
      IK=2*I-2

      IF(I.EQ.1)THEN
        LEG1=LG1
        T1MIN=DJ1 +DT1 -.5*TI12(1)
        T1MAX=T1MIN +TI12(1)
        IF(T1MIN.LT.DJ1) T1MIN=DJ1
      ELSE
        LEG1=LEG1+2
        T1MIN=T2MAX +1.E-5
        T1MAX=T2MAX +.5*(TI12(IK) +TI12(IK+1))
      END IF

      IF(I.EQ.ISTOP)THEN
        T2MAX=T1MAX +TI12(IK+1)
        IF(T2MAX.GT. DJ2 +TSTAY) T2MAX=DJ2 +TSTAY
        EPOC2=ODJ(IORB2)
      ELSE
        T2MAX=T1MAX +.5*(TI12(IK+1) +TI12(IK+2))
      END IF

      LEG2=LEG1+2
      T2MIN=T1MAX +1.E-5


      DO 655 J=1,6
      E1(J)=EML(LEG1,J)

      IF(I.LT.ISTOP)THEN
        E2(J)=EML(LEG2,J)
      ELSE
        E2(J)=OEM(IORB2,J)
      END IF

  655 CONTINUE

C     A GLOBAL SEARCH ACROSS T1 AND T2 FOR MINIMUM DELTA-V IS FIRST MADE
C     WITH T1 VARIED BETWEEN T1MIN AND T1MAX IN STEPS CORRESPONDING TO
C     APPROXIMATELY 10 DEGREES IN MEAN ANOMALY.  FOR EACH VALUE OF T1,
C     T2 IS VARIED BETWEEN VALUES CORRESPONDING TO MEAN ANOMALIES OF
```

```
C     APPROXIMATELY 135 AND 225 DEGREES W.R.T THE ANOMALY CORRESPONDING
C     TO T1.  EACH T2 STEP ALSO CORRESPONDS TO APPROXIMATELY 10 DEGREES.
C     ------------------------------------------------------------------
      DVMIN=1.E20
      RPHIGH=0.
      FAC=(T2MAX-T2MIN)/(T1MAX-T1MIN)
      DTGRD1=.05*(T1MAX-T1MIN)
      DTGRD2=FAC*DTGRD1
      T1=T1MIN-DTGRD1

      DO 665 IGRID=1,21
      T1=T1+DTGRD1

      T2=T2MIN +FAC*(T1-T1MIN) -6.*DTGRD2

      DO 660 JGRID=1,11
      T2=T2+DTGRD2
      IF(T2.LT.T2MIN.OR.T2.GT.T2MAX) GO TO 660

      RPMIN=RPMIN-185200.
      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
      RPMIN=RPMIN+185200.

      IF(KV.NE.1) GO TO 660
      DVTOT=VMAG(DV1) +VMAG(DV2)
      RPER=ET(1)/(1.+ET(2))

      IF(DVTOT.LT.DVMIN.AND.RPER.GE.RPMIN)THEN
        DVMIN=DVTOT
        T1OPT=T1
        T2OPT=T2
      END IF

      IF(RPER.GT.RPHIGH)THEN
        RPHIGH=RPER
        T1HIGH=T1
        T2HIGH=T2
      END IF

 660    CONTINUE
 665  CONTINUE

      IF(DVMIN.GT..99E20)THEN

        IF(RPHIGH.LT..01)THEN
          CALL MSG('***HALT: SW01 GLOBAL SEARCH FAILED',3)
          STOP
        ELSE
          CALL MSG('GLOBAL-SEARCH PERIGEE TOO HIGH',1)
          T1OPT=T1HIGH
          T2OPT=T2HIGH
        END IF

      END IF
```

```
C     MAKE FIVE-LEVEL SEARCH, STARTING WITH THE T1OPT AND T2OPT
C     VALUES FROM THE GLOBAL SEARCH, TO REFINE THE OPTIMUM POINT.
C     ------------------------------------------------------------
      DO 675 NLEV=1,5
      DTGRD1=.5*DTGRD1
      T1REF=T1OPT

      DTGRD2=.5*DTGRD2
      T2REF=T2OPT

      DO 670 IJGRID=1,16
      T1=T1REF +IMUL(IJGRID)*DTGRD1
      T2=T2REF +JMUL(IJGRID)*DTGRD2
      IF(T2.GT.T2MAX) T2=T2MAX

      RPMIN=RPMIN-185200.
      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
      RPMIN=RPMIN+185200.

      IF(KV.NE.1) GO TO 670
      DVTOT=VMAG(DV1) +VMAG(DV2)
      RPER=ET(1)/(1.+ET(2))

      IF(DVTOT.LT.DVMIN.AND.RPER.GE.RPMIN)THEN
        DVMIN=DVTOT
        T1OPT=T1
        T2OPT=T2
      END IF

      IF(RPER.GT.RPHIGH)THEN
        RPHIGH=RPER
        T1HIGH=T1
        T2HIGH=T2
      END IF

  670 CONTINUE

      IF(DVMIN.GT..99E20)THEN
        T1OPT=T1HIGH
        T2OPT=T2HIGH
      END IF

  675 CONTINUE

      T1=T1OPT
      T2=T2OPT

C     RELAX RPMIN FOR TRNSFR MODE 3 CALCULATIONS
C     ------------------------------------------
      RPMIN=RPMIN -18520.

      CALL TRNSFR(EPOC1,E1,EPOC2,E2,T1,T2,1,1,ET,DV1,DV2,KV)
```

```
      IF(KV.NE.1)THEN
        IF(KV.EQ.-1)CALL MSG('NO FPO TRNSFR WITH ACCEPTABLE PERIGEE',3)
        IF(KV.NE.-1)CALL MSG('FAILURE TO FIND FPO TRNSFR',3)
        RPMIN=RPMIN +18520.
        STOP
      END IF

C     RESTORE RPMIN TO NOMINAL VALUE
C     ------------------------------
      RPMIN=RPMIN +18520.


C     STORE PARAMETER VALUES FOR THE TRANSFER LEGS.  FIRST, PROJECT
C     THE ET VALUES TO T2. THIS SECTION APPLIES TO LEGS LG1+1, LG1+3,
C     LG1+5, ...LG2.
C     -----------------------------------------------------------------
      LEG=LEG1 +1
      DJL(LEG)=T2
      SIT=SIN(ET(5))
      CIT=COS(ET(5))
      CALL ORBMOVE(ET(1),ET(2),SIT,CIT,RAD,APD,XMD)
      DELDJ=T2-T1
      ET(3)=ET(3) +XMD*DELDJ
      ET(4)=ET(4) +APD*DELDJ
      ET(6)=ET(6) +RAD*DELDJ

      DO 685 J=1,6
  685 EML(LEG,J)=ET(J)

      RADOT(LEG)=RAD
      APDOT(LEG)=APD
      XMDOT(LEG)=XMD

      DO 690 J=1,3
  690 DVCL(LEG,J)=DV2(J)


C     ADJUST PARAMETER VALUES FOR LEGS PRECEDING THE TRANSFER LEGS
C     AND STORE DELTA-V COMPONENTS.  THIS SECTION APPLIES TO LEG=LG1,
C     LG1+2, LG1+4, ...LG2-1.
C     -------------------------------------------------------
      LEG=LEG1
      DJL(LEG)=T1

      IF(LEG.EQ.LG1)THEN
        RAD=DRADT(IORB1)
        APD=DAPDT(IORB1)
        XMD=DMADT(IORB1)
      ELSE
        SIT=SIN(EML(LEG,5))
        CIT=COS(EML(LEG,5))
        CALL ORBMOVE(EML(LEG,1),EML(LEG,2),SIT,CIT,RAD,APD,XMD)
```

```
      END IF

      DELDJ=DJL(LEG) -DJLREF
      EML(LEG,3)=ANG(EML(LEG,3) +XMD*DELDJ)
      EML(LEG,4)=EML(LEG,4) +APD*DELDJ
      EML(LEG,6)=EML(LEG,6) +RAD*DELDJ

      RADOT(LEG)=RAD
      APDOT(LEG)=APD
      XMDOT(LEG)=XMD

      DO 695 J=1,3
  695 DVCL(LEG,J)=DV1(J)

  720 CONTINUE
C     <<<<<<<<<<<<<<<<<  END LONG LOOP  <<<<<<<<<<<<<<<<<<<<<<<<<<<


C     ******************
C     SET DJ2 TO DJL(LG2)
C     ******************
      DJ2=DJL(LG2)



C     ******* OUTPUT FOR CHECKOUT ONLY *******************
      IF(MPRINT.NE.0)THEN
        WRITE(11,708)
  708   FORMAT(1X,'LEG',3X,'DELTAV',4X,'TLAPSE',11X,'SLR',8X,'ECC',8X,
     &             'XMA',8X,'ARP',8X,'INC',8X,'RAC'/)

        DO 750 LEG=LG1,LG2
        TLAPSE=DJL(LEG) -DJ1
        DO 740 J=1,3
  740   DV1(J)=DVCL(LEG,J)
        DELTAV=VMAG(DV1)
        WRITE(11,709)LEG,DELTAV,TLAPSE,(EML(LEG,J),J=1,6)
  709   FORMAT(1X,I3,F9.2,F10.5,F14.0,F11.6,4F11.4)
  750   CONTINUE

      END IF
C     *****************************************************



C     COMPUTE THE PROPELLANT REQUIREMENTS FOR THE VARIOUS LEGS AND
C     STORE DATA INTO ARRAYS NEEDED BY THE CALLING PROGRAM TO PRODUCE
C     THE OUTPUT FILES.
C     ------------------------------------------------------------
      NLTRAN(ITRAN1)=NB12

      LEG=LG1-1
      BPF=0.
      TLAST=DJ1
```

```
      BTLAST=BT0

      DO 760 M=1,6
  760 PIN(M)=PROP1(M)

C     ***********************************
C     CHECKOUT PRINTOUT
C     -----------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,900)
  900 FORMAT(/1X,'LEG',4X,'COAST TIME',5X,'BURN TIME',17X,'PROPELLANTS A
     &T END OF COAST',16X,'PROPELLANTS AT END OF BURN'/)
      END IF
C     ***********************************


C     ***********************************
C     CHECKOUT PRINTOUT
C     -----------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,901)ITRAN1
  901 FORMAT(/1X,'BEGIN TRANSFER NO. ',I3/)
      END IF
C     ***********************************

      DO 790 LIT=1,NLTRAN(ITRAN1)
      LEG=LEG+1

      DO 765 J=1,3
  765 DV1(J)=DVCL(LEG,J)

      DELV=VMAG(DV1)
      DTIP=DJL(LEG) -TLAST -.5*BTLAST

      CALL CSTBRN(ITRAN1,LIT,DTIP,BPF,DELV,PIN,PCOUT,PROP2,CTL,BTL)
      TCSTL(LEG)=CTL
      TBRNL(LEG)=BTL
      TLAST=DJL(LEG)
      BTLAST=BTL

      DO 780 M=1,6
      PLEGC(LEG,M)=PCOUT(M)
      PLEGB(LEG,M)=PROP2(M)
  780 PIN(M)=PROP2(M)

C     ******************************************
C     CHECKOUT PRINTOUT
C     ------------------------------------------
      IF(MPRINT.NE.0)THEN
      WRITE(11,902)LEG,TCSTL(LEG),TBRNL(LEG),(PLEGC(LEG,M),M=1,3),
     &                (PLEGB(LEG,M),M=1,3)
  902 FORMAT(1X,I3,2E14.5,2X,3E14.5,2X,3E14.5)
      END IF
```

```
C       ********************************************

  790 CONTINUE

      BTF=BTLAST

      RETURN
      END
```

```
      SUBROUTINE    SW03(MD,ITRAN1,LG1,BTO,DJ1,PROP1,DJ2,PROP2,LG2,BTF,KFB)

      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL*4 LFILL, LGEOM, LATASC

C     SW03 IS THE SEGMENT WORKER THAT SOLVES THE MISSION SEGMENT
C     CONSISTING OF A ONE-BURN TRANSFER (1BRC) TO A REENTRY CONIC
C     AND A TWO-BURN TRANSFER (2BSO) TO A SAVING ORBIT.
C     ****************************************************************
C     *** NOTE THAT THIS VERSION DOES NOT COMPUTE CONTINUOUS BURNS ***
C     ****************************************************************

C     SW03 IS RESPONSIBLE FOR COMPUTING THE MEAN ANOMALY, INCLINATION,
C     AND RIGHT ASCENSION OF THE SAVING ORBIT.  THE ENTIRE SEGMENT IS
C     ACCOMPLISHED WITHOUT ANY OUT-OF-PLANE IMPULSE COMPONENT.

C     INPUTS: ---------------------------------------------------------

C     MD    = 1    INSTRUCTS SW03 TO ESTIMATE THE MINIMUM-PROPELLANT
C                  MISSION SEGMENT, BEGINNING AT DJ1.  DJ2 IS AN OUTPUT.
C                  (THIS SOLUTION IS ALSO CONSIDERED TO BE MINIMUM-TIME)

C     MD    = 2    INSTRUCTS SW03 TO ESTIMATE THE MINIMUM-PROPELLANT
C                  MISSION SEGMENT BETWEEN DJ1 AND DJ2.  DJ2 IS AN INPUT,
C                  BUT SW03 WILL ADJUST IT TO THE NEAREST OPPORTUNITY.
C                      ----------------------------------------------
C                      NOTE: IN MODE 2, THIS SEGMENT WORKER SETS THE STAY
C                      TIME IN THE INITIAL ORBIT EQUAL TO "TSTAY" AND
C                      PASSES IT TO THE TOP-LEVEL OPTIMIZER THRU COMMON
C                      BLOCK IMA60.  THIS TIME MAY BE USED TO RESTRICT
C                      THE DJ2 VARIATION IN A PRECEDING 2BTO SEGMENT.
C                      ----------------------------------------------

C     MD    = 3    SAME AS MD=2, EXCEPT THAT SOME COMPUTATIONS ARE MORE
C                  ACCURATE AND OUTPUT DATA IS GENERATED.

C     ****************** NOTE *****************************************
C     *****************************************************************
C     *  SW03 MINIMIZES THE DELTA-V OF THE 1BRC IMPULSE IN DEFINING   *
C     *  THE REENTRY CONIC.  ONCE THE REENTRY CONIC IS DEFINED, THE   *
C     *  2BSO TRANSFER IS OPTIMIZED TO MINIMIZE THE DELTAV OF THAT    *
C     *  TRANSFER.  THE DESCENDING COAST TIME, AFTER THE 1BRC IMPULSE,*
C     *  IS SET EQUAL TO THE "COAST TIME BEFORE SAVING IMPULSE"(CTBSI)*
C     *  SPECIFIED BY THE USER.  THIS PIECE-WISE OPTIMIZATION OF THE  *
C     *  TOTAL DELTA-V SHOULD BE A GOOD APPROXIMATION OF A MINIMUM-   *
C     *  PROPELLANT SOLUTION, GIVEN THE FACT THAT A PAYLOAD IS        *
C     *  USUALLY DISCARDED AFTER THE 1BRC IMPULSE.                    *
C     *****************************************************************

C     ITRAN1    = NUMBER OF THE FIRST TRANSFER IN THE SEGMENT
C     LG1       = NUMBER OF FIRST LEG IN THE SEGMENT
C     BT0       = BURN TIME JUST PRIOR TO START OF THE SEGMENT
C     DJ1       = JULIAN DATE AT START OF THE SEGMENT
C     PROP1(M)  = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ1
```

```
C     DJ2       = JULIAN DATE AT END OF SEGMENT (INPUT IF MD.GE.2)

C     OUTPUTS:---------------------------------------------------------------
C     DJ2       = JULIAN DATE AT END OF SEGMENT (OUTPUT IF MD.LT.2)
C     PROP2(M) = PROPELLANT REMAINING IN PROPULSION SUBSYSTEM M AT DJ2
C     LG2       = NUMBER OF LAST LEG IN THE SEGMENT
C     BTF       = FINAL BURN TIME IN THIS SEGMENT

C     KFB   =0
C     EVERYTHING IS O.K.

C     KFB   =1
C     THE INITIAL-ORBIT STAY TIME RESTRICTIONS DO NOT ALLOW A FEASIBLE
C     OPPORTUNITY (RARE OCCURRENCE).

C     ******************************************************
C     NOTE: ALL TIMES IN THIS SUBROUTINE ARE EXPRESSED IN DAYS
C     ******************************************************

      COMMON/IMA04/XMU,XJ2,XJ3,XJ4,REQ,RPL
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA20/NORB0,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA40/RPMIN
      COMMON/IMA42/MPRINT
      COMMON/IMA44/DJL(40),EML(40,6),DVCL(40,3),TCSTL(40),TBRNL(40)
      COMMON/IMA46/NLTRAN(15),PLEGC(40,6),PLEGB(40,6)
      COMMON/IMA48/RADOT(40),APDOT(40),XMDOT(40)
      COMMON/IMA58/NBIT(15),NBITS(15)
      COMMON/IMA60/ORBMIN(15),ORBMAX(15), TSTAY

      DIMENSION PROP1(6), PROP2(6)

      DIMENSION E1(6), ERE(6), E2(6), DV1(3), DV2(3)
      DIMENSION PIN(6), PCOUT(6)
      DIMENSION VTC(2), VTR(2), ESC(6)
      DIMENSION DVMINT(16)
      DIMENSION X1(6), X2(6)

      DATA KERR/0/, DTTOL/.00002/
      DATA ANGL0/1.57079633/, DANGL/.26179939/
C     ----------------------------------------------------------

      KFB=0

C     COMPUTE I.D. NUMBERS OF INITIAL AND FINAL ORBITS AND GIVE LOCAL
C     NAMES TO THE SINES AND COSINES OF THE MEAN ORBITAL INCLINATIONS
C     ----------------------------------------------------------------
      IF(ITRAN1.EQ.1) IORB1=NORB0
      IF(ITRAN1.GT.1) IORB1=NTORB(ITRAN1-1)
```

```
      IORB2=NTORB(ITRAN1+1)

      IF(KIND(IORB2).EQ.3.OR.KIND(IORB2).EQ.5.OR.KIND(IORB2).EQ.7)THEN
        CALL MSG('***HALT: SAVING ORBIT FOR 2BSO TRANSFER CANNOT HAVE A
     &FREE ARGUMENT OF PERIGEE OR A FREE GMT',3)
        STOP
      END IF

      IF(KIND(IORB2).NE.6.AND.KERR.EQ.0)THEN
        CALL MSG('WARNING: TOO MANY PARAMETERS HAVE BEEN SET FOR A 2BSO
     &SAVING ORBIT.  ANY SET VALUE FOR THE ANOMALY, INCLINATION, OR RIGH
     &T ASCENSION WILL BE IGNORED',1)
        KERR=1
      END IF


      SI=SININC(IORB1)
      CI=COSINC(IORB1)

C     COMPUTE THE AVERAGE SECULAR RATE OF ARGUMENT OF LATITUDE, THE
C     "ARGUMENT-OF-LATITUDE" PERIOD, AND THE MINIMUM AND MAXIMUM
C     ALLOWABLE STAY TIMES FOR THE INITIAL ORBIT
C     ------------------------------------------------------------
C     NOTE: THE MINIMUM AND MAXIMUM ALLOWABLE STAY TIMES ARE BASED
C     ON IMPULSIVE DELTA-V ASSUMPTIONS.  THE ACTUAL COASTING STAY
C     TIMES WILL BE LESS THAN THE TIMES BETWEEN IMPULSES SO AS TO
C     PROVIDE ROOM FOR THE FINITE BURN TIMES.
C     ------------------------------------------------------------
      DAL1DT =DAPDT(IORB1) +DMADT(IORB1)
      PERIOD1=TWOPI/DAL1DT
      DTMIN=PERIOD1*ORBMIN(ITRAN1)
      DTMAX=PERIOD1*ORBMAX(ITRAN1)


C     COMPUTE THE FINAL LEG NUMBER FOR THE SEGMENT, AND SET THE NUMBER
C     OF LEGS IN EACH TRANSFER
C     -------------------------
      LG2=LG1 +2
      NLTRAN(ITRAN1)=1
      NLTRAN(ITRAN1+1)=2


C     SET THE INCLINATION OF IORB2 EQUAL TO THAT OF IORB1,
C     COMPUTE SINE AND COSINE OF THE INCLINATION,
C     AND COMPUTE THE SECULAR RATES FOR ORBIT NO. IORB2
C     ----------------------------------------------
      OEM(IORB2,5)=OEM(IORB1,5)
      SININC(IORB2)=SIN(OEM(IORB2,5))
      COSINC(IORB2)=COS(OEM(IORB2,5))
      SLR2=OEM(IORB2,1)
      ECC2=OEM(IORB2,2)
      CALL ORBMOVE(SLR2,ECC2,SI,CI,RAD,APD,XMD)
      DRADT(IORB2) =RAD
      DAPDT(IORB2) =APD
```

```
      DMADT(IORB2) =XMD

C     SET LOCAL TRANSFER GEOMETRY PARAMETERS
C     --------------------------------------
      RRE=GEOM(ITRAN1,5) +REQ
      FPARE=GEOM(ITRAN1,6)

      IF(FPARE.GE.0.)THEN
        CALL MSG('*** HALT: REENTRY FLIGHT PATH ANGLE MUST BE NEGATIVE',
     &             3)
        STOP
      END IF

      SFPARE=SIN(FPARE)
      CFPARE=COS(FPARE)
      XLTRE=GEOM(ITRAN1,7)
      LATASC=.TRUE.
      IF(GEOM(ITRAN1,8).LT.0.) LATASC=.FALSE.
      CTBSI=GEOM(ITRAN1+1,5)/86400.
      STBUF=GEOM(ITRAN1+1,6)/86400.

      IF(ABS(XLTRE).GT.ASIN(SI))THEN
        CALL MSG('*** HALT: REENTRY LATITUDE CANNOT BE REACHED',3)
        STOP
      END IF

      SARLRE=SIN(XLTRE)/SI
      ARLRE=ASIN(SARLRE)
      IF(.NOT.LATASC) ARLRE= PI-ARLRE


C     COMPUTE THE SEMIMAJOR AXES OF THE INITIAL AND SAVING ORBITS
C     -----------------------------------------------------------
      SMA1=OEM(IORB1,1)/(1. -OEM(IORB1,2)**2)
      SMA2=OEM(IORB2,1)/(1. -OEM(IORB2,2)**2)


      DO 10 J=1,6
   10 E1(J)=OEM(IORB1,J)


C     INITIALIZE FOR ITERATIVE PROCESS TO DETERMINE REENTRY CONIC
C     -----------------------------------------------------------
      T1MIN=DJ1 +DTMIN
      T1MAX=DJ1 +DTMAX
      T1START=T1MIN

C     LOOP BACK TO THIS POINT IF STAY TIME IN STARTING ORBIT MUST
C     BE CHANGED TO PRODUCE END TIME CLOSE TO INPUT VALUE OF DJ2
C     ----------------------------------------------------------
   15 T1=T1START
      T1STOP=T1START +PERIOD1
      IF(T1STOP.GT.T1MAX) T1STOP=T1MAX
      DT1=.002
```

```
      NPHASE=1
      DVMIN=9.E9

      IF(MPRINT.EQ.1)THEN
        WRITE(11,1001)T1MIN,T1MAX,T1STOP
 1001   FORMAT(1X,'T1MIN, MAX, STOP =',3F17.8//1X,'NPHASE',13X,'T1',
     &  10X,'SLRRE',10X,'ECCRE',10X,'TRARE',11X,'TRA1',11X,
     &  'RLIM',13X,'DV',2X,'LEVEL'/)
      END IF

C     COARSE (NPHASE=1) AND FINE (NPHASE=2) SEARCH PHASES LOOP BACK
C     TO THIS POINT IN ITERATIVE PROCESS TO DETERMINE THE REENTRY CONIC
C     -----------------------------------------------------------------
   20 DELDJ=T1-ODJ(IORB1)
      E1(3)=ANG(OEM(IORB1,3) +DMADT(IORB1)*DELDJ)
      CALL KEPLE(E1(2),E1(3),DUM,TRA11)
      E1(4)=OEM(IORB1,4) +DAPDT(IORB1)*DELDJ
      E1(6)=OEM(IORB1,6) +DRADT(IORB1)*DELDJ

      R1=OEM(IORB1,1)/(1. +E1(2)*COS(TRA11))
      DARL=ANG(ARLRE -TRA11 -E1(4))
      SPHI=R1*SFPARE*(COS(DARL)-1.)
      CPHI=R1*(CFPARE +SFPARE*SIN(DARL)) -RRE*CFPARE

      TRARE= -ATAN2(SPHI,CPHI)
      IF(TRARE.GT.0.) TRARE =TRARE-PI
      LEVEL=0
      IF(TRARE.GT.(FPARE-PIO2).AND.TRARE.LT.FPARE)THEN
        ECCRE=SFPARE/SIN(TRARE-FPARE)
        LEVEL=1
        IF(ECCRE.GT.0.AND.ECCRE.LE..99)THEN
          TRA1=ANG(TRARE-DARL)
          SLRRE=R1*(1.+ECCRE*COS(TRA1))
          SMARE=SLRRE/(1.-ECCRE**2)
          RPRE=SLRRE/(1.+ECCRE)
          XMMRE=SQRT(XMU/SMARE**3)*86400.
          XMARE1=ANG(XMANOM(ECCRE,TRA1))
          DTPRE=(TWOPI-XMARE1)/XMMRE
          LEVEL=2
          IF((CTBSI+STBUF).LT.DTPRE)THEN
            XMALIM=XMARE1 +(CTBSI+STBUF)*XMMRE
            CALL KEPLE(ECCRE,XMALIM,DUM,TRALIM)
            RLIM=SLRRE/(1.+ECCRE*COS(TRALIM))
            LEVEL=3
            IF(RLIM.GT.RPMIN)THEN
C           COMPUTE DELTA-V AND COMPARE WITH MINIMUM
C           ----------------------------------------
              FAC1=SQRT(XMU/E1(1))
              FACRE=SQRT(XMU/SLRRE)
              VC1=FAC1*(1.+E1(2)*COS(TRA11))
              VCRE=FACRE*(1.+ECCRE*COS(TRA1))
              VR1=FAC1*E1(2)*SIN(TRA11)
              VRRE=FACRE*ECCRE*SIN(TRA1)
```

```
                DV=SQRT((VCRE-VC1)**2 +(VRRE-VR1)**2)
                LEVEL=4
                IF(DV.LT.DVMIN)THEN
                  DVMIN=DV
                  T1OPT=T1
                  ERE(1)=SLRRE
                  ERE(2)=ECCRE
                  ERE(3)=XMARE1
                  ERE(4)=TRA11 +E1(4) -TRA1
                  ERE(5)=E1(5)
                  ERE(6)=E1(6)
                  LEVEL=5
                END IF
              END IF
            END IF
          END IF
        END IF

        IF(MPRINT.EQ.1)THEN
          WRITE(11,1002)NPHASE,T1,SLRRE,ECCRE,TRARE,TRA1,RLIM,DV,LEVEL
 1002     FORMAT(1X,I7,7E15.5,I7)
        END IF

        IF(NPHASE.EQ.1)THEN
          IF(T1.EQ.T1STOP)THEN
            IF(DVMIN.EQ.9.E9)THEN
              CALL MSG('*** HALT: SW03 UNABLE TO FIND FEASIBLE REENTRY CON
     &IC',3)
              STOP
            END IF
            NPHASE=2
            DT1=.5*DT1
            T1=T1OPT-DT1
            NEGSTP=1
          ELSE
            T1=T1+DT1
            IF(T1.GT.T1STOP) T1=T1STOP
          END IF
          GO TO 20
        END IF


C       PHASE 2 CONTROL LOGIC
C       ---------------------
        IF(NEGSTP.EQ.0)THEN
          IF(DT1.LE.DTTOL) GO TO 100
          T1=T1OPT-DT1
          NEGSTP=1
        ELSE
          T1=T1+2.*DT1
          DT1=.5*DT1
          NEGSTP=0
        END IF
        GO TO 20
```

```
  100 CONTINUE

      IF(MD.GE.2)THEN
C     SEE IF MORE OR LESS ORBITS IN THE STARTING ORBIT ARE REQUIRED
C     TO GET A SEGMENT END TIME CLOSE TO THE INPUT VALUE OF DJ2.  IF
C     SO, LOOP BACK AND GET NEW SOLUTION FOR REENTRY CONIC.
C     -----------------------------------------------------
        T2EST=T1OPT +PERIOD1
        OSTAY=(DJ2-T2EST)/PERIOD1
        NSTAY=OSTAY+.5
        IF(NSTAY.GT.0)THEN
          T1START=T1START +NSTAY*PERIOD1
          IF(T1START.GT.(T1MAX-PERIOD1))THEN
            KFB=1
            RETURN
          END IF
          GO TO 15
        END IF
      END IF



C     SAVE MINIMUM DELTA-V FOR REENTRY IMPULSE
C     AND PROJECT E1 VALUES TO T1OPT
C     ------------------------------
      DV1RE =DVMIN
      DELDJ=T1OPT-ODJ(IORB1)
      E1(3)=OEM(IORB1,3) +DMADT(IORB1)*DELDJ
      E1(4)=OEM(IORB1,4) +DAPDT(IORB1)*DELDJ
      E1(6)=OEM(IORB1,6) +DRADT(IORB1)*DELDJ

C     COMPUTE THE DV1RE VECTOR COMPONENTS IN ECID COORDINATES
C     -----------------------------------------------------
      CALL KEPLE(E1(2),E1(3),DUM,E1(3))
      CALL KEPLE(ERE(2),ERE(3),DUM,ERE(3))
      CALL OITRAN(E1,X1,1)
      CALL OITRAN(ERE,X2,1)
      DO 105 J=1,3
  105 DVCL(LG1,J) = X2(J)-X1(J)

      E1(3)=XMANOM(E1(2),E1(3))
      ERE(3)=XMANOM(ERE(2),ERE(3))

C     CHECK RESULTS
C     -------------
      IF(MPRINT.EQ.1)THEN
        DV1RECK=SQRT(DVCL(LG1,1)**2 +DVCL(LG1,2)**2 +DVCL(LG1,3)**2)
        DVERR=DV1RECK-DV1RE
        WRITE(11,555)DVERR
  555   FORMAT(/1X,'**** DVERR=',F10.5)
      END IF
```

```
C     REENTRY CONIC HAS BEEN DEFINED IN ERE(J) ARRAY,
C     WITH REFERENCE TIME = T1OPT.  NOW DETERMINE STARTING STATE
C     FOR THE TWO-IMPULSE 2BSO TRANSFER BY EXTRAPOLATING
C     ERE(J) TO TIME OF SAVING IMPULSE (TSI).
C     --------------------------------------------------
      CALL ORBMOVE(ERE(1),ERE(2),SI,CI,RAD2,APD2,XMD2)
      TSI=T1OPT +CTBSI
      ERE(3)=ERE(3) +XMD2*CTBSI
      ERE(4)=ERE(4) +APD2*CTBSI
      ERE(6)=ERE(6) +RAD2*CTBSI
C     ************************************************************
C     NOTE THAT THE UNCOMPENSATED SECULAR ROTATION OF THE REENTRY
C     CONIC'S ARGUMENT OF PERIGEE WILL PRODUCE A SMALL ERROR IN THE
C     LATITUDE OF THE REENTRY POINT.  THIS ERROR IS VERY SMALL.
C     ************************************************************

      IF(MPRINT.EQ.1)THEN
        WRITE(11,7654)T1OPT,E1
 7654   FORMAT(/1X,' T1,  E1 =',F16.8,F14.2,5F14.9)
        WRITE(11,1003)TSI,ERE
        WRITE(11,1007)DV1RE
 1003   FORMAT(/1X,'TSI, ERE =',F16.8,F14.2,5F14.9)
 1007   FORMAT(1X,'DV1RE = ',F10.2)
      END IF

      CALL KEPLE(ERE(2),ERE(3),DUM,TRASI)
      CTRASI=COS(TRASI)
      RSI=ERE(1)/(1.+ERE(2)*CTRASI)
      ARLSI=ERE(4) +TRASI
      FAC=SQRT(XMU/ERE(1))
      VREC=FAC*(1.+ERE(2)*CTRASI)
      VRER=FAC*ERE(2)*SIN(TRASI)


C     PERFORM TWO-VARIABLE COARSE SEARCH (VARYING THE TRANSFER TIME
C     AND ANGLE) TO APPROXIMATELY DEFINE THE MINIMUM-DELTA-V 2BSO
C     TRANSFER.  IGNORE SECULAR ROTATIONS OF THE TRANSFER CONIC.
C     -----------------------------------------------------------
      PERAV=.5*TWOPI*(1./(DMADT(IORB2) +DAPDT(IORB2))
     &               +1./(XMD2 +APD2))
      FAC=SQRT(XMU/OEM(IORB2,1))
      DVMIN=9.E9
      ANGL=ANGL0-DANGL
      ISTOP=16
      IOP=0

  110 DO 200 I=1,ISTOP
      ANGL=ANGL+DANGL
      ARL22=ARLSI +ANGL
      TMEREF=ANGL*PERAV/TWOPI
      TME0=.80*TMEREF
      TMESTP=1.2*TMEREF
```

```
      DTME=(TMESTP-TME0)/16.
      TME=TME0
      NPHASE=1
      DVMINT(I)=9.E9

  120 T2=TSI +TME
      ARP22=OEM(IORB2,4) +DAPDT(IORB2)*(T2-ODJ(IORB2))
      TRA22=ARL22 -ARP22
      CTRA22=COS(TRA22)
      V2C=FAC*(1. +OEM(IORB2,2)*CTRA22)
      V2R=FAC*OEM(IORB2,2)*SIN(TRA22)
      R2=OEM(IORB2,1)/(1. +OEM(IORB2,2)*CTRA22)
C     -----------------------------------------------------------
      CALL GAUSS(RSI,R2,ANGL,TME,VTC,VTR,TRASC,SLRSC,ECCSC,KGERR)
C     -----------------------------------------------------------
      IF(KGERR.EQ.0)THEN
        DVRESC=SQRT((VTC(1)-VREC)**2 +(VTR(1)-VRER)**2)
        DVSC2=SQRT((V2C-VTC(2))**2  +(V2R-VTR(2))**2)
        DV=DVRESC +DVSC2

        IF(MPRINT.EQ.1)THEN
          WRITE(11,1004)I,DV,ANGL,TME
 1004     FORMAT(1X,'I, DV, ANGL, TME = ',I3,F10.2,F10.5,F10.6)
        END IF

        IF(DV.LT.DVMINT(I))THEN
          DVMINT(I)=DV
          ANGLT=ANGL
          TMET=TME
        END IF
      END IF

      IF(NPHASE.EQ.1)THEN

        IF((TME+DTME).GT.TMESTP)THEN
          NPHASE=2
          DTME=.5*DTME
          TME=TMET-DTME
          NEGSTP=1
        ELSE
          TME=TME+DTME
        END IF
        GO TO 120
      END IF


C     PHASE 2 CONTROL LOGIC
C     ---------------------
      IF(NEGSTP.EQ.0)THEN
        IF(DTME.LE.DTTOL) GO TO 190
        TME=TMET-DTME
        NEGSTP=1
      ELSE
        TME=TME+2.*DTME
```

```
        DTME=.5*DTME
        NEGSTP=0
      END IF
      GO TO 120


  190 IF(DVMINT(I).LT.DVMIN)THEN
        IOP=I
        DVMIN=DVMINT(I)
        ANGLOP=ANGLT
        TMEOP=TMET
      END IF

  200 CONTINUE


      IF(ISTOP.EQ.16)THEN

        IF(IOP.EQ.0)THEN
          CALL MSG('*** HALT: SW03 UNABLE TO FIND FEASIBLE SAVING TRANSF
     &ER',3)
          STOP
        END IF

        IF(IOP.GT.1.AND.IOP.LT.16.AND.DVMINT(IOP-1).NE.9.E9.
     &         AND.DVMINT(IOP+1).NE.9.E9)THEN

C       USE PARABOLIC FIT TO DETERMINE THE OPTIMUM TRANSFER ANGLE
C       ---------------------------------------------------------
          DENOM=DVMINT(IOP-1) -2.*DVMIN +DVMINT(IOP+1)
          ANGL=ANGLOP +.5*(DVMINT(IOP-1) -DVMINT(IOP+1))*DANGL/DENOM
     &          -DANGL
          ISTOP=1
          GO TO 110
        END IF
      END IF


C     COMPUTE THE MINIMUM DELTA-V
C     AND OPTIMUM ELEMENTS FOR THE SAVING CONIC
C     -----------------------------------------
      TME=TMEOP
      ANGL=ANGLOP
      ARL22=ARLSI +ANGL
      T2=TSI +TME
      ARP22=OEM(IORB2,4) +DAPDT(IORB2)*(T2-ODJ(IORB2))
      TRA22=ARL22 -ARP22
      CTRA22=COS(TRA22)
      V2C=FAC*(1. +OEM(IORB2,2)*CTRA22)
      V2R=FAC*OEM(IORB2,2)*SIN(TRA22)
      R2=OEM(IORB2,1)/(1. +OEM(IORB2,2)*CTRA22)
C     ---------------------------------------------------------
      CALL GAUSS(RSI,R2,ANGL,TME,VTC,VTR,TRASC,SLRSC,ECCSC,KGERR)
C     ---------------------------------------------------------
```

```
      DVRESC=SQRT((VTC(1)-VREC)**2 +(VTR(1)-VRER)**2)
      DVSC2=SQRT((V2C-VTC(2))**2  +(V2R-VTR(2))**2)

C     LOAD SAVING-CONIC ELEMENTS, VALID AT T2, INTO ESC(I)
C     ----------------------------------------------------
      CALL ORBMOVE(SLRSC,ECCSC,SI,CI,RAD,APD,XMD)
      ESC(1)=SLRSC
      ESC(2)=ECCSC
      ESC(3)=ANG(XMANOM(ECCSC,TRASC) +XMD*TME)
      CALL KEPLE(ECCSC,ESC(3),DUM,TRASC2)
      ESC(4)=ARL22-TRASC2
      ESC(5)=OEM(IORB1,5)
      ESC(6)=ERE(6) +RAD*TME

      IF(MPRINT.EQ.1)THEN
        WRITE(11,1005)DVMIN,ANGLOP,TMEOP
        WRITE(11,1006)T2,ESC
        WRITE(11,1008)DVRESC, DVSC2
 1005   FORMAT(/1X,'DVMIN, ANGLOP, TMEOP = ',F10.2,F10.5,F10.6)
 1006   FORMAT(1X,'T2, ESC(I) = ',F16.8,F14.2,5F14.9)
 1008   FORMAT(1X,'DVRESC, DVSC2 = ',2F10.2)
      END IF

      DJ2=T1OPT +CTBSI +TMEOP

C     COMPUTE ELEMENTS OF SAVING ORBIT, VALID AT ODJ(IORB2).  THE RIGHT
C     ASCENSION IS AN APPROXIMATION THAT WILL BE IMPROVED  WHEN MD=3.
C     ---------------------------------------------------------------
      DELDJ=T2-ODJ(IORB2)
      OEM(IORB2,3)=XMANOM(OEM(IORB2,2),TRA22) -DMADT(IORB2)*DELDJ
      OEM(IORB2,6)=ESC(6) -DRADT(IORB2)*DELDJ



      IF(MD.EQ.3)THEN
C     ********************************************
C     THE FOLLOWING COMPUTATIONS ARE FOR MD=3 ONLY
C     ********************************************

C       COMPUTE MEAN ELEMENTS OF SAVING ORBIT, VALID AT T2, EXCEPT
C       SET RIGHT ASCENSION EQUAL TO THAT OF THE REENTRY CONIC
C       ------------------------------------------------------
        E2(1)=OEM(IORB2,1)
        E2(2)=OEM(IORB2,2)
        E2(3)=XMANOM(E2(2),TRA22)
        E2(4)=ARP22
        E2(5)=OEM(IORB2,5)
        E2(6)=ERE(6)

C       RELAX RPMIN FOR MODE 3 CALCULATIONS
C       -----------------------------------
```

```
      RPMIN=RPMIN -18520.
C     COMPUTE ACCURATE SAVING TRANSFER, EXCLUDING THE SECULAR
C     RIGHT ASCENSION RATES
C     ---------------------------------------------------
      CALL TRNSFR(TSI,ERE,T2,E2,TSI,T2,1,0,ESC,DV1,DV2,KV)
      DVRESC=VMAG(DV1)
      DVSC2=VMAG(DV2)


C     PROJECT ESC(I) TO T2
C     --------------------
      CALL ORBMOVE(ESC(1),ESC(2),SI,CI,RAD3,APD3,XMD3)
      DELDJ=T2-TSI
      ESC(3)=ANG(ESC(3) +XMD3*DELDJ)
      ESC(4)=ESC(4) +APD3*DELDJ
      ESC(6)=ESC(6) +RAD3*DELDJ

      IF(MPRINT.EQ.1)THEN
        WRITE(11,1006)T2,ESC
        WRITE(11,1008)DVRESC, DVSC2
      END IF

C     SET RIGHT ASCENSION OF SAVING ORBIT TO SAVING-CONIC VALUE
C     ----------------------------------------------------------
      OEM(IORB2,6) =ESC(6) -DRADT(IORB2)*(T2-ODJ(IORB2))


C     CHECK TRNSFR SOLUTION
C     ---------------------
      DO 220 J=1,6
  220 E2(J)=OEM(IORB2,J)
      EP2=ODJ(IORB2)
      CALL TRNSFR(TSI,ERE,EP2,E2,TSI,T2,1,1,ESC,DV1,DV2,KV)
      CALL ORBMOVE(ESC(1),ESC(2),SI,CI,RAD,APD,XMD)
      DELDJ=T2-TSI
      ESC(3)=ANG(ESC(3) +XMD*DELDJ)
      ESC(4)=ESC(4) +APD*DELDJ
      ESC(6)=ESC(6) +RAD*DELDJ
      DVRESC=VMAG(DV1)
      DVSC2=VMAG(DV2)

      IF(MPRINT.EQ.1)THEN
        WRITE(11,1006)T2,ESC
        WRITE(11,3322)EP2,E2
 3322   FORMAT(/1X,'EPOCH2, E2 =',F16.8,F14.2,5F14.9)
        WRITE(11,1008)DVRESC, DVSC2
      END IF

      DJL(LG1)=T1OPT
      DJL(LG1+1)=TSI
      DJL(LG1+2)=T2

      DO 230 J=1,6
```

```
      EML(LG1,J)=E1(J)
      EML(LG1+1,J)=ERE(J)
  230 EML(LG1+2,J)=ESC(J)

      RADOT(LG1)=DRADT(IORB1)
      APDOT(LG1)=DAPDT(IORB1)
      XMDOT(LG1)=DMADT(IORB1)

      RADOT(LG1+1)=RAD2
      APDOT(LG1+1)=APD2
      XMDOT(LG1+1)=XMD2

      RADOT(LG1+2)=RAD3
      APDOT(LG1+2)=APD3
      XMDOT(LG1+2)=XMD3

      IF(MPRINT.EQ.1)THEN
        DO 1234 LG=1,3
 1234   WRITE(11,1235)LG,RADOT(LG),APDOT(LG),XMDOT(LG)
        LG=4
        WRITE(11,1235)LG,DRADT(IORB2), DAPDT(IORB2), DMADT(IORB2)
 1235   FORMAT(/1X,'LG, RADOT, APDOT, XMDOT = ',I3,3F12.7)
      END IF

      DO 235 J=1,3
      DVCL(LG1+1,J)=DV1(J)
  235 DVCL(LG1+2,J)=DV2(J)

C     RESTORE RPMIN TO NOMINAL VALUE
C     ------------------------------
      RPMIN=RPMIN +18520.

      END IF
C     ***********  END MODE 3 BLOCK  *************************

      CTIP=T1OPT -DJ1 -.5*BT0
      CALL CSTBRN(ITRAN1,1,CTIP,0.,DV1RE,PROP1,PCOUT,PROP2,CTL,BTL)
      IF(MD.EQ.3)THEN
        TCSTL(LG1)=CTL
        TBRNL(LG1)=BTL
        DO 240 M=1,6
        PLEGC(LG1,M)=PCOUT(M)
  240   PLEGB(LG1,M)=PROP2(M)
      END IF

      IF(MPRINT.EQ.1)THEN
        WRITE(11,1009)CTL,BTL,PROP2(1)
 1009   FORMAT(/1X,'LEG 1  CT, BT, PROP2(1)=',2F15.7,F15.2)
      END IF

      ITRAN=ITRAN1+1
      CTIP=CTBSI -.5*BTL
      CALL CSTBRN(ITRAN,1,CTIP,0.,DVRESC,PROP2,PCOUT,PIN,CTL,BTL)
      IF(MD.EQ.3)THEN
```

```
          TCSTL(LG1+1)=CTL
          TBRNL(LG1+1)=BTL
          DO 245 M=1,6
          PLEGC(LG1+1,M)=PCOUT(M)
  245     PLEGB(LG1+1,M)=PIN(M)
        END IF

        IF(MPRINT.EQ.1)THEN
          WRITE(11,1010)CTL,BTL,PIN(1)
 1010     FORMAT(1X,'LEG 2  CT, BT, PROP2(1)=',2F15.7,F15.2)
        END IF

        CTIP=DJ2 -TSI -.5*BTL
        CALL CSTBRN(ITRAN,2,CTIP,0.,DVSC2,PIN,PCOUT,PROP2,CTL,BTF)
        IF(MD.EQ.3)THEN
          TCSTL(LG1+2)=CTL
          TBRNL(LG1+2)=BTF
          DO 250 M=1,6
          PLEGC(LG1+2,M)=PCOUT(M)
  250     PLEGB(LG1+2,M)=PROP2(M)
        END IF

        IF(MPRINT.EQ.1)THEN
          WRITE(11,1011)CTL,BTF,PROP2(1)
 1011     FORMAT(1X,'LEG 3  CT, BT, PROP2(1)=',2F15.7,F15.2)
        END IF


        IF(MD.EQ.2)THEN
C       SET TSTAY TIME FOR POSSIBLE USE IN PRECEDING 2BTO SEGMENT
C       -----------------------------------------------------------
        TSTAY=T1OPT-DJ1
        END IF


        RETURN
        END
```

```
      SUBROUTINE   TLOPT
      IMPLICIT REAL*8(A-H,O-Z)
      logical lfill,lobj,lorbele,lgeom,LTMAX
      integer*2 key,inkey$
C
C COMMON BLOCKS
C
      COMMON/IMA06/PI,TWOPI,PIO2
      COMMON/IMA11/NGMTMN(5),GMTMNS
      COMMON/IMA12/NORBIT,KTYPE(12),NGMT(12,5),GMTSEC(12),LGMT(12),
     1               ORBELE(12,8),LORBELE(12,8)
      COMMON/IMA13/NSEG,IESEG,IBSEG,IBLEG,KSEG
      COMMON/IMA14/NPL,PLMASS(12),ACCLIM(12)
      COMMON/IMA15/TMAXL,PMIN,LORB
      COMMON/IMA18/VEHMASS,NPROP,PCAP(6),PMTHRST(6),THMISP(6),THOISP(6)
      COMMON/IMA20/NORBO,FILL(6),LFILL(6),RESERV(6)
      COMMON/IMA22/NTRAN,KTRAN(15),NTORB(15),GEOM(15,14),LGEOM(15,14)
      COMMON/IMA26/NPLD(15),IPLD(15,12)
      COMMON/IMA28/LOBJ,TMAX,LTMAX,WFACT(6)
      COMMON/IMA32/LREF,NTSHIFT,TSHIFTS
      COMMON/IMA34/OEM(12,6),ODJ(12),KIND(12)
      COMMON/IMA36/DRADT(12),DAPDT(12),DMADT(12),SININC(12),COSINC(12)
      COMMON/IMA42/MPRINT
      COMMON/IMA58/NBIT(15),NBITS(15)
      COMMON/IMA60/ORBMIN(15),ORBMAX(15),TSTAY
C
      LOGICAL LPROP,LBET,LREF,LFEAS(20),LOPT(7),LNFP(7),LORB(12),LMODE
C
      DIMENSION IESEG(15),IBSEG(15),T(15),NLGMT(5),PMASS(15,7),IMODE(7),
     1          PSMASS(7),SEGE(7),SEGMIN(7),PEMASS(7),TB(10),PEMASS2(7),
     2          PTMASS(15,7),IOPT(7),IMULT(7),PBMASS(15,7),NTSHIFT(3),
     3          IBLEG(7),PMIN(7),TM(7),COST(39),PSMASS1(7),PEMASS1(7),
     4          FHIGH(7),FLOW(7),BT(15),FT(39),IC(7),TL(7),TH(7),
     5          TSUN(100),TCOM(100),TSUNV(100),TCOMV(100),XJOINT(100),
     6          WTRY(7),XJOINT1(100),TLAT(100),THOLDL(15),OHOLDL(12),
     7          PHOLDL(15,7),BTHOLDL(15),THOLDH(15),OHOLDH(12),
     8          PHOLDH(15,7),BTHOLDH(15),THOLDT(15),OHOLDT(12),
     9          PHOLDT(15,7),BTHOLDT(15),TADJ(15),STAY(7),TK(15),
     1          KSEG(7)
C
C THIS SUBROUTINE IS THE IMA TOP LEVEL OPTIMIZER. IT'S JOB IS TO ASSIGN
C VALUES FOR THE TIMES THAT THE INDIVIDUAL SEGMENTS END ON, AN INITIAL
C TIME IF THE INITIAL ORBIT IS LI WITH A FREE GMT, AND FREE ORBITAL
C PARAMETERS IF TARGET ORBITS HAVE THEM.
C
C DETERMINE THE NUMBER OF SEGMENTS AND ON WHICH TRANSFER THEY END ON
C
C INITIALIZE THE NUMBER OF SEGMENTS TO ZERO
C
      LMODE = .true.
      MPRINT1=1
      DO I = 1,12
         LORB(I) = LORBELE(I,4)
      END DO
```

```
      DO I = 1,NTRAN
         NBIT(I) = 2
         NBITS(I) = 2
      END DO
      JADJ = 0
      MPRINT=0
      BCOSTL = 9.0E9
      NSEG = 0
      IBLEG(1) = 1
C
C CALCULATE THE PROPELLANT FLOOR
C
      DO 5 I = 1,NPROP
         PMIN(I) = PCAP(I) * FILL(I) * RESERV(I)
         EPS = EPS + WFACT(I) / PCAP(I)
 5    CONTINUE
C
      DO 10 I=1,NTRAN
C
C IS THE TRANSFER A TARGET ORBIT OR TARGET ORBIT RENDEZVOUS ?
C
         IF (KTRAN(I) .GE. 3 .AND. KTRAN(I) .LE. 5) THEN
C
            IOK = 0
            IF (KTRAN(I) .EQ. 4) THEN
               IF (I .EQ. 1) THEN
                  IOK = 1
               ELSE IF (KTRAN(I-1) .GE. 3 .AND. KTRAN(I-1) .LE. 5) THEN
                  IOK = 1
               END IF
               KSEG(NSEG+1) = 2
            ELSE IF (KTRAN(I) .EQ. 5) THEN
               IF (I .GE. 3 .AND. KTRAN(I-1) .EQ. 2 .AND. KTRAN(I-2)
     1            .EQ. 2) THEN
                  IF (I .EQ. 3) THEN
                     IOK = 1
                  ELSE IF (KTRAN(I-3) .GE. 3 .AND. KTRAN(I-3) .LE. 5)
     1                     THEN
                     IOK = 1
                  END IF
               END IF
               KSEG(NSEG+1) = 1
            ELSE
               IF (I .GE. 2 .AND. KTRAN(I-1) .EQ. 1) THEN
                  IOK = 1
               ELSE IF (KTRAN(I-2) .GE. 3 .AND. KTRAN(I-2) .LE. 5) THEN
                  IOK = 1
               END IF
               KSEG(NSEG+1)=3
            END IF
            IF (IOK .EQ. 0) THEN
               CALL MSG('***HALT: IMPROPER GROUPING OF TRANSFERS',3)
               STOP
            END IF
```

```
C
C INCREMENT THE NUMBER OF SEGMENTS
C
            NSEG = NSEG + 1
C
C SAVE THE NUMBER OF THE TRANSFER
C
            IESEG(NSEG) = I
C
         END IF
C
 10   CONTINUE
C
      IF (NSEG .EQ. 0) THEN
C
         CALL MSG('***HALT: NO LEGAL SEGMENTS ENTERED',3)
         STOP
C
      END IF
C
C FIND THE TRANSFERS THAT SEGMENTS BEGIN ON
C
      IBSEG(1) = 1
C
      DO 20 I = 2,NSEG
C
         IBSEG(I) = IESEG(I - 1) + 1
C
 20   CONTINUE
C
C SEE IF FILL FRACTIONS HAVE BEEN ENTERED
C
      DO 30 I = 1,NPROP
C
         IF (.NOT. LFILL(I)) THEN
C
            CALL MSG('***HALT: FILL FRACTIONS REQUIRED BY THIS PROGRAM V
     1ERSION',3)
C
            STOP
C
         ELSE
C
C CALULATE INITIAL PROPELLANT MASSES
C
            PMASS(1,I) = PCAP(I) * FILL(I)
C
         END IF
C
 30   CONTINUE
C
      DO 70 I = 1,NSEG + 1
C
         IMODE(I) = 1
```

```
            LNFP(I) = .FALSE.
            IC(I) = 0
C
 70   CONTINUE
C
C START SUBROUTINE CALL
C
      I = 1
 100  CONTINUE
C
      DO 50 J = 1,NPROP
C
         PSMASS(J) = PMASS(I,J)
C
 50   CONTINUE
C
C IS THE INITIAL ORBIT NOT AN LI WITH FREE GMT ?
C
      IF (KIND(NORB0) .EQ. 7) THEN
C
         CALL TLNCH
         T(1) = ODJ(NORB0)
C
      ELSE
C
C
C LOAD THE GMT
C
         DO I = 1,5
C
            NLGMT(I) = NGMT(NORB0,I)
C
         END DO
C
C GET JULIAN DAY
C
         T(1) = DJUL(NLGMT,GMTSEC(NORB0))
C
      END IF
C
C SET STARTING TIME TO 2
C
         ISTRT = 2
C
C IS THERE A MISSION REFERENCE TIME SPECIFIED ?
C
         IF (LREF) THEN
C
C COMPUTE THE TIME SHIFT IN DAYS
C
            SHIFT = NTSHIFT(1) + FLOAT(NTSHIFT(2)) / 24.0 +
     1              FLOAT(NTSHIFT(3)) / 1440.0 + TSHIFTS / 86400.0
C
C SHIFT MISSION TIME ZERO
```

```
C
            T(1) = T(1) + SHIFT
C
         END IF
C
C SET MAXIMUM MISSION TIME
C
         IF (LOBJ) THEN
            TMAXL = T(1) + TMAX / 86400.0
         ELSE
            TMAXL = T(1) + 100.0
         END IF
C
C WORK MINIMUM TIME PROBLEM FOR SEGMENT 1
C
         I = 1
C
C DOES SEGMENT END WITH TOR ?
C
         IF (KTRAN(IESEG(I)) .EQ. 5) THEN
C
C SET MODE TO MINUMUM TIME
C
            IMODE(1) = 0
C
C CALL SEGMENT WORKER #1
C
            CALL SW01(IMODE(I),IBSEG(I),IBLEG(I),BT(I),T(I),PSMASS,
     1                TM(I+1),PEMASS,IELEG,BT(I+1),KERR)
C
C DOES TRANSFER END WITH TO ?
C
         ELSE IF (KTRAN(IESEG(I)) .EQ. 4) THEN
C
C IS IT'S RA FREE ?
C
            IF (.NOT. LORB(NTORB(IESEG(I)))) THEN
C
C SET MODE TO MINIMUM WITH FREE RA
C
               IMODE(1) = -1
C
            ELSE
C
C SET MODE TO MINIMUM TIME WITH FIXED RA
C
               IMODE(1) = 0
C
            END IF
C
C CALL SEGMENT WORKER # 2
C
            CALL SW02(IMODE(I),IBSEG(I),IBLEG(I),BT(I),T(I),PSMASS,
     1                TM(I+1),PEMASS,IELEG,BT(I+1),KERR)
```

```
C
          ELSE IF (KTRAN(IESEG(I)) .EQ. 3) THEN
C
             IMODE(1) = 1
             CALL SW03(IMODE(I),IBSEG(I),IBLEG(I),BT(I),T(I),PSMASS,
     1                 TM(I+1),PEMASS,IELEG,BT(I+1),KERR)

C
          END IF
             IF (MPRINT1 .EQ. 1)
     1       WRITE(20,999) 1,IMODE(1),IBSEG(1),T(I),PSMASS(1),
     1       PSMASS(2),TF,TM(I+1),PEMASS(1),PEMASS(2),KERR
 999  FORMAT(' ',I1,1X,I2,1X,I1,1X,F15.7,1X,F11.5,1X,F11.5,1X,F15.7,1X,
     1        F15.7,1X,F11.5,1X,F11.5,1X,I1)
C
          IF (KERR .NE. 0) THEN
             TM(I+1) = T(I)
          END IF
C
 1700 DO J = 1,NSEG + 1
          LOPT(J) = .TRUE.
      END DO
      JR = 0
C
C HERE IS THE OPTIMIZATION PORTION
C
      K = ISTRT - 1
C
C INCREMENT TIME COUNTER
C
 1130 K = K + 1
C
C IS THIS NOT THE LAST TIME TO BE VARIED ?
C
 1131 CONTINUE
      key=inkey$()
      if(key.eq.27)then
 7777   key=inkey$()
        if(key.ne.0)go to 7777
        print*, ' '
        call msg(
     *      '*** RESPONDING TO USER REQUEST TO TERMINATE EXECUTION **',3)
      endif
      IF (K .NE. NSEG + 1) THEN
C
C INITIALIZE RA COUNTER AND COST
C
          OCOST = 9.99E10
          IR = 0
          ICHK = 0
C
C LOAD INITIAL MASS
C
          DO I = 1,NPROP
```

```
            PSMASS(I) = PMASS(K-1,I)
         END DO
C
C ARE WE IN AN OPTIMIZATION MODE ?
C
         IF (LOPT(K)) THEN
C
 650        FORMAT(' ','** OPTIMIZING JOINT TIME #',I1,' **')
 651        FORMAT(' ','*** OPTIMUM SOLUTION FOUND FOR JOINT TIME#',I1,
     1             ' ***')
 652           FORMAT(' ','** ALTERNATE SOLUTION CHOSEN FOR TIME#',I1,
     1                   ' **')

            WRITE(6,650) K
            WRITE(20,650) K
C
C IS THE SEGMENT A TO ?
C
            IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
C
C IS THE RA FIXED ?
C
               IF (LORB(NTORB(IESEG(K-1)))) THEN
                  ITEMP = 1
               ELSE
                  ITEMP = -1
               END IF
C
C CALL THE 1ST SEGMENT IN A MINIMUM PROP MODE
C
               CALL SW02(ITEMP,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,TP,PEMASS,IELEG,BT(K),KERR)
            IF (MPRINT1 .EQ. 1)
     1      WRITE(20,999) K-1,ITEMP,IBSEG(K-1),T(K-1),PSMASS(1),
     1      PSMASS(2),TP,TM(K),PEMASS(1),PEMASS(2),KERR
C
C IS THE SEGMENT A TOR ?
C
            ELSE IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
C
               CALL SW01(1,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,TP,PEMASS,IELEG,BT(K),KERR)
            IF (MPRINT1 .EQ. 1)
     1      WRITE(20,999) K-1,1,IBSEG(K-1),T(K-1),PSMASS(1),
     1      PSMASS(2),TP,TM(K),PEMASS(1),PEMASS(2),KERR
C
            ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
C
               CALL SW03(1,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,TP,PEMASS,IELEG,BT(K),KERR)
            IF (MPRINT1 .EQ. 1)
     1      WRITE(20,999) K-1,1,IBSEG(K-1),T(K-1),PSMASS(1),
     1      PSMASS(2),TP,TM(K),PEMASS(1),PEMASS(2),KERR
C
```

```
            END IF
C
C IS THERE AN ERROR FOR THE 1ST SEG MIN PROP ?
C
            IF (KERR .NE. 0) THEN
C
C GO BACK AND WORK ON PREVIOUS T(K)
C
               IF (K .EQ. ISTRT) THEN
               CALL MSG('***HALT: NO FEASIBLE SOLUTION WITHIN MAXIMUM MI
     1SSION CONSTRAINTS CAN BE FOUND',3)
               STOP
               ELSE
               WRITE(6,653) K,K-1
               WRITE(20,653) K,K-1
 653           FORMAT(' ','** NO SOLUTION FOR TIME#',I1,' GIVEN THE CURR
     1ENT TIME#',I1,'. GO BACK 1 TIME **')
               K = K - 1
               LOPT(K) = .FALSE.
               GOTO 1131
               END IF
C
            END IF
C
C PREPARE TO RUN THE 1ST SEGMENT IN FIXED TIME MODE
C
            IF (ITEMP .EQ. -1) THEN
               ITEMP = -2
            ELSE
               ITEMP = 2
            END IF
C
C INITIALIZE BEST COST AND RANGE TO SEARCH FOR T(K)
C
 1004       BCOST = 9.99E8
            THIGH = TMAXL
            TLOW = TM(K)
C
            IT = 39
C
C CALCULATE DELTA
C
 1002       DELTA = (THIGH - TLOW) / 11.0
C
C LOOP FOR EACH ATTEMPT AT T(K)
C
            DO I = 1,IT
C
               IF (IT .EQ. 10) THEN
                  TTRY = TLOW + I * DELTA
               ELSE
                  IF (I .LE. 10) THEN
                     TTRY = TLOW + (I-1) * 0.01
                  ELSE IF (I .LE. 30) THEN
```

```
                  TTRY = TLOW + (I-10) * 0.1
               ELSE IF (I .LE. 38) THEN
                  TTRY = TLOW + (I-28) * 1.0
               ELSE
                  TTRY = TP
               END IF
               IF (TTRY .GT. THIGH) THEN
                  COST(I) = 9.99E9
                  GOTO 1003
               END IF
            END IF
C
C IS THE 1ST SEGMENT A TOR ?
C
            IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
C
C CALL IN FIXED END TIME MODE
C
               CALL SW01(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
C
            ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
C
C CALL IN FIXED END TIME MODE
C
               CALL SW03(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
C
            ELSE IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
C
C IS THE RA FREE AND OVERALL OBJ = MIN PROP
C
C              IF (ITEMP .EQ. -2 .AND. LOBJ) THEN
               IF (ITEMP .EQ. -2) THEN
C
C CALL TO IN FREE RA FIXED TIME MODE
C
                  CALL SW02(-2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                      PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
         IF (MPRINT1 .EQ. 1)
     1   WRITE(20,999) K-1,-2,IBSEG(K-1),T(K-1),PSMASS(1),
     1   PSMASS(2),TTRY,TM(K),PEMASS(1),PEMASS(2),KERR
C
C IS THERE AN ERROR ?
C
                  IF (KERR .NE. 0) THEN
                     COST(I) = 9.99E9
                     GOTO 1003
                  END IF
C
C CALCULATE WHAT RA OF TO SHOULD BE
C
                  DIFF = TMAX / 86400.0 *
     1                   ABS(DRADT(NTORB(IESEG(K-1))) -
```

```
     2                           DRADT(NTORB(IESEG(K))))
C
                      IF (XDIFF .LT. 0.0) THEN
C
                         OEM(NTORB(IESEG(K-1)),6) =
     1                                              OEM(NTORB(IESEG(K-1)),6)
     2                                              - IR * 0.001745
C
                      ELSE
C
                         OEM(NTORB(IESEG(K-1)),6) =
     1                                              OEM(NTORB(IESEG(K-1)),6)
     2                                              + IR * 0.001745
                      END IF
C
                   END IF

C
C RECALCULATE TTRY
C
                IF (IT .EQ. 10) THEN
                   TTRY = TLOW + I * DELTA
                ELSE
                   IF (I .LE. 10) THEN
                      TTRY = TLOW + (I-1) * 0.01
                   ELSE IF (I .LE. 30) THEN
                      TTRY = TLOW + (I-10) * 0.1
                   ELSE IF (I .LE. 38) THEN
                      TTRY = TLOW + (I-28) * 1.0
                   ELSE
                      TTRY = TP
                   END IF
                   IF (TTRY .GT. THIGH) THEN
                      COST(I) = 9.99E9
                      GOTO 1003
                   END IF
                END IF
C
C CALL TO WITH FIXED RA
C
                   CALL SW02(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                       PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
C
                END IF
             IF (MPRINT1 .EQ. 1)
     1       WRITE(20,999) K-1,2,IBSEG(K-1),T(K-1),PSMASS(1),
     1       PSMASS(2),TTRY,TM(K),PEMASS(1),PEMASS(2),KERR
C
C IS THERE AN ERROR ?
C
                IF (KERR .NE. 0) THEN
                   COST(I) = 9.99E9
                   GOTO 1003
                END IF
```

```
C
C INCREMENT LEG COUNTER
C
                IBLEG(K) = IELEG + 1
                BTS = BT(K)
C
                CALL FINISH (K+1,TTRY,BTS,PEMASS,COST(I),FTBL)
            WRITE(20,*) COST(I)

 1003       IF (MPRINT1 .EQ. 1)WRITE(20,*) COST(I) ,OEM(4,6)
C
            END DO
C
C INITIALIZE BETTER FLAG TO FALSE
C
            LBET = .FALSE.
C
            DO I = 1,IT
C
                IF (COST(I) .LT. BCOST) THEN
C
C SAVE THE STUFF
C
                    BCOST = COST(I)
                    IBEST = I
                    LBET = .TRUE.
C
                END IF
C
            END DO
C
C WERE THERE NO SUCCESSFUL COMBINED RUNS ?
C
            IF (BCOST .EQ. 9.99E8) THEN
C
                        IF (K .EQ. ISTRT) THEN
                CALL MSG('***HALT: NO FEASIBLE SOLUTION WITHIN MAXIMUM MI
     1SSION CONSTRAINTS CAN BE FOUND',3)
                        STOP
                        ELSE
                        WRITE(6,655) K,K-1
                        WRITE(20,655) K,K-1
 655  FORMAT(' ','** NO SOLUTION FOR TIME#',I1,' WITHIN TIME CONSTRAINTS
     1 GIVEN TIME#',I1,'. BACKING UP 1 TIME **')
                        K = K - 1
                        LOPT(K) = .FALSE.
                        GOTO 1131
                        END IF
C
            ELSE
C
C WAS THE BEST BETTERED ?
C
                IF (LBET) THEN
```

```
C
C RESET THE TIME AND RANGE
C
               IF (IT .EQ. 10) THEN
                  T(K) = TLOW + IBEST * DELTA
                  THIGH = TLOW + (IBEST + 1) * DELTA
                  TLOW = TLOW + (IBEST - 1) * DELTA
               ELSE
                  IF (IBEST .LE. 10) THEN
                  T(K) = TLOW + (IBEST-1) * 0.01
                  THIGH = TLOW + (IBEST) * 0.01
                  TLOW = TLOW + (IBEST - 2) * 0.01
                  DELTA = .01
                  ELSE IF (IBEST .LE. 30) THEN
                  T(K) = TLOW + (IBEST-10) * 0.1
                  THIGH = TLOW + (IBEST-9) * 0.1
                  TLOW = TLOW + (IBEST-11) * 0.1
                  DELTA = .1
                  ELSE IF (IBEST .LE. 38) THEN
                  T(K) = TLOW + (IBEST-28) * 1.0
                  THIGH = TLOW + (IBEST-27) * 1.0
                  TLOW = TLOW + (IBEST-29) * 1.0
                  DELTA = 1.0
                  ELSE
                  T(K) = TP
                  THIGH = TP + .5
                  TLOW = TP - .5
                  DELTA = .1
                  END IF
               END IF
C
               ELSE
C
C NARROW THE RANGE
C
                  TLOW = T(K) - DELTA
                  THIGH = T(K) + DELTA
C
               END IF
C
C WAS THE DELTA ABOVE TOLERANCE ?
C
               IF (DELTA .GT. 0.01) THEN
C
C GO BACK AND TRY MORE POINTS
C
                  IT = 10
                  GOTO 1002
C
               END IF
C
            END IF
C
C
```

```
C CHECK TO SEE IF I NEED TO TRY ANOTHER RA
C
            IF (KTRAN(IESEG(K-1)) .EQ. 4 .AND. .NOT.
     1         LORB(NTORB(IESEG(K-1))) .AND.
     1         (LORB(NTORB(IESEG(K))) .OR. KTRAN(IESEG(K)) .EQ. 5)
     1         .AND. LOBJ) THEN
C
                  WRITE(6,654) IR,K-1
                  WRITE(20,654) IR,K-1
                  IF (MPRINT1 .EQ. 1) WRITE(20,*)
     1                 OEM(NTORB(IESEG(K-1)),6)
 654  FORMAT(' ','**** RIGHT ASCENSION #',I3,' ATTEMPTED FOR SEGMENT #',
     1I1,' ****')

C              IF (IR .EQ. 0) THEN
C                 OCOST = BCOST
C                 IR = IR + 1
C                 TBEST = T(2)
C                 GOTO 1004
C              ELSE IF (OCOST - BCOST .GT. 0.0) THEN
C                 OCOST = BCOST
C                 IR = IR + 1
C                 TBEST = T(2)
C                 IF (IR .LE. 20) GOTO 1004
C              END IF
             IF (IR .EQ. 0) THEN
                ZCOSTH = 9.0E9
                ZCOSTL = BCOST
                   XDIFF = OEM(NTORB(IESEG(K)),6) + (FTBL -
     1                     ODJ(NTORB(IESEG(K)))) *
     2                     DRADT(NTORB(IESEG(K))) -
     3                     (OEM(NTORB(IESEG(K-1)),6) + (FTBL -
     1                     ODJ(NTORB(IESEG(K-1)))) *
     2                     DRADT(NTORB(IESEG(K-1))))
                ITOT = INT(ABS(XDIFF)/ 0.0017) + 2
                ILOW = 0
 1964           IHIGH = ITOT
                TVL = T(K)
                IF1 = 1
             END IF
 1888        IF (IF1 .EQ. 1) THEN
                ITRY = (IHIGH + ILOW) * 0.5
                IF (ITRY .EQ. ILOW) GOTO 1889
                IR = ITRY
                IF1 = 2
             ELSE IF (IF1 .EQ. 2) THEN
                IF (BCOST .GT. ZCOSTH) THEN
                   ILOW = ITRY
                   ZCOSTL = BCOST
                   TVL = T(K)
                   IF1 = 1
                   GOTO 1888
                ELSE IF (BCOST .GT. ZCOSTL) THEN
                   IHIGH = ITRY
```

```
                     ZCOSTH = BCOST
                     TVH = T(K)
                     IF1 = 1
                     GOTO 1888
                  END IF
                  TCOST = BCOST
                  TVT = T(K)
                  IDIR = ITRY + 1
                  IF (IDIR .EQ. IHIGH) GOTO 1889
                  IR = IDIR
                  IF1 = 3
               ELSE IF (IF1 .EQ. 3) THEN
                  IF (TCOST .LT. BCOST) THEN
                     IHIGH = ITRY
                     ZCOSTH = TCOST
                     TVH = TVT
                  ELSE
                     ILOW = IDIR
                     ZCOSTL = BCOST
                     TVL = T(K)
                  END IF
                  IF1 = 1
                  GOTO 1888
               END IF
               GOTO 1004
 1889          IF (IHIGH .EQ. ITOT) THEN
                  ITOT = ITOT + 2
                  GOTO 1964
               END IF
               IF (ZCOSTL .LT. ZCOSTH) THEN
                  IKEEP = ILOW
                  T(K) = TVL
                  BCOST = ZCOSTL
               ELSE
                  IKEEP = IHIGH
                  T(K) = TVH
                  BCOST = ZCOSTH
               END IF
               IF (IF1 .EQ. 2 .AND. TCOST .LT. ZCOSTL .AND. TCOST .LT.
     1             ZCOSTH) THEN
                   IKEEP = ITRY
                   T(K) = TVT
                   BCOST = TCOST
               END IF
C
            ELSE
               IKEEP = 0
C
            END IF
C
         ELSE
C
C IS THIS THE 1ST PASS ?
C
```

```
            IF (.NOT. LNFP(K)) THEN
C
               LNFP(K) = .TRUE.
               DO L = K + 1,NSEG + 1
                  LOPT(L) = .TRUE.
                  IC(L) = 0
               END DO
               IC(K)=0
C
C FIND LOW LIMIT FOR T(K)
C
               TL(K) = T(K)
               TH(K) = T(K)
               TBOT = T(K-1)
               TTOP = TMAXL
 601           DELTA = (TL(K) - TBOT) / 11.0
               I = 1
 600           TTRY = TL(K) - I * DELTA
               IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
                  CALL SW02(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                      PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
               ELSE IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
                  CALL SW01(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                      PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
               ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
                  CALL SW03(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                      PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
               END IF
               IWORK = 1
               IF (KERR .NE. 0) THEN
                  IWORK = 0
               END IF
               IF (IWORK .EQ. 1) THEN
                  I = I + 1
                  IF (I .LE. 10) GOTO 600
                  TL(K) = TL(K) - 10 * DELTA
               ELSE
                  TL(K) = TL(K) - (I - 1) * DELTA
               END IF
               IF (DELTA .GT. 0.01) THEN
                  TBOT = TL(K) - DELTA
                  GOTO 601
               END IF
C
C GET HIGH LIMIT
C
 603           DELTA = (TTOP - TH(K)) / 11.0
               I = 1
 602           TTRY = TH(K) + I * DELTA
               IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
                  CALL SW02(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                      PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
               ELSE IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
                  CALL SW01(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
```

```
     1                          PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
                ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
                   CALL SW03(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                          PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
                END IF
                IWORK = 1
                IF (KERR .NE. 0) THEN
                   IWORK = 0
                END IF
                IF (IWORK .EQ. 1) THEN
                   I = I + 1
                   IF (I .LE. 10) GOTO 602
                   TH(K) = TH(K) + 10 * DELTA
                ELSE
                   TH(K) = TH(K) + (I - 1) * DELTA
                END IF
                IF (DELTA .GT. 0.01) THEN
                   TTOP = TH(K) + DELTA
                   GOTO 603
                END IF
C
             END IF
C
             DELTA = (TH(K) - TL(K)) / 11.0
             IC(K) = IC(K) + 1
             IF (IC(K) .EQ. 11) THEN
                IF (K .EQ. ISTRT) THEN
                CALL MSG('***HALT: NO FEASIBLE SOLUTION WITHIN MAXIMUM MI
     1SSION CONSTRAINTS CAN BE FOUND',3)
                STOP
                ELSE
                   WRITE(6,657) K
                   WRITE(20,657) K
 657  FORMAT(' ','**ALL ATTEMPTS AT TIME#',I1,' FAILED. BACKING UP ANOTH
     1ER TIME **')
                   LNFP(K) = .FALSE.
                   K = K -1
                   LOPT(K) = .FALSE.
                   GOTO 1131
                END IF
             END IF
             WRITE(6,652) K
             WRITE(20,652) K
             T(K) = TL(K) + IC(K) * DELTA
C
          END IF
C
C RUN THE 1ST SEGMENT AGAIN TO GET CORRECT PROPELLANT AND TIME
C
             TK(K) = T(K)
             IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
             CALL SW01(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                   PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
             ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
```

```
            CALL SW03(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1               PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
            ELSE IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
            TSAVE = T(K)
C           IF (.NOT. LORB(NTORB(IESEG(K-1))).AND.LOBJ) THEN
            IF (.NOT. LORB(NTORB(IESEG(K-1)))) THEN
            CALL SW02(-2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1               PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
            DIFF = TMAX / 86400.0 * ABS(DRADT(NTORB(IESEG(K-1))) -
     1            DRADT(NTORB(IESEG(K))))
            IF (XDIFF .LT. 0.0) THEN
               OEM(NTORB(IESEG(K-1)),6) = OEM(NTORB(IESEG(K-1)),6) -
     1                                    IKEEP * 0.001745
            ELSE
               OEM(NTORB(IESEG(K-1)),6) = OEM(NTORB(IESEG(K-1)),6) +
     1                                    IKEEP * 0.001745
            END IF
            END IF
            T(K) = TSAVE
            CALL SW02(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1               PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
            END IF
            IF (MPRINT1 .EQ. 1)
     1      WRITE(20,999) K-1,2,IBSEG(K-1),T(K-1),PSMASS(1),
     1      PSMASS(2),T(K),TM(K),PEMASS(1),PEMASS(2),KERR


C
C INCREMENT RUNNING LEG COUNT
C
            IBLEG(K) = IELEG + 1
C
C ENTER MASS INTO OFFICIAL ARRAY
C
            DO I = 1,NPROP
               PMASS(K,I) = PEMASS(I)
            END DO
C
C RUN FINAL SEGMENT TO GET MINIMUM TIME FOR IT
C
            IF (ITEMP1 .NE. -1) THEN
C
               ITEMP1 = 0
C
            END IF
C
            IF (KTRAN(IESEG(K)) .EQ. 4) THEN
               CALL SW02(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
     1                  TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
            ELSE IF (KTRAN(IESEG(K)) .EQ. 5) THEN
               CALL SW01(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
     1                  TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
            ELSE IF (KTRAN(IESEG(K)) .EQ. 3) THEN
               CALL SW03(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
```

```
     1                    TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
             END IF
             IF (MPRINT1 .EQ. 1)
     1       WRITE(20,999) K,ITEMP1,IBSEG(K),T(K),PEMASS(1),
     1       PEMASS(2),TM(K+1),TM(K),PEMASS1(1),PEMASS1(2),KERR
C
             WRITE(6,651) K
             WRITE(20,651) K
C
         ELSE
C
           WRITE(6,650) K
           WRITE(20,650) K
C
C SET LAST SEGMENT TO A FIXED TIME MODE
C
           IF (KTRAN(IESEG(K-1)) .EQ. 4 .AND. .NOT.
     1        LORB(NTORB(IESEG(K-1)))) THEN
             ITEMP = -2
           ELSE
             ITEMP = 2
           END IF
           IT = 39
C
C LOAD UP INITIAL PROPELLANT
C
           DO I = 1,NPROP
             PSMASS(I) = PMASS(K-1,I)
           END DO
C
C SET INITIAL COST VERY HIGH
C
           BCOST = 9.99E8
C
C SET RANGE
C
           THIGH = TMAXL
           TLOW = TM(K)
C
C CALCULATE DELTA TIME
C
 1010      DELTA = (THIGH - TLOW) / 11.0
C
C LOOP FOR THE 10 ATTEMPTS
C
           DO I = 1,IT
C
C CALCULATE TIME TO TRY
C
             IF (IT .EQ. 10) THEN
               TTRY = TLOW + I * DELTA
             ELSE
               IF (I .LE. 10) THEN
                 TTRY = TLOW + (I-1) * .01
```

```
                ELSE IF (I .LE. 30) THEN
                   TTRY = TLOW + (I-10) * .1
                ELSE IF (I .LE. 39) THEN
                   TTRY = TLOW + (I-28) * 1.
                END IF
             END IF
C
C CALL THE SEGMENT
C
             IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
                CALL SW01(ITEMP,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
             ELSE IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
                CALL SW02(ITEMP,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
             ELSE
                CALL SW03(ITEMP,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                PSMASS,TTRY,PEMASS,IELEG,BT(K),KERR)
             END IF
             IF (MPRINT1 .EQ. 1)
     1       WRITE(20,999) K,ITEMP,IBSEG(K-1),T(K-1),PSMASS(1),
     1       PSMASS(2),TTRY,TM(K),PEMASS(1),PEMASS(2),KERR
C
C IS THERE AN ERROR ?
C
             IF (KERR .NE. 0) THEN
C
C SET THE COST VERY HIGH
C
                COST(I) = 9.99E9
C
             ELSE
C
C IS THE OBJECTIVE MINIMUM PROP ?
C
                IF (LOBJ) THEN
C
C CALCULATE WEIGHTED COST FUNCTION
C
                   COST(I) = 0.0
                   DO J = 1,NPROP
                      COST(I) = COST(I) + (PMASS(1,J) - PEMASS(J)) /
     1                          PCAP(J) * WFACT(J)
                   END DO
C
                ELSE
C
                   COST(I) = TTRY
                   DO J = 1,NPROP
                      IF (PEMASS(J) .LT. PMIN(J)) THEN
                         COST(I) = 9.99E9
                      END IF
                   END DO
C
```

```
                  END IF
C
               END IF
C
            IF (MPRINT1 .EQ. 1)WRITE(20,*) COST(I),OEM(4,6)
            END DO
C
C INITIALIZE BETTER FLAG TO FALSE
C
            LBET = .FALSE.
C
            DO I = 1,IT
C
               IF (COST(I) .LT. BCOST) THEN
C
C SAVE THE STUFF
C
                  BCOST = COST(I)
                  IBEST = I
                  LBET = .TRUE.
C
               END IF
C
            END DO
C
            IF (BCOST .EQ. 9.99E8) THEN
C
                  WRITE(6,653)K,K-1
                  WRITE(20,653)K,K-1
                  K = K -1
                  LOPT(K) = .FALSE.
                  GOTO 1131
C
            ELSE
C
               IF (LBET) THEN
C
                  IF (IT .EQ. 10) THEN
                     T(K) = TLOW + IBEST * DELTA
                     THIGH = TLOW + (IBEST + 1) * DELTA
                     TLOW = TLOW + (IBEST -1) * DELTA
                  ELSE
                  IF (IBEST .LE. 10) THEN
                     T(K) = TLOW + (IBEST-1) * .01
                     THIGH = TLOW + (IBEST) * .01
                     TLOW = TLOW + (IBEST - 2) * .01
                  ELSE IF (IBEST .LE. 30) THEN
                     T(K) = TLOW + (IBEST-10) * .1
                     THIGH = TLOW + (IBEST-9) * .1
                     TLOW = TLOW + (IBEST -11) * .1
                  ELSE
                     T(K) = TLOW + (IBEST-28) * 1.0
                     THIGH = TLOW + (IBEST-28) * 1.0
                     TLOW = TLOW + (IBEST-28) * 1.0
```

```
                END IF
                END IF
C
             ELSE
C
                TLOW = T(K) - DELTA
                THIGH = T(K) + DELTA
C
             END IF
C
             IF (DELTA .GE. 0.01) THEN
C
                IT = 10
                GOTO 1010
C
             END IF
             TK(K) = T(K)
C
          END IF
C
       END IF
C
C CHECK MULTIPLE BURN PROBLEMS
C
C      IADJ = 0
C      DO I = 1,NTRAN
C
C         IF (NBITS(I) .GT. 12) THEN
C
C            NBITS(I) = 12
C            CALL MSG('***MAXIMUM BURN TIME LIMITED EXCEEDED. RESULT IS F
C    1OR THE 12 BURN MAXIMUM',1)
C
C         END IF
C
C
C         IF (NBIT(I) .NE. NBITS(I)) THEN
C
C            IADJ = 1
C
C         END IF
C
C      END DO
C
C      IF (IADJ .EQ. 1 .AND. JADJ .LE. 3) THEN
C
C         DO I = 1,NTRAN
C            NBIT(I) = NBITS(I)
C         END DO
C         JADJ = JADJ + 1
C         WRITE(6,658)
C         WRITE(20,658)
 658  FORMAT(' ','* NUMBER OF BURNS IN A TRANSFER MUST BE CHANGED. REWOR
    1K PROBLEM *')
```

```
C         GOTO 1700
C
C      END IF
C
      IF (K .GE. 4 .AND. LOBJ) THEN
C
         IZ = 1
C
         DO L = K-3,1,-1
C
            IF (NTORB(IESEG(K-2)) .EQ. NTORB(IESEG(L)) .AND.
     1          KTRAN(IESEG(L)) .EQ. 4 .AND. .NOT.
     2          LORBELE(NTORB(IESEG(L)),4)) THEN
C
               LS = L
               IZ = 0
            END IF
         END DO
         IF (IZ .EQ. 0) THEN
            IF (JR .EQ. 0) THEN
               KT = K-1
 2122          IF (.NOT. LORBELE(NTORB(IESEG(KT)),4)) THEN
                  KT = KT + 1
                  GOTO 2122
               END IF
               YCOSTH = 9.0E9
               YCOSTL = BCOST
               YDIFF = ABS(OEM(NTORB(IESEG(KT)),6) + (T(K) -
     1                 ODJ(NTORB(IESEG(KT)))) *
     2                 DRADT(NTORB(IESEG(KT))) -
     3                 (OEM(NTORB(IESEG(LS)),6) + (T(K) -
     4                 ODJ(NTORB(IESEG(LS)))) *
     5                 DRADT(NTORB(IESEG(LS)))))
               OEMREF1 = OEM(NTORB(IESEG(LS)),6)
               JTOT = INT(YDIFF / 0.001745) + 2
               JHIGH = JTOT
               JLOW = 0
               DO M = 1,15
                  THOLDL(M) = T(M)
                  BTHOLDL(M) = BT(M)
                  DO J = 1,NPROP
                     PHOLDL(M,J) = PMASS(M,J)
                  END DO
               END DO
               DO M = 1,12
                  OHOLDL(M) = OEM(M,6)
               END DO
               JF1 = 1
            END IF
 1988       IF (JF1 .EQ. 1) THEN
               JTRY = (JHIGH + JLOW) * 0.5
               IF (JTRY .EQ. JLOW) GOTO 1989
               JR = JTRY
               JF1 = 2
```

```
ELSE IF (JF1 .EQ. 2) THEN
   IF (BCOST .GT. YCOSTL) THEN
      JHIGH = JTRY
      YCOSTH = BCOST
      DO M = 1,15
         THOLDH(M) = T(M)
         BTHOLDH(M) = BT(M)
         DO J = 1,NPROP
            PHOLDH(M,J) = PMASS(M,J)
         END DO
      END DO
      DO M = 1,12
         OHOLDH(M) = OEM(M,6)
      END DO
      JF1 = 1
      GOTO 1988
   ELSE IF (BCOST .GT. YCOSTH) THEN
      JLOW = JTRY
      YCOSTL = BCOST
      DO M = 1,15
         THOLDL(M) = T(M)
         BTHOLDL(M) = BT(M)
         DO J = 1,NPROP
            PHOLDL(M,J) = PMASS(M,J)
         END DO
      END DO
      DO M = 1,12
         OHOLDL(M) = OEM(M,6)
      END DO
      JF1 = 1
      GOTO 1988
   END IF
   SCOST = BCOST
   DO M = 1,15
      THOLDT(M) = T(M)
      BTHOLDT(M) = BT(M)
      DO J = 1,NPROP
         PHOLDT(M,J) = PMASS(M,J)
      END DO
   END DO
   DO M = 1,12
      OHOLDT(M) = OEM(M,6)
   END DO
   JDIR = JTRY + 1
   IF (JDIR .EQ. JHIGH) GOTO 1989
   JR = JDIR
   JF1 = 3
ELSE IF (JF1 .EQ. 3) THEN
   IF (SCOST .LT. BCOST) THEN
      JHIGH = JTRY
      DO M = 1,15
         THOLDH(M) = T(M)
         BTHOLDH(M) = BT(M)
         DO J = 1,NPROP
```

```
                 PHOLDH(M,J) = PMASS(M,J)
              END DO
           END DO
           DO M = 1,12
              OHOLDH(M) = OEM(M,6)
           END DO
           YCOSTH = SCOST
        ELSE
           JLOW = JDIR
           YCOSTL = BCOST
           DO M = 1,15
              THOLDL(M) = T(M)
              BTHOLDL(M) = BT(M)
              DO J = 1,NPROP
                 PHOLDL(M,J) = PMASS(M,J)
              END DO
           END DO
           DO M = 1,12
              OHOLDL(M) = OEM(M,6)
           END DO
        END IF
        JF1 = 1
        GOTO 1988
     END IF
2121    IF (DRADT(NTORB(IESEG(LS))).LT. DRADT(NTORB(IESEG(KT))))
     1     THEN
           OEM(NTORB(IESEG(LS)),6) = OEMREF1 +
     1                                  JR * 0.001745
        ELSE
           OEM(NTORB(IESEG(LS)),6) = OEMREF1 -
     1                                  JR * 0.001745
        END IF
        DO M = 1,NPROP
           PSMASS(M) = PMASS(LS,M)
        END DO
        CALL SW02(2,IBSEG(LS),IBLEG(LS),BT(LS),T(LS),
     1          PSMASS,T(LS+1),PEMASS,IELEG,BT(LS+1),KERR)
        DO M = 1,NPROP
           PMASS(LS+1,M) = PEMASS(M)
        END DO
        K = LS + 2
        LORB(NTORB(IESEG(LS))) = .TRUE.
        WRITE(6,*) 'JUMPING TO TRY INOVATION'
        WRITE(6,*) JR,BCOST
        GOTO 1131
1989 IF (YCOSTL .LT. YCOSTH) THEN
        DO M = 1,15
           T(M) = THOLDL(M)
           BT(M) = BTHOLDL(M)
           DO J = 1,NPROP
              PMASS(M,J) = PHOLDL(M,J)
           END DO
        END DO
        DO M = 1,12
```

```
                  OEM(M,6) = OHOLDL(M)
               END DO
            ELSE
               DO M = 1,15
                  T(M) = THOLDH(M)
                  BT(M) = BTHOLDH(M)
                  DO J = 1,NPROP
                     PMASS(M,J) = PHOLDH(M,J)
                  END DO
               END DO
               DO M = 1,12
                  OEM(M,6) = OHOLDH(M)
               END DO
            END IF
            IF (JF1 .EQ. 2 .AND. SCOST .LT. YCOSTL .AND. SCOST .LT.
     1          YCOSTH) THEN
               DO M = 1,15
                  T(M) = THOLDT(M)
                  BT(M) = BTHOLDT(M)
                  DO J = 1,NPROP
                     PMASS(M,J) = PHOLDT(M,J)
                  END DO
               END DO
               DO M = 1,12
                  OEM(M,6) = OHOLDT(M)
               END DO
            END IF
               IF (K .NE. NSEG + 1) THEN
               CALL SW02(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1              PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
            DO I = 1,NPROP
               PMASS(K,I) = PEMASS(I)
            END DO
               IBLEG(K) = IELEG + 1
C
C RUN FINAL SEGMENT TO GET MINIMUM TIME FOR IT
C
            IF (ITEMP1 .NE. -1) THEN
C
               ITEMP1 = 0
C
            END IF
C
            IF (KTRAN(IESEG(K)) .EQ. 4) THEN
               CALL SW02(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
     1                   TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
            ELSE IF (KTRAN(IESEG(K)) .EQ. 5) THEN
               CALL SW01(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
     1                   TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
            ELSE IF (KTRAN(IESEG(K)) .EQ. 3) THEN
               CALL SW03(ITEMP1,IBSEG(K),IBLEG(K),BT(K),T(K),PEMASS,
     1                   TM(K+1),PEMASS1,IELEG,BT(K+1),KERR)
            END IF
```

```
              END IF
           END IF
        END IF
C
C DOES THE SEGMENT END WITH A TOR ?
C
        IF (K .NE. 1) THEN
C
C
           IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
           WRITE(6,659) K
           WRITE(20,659) K
 659    FORMAT(' ','***** CALCULATING WINDOW CONSTRAINTS ON TIME#',I1,' **
       1***')
C
C GET WINDOWS AROUND T(K)
C
              TSTART = T(K) - 0.25
              TEND = T(K) + 0.25
 850          IF (TSTART .LT. T(K-1)) THEN
                 TSTART = T(K-1)
              END IF
              IF (TEND .GT. TMAXL) THEN
                 TEND = TMAXL
              END IF
              JSUN = 0
              JCOM = 0
              JLAT = 0
C
C ARE THERE SPECIFIED SUNLIGHT WINDOWS ?
C
              IF (LGEOM(IESEG(K-1),6) .AND. LGEOM(IESEG(K-1),7)) THEN
C
C GET RAW WINDOWS
C
                 JSUN = 1
                 CALL SUNWIN (NTORB(IESEG(K-1)),TSTART,TEND,TSUN)
C
C CALCULATE FEASIBLE T(K) REGION
C
                 I = 1
                 J = 1
 700             IF (TSUN(I) .NE. 9.0E9) THEN
                    IF (TSUN(I) .NE. TSTART) THEN
                       TSUNV(J) = TSUN(I) + GEOM(IESEG(K-1),6) /86400.
                       TSUNV(J+1) = TSUN(I) + GEOM(IESEG(K-1),7) /86400.
                       J = J + 2
                    END IF
                    IF (TSUN(I+1) .EQ. TEND) THEN
                       TSUNV(J) = 9.0E9
                    ELSE
                       I = I + 2
                       IF (I .NE. 101) GOTO 700
                    END IF
```

```
               ELSE
                  TSUNV(J) = 9.0E9
               END IF
            ELSE IF (LGEOM(IESEG(K-1),6) .OR. LGEOM(IESEG(K-1),7)) THEN
            CALL MSG('***HALT: YOU MUST ENTER BOTH SUNLIGHT CONSTRAINTS
     1OR NEITHER',3)
            STOP
            END IF
C
C ARE THERE SPECIFIED TRACKING WINDOWS ?
C
            IF (LGEOM(IESEG(K-1),10) .AND. LGEOM(IESEG(K-1),11)) THEN
               JCOM = 1
               CALL TRKWIN (NTORB(IESEG(K-1)),TSTART,TEND,TCOM)
C
C CALCULATE MINIMUM WINDOW LENGTH
C
               XCOM = (GEOM(IESEG(K-1),10) + GEOM(IESEG(K-1),11))/86400.
C
C CALCULATE FEASIBLE T(K) REGION
C
               I = 1
               J = 1
 701           IF (TCOM(I) .NE. 9.0E9) THEN
                  IF (TCOM(I+1) - TCOM(I) .GE. XCOM) THEN
                     TCOMV(J) = TCOM(I) + (GEOM(IESEG(K-1),10)) /86400.
                     TCOMV(J+1) = TCOM(I+1) - GEOM(IESEG(K-1),11)/86400.
                     J = J + 2
                  END IF
                  IF (TCOM(I+1) .EQ. TEND) THEN
                     TCOMV(J) = 9.0E9
                  ELSE
                     I = I + 2
                     IF (I .NE. 101) GOTO 701
                  END IF
               ELSE
                  TCOMV(J) = 9.0E9
               END IF
            ELSE IF (LGEOM(IESEG(K-1),10) .OR. LGEOM(IESEG(K-1),11))
     1               THEN
            CALL MSG('***HALT: YOU MUST ENTER BOTH TRACKING CONSTRAINTS
     1OR NEITHER',3)
            STOP
            END IF
            IF (LGEOM(IESEG(K-1),8) .AND. LGEOM(IESEG(K-1),9)) THEN
               JLAT = 1
               TASTART = TANOM(NTORB(IESEG(K-1)),TSTART)
               WSTART = ANG(OEM(NTORB(IESEG(K-1)),4) +
     1                  (TSTART - ODJ(NTORB(IESEG(K-1)))) *
     2                  DAPDT(NTORB(IESEG(K-1))))

               ALSTART = ANG(TASTART + WSTART)
               ALAT1 = ANG(GEOM(IESEG(K-1),8))
               ALAT2 = ANG(GEOM(IESEG(K-1),9))
```


C-3

```
                  IF (ALAT2 .LT. ALAT1) THEN
                     ALAT2 = ALAT2 + TWOPI
                  END IF
                  DAL1 = ALAT1 - ALSTART
                  DAL2 = ALAT2 - ALSTART
                  CALL ARC(ALSTART,WSTART,OEM(NTORB(IESEG(K-1)),2),
     1                     DMADT(NTORB(IESEG(K-1))),
     2                     DAPDT(NTORB(IESEG(K-1))),DT,DAL1,-1)
                  T1 = TSTART + DT
                  CALL ARC(ALSTART,WSTART,OEM(NTORB(IESEG(K-1)),2),
     1                     DMADT(NTORB(IESEG(K-1))),
     2                     DAPDT(NTORB(IESEG(K-1))),DT,DAL2,-1)
                  T2 = TSTART + DT
                  IF (TSTART .LT. T1) THEN
                     L = 0
                     J = 1
                  ELSE IF (TSTART .LT. T2) THEN
                     TLAT(1) = TSTART
                     TLAT(2) = T2
                     J = 3
                     L = 1
                  ELSE
                     J = 1
                     L = 1
                  END IF
 202              CALL ARC(ALSTART,WSTART,OEM(NTORB(IESEG(K-1)),2),
     1                     DMADT(NTORB(IESEG(K-1))),
     2                     DAPDT(NTORB(IESEG(K-1))),DT,DAL1 + TWOPI * L,-1)
                  TLAT(J) = TSTART + DT
                  CALL ARC(ALSTART,WSTART,OEM(NTORB(IESEG(K-1)),2),
     1                     DMADT(NTORB(IESEG(K-1))),
     2                     DAPDT(NTORB(IESEG(K-1))),DT,DAL2 + TWOPI * L,-1)
                  TLAT(J+1) = TSTART + DT
                  IF (TLAT(J) .GT. TEND) THEN
                     TLAT(J) = 9.0E9
                  ELSE IF (TLAT(J+1) .GT. TEND) THEN
                     TLAT(J+1) = TEND
                     TLAT(J+2) = 9.0E9
                  ELSE
                     L = L + 1
                     J = J + 2
                     GOTO 202
                  END IF
               ELSE IF (LGEOM(IESEG(K-1),8) .OR. LGEOM(IESEG(K-1),9)) THEN
               CALL MSG('***HALT: YOU MUST ENTER BOTH LATITUDE CONSTRAINTS
     1OR NEITHER',3)
               STOP
               END IF
C               DO J = 1,50
C                  WRITE(21,998) J,TSUN(J),TCOM(J),TSUNV(J),TCOMV(J),TLAT(J)
C               END DO
 998           FORMAT(' ',I3,5F20.7)
C
               IF (JSUN .EQ. 1) THEN
```

```
C
                  IF (JCOM .EQ. 1) THEN
C
C CALCULATE THE INTERSECTION OF 2 WINDOW ARRAYS
C
                     CALL WINDINT(TSUNV,TCOMV,XJOINT,INDX)
C
                  ELSE
C
                     J = 0
 710                 J = J + 1
                     XJOINT(J) = TSUNV(J)
                     IF (J .NE. 100 .AND. TSUNV(J) .NE. 9.0E9) GOTO 710
                     INDX = J
C
                  END IF
C
               ELSE IF (JCOM .EQ. 1) THEN
C
                  J = 0
 711              J = J + 1
                  XJOINT(J) = TCOMV(J)
                  IF (J .NE. 100 .AND. TCOMV(J) .NE. 9.0E9) GOTO 711
                  INDX = J
C
               ELSE
C
                  XJOINT(1) = TSTART
                  XJOINT(2) = TEND
                  XJOINT(3) = 9.0E9
                  INDX = 3
C
               END IF
C
               IF (JLAT .EQ. 1) THEN
C
                  CALL WINDINT(XJOINT,TLAT,XJOINT1,INDX)
                  DO J = 1,INDX
                     XJOINT(J) = XJOINT1(J)
                  END DO
               END IF
C
C               DO J = 1,INDX
C                  WRITE(21,*) J,XJOINT(J)
C               END DO
               I = 1
 702           IF (XJOINT(I) .LT. T(K)) THEN
                  I = I + 1
                  GOTO 702
               END IF
               IF (I - 2.0 * INT(I/2) .NE. 0) THEN
                  J = 0
 705              IF (J .LT. I-1) THEN
                     WTRY(1) = XJOINT(I-1-J)
```

```
                  WTRY(2) = (XJOINT(I-1-J) + XJOINT(I-2-J)) * 0.5
                  WTRY(3) = XJOINT(I-2-J)
                  L = 1
               ELSE
                  L = 4
               END IF
               IF (J .LT. INDX-I) THEN
                  WTRY(4) = XJOINT(I+J)
                  WTRY(5) = (XJOINT(I+J) + XJOINT(I+1+J)) * 0.5
                  WTRY(6) = XJOINT(I+1+J)
                  M = 6
               ELSE
                  M = 3
               END IF
               IF (L .GT. M) THEN
                  IF (IFULL .EQ. 0) THEN
                  TSTART= T(K) - 1.0
                  TEND = T(K) + 1.0
                  IFULL = 1
                  GOTO 850
                  ELSE
                  IFULL=0
                  IF (K .EQ. ISTRT) THEN
               CALL MSG('***HALT: NO FEASIBLE SOLUTION WITHIN MAXIMUM MI
     1SSION CONSTRAINTS CAN BE FOUND',3)
                  STOP
                  ELSE
                  WRITE(6,660)
                  WRITE(20,660)
 660  FORMAT(' ','*** NO FEASIBLE WINDOWS FOUND WITHIN 1 DAY OF SELECTED
     1 TIME. BACK UP 1 TIME ***')
                  K = K - 1
                  LOPT(K) = .FALSE.
                  GOTO 1131
                  END IF
                 END IF
               ELSE
                  IFULL = 0
               END IF
 706           CONTINUE
C              WRITE(21,*) WTRY(L)
               CALL SW01(2,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1                    PSMASS,WTRY(L),PEMASS,IELEG,BT(K),KERR)
               IWK = 1
               IF (KERR .NE. 0) THEN
                  IWK = 0
               END IF
               IF (IWK .EQ. 0) THEN
                  L = L + 1
                  IF (L .LE. M) GOTO 706
                  J = J + 2
                  GOTO 705
               END IF
               T(K) = WTRY(L)
```

[redacted]

```
                TK(K) = T(K)
                DO I = 1,NPROP
                   PMASS(K,I) = PEMASS(I)
                END DO
             END IF
          WRITE(6,662) K
          WRITE(20,662) K
 662      FORMAT(' ','*** WINDOW FITTED FOR TIME#',I1,' ***')
          END IF
       END IF
C
C IS THIS NOT THE LAST FREE TIME ?
C
 661   IF (K .NE. NSEG + 1) GOTO 1130
C
C CHECK TO SEE ABOUT BURNS
C
 1850 IADJ = 0
       DO I = 1,NTRAN
C
          IF (NBITS(I) .GT. 12) THEN
C
             NBITS(I) = 12
             CALL MSG('***MAXIMUM BURN TIME LIMITED EXCEEDED. RESULT IS F
     1OR THE 12 BURN MAXIMUM',1)
C
          END IF
C

          IF (NBIT(I) .NE. NBITS(I)) THEN
C
             IADJ = 1
C
          END IF
C
       END DO
C
       IF (IADJ .EQ. 1 .AND. JADJ .LE. 3) THEN
C
          DO I = 1,NTRAN
             NBIT(I) = NBITS(I)
          END DO
          JADJ = JADJ + 1
          WRITE(6,658)
          WRITE(20,658)
          GOTO 1700
C
       END IF
C
C TURN PRINTOUT ON
       MPRINT = 1
       WRITE(6,663)
       WRITE(20,663)
 663   FORMAT(' ','----FINALIZING SOLUTION-------')
```

```
C
C RUN SEGMENTS THROUGH IN MODE 3
C
      TK(1) = T(1)
      PDMIN=ORBMIN(1)*TWOPI/(DAPDT(NORB0) +DMADT(NORB0))
      DO IMODE1 = 2,3
      IF (IMODE1 .EQ. 3 .AND. .NOT. LREF) THEN
         T(1) = T(1) + STAY(1) - PDMIN - 0.007
      END IF
      DO K = 2,NSEG+1
      T(K) = TK(K)
      END DO
      DO K = 2,NSEG + 1
        DO I = 1,NPROP
           PSMASS(I) = PMASS(K-1,I)
        END DO
        IF (IMODE1 .EQ. 3) THEN
           TSTAY = STAY(K)
        END IF
        IF (KTRAN(IESEG(K-1)) .EQ. 5) THEN
        CALL SW01(IMODE1,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1             PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
        ELSE IF (KTRAN(IESEG(K-1)) .EQ. 4) THEN
        CALL SW02(IMODE1,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1             PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
        ELSE IF (KTRAN(IESEG(K-1)) .EQ. 3) THEN
        CALL SW03(IMODE1,IBSEG(K-1),IBLEG(K-1),BT(K-1),T(K-1),
     1             PSMASS,T(K),PEMASS,IELEG,BT(K),KERR)
        END IF
        IF (IMODE1 .EQ. 2) THEN
           STAY(K-1) = TSTAY
           TADJ(K) = T(K)
        END IF
        IF (KERR .NE. 0) THEN
            WRITE(20,*) IMODE1,K,T(K)
            CALL MSG('***HALT: ERROR ENCOUNTERED IN MODE 3 CALCULATIONS'
     1,3)
            STOP
        END IF
        DO I = 1,NPROP
           PMASS(K,I) = PEMASS(I)
        END DO
        IBLEG(K) = IELEG + 1
        IF (T(K) .NE. TADJ(K) .AND. K .NE. NSEG + 1 .AND. IMODE1 .EQ. 3)
     1     THEN
           ORBMIN(IBSEG(K)) = ORBMIN(IBSEG(K)) + TADJ(K) - T(K)
           IF (ORBMIN(IBSEG(K)) .LT. 0.0) ORBMIN(IBSEG(K)) = 0.0
           ORBMAX(IBSEG(K)) = ORBMAX(IBSEG(K)) + TADJ(K) - T(K)
        END IF
      END DO
      END DO
      CALL OUTPUT
      RETURN
      END
```