Kansas Geological Survey
Open File Report 96-66

Source Code for the LEO System for Automated Conversion Between Public Land Survey and Geographic Reference Systems In Kansas

Version 3.4

December 29, 1996
(Revised 2/14/97)

by

David R. Collins, Ph.D.
and
Robert Sampson

CONTENTS

```
C
C               * * * * *     L E O 3 S U N     * * * * *
C
C               **  Reference conversion calling program for the
C               **  LEO3CVT routine, converting between Legal and
C               **  Geographic reference in Kansas, and operating
C               **  interactively or with data in disk files.
C               **  **  This version is for the SUN/UNIX environment **
C
C               **  This version includes proper handling of the four
C               **  corners OPTION and the side lengths OPTION for both
C               **  interactive and batch processing
C
C       Additional references:
C       Collins, D. R., 1996, Automated conversion between public land survey
C               and _geographic reference systems in Kansas, KGS Open File C
                Report 96-64.
C       Collins, D. R., 1996, User manual for the LEO System, Version 3, KGS
C               Open File Report 96-65.
C
C
C
C
C               **  additional subroutines (for compiling):
C                  se.f -- subdivision of sections on state edges
C                  slack.f -- subdivision of sections
C                           1-6,7,18,19,30,31
C                  utmsub.f -- utm/geographic conversions
c
C               **  Declarations
C
        IMPLICIT NONE
        CHARACTER EW*1, SUBD*2(4), CORNR*2, IFILE*60, OFILE*60,
     &     VERSION*4, BLANK2*2, DBNAME*60, COMP*12,cornrg1*2
        CHARACTER*2 sub1,sub2,sub3,sub4
        INTEGER*2 TSHIP, RANGE, SECT, STATUS, SOURCE, OPTION, I, J,
     &     NONSTD, UNL/1/, UNI/2/, UNO/3/, BATCH, INXTRA, RECORD,
     &     REMAIN, TFLAG, SFLAG(36),stemp,SCOR(4)
        INTEGER*2 maxlvl,OPTION2,ALPHA,BETA,GAMMA,
     &     STLINE,RIVER,RESERVE,TREATY,SLACKN,SLACKW,SLACKNW,SLACK,
     &     utmform,zone,utmdir,result,PLSSMAT
        LOGICAL DONE, EKO, FOURSIDE, FOURCORN, XTRA, TRACE
        REAL PLON, PLAT, FTNS, FTEW, LONX, LATX, latlon(49,2),
     &     sfactor,fm
        double precision  utmx,utmy,utmk,lambda,phi
        common/leotest/trace
        COMMON /LEOXTR/LONX(4,3),LATX(4,3),FTNS,FTEW,NONSTD,CORNR,
     &         maxlvl,OPTION2,cornrg1
        COMMON/LEOTYPE/STLINE,RIVER,RESERVE,TREATY,SLACKN,SLACKW,SLACKNW
        COMMON /LEOFLG/ TFLAG,SFLAG,SCOR
        DATA VERSION/'3.4'/, BLANK2/'  '/, DBNAME/'LEO3BASE'/,
     &    COMP/'2 - 14 - 97'/
c       conversion factor from feet to meters:
        fm=3.28083
C
C
```

```
C ***************************** INITIALIZATION *************************
C
C
C ***************  Determine mode and open disk files if requested *****
C
C
       UNL=1
       OPEN (UNL, FILE=DBNAME,ACCESS='DIRECT',
     &          FORM='UNFORMATTED', RECL=466)
c
c              **  check version from date in record 68, sflag(1-3)
c
       read (unl,rec=68) latlon, tflag, sflag
       write (*,*) '         * * * * *   N O T E   * * * * *'
       write (*,*) ' '
       write (*,*) 'This version of LEO3BASE was compiled on ',
     &         sflag(1),' -',sflag(2),' -',sflag(3)
       write (*,*)
     &    'This is Version ',version,' of the LEO System, compiled on ',
     &         comp
       write (*,*) ' '
       write (*,*) 'Copyright, Kansas Geological Survey, 1997'
       write (*,*) ' '
       write (*,*) 'David R. Collins, Ph.D.'
       write (*,*) 'Robert J. Sampson'
       write (*,*) 'Kansas Geological Survey  --  913-864-3965'
       write (*,*) ' '
       write (*,*) '         * * * * * * * * * * * * * * * * *'
       write (*,*) ' '
       CLOSE(UNL)
C
```

```
10      continue
        WRITE (*,*) ' '
        WRITE (*,*) 'Enter mode, 0 = Interactive, 1 = Disk Files: '
        READ (*,*,ERR=10) BATCH
        IF (BATCH.LT.0 .OR. BATCH.GT.1) STOP
c
c       Set-up for batch mode
c
        IF (BATCH.EQ.1) THEN
20       Continue
         WRITE (*,*) ' '
         WRITE (*,*) 'Enter filename for input data: '
         READ (*,'(A)',ERR=20) IFILE
C
C
30       continue
c
c  ********************************************************************
         EKO = .FALSE.
         TRACE = .FALSE.
C  ********************************************************************
C
         WRITE (*,*) ' '
         WRITE (*,*) 'Enter 1 to echo input data, 0 if not: '
         READ (*,'(I1)',ERR=30) I
         IF (I.EQ.1) THEN
          EKO = .TRUE.
35        continue
          WRITE (*,*)'Enter 1 for Trace to output file, 0 if not: '
          READ (*,*,ERR=35) I
          IF (I.EQ.1) THEN
           TRACE = .TRUE.
          ELSE
           TRACE = .FALSE.
          ENDIF
         ELSE
          EKO= .FALSE.
          TRACE = .FALSE.
         ENDIF
c
c  ********************************************************************
c
40       continue
         WRITE (*,*) ' '
         WRITE (*,*) 'Enter filename for output values: '
         READ (*,'(A)',ERR=40) OFILE
         OPEN (UNI,FILE=IFILE,STATUS='OLD')
         OPEN (UNO,FILE=OFILE,STATUS='NEW')
        ENDIF
C
C               **  Open the LEO3 database
C
        UNL=1
        OPEN (UNL, FILE=DBNAME,ACCESS='DIRECT',
     &          FORM='UNFORMATTED', RECL=466)
c
```

```
C       SET SOURCE
c
50      continue
        WRITE (*,*)' '
        WRITE (*,*)'Data source options:'
        WRITE (*,*)'  -2 = STOP'
        write (*,*)'  -1 = Reset mode (Interactive/Disk)'
        WRITE (*,*)'   1 = LEGAL'
        WRITE (*,*)'   2 = LONGITUDE / LATITUDE'
        write (*,*)'   3 = UTM [Universal Transverse Mercator]'
        WRITE (*,*) 'Select source: '
        READ (*,*,ERR=50) SOURCE
        if(source.lt.-2.or.source.gt.3.or.source.eq.0) goto 50
        if(source.eq.-2)stop
        if(source.eq.-1) GOTO 10
c
c       Set-up for batch with PLSS source
C
        if(batch.eq.1.and.source.eq.1)then
C
1142     CONTINUE
C
         WRITE(*,*)'INPUT FORMAT OPTIONS'
         WRITE(*,*)'          -- for Public Land Survey locations'
         WRITE(*,*)' -1 = Reset all options'
         WRITE(*,*)'  1 = From largest to smallest designation:'
         WRITE(*,*)
     &      '        TSHP, RANGE, E/W, SECT, SUBD(4) large to small,'
         WRITE(*,*)'          CORNER,FOOTAGE N/S, FOOTAGE E/W'
         WRITE(*,*)
     &      '     FORMAT(2I3,1X,A1,I3,4(1X,A2),1X,A2,2F8.0)'
C
         WRITE(*,*)'  2 = From smallest to largest designation:'
         WRITE(*,*)'       FOOTAGE N/S, FOOTAGE E/W, CORNER,'
         WRITE(*,*)
     &      '          SUBD(4) small to large, SECT, TSHP, RANGE, E/W,'
         WRITE(*,*)
     &      '     FORMAT(2F8.0,1X,A2,4(1X,A2),3I3,1X,A1)'
         WRITE(*,*)' '
         WRITE(*,*)'ENTER APPROPRIATE FORMAT OPTION:  '
C
         READ(*,*,ERR=1142)PLSSMAT
         IF((PLSSMAT.LT.1.AND.PLSSMAT.NE.-1).
     &                 OR.PLSSMAT.GT.2)GOTO 1142
         IF(PLSSMAT.eq.-1)goto 10
        endif
C
C
```

```
53      CONTINUE
        write (*,*)' '
        write (*,*)'Data output type options:'
        write (*,*)' -1 = Reset source option'
        if(source.eq.2.or.source.eq.3)then
c        source is geographic or utm
         write(*,*)'  1 = Legal [UTM or Lon/Lat if outside KS PLSS]'
c
c        NOTE:  If legal output format is requested where the PLSS does
c               not apply; Lon/Lat input will be converted to UTM
c               output, while UTM input will be converted to Lon/Lat.
c
        endif
        if(source.eq.1.or.source.eq.3)then
c        source is legal or utm
         write(*,*)'  2 = Lon/Lat'
        endif
        if(source.eq.1.or.source.eq.2)then
c        source is legal or geographic
         write(*,*)'  3 = UTM'
        endif
        WRITE(*,*)'Select data output type:  '
        read(*,*,err=53)RESULT
        if(result.lt.1.or.result.gt.3)goto 50
        if(source.eq.result)goto 50
c        assume source is utm then direction is NOT toward utm
        utmdir=0
c        but if source is not utm, set direction toward utm
        if(source.ne.3)utmdir=1
```

```
51      continue
c ***********************************************************************
C
C **** LEGAL TO GEOGRAPHIC or LEGAL TO UTM CONVERSIONS ***************
C
C***********************************************************************
c
        IF(SOURCE.EQ.1)THEN
         if(result.eq.3)then
          write(*,*)' '
          write(*,*)'Output options for UTM coordinates:'
          write(*,*)
     &      ' -1 = Reset source option'
          write(*,*)
     &      '  0 = find X, Y, UTM zone, scale factor for each location'
          write(*,*)
     &      '  1 = find X, Y, UTM zone for each location'
          write(*,*)
     &      '  2 = find X, Y, scale factor relative to selected UTM zone'
          write(*,*)
     &      '  3 = find X,Y relative to selected UTM zone'
          write(*,*)'Select UTM output format: '
          read(*,*,err=51)utmform
          if(utmform.lt.-1.or.utmform.gt.3)goto 51
          if(utmform.eq.-1)goto 50
52        continue
          zone=0
          if(utmform.gt.1)then
           write(*,*)'Select the UTM zone common to the most'
           write(*,*)'  significant portion of your data:'
           write(*,*)'     zone 15 -- 90-96 deg west longitude'
           write(*,*)'     zone 14 -- 96-102 deg west longitude'
           write(*,*)'     zone 13 -- 102-108 deg west longitude'
           write(*,*)'Specify UTM zone [13 CO, 14 W KS , 15 E KS]: '
           read(*,*,err=52)zone
           if(zone.lt.13.or.zone.gt.15)goto 52
          endif
         endif
C
C
C
C
C
```

```
C       ********************** BATCH CONVERSION ******************
C
C
        IF (BATCH.EQ.1) THEN
C             **  Batch Initialization  **
C        ** SET OPTION
C
1140     WRITE(*,*)' '
         WRITE(*,*)'DATA CONVERSION OPTIONS'
         WRITE(*,*)'  -2 = Stop program'
         WRITE(*,*)'  -1 = Reset all options'
         if(result.eq.2)then
          WRITE(*,*)
     &      '* Options 0-3 (decimal degrees) == Options 4-7 (DMS) *'
         else
          write(*,*)'* Options 0-3 (meters)'
         endif
         WRITE(*,*)
     &   ' 0/4 = Return four corners of specified subdivision or, if'
         WRITE(*,*)
     &   '       requested, its center, a corner or a side midpoint'
         WRITE(*,*)
     &     '       (not used if section = 0, implying township)'
c            Options 0/4 will return four corners if a point location
c            (i.e., center, corner or side mid-point) is not explicitly
c            requested.  With these options, "C" means 'CENTER.'
         WRITE(*,*)' 1/5 = Select section corner or side midpoint'
c            Options 1/5 will return four corners if a point location
c            (i.e., center, corner or side mid-point) is not explicitly
c            requested.  With these options, "C" means 'SW CORNER.'
c            Options 1/5 will NOT return a CENTER location.
         WRITE(*,*)
     &   ' 2/6 = Apply footages from section corner to point location'
         WRITE(*,*)
     &     '       (not used if section = 0, implying township)'
         WRITE(*,*)' 3/7 = Return locations of the section corners'
         WRITE(*,*) ' '
         WRITE(*,*) 'Enter Option Number (-2 to 7): '
         READ (*,*,ERR=1140) OPTION
         IF(OPTION.GT.7.or.OPTION.lt.-2)GOTO 1140
         if(OPTION.eq.-2)stop
1141     continue
         IF (OPTION.EQ.-1) GOTO 50
C
C
60       WRITE (*,*) 'Enter 0=OMIT or 1=INCLUDE in the output file'
         WRITE (*,*) '  the Status and Nonstandard Section indicators?'
         READ (*,*,ERR=60) INXTRA
         IF (INXTRA.EQ.1) THEN
          XTRA = .TRUE.
         ELSE
          XTRA = .FALSE.
         ENDIF
C
C
C
C
```

```
          RECORD = 0
C
C               **  First read and write records for the disk mode
C
401       FORMAT (2I3,1X,A1,1X,I2,5(1X,A2),2F8.0)
402       FORMAT (2F8.0,5(1X,A2),1X,I2,2I3,1X,A1)
75        CONTINUE
          slack=0
          slackn=0
          slackw=0
          slacknw=0
          IF(PLSSMAT.EQ.1)THEN
           READ (UNI,401,END=90) TSHIP, RANGE, EW, SECT, SUBD,
     &          CORNR, FTNS, FTEW
          ELSE
           READ (UNI,402,END=90) FTNS, FTEW, CORNR, SUBD, SECT,
     &             TSHIP, RANGE, EW
          ENDIF
          RECORD = RECORD + 1
          IF (EKO) THEN
           IF(PLSSMAT.EQ.1)THEN
            WRITE (*,81) RECORD, OPTION, TSHIP, RANGE, EW,
     &       SECT, SUBD, CORNR, FTNS, FTEW
           ELSE
            WRITE (*,82) RECORD, OPTION, FTNS, FTEW, CORNR,
     &       SUBD, SECT, TSHIP, RANGE, EW
           ENDIF
          endif
          IF (TRACE) THEN
           IF(PLSSMAT.EQ.1)THEN
            WRITE (uno,81) RECORD, OPTION, TSHIP, RANGE, EW,
     &       SECT, SUBD, CORNR, FTNS, FTEW
           ELSE
            WRITE (uno,82) RECORD, OPTION, FTNS, FTEW, CORNR,
     &       SUBD, SECT, TSHIP, RANGE, EW
           ENDIF
          ENDIF
C
81        FORMAT (' Record',I5,' Opt=',I3,5X,2I4,A1,2X,I3,
     &                  5A3,2F8.1)
82        FORMAT (' Record',I5,' Opt=',5X,2F8.1,5A3,2X,I3,
     &                  2I4,A1)
C
          IF(PLSSMAT.EQ.2)THEN
           sub1=SUBD(4)
           sub2=SUBD(3)
           sub3=SUBD(2)
           sub4=SUBD(1)
           SUBD(1)=sub1
           SUBD(2)=sub2
           SUBD(3)=sub3
           SUBD(4)=sub4
          ENDIF
C
C
```

```
C       ***** CONVERT REFERENCE SYSTEMS: PLSS to GEOGRAPHIC
C
          CALL LEO3CVT (TSHIP, RANGE, EW, SECT, OPTION, SUBD,
     &     PLON, PLAT, UNL, STATUS, SOURCE)
C
C          NONSTD
C              = 2, if STLINE = 1
C              = 4, if RIVER = 1
C              = 6, if STLINE = 1, and RIVER = 1
C              = 8, if RESERVE = 1
C              = 10, if STLINE = 1, and RESERVE = 1
C              = 12, if RIVER = 1, and RESERVE = 1
C              = 14, if STLINE = 1, and RIVER = 1, and RESERVE = 1
C              = 16, if TREATY = 1
C              = 18, if STLINE = 1, and TREATY = 1
C              = 20, if RIVER = 1, and TREATY = 1
C              = 22, if STLINE = 1, and RIVER = 1, and TREATY = 1
C              = 24, if RESERVE = 1, and TREATY = 1
C              = 26, if STLINE = 1, and RESERVE = 1, and TREATY = 1
C              = 28, if RIVER = 1, and RESERVE = 1, and TREATY = 1
C              = 30, if STLINE = 1, and RIVER = 1, and RESERVE = 1,
C     &             and TREATY = 1
C              = 32, if SLACKN = 1, or SLACKW = 1, or SLACKNW =1
C              = 34, if STLINE = 1 & SLACK
C              = 36, if RIVER = 1 & SLACK
C              = 38, if STLINE = 1, and RIVER = 1 & SLACK
C              = 40, if RESERVE = 1 & SLACK
C              = 42, if STLINE = 1, and RESERVE = 1 & SLACK
C              = 44, if RIVER = 1, and RESERVE = 1 & SLACK
C              = 46, if STLINE=1, and RIVER=1, and RESERVE=1 & SLACK
C              = 48, if TREATY = 1 & SLACK
C              = 50, if STLINE = 1, and TREATY = 1 & SLACK
C              = 52, if RIVER = 1, and TREATY = 1 & SLACK
C              = 54, if STLINE=1, and RIVER=1, and TREATY=1 & SLACK
C              = 56, if RESERVE = 1, and TREATY = 1 & SLACK
C              = 58, if STLINE=1, and RESERVE=1, and TREATY=1 & SLACK
C              = 60, if RIVER=1, and RESERVE=1, and TREATY=1 & SLACK
C              = 62, if STLINE = 1, and RIVER = 1, and RESERVE = 1,
C     &             and TREATY = 1 & SLACK
C
          IF(SLACKN.EQ.1.OR.SLACKW.EQ.1.OR.SLACKNW.EQ.1)SLACK=1
          IF(NONSTD.EQ.1.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &         RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
           NONSTD=0
           IF(STLINE.EQ.1)NONSTD=NONSTD+2
           IF(RIVER.EQ.1)NONSTD=NONSTD+4
           IF(RESERVE.EQ.1)NONSTD=NONSTD+8
           IF(TREATY.EQ.1)NONSTD=NONSTD+16
           IF(SLACK.EQ.1)NONSTD=NONSTD+32
          ELSEIF(NONSTD.EQ.0.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &         RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
           IF(STLINE.EQ.1)NONSTD=NONSTD+2
           IF(RIVER.EQ.1)NONSTD=NONSTD+4
           IF(RESERVE.EQ.1)NONSTD=NONSTD+8
           IF(TREATY.EQ.1)NONSTD=NONSTD+16
           IF(SLACK.EQ.1)NONSTD=NONSTD+32
          ENDIF
C
```

```
C               **  Determine output option and output the results to
C               **  the batch output file
C
c
c       ***** SET FOURCORN and FOURSIDE
c
c
          FOURCORN = .FALSE.
          FOURSIDE = .FALSE.
          IF ((OPTION.EQ.0.OR.OPTION.EQ.4).AND.CORNR.EQ.'   ')THEN
           FOURCORN = .TRUE.
          endif
          IF ((OPTION.EQ.1.OR.OPTION.EQ.5).AND.CORNR.EQ.'   ')THEN
           FOURCORN = .TRUE.
          ELSEIF (OPTION.EQ.3 .OR. OPTION.EQ.7) THEN
           FOURCORN = .TRUE.
          ELSEIF (OPTION.EQ.2 .OR. OPTION.EQ.6) THEN
           IF((ABS(FTNS)+ABS(FTEW)).LT.1.) FOURSIDE=.TRUE.
          ENDIF
c
```

```
c              Output results in geographic if RESULT = 2 was
C              selected, otherwise output in UTM coordinates
c
          IF(result.eq.2)THEN
c
           IF (FOURCORN) THEN
C
C ********************* Section or subarea four corners OPTION *********
C
            IF (XTRA) THEN
C
C                 **  Output 4 corners of section or subarea with Status
C                 **  and Non-std section values, in DMS then in Decimal
C                 **  degrees, in the order NW, NE, SW, SE
C
              IF (OPTION.GT.3) THEN
               WRITE(UNO,'(1X,2(2(F5.0,F4.0,F7.3),2X),2I4,2X,I5)')
     &                    ((LONX(I,J),J=1,3),(LATX(I,J),J=1,3),I=1,2),
     &                          STATUS,NONSTD,SFLAG(SECT)
               WRITE (UNO,'(1X,2(2(F5.0,F4.0,F7.3),2X))')
     &                    ((LONX(I,J),J=1,3),(LATX(I,J),J=1,3),I=3,4)
              ELSE
               WRITE (UNO,'(1X,2(F11.5,F9.5),3X,2I4,2X,I5)')
     &               (LONX(I,1),LATX(I,1),I=1,2),STATUS,NONSTD,SFLAG(SECT)
                WRITE (UNO,'(1X,2(F11.5,F9.5))')
     &                    (LONX(I,1),LATX(I,1),I=3,4)
              ENDIF
             ELSE
C
C            **  Output 4 section or subarea corners w/o extra values
C
              IF (OPTION.GT.3) THEN
               WRITE (UNO,'(1X,2(2(F5.0,F4.0,F7.3),2X))')
     &                    ((LONX(I,J),J=1,3),(LATX(I,J),J=1,3),I=1,2)
               WRITE (UNO,'(1X,2(2(F5.0,F4.0,F7.3),2X))')
     &                    ((LONX(I,J),J=1,3),(LATX(I,J),J=1,3),I=3,4)
              ELSE
               WRITE (UNO,'(1X,2(F11.5,F10.5),4X)')
     &                    (LONX(I,1),LATX(I,1),I=1,2)
               WRITE (UNO,'(1X,4(F11.5,F10.5),4X)')
     &                    (LONX(I,1),LATX(I,1),I=3,4)
              ENDIF
             ENDIF
            ELSEIF (FOURSIDE) THEN
C
C                 **  Output 4 side lengths in feet in the order N,E,W,S
C
             IF (XTRA) THEN
              WRITE(UNO,'(1X,4F8.1,2I4,2X,I5)') (LONX(I,1),I=1,4),
     &                    STATUS, NONSTD, SFLAG(SECT)
             ELSE
              WRITE (UNO,'(1X,4F8.1)') (LONX(I,1),I=1,4)
             ENDIF
            ELSE
C
```

```
C               **  Output a single point for all other options (0 and 4
C               **  with CORNR nonblank, 1 and 5, and 2 and 6) first with
C               **  the extra Status and Nonstd values, then without.
C               **  For each, do Decimal Degrees and then DMS options
C
             IF (XTRA) THEN
              IF (OPTION.LT.4) THEN
               WRITE (UNO,'(1X,F11.5,F9.5,2I4,2X,I5)')
     &                     PLON,PLAT,STATUS,NONSTD,SFLAG(SECT)
              ELSE
               WRITE (UNO,'(1X,2(F5.0,F4.0,F7.3,2X),2I4,2X,I5)')
     &                     (LONX(1,J),J=1,3),(LATX(1,J),J=1,3),
     &                     STATUS,NONSTD,SFLAG(SECT)
              ENDIF
             ELSE
              IF (OPTION.LT.4) THEN
               WRITE (UNO,'(1X,F11.5,F9.5)') PLON,PLAT
              ELSE
               WRITE (UNO,'(1X,2(F5.0,F4.0,F7.3,2X))')
     &                     (LONX(1,J),J=1,3),(LATX(1,J),J=1,3)
              ENDIF
             ENDIF
            ENDIF
C
```

```
c         ***  Output in UTM coordinates  ***
c
C         Output options for UTM coordinates:
C          0 = find X, Y, UTM zone, scale factor for each location
C          1 = find X, Y, UTM zone for each location
C          2 = find X, Y, scale factor relative to selected UTM zone
C          3 = find X,Y relative to selected UTM zone
C
           ELSE
c
            IF (FOURCORN) THEN
C
C ********************* Section or subarea four corners option *********
C
C                  **  Convert locations from decimal degrees to UTM
C
             do 666 i=1,4
                lambda=lonx(i,1)
                phi=latx(i,1)
c               write(*,*)'zone = ',zone,'  longitude = ',
c    &                    lambda,'  latitude = ',phi
                call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &                    utmdir,utmform)
                sfactor=utmk
                lonx(i,1)=utmx
                latx(i,1)=utmy
                lonx(i,2)=zone
666          continue
c
            IF (XTRA) THEN
C
C                  **  Output 4 corners of section or subarea with Status
C                  **  and Non-std section values, in UTM coordinates
C                 **  in the order NW, NE, SW, SE
C
             if(utmform.eq.1.or.utmform.eq.3)then
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x),1X,2I4,2X,I5)')
     &             (LONX(I,1),LATX(I,1),lonx(i,2),I=1,2),
     &             STATUS,NONSTD,SFLAG(SECT)
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=3,4)
             else
              WRITE (UNO,
     &                '(2(2f12.2,1x,f2.0,2x),1x,f6.4,3X,2I4,2X,I5)')
     &             (LONX(I,1),LATX(I,1),lonx(i,2),I=1,2),sfactor,
     &             STATUS,NONSTD,SFLAG(SECT)
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=3,4)
             endif
c
            ELSE
```

```
C
C          **  Output 4 section or subarea corners w/o extra values
C
             if(utmform.eq.1.or.utmform.eq.3)then
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=1,2)
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=3,4)
             else
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x),1x,f6.4)')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=1,2),sfactor
              WRITE (UNO,'(2(2f12.2,1x,f2.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=3,4)
             endif
            ENDIF
           ELSEIF (FOURSIDE) THEN
C
C          **  Output 4 side lengths in meters in the order:  N,W,E,S
C
            IF (XTRA) THEN
             WRITE(UNO,'(1X,4F8.1,2I4,2X,I5)')
     &                (LONX(I,1)/fm,I=1,4),STATUS, NONSTD, SFLAG(SECT)
            ELSE
             WRITE (UNO,'(1X,4F8.1)') (LONX(I,1)/fm,I=1,4)
            ENDIF
           ELSE
C
```

```
C               **  Output a single point for all other options (0 and 4
C               **  with CORNR nonblank, 1 and 5, and 2 and 6) first with
C               **  the extra Status and Nonstd values, then without.
C               **
C
c
              lambda=plon
              phi=plat
              if (TRACE) then
               write(*,*)'zone = ',zone,'  longitude = ',
     &                  lambda,'  latitude = ',phi
              endif
              call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &              utmdir,utmform)
              sfactor=utmk
              if (TRACE) then
               write(*,*)'utmx = ',utmx,'  utmy = ',utmy
              endif
667           continue
c
              IF (XTRA) THEN
               if(utmform.eq.1.or.utmform.eq.3)then
                WRITE (UNO,'(2f12.2,1x,i2,3x,2I4,2X,I5)')
     &                      utmx,utmy,zone,STATUS,NONSTD,SFLAG(SECT)
               else
                WRITE (UNO,'(2f12.2,1x,i2,1x,f6.4,,3x,2I4,2X,I5)')
     &              utmx,utmy,zone,sfactor,STATUS,NONSTD,SFLAG(SECT)
               endif
              ELSE
               if(utmform.eq.1.or.utmform.eq.3)then
                WRITE (UNO,'(2f12.2,1x,i2)') utmx,utmy,zone
               else
                WRITE (UNO,'(2f12.2,1x,i2,1x,f6.4)')
     &                    utmx,utmy,zone,sfactor
               endif
              ENDIF
             ENDIF
            ENDIF
C
C                 **  Go get the next input record
C
            GOTO 75
90          WRITE (*,*) ' '
            WRITE (*,*) '** Reference Conversion Completed **'
            WRITE (*,*)' '
            STOP
           ENDIF
c
c          end of batch section
C
C
```

```
C ************************  INTERACTIVE CONVERSION ********************
C
C                   **********  Legal to Geographic or UTM **********
C
C
C     ********  Set-up  *********
C
100      WRITE (*,*)' '
         WRITE (*,*)'Enter Township number: '
         READ (*,*,ERR=100) J
         IF (J.EQ.0) GOTO 140
         TSHIP = J
110      continue
          WRITE (*,*)' '
         WRITE (*,*)'Enter Range number: '
         READ (*,*,ERR=110)RANGE
         IF (RANGE.GT.25) THEN
          EW = 'W'
          write(*,*)'Range direction = W'
         ELSE
120       continue
          WRITE (*,*)' '
          WRITE (*,*)'Enter Range Direction -- E(ast) or W(est): '
          READ (*,'(A1)',ERR=120)EW
         ENDIF
C
         IF (TSHIP.LT.1 .OR. TSHIP.GT.35) THEN
           write(*,*)'Township value must be in [1,35].'
           goto 100
         ENDIF
         IF (EW.EQ.'E' .OR. EW.EQ.'e') THEN
           IF (RANGE.LT.1 .OR. RANGE.GT.25) THEN
            write(*,*)'East range value must be in [1-25].'
            goto 110
           ENDIF
           EW='E'
         ELSEIF (EW.EQ.'W' .OR. EW.EQ.'w') THEN
           IF (RANGE.LT.1 .OR. RANGE.GT.43) THEN
            write(*,*)'West range value must be in [1-43].'
            goto 110
           ENDIF
           EW='W'
         ELSE
           write(*,*)'Incorrect range direction specified (E or W).'
           goto 120
         ENDIF
C
```

```
130      continue
         WRITE (*,*)' '
         WRITE (*,*)'Enter Section number (1-36), or 0 for Township: '
         READ (*,*,ERR=130)SECT
         IF (SECT.LT.0.OR.SECT.GT.36) GOTO 50
140       continue
         WRITE(*,*)' '
         WRITE(*,*)'       OPTIONS'
         WRITE(*,*)'  -2 = Stop program'
         WRITE(*,*)'  -1 = Select a new section or reset all options'
         if(sect.ge.1.and.sect.le.36)then
          if(result.eq.2)then
           WRITE(*,*)
     &       '* Options 0-3 (decimal degrees) == Options 4-7 (DMS) *'
          else
           write(*,*)'* Options 0-3 (meters)'
          endif
         else
          if(result.eq.2)then
           WRITE(*,*)
     &    '* Options 1 or 3 (decimal degrees) == Options 5 or 7 (DMS) *'
          else
           write(*,*)'* Options 1 or 3 (meters)'
          endif
         endif
         if(sect.ge.1.and.sect.le.36)then
          WRITE(*,*)
     &    ' 0/4 = Return four corners of specified subdivision or, if'
          WRITE(*,*)
     &    '       requested, its center, a corner or a side midpoint'
c             Options 0/4 will return four corners if a point location
c             (i.e., center, corner or side mid-point) is not explicitly
c             requested.  With these options, "C" means 'CENTER.'
          WRITE(*,*)' 1/5 = Select section corner or side midpoint'
c             Options 1/5 will return four corners if a point location
c             (i.e., center, corner or side mid-point) is not explicitly
c             requested.  With these options, "C" means 'SW CORNER.'
c             Options 1/5 will NOT return a CENTER location.
          WRITE(*,*)
     &    ' 2/6 = Apply footages from section corner to point location'
          WRITE(*,*)' 3/7 = Return locations of the section corners'
          WRITE(*,*) ' '
         else
          WRITE(*,*)
     &      ' 1/5 = Select township corner or approximate side midpoint'
          WRITE(*,*)' 3/7 = Return locations of the township corners'
          WRITE(*,*) ' '
         endif
         WRITE(*,*) 'Enter Option Number (-2 to 7): '
         READ (*,*,ERR=140) OPTION
         IF(OPTION.GT.7.or.OPTION.lt.-2)GOTO 140
         if(OPTION.eq.-2)stop
```

```
141      continue
         IF (OPTION.EQ.-1) THEN
          WRITE(*,*)' '
          WRITE(*,*)'  -1 = Reset all OPTIONs'
          WRITE(*,*)'   1 = Reset section number'
          WRITE(*,*)' '
          WRITE(*,*)'Enter Option Number (-1 or 1): '
          READ (*,*,ERR=141) OPTION
          IF(OPTION.EQ.-1)GOTO 50
          IF(OPTION.EQ.1)GOTO 130
          GOTO 140
         ENDIF
         if(sect.eq.0.and.(OPTION.ne.1.and.OPTION.ne.5.
     &    and.OPTION.ne.3.and.OPTION.ne.7)) GOTO 140
C
C                **  Set subdivision and corner to blank, Set logical
C                **  variables FOURSIDE (=true when the four side lengths
C                **  OPTION is on) and FOURCORN (=true when 4 corners of
C                **  a section OPTION is on or when subdivision does not
C                **  specify a CORNR value, i.e. = blank) to false
C
         SUBD(1) = BLANK2
         CORNR = BLANK2
         FOURSIDE = .FALSE.
         FOURCORN = .FALSE.
C
C                **  Set remain to [mod(OPTION,4)] to divide options into
C                **  classes (remainder is = for 0/4, 1/5, 2/6 and 3/7)
C
         REMAIN = MOD (INT (OPTION), 4)
C
C                **  Subdivide for options 0/4 and get CORNR value
C                **  for options 0/4 and options 1/5
C
         IF (REMAIN.LT.2) THEN
          IF (REMAIN.EQ.0) THEN
C
```

```
C               **  Get subdivision values for options 0/4
C
           write(*,*)' '
                write(*,*)
     &     'Subdivisions may be specified as quarter or half divisions.'
           write(*,*)
     &    '  Quarter divisions - (NE, NW, SW, or SE) or (A, B, C, or D)'
           write(*,*)
     &           '  Half divisions    - (N, W, S, or E)'
           DONE = .FALSE.
              write(*,*)'
     &            Enter subdivisions from largest to smallest.'
           DO 200 I = 1,4
150         SUBD(I) = BLANK2
            IF (.not.DONE) THEN
             WRITE (*,*) ' '
             WRITE (*,*) 'Enter Subdivision ', I, ': '
             READ (*,'(A2)',ERR=150) SUBD(I)
             IF (SUBD(I).EQ.BLANK2) then
              DONE = .TRUE.
              goto 200
             ENDIF
             IF(SUBD(I)(1:1).EQ.' ') THEN
               SUBD(I)=SUBD(I)(2:2) // SUBD(I)(1:1)
             ENDIF
             IF (SUBD(I).EQ.'D '.OR.SUBD(I).EQ.'SE')THEN
               SUBD(I)='SE'
             ELSEIF (SUBD(I).EQ.'d '.OR.SUBD(I).EQ.'se')THEN
               SUBD(I)='SE'
             ELSEIF (SUBD(I).EQ.'A '.OR.SUBD(I).EQ.'NE')THEN
               SUBD(I)='NE'
             ELSEIF (SUBD(I).EQ.'a '.OR.SUBD(I).EQ.'ne')THEN
               SUBD(I)='NE'
             ELSEIF (SUBD(I).EQ.'B '.OR.SUBD(I).EQ.'NW')THEN
               SUBD(I)='NW'
             ELSEIF (SUBD(I).EQ.'b '.OR.SUBD(I).EQ.'nw')THEN
               SUBD(I)='NW'
             ELSEIF (SUBD(I).EQ.'C '.OR.SUBD(I).EQ.'SW')THEN
               SUBD(I)='SW'
             ELSEIF (SUBD(I).EQ.'c '.OR.SUBD(I).EQ.'sw')THEN
               SUBD(I)='SW'
             ELSEIF (SUBD(I).EQ.'N '.OR.SUBD(I).EQ.'n ')THEN
               SUBD(I)='N '
             ELSEIF (SUBD(I).EQ.'W '.OR.SUBD(I).EQ.'w ')THEN
               SUBD(I)='W'
             ELSEIF (SUBD(I).EQ.'S '.OR.SUBD(I).EQ.'s ')THEN
               SUBD(I)='S'
             ELSEIF (SUBD(I).EQ.'E '.OR.SUBD(I).EQ.'e ')THEN
               SUBD(I)='E'
             ELSE
               WRITE (*,*) 'Unrecognized subdivision ',I,' spec:'
               goto 150
             ENDIF
            ENDIF
200        CONTINUE
C
C
C       *********************************
```

```
C
C               **  Get CORNR value for options 0/4 and 1/5
C
210        continue
           WRITE(*,*)' '
           write(*,*)
     &         'All section or subdivision corners will be given unless'
           write(*,*)
     &         'you request the center, a specific corner, or the '
           write(*,*)'mid-point of a side.'
           WRITE(*,*)' '
           WRITE(*,*)'  center: (C, CC, or CR)'
           WRITE(*,*)'  corners: (NE, NW, SW, or SE)'
           WRITE(*,*)'  mid-point of sides: (N, W, S, or E)'
           WRITE(*,*)' '
           write(*,*)
     &           'Enter center, corner, or mid-point of a side:'
          endif
          if(OPTION.eq.1.or.OPTION.eq.5)then
           WRITE(*,*)' '
           write(*,*)'All section corners will be given unless'
           write(*,*)
     &         'you request a specific corner or mid-point of a side.'
           WRITE(*,*)' '
           WRITE(*,*)
     &           '  corners: (NE, NW, SW, or SE) or (A, B, C, or D)'
           WRITE(*,*)'  side mid-points: (N, W, S, or E)'
           WRITE(*,*)' '
           write(*,*)'Enter corner, or mid-point of a side: '
          endif
          READ (*,'(A2)',ERR=210) CORNR
          IF (OPTION.EQ.0 .AND. CORNR.EQ.'  ') FOURCORN = .TRUE.
          IF (OPTION.EQ.1 .AND. CORNR.EQ.'  ') FOURCORN = .TRUE.
          IF (OPTION.EQ.4 .AND. CORNR.EQ.'  ') FOURCORN = .TRUE.
          IF (OPTION.EQ.5 .AND. CORNR.EQ.'  ') FOURCORN = .TRUE.
         ELSEIF (REMAIN.EQ.2) THEN
C
```

```
C               **  Get footages for options 2/6, check for 4 side opt
C
          WRITE (*,*) ' '
          WRITE (*,*)
     &        'Positive values imply measurements to the north or east.'
          WRITE (*,*)
     &        'Negative values imply measurements to the south or west.'
          WRITE (*,*) ' '
          WRITE (*,*)
     &        'Unless specified explicitly after giving footage values,'
          WRITE (*,*)
     &    'the corner of origin is implied by the sign of the footages.'
          WRITE (*,*) '     ft-NS<0 and ft-EW<0 imply CORNER=NE'
          WRITE (*,*) '     ft-NS<0 and ft-EW>0 imply CORNER=NW'
          WRITE (*,*) '     ft-NS>0 and ft-EW>0 imply CORNER=SW'
          WRITE (*,*) '     ft-NS>0 and ft-EW<0 imply CORNER=SE'
          WRITE (*,*) ' '
          WRITE (*,*)
     &        'Enter footages = 0,0 to get section side lengths.'
          WRITE (*,*) ' '
220        WRITE (*,*) 'Enter Footage-NS, Footage-EW: '
          READ (*,*,ERR=220) FTNS, FTEW
221       continue
          WRITE (*,*) 'Corner of origin for measurements (CORNR): '
          READ (*,'(A2)',ERR=221) CORNR
          IF (ABS(FTNS).LT.1..AND.ABS(FTEW).LT.1.) FOURSIDE = .TRUE.
         ELSEIF (REMAIN.EQ.3) THEN
          FOURCORN = .TRUE.
         ENDIF
C
C               **  Call conversion routine and print out the results
C
         slack=0
         slackn=0
         slackw=0
         slacknw=0
         CALL LEO3CVT (TSHIP,RANGE,EW,SECT,OPTION,SUBD,PLON,PLAT,
     &        UNL, STATUS, SOURCE)
C
C***********
C              This is the point of RETURN from Subroutine LEO3CVT
C***********
C
         IF(SLACKN.EQ.1.OR.SLACKW.EQ.1.OR.SLACKNW.EQ.1)SLACK=1
         IF(STLINE.EQ.1)NONSTD=NONSTD+2
         IF(RIVER.EQ.1)NONSTD=NONSTD+4
         IF(RESERVE.EQ.1)NONSTD=NONSTD+8
         IF(TREATY.EQ.1)NONSTD=NONSTD+16
         IF(SLACK.EQ.1)NONSTD=NONSTD+32
         WRITE (*,*)' '
         IF (STATUS.GT.0) THEN
          stemp=sflag(sect)
          if(sect.ge.1.and.sect.le.36)write(*,
     &        '(5x,A1,i2,a4,i2,a1,2x,i2,2x,4(a2,1x),5x,a2,10x,A8,i5/)')
     &         'T',tship,'S  R',range,ew,sect,subd,cornr,'SFLAG = ',stemp
```

```
          if(sect.eq.0)then
c
c
c       **** SET CORNER FLAGS FOR TOWNSHIP
c
c
           write(*,
     &        '(5x,A1,i2,a4,i2,a1,2x,2x,2x,4(a2,1x),5x,a2,10x,A8,i5/)')
     &         'T',tship,'S  R',range,ew,subd,cornr,'TFLAG = ',stemp
           if(sflag(1).eq.0)then
            scor(1)=0
           else
            ALPHA=ABS(SFLAG(1))
            SCOR(1)=INT(ALPHA/1000)
           endif
           if(sflag(6).eq.0)then
            scor(2)=0
           else
            ALPHA=ABS(SFLAG(6))
            BETA=ALPHA-1000*INT(ALPHA/1000)
            SCOR(2)=INT(BETA/100)
           endif
           if(sflag(31).eq.0)then
            scor(3)=0
           else
            ALPHA=ABS(SFLAG(31))
            BETA=ALPHA-1000*INT(ALPHA/1000)
            GAMMA=BETA-100*INT(BETA/100)
            SCOR(3)=INT(GAMMA/10)
           endif
           write(*,*)'sflag(36) = ',sflag(36)
           if(sflag(36).eq.0)then
            scor(4)=0
           else
            ALPHA=ABS(SFLAG(36))
            BETA=ALPHA-1000*INT(ALPHA/1000)
            GAMMA=BETA-100*INT(BETA/100)
            SCOR(4)=GAMMA-10*INT(GAMMA/10)
           endif
           do 201 i=1,4
            IF(SCOR(i).GT.5)SCOR(i)=SCOR(i)-5
201         continue
          endif
          write(*,*)'corner flags are ',scor
          WRITE (*,*) '* S U C C E S S *'
          WRITE(*,*)'   STATUS = ',STATUS,'  (NONSTD = ',NONSTD, ')'
c
c
c
c
```

```
      if(status.eq.16)then
       write(*,*)'N-S footage exceeds average distance'
       write(*,*)'between north and south section corners.'
      endif
      if(status.eq.17)then
       write(*,*)'E-W footage exceeds average distance'
       write(*,*)'between east and west section corners.'
      endif
      if(status.eq.34)then
       write(*,*)
     &   'Part of the requested subdivision lies outside the state.'
       write(*,*)'Information is provided for the portion '
       write(*,*)'of the subdivision in Kansas.'
      endif
      if(status.eq.35)then
       write(*,*)
     &   'Part of the requested subdivision lies outside the state.'
       write(*,*)
     &   'The requested point in that subdivision is in Kansas.'
      endif
       if(status.eq.36)then
       write(*,*)
     &   'Part of the requested subdivision lies outside the state.'
       write(*,*)
     &    'The requested point also lies outside Kansas.'
       write(*,*)
     &   'Information is provided for the corresponding point'
       write(*,*)'in the partial subdivision.'
      endif
      if(status.ge.41.and.status.le.44)then
       write(*,*)
     &   'Footage is measured from protracted section corner.'
      endif
      if(status.ge.45.and.status.le.52)then
       write(*,*)
     &   'Footage is measured from protracted section corner, and'
       write(*,*)
     &   'footage exceeds average distance between section corners.'
      endif
c
```

```
c            Output results in geographic if RESULT = 2 was
C            selected, otherwise output in UTM coordinates
c
          if(result.eq.2)THEN
c
           IF (FOURCORN) THEN
C
C                ** Section/subarea corners NW, NE, SW, SE -- DD and DMS
C
            IF (OPTION.LT.4) THEN
             WRITE (*,'(/1X,2F12.5,10X,2F12.5/)')
     &              (LONX(I,1), LATX(I,1), I=1,4)
            ELSE
             WRITE (*,'(/1X,2(2(F5.0,F4.0,F7.3,2X),6X)/1X)')
     &              ((LONX(I,J),J=1,3),(LATX(I,J),J=1,3),I=1,4)
            ENDIF
c
           ELSEIF (FOURSIDE) THEN
C
C                ** Four side lengths in feet N, W, E, S
C
            WRITE(*,'(/11X,F8.2/2x,F8.2,9X,F8.2/11X,F8.2/)')
     &              LONX(1,1),LONX(3,1),LONX(2,1),LONX(4,1)
           ELSE
            IF (OPTION.LT.3) THEN
C
C                ** Single Point for options 0-2 in decimal degrees
C
             WRITE (*,*) ' '
             WRITE (*,'('' LON, LAT = '',2F12.5)') PLON, PLAT
            ELSE
C
C                ** Single Point for options 4-6 in DMS
C
             WRITE(*,'(/1X,F6.1,F5.1,F7.3,'' , '',F6.1,F5.1,F7.3)')
     &            (LONX(1,J), J=1,3), (LATX(1,J), J=1,3)
            ENDIF
           ENDIF
c
```

```
c            Output results in UTM coordinates
c
C          Output options for UTM coordinates:
C           0 = find X, Y, UTM zone for each location, scale factor
C           1 = find X, Y, UTM zone for each location
C           2 = find X, Y relative to selected UTM zone, scale factor
C           3 = find X,Y relative to selected UTM zone
C
          ELSE
c
           IF (FOURCORN) THEN
C
C ********************** Section or subarea four corners option *********
C
C                   **  Convert locations from decimal degrees to UTM
c
            do 668 i=1,4
             lambda=lonx(i,1)
             phi=latx(i,1)
             call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &                   utmdir,utmform)
             lonx(i,1)=utmx
             latx(i,1)=utmy
             lonx(i,2)=zone
             lonx(i,3)=utmk
668         continue
C
C            **  Output 4 section or subarea corners w/o extra values
C
            if(utmform.eq.1.or.utmform.eq.3)then
             WRITE (*,'(2(2f12.2,1x,f3.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=1,2)
             WRITE (*,'(2(2f12.2,1x,f3.0,2x))')
     &                    (LONX(I,1),LATX(I,1),lonx(i,2),I=3,4)
            else
             WRITE (*,'(2(2f12.2,1x,f3.0,1x,f6.4,2x))')
     &               (LONX(I,1),LATX(I,1),lonx(i,2),lonx(i,3),I=1,2)
             WRITE (*,'(2(2f12.2,1x,f3.0,1x,f6.4,2x))')
     &               (LONX(I,1),LATX(I,1),lonx(i,2),lonx(i,3),I=3,4)
            endif
c
          ELSEIF (FOURSIDE) THEN
C
C            **  Output 4 side lengths in meters in the order:  N,W,E,S
C
            WRITE(*,*)' '
            WRITE(*,*)'Section dimensions in meters'
C
            WRITE(*,'(/11X,F8.2/2x,F8.2,9X,F8.2/11X,F8.2/)')
     &               LONX(1,1)/fm,LONX(3,1)/fm,LONX(2,1)/fm,LONX(4,1)/fm
```

```
          ELSE
C
C
C               **  Single Point for options 0-2 in UTM coordinates
C
           lambda=plon
           phi=plat
           call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &             utmdir,utmform)
           sfactor=utmk
           plon=utmx
           plat=utmy
669        continue
           if(utmform.eq.1.or.utmform.eq.3)then
            WRITE(*,*)' '
            WRITE(*,'('' X,Y,zone = '',2f12.2,1x,i2)')utmx,utmy,zone
           else
            WRITE (*,'('' X,Y,zone,scale-factor = '',
     &           (2f12.2,1x,i2,1x,f6.4))')
     &           utmx,utmy,zone,sfactor
           endif
          ENDIF
         ENDIF
        ELSE
          stemp=sflag(sect)
          if(sect.ge.1.and.sect.le.36)write(*,
     &          '(5x,A1,i2,a4,i2,a1,2x,i2,2x,4(a2,1x),5x,a2,10x,A8,i5/)')
     &          'T',tship,'S  R',range,ew,sect,subd,
     &           cornr,'SFLAG = ',stemp
          if(sect.eq.0)write(*,
     &        '(5x,A1,i2,a4,i2,a1,2x,2x,2x,4(a2,1x),5x,a2,10x,A8,i5/)')
     &         'T',tship,'S  R',range,ew,subd,cornr,'TFLAG = ',stemp
```

```
      WRITE (*,*)' ** Error, STATUS = ', STATUS, ' **'
      IF (STATUS.EQ.-1) THEN
       WRITE (*,*)  'Township number out of RANG'
      ELSEIF (STATUS.EQ.-2) THEN
       WRITE (*,*) 'RANGE Number out of RANGE'
      ELSEIF (STATUS.EQ.-3) THEN
       WRITE (*,*) 'RANGE direction (EW) not E or W'
      ELSEIF (STATUS.EQ.-4) THEN
       WRITE (*,*) 'Section number out of RANGE'
      ELSEIF (STATUS.EQ.-5) THEN
       WRITE (*,*) 'Illegal/unrecognized subdivision spec  #1'
      ELSEIF (STATUS.EQ.-6) THEN
       WRITE (*,*) 'Illegal/unrecognized subdivision spec  #2'
      ELSEIF (STATUS.EQ.-7) THEN
       WRITE (*,*) 'Illegal/unrecognized subdivision spec  #3'
      ELSEIF (STATUS.EQ.-8) THEN
       WRITE (*,*) 'Illegal/unrecognized subdivision spec  #4'
      ELSEIF (STATUS.EQ.-9) THEN
       WRITE (*,*) 'Illegal conversion direction: SORCE value'
      ELSEIF (STATUS.EQ.-10) THEN
       WRITE (*,*) 'Invalid Longitude for LEO3/KGS'
      ELSEIF (STATUS.EQ.-11) THEN
       WRITE (*,*) 'Invalid Latitude for LEO3/KGS'
      ELSEIF (STATUS.EQ.-12) THEN
       WRITE (*,*) 'Invalid value for OPT (OPTION)'
      ELSEIF (STATUS.EQ.-13) THEN
       WRITE (*,*) 'Database (LEO3BASE) I/O error'
      ELSEIF (STATUS.EQ.-14) THEN
       WRITE (*,*) 'Thrashing error on the Township level'
      ELSEIF (STATUS.EQ.-15) THEN
       WRITE (*,*) 'Thrashing error on the Section level'
      ELSEIF (STATUS.EQ.-16) THEN
       WRITE (*,*) 'Illegal footages from explicit corner'
      ELSEIF (STATUS.EQ.-17) THEN
       WRITE (*,*) 'Error in degrees,minutes,or seconds value'
      ELSEIF (STATUS.EQ.-18) THEN
       WRITE (*,*) 'Missing SUBD value for OPT=1 or 5'
      ELSEIF (STATUS.EQ.-19) THEN
       WRITE (*,*) 'Township record has no usable sections'
      ELSEIF (STATUS.EQ.-20) THEN
       WRITE (*,*) 'Point-in-Polygon Error'
      ELSEIF (STATUS.EQ.-21) THEN
       WRITE (*,*)
     &     'At least one township corner missing from ',cornr,'side.'
      ELSEIF (STATUS.EQ.-22) THEN
       WRITE (*,*) 'Two township corners missing'
      ELSEIF (STATUS.EQ.-23) THEN
       WRITE (*,*) 'Three township corners missing'
      ELSEIF (STATUS.EQ.-24) THEN
       WRITE (*,*) 'Four township corners missing'
      ELSEIF (STATUS.EQ.-25) THEN
       WRITE (*,*) 'Illegal or unrecognized CORNER spec'
       goto 210
      ELSEIF (STATUS.EQ.-26) THEN
       WRITE (*,*) 'Option is not allowed for townships'
      ELSEIF (STATUS.EQ.-27) THEN
       WRITE (*,*) 'Illegal record # in township search'
```

```
          ELSEIF (STATUS.EQ.-28) THEN
           WRITE (*,*) 'Failed search within incomplete township'
          ELSEIF (STATUS.EQ.-31) THEN
           WRITE (*,*)
     &        'One or more section corner protracted or now in Missouri'
          ELSEIF (STATUS.EQ.-32) THEN
           WRITE (*,*)
     &     'Section is not within the Kansas PLSS or no longer in Kansas'
          ELSEIF (STATUS.EQ.-33) THEN
           WRITE (*,*)
     &              'Requested subdivision is entirely outside the state'
          ELSEIF (STATUS.EQ.-37) THEN
           WRITE (*,*)
     &              'The requested point lies outside the state.'
           write (*,*)
     &     'Only one corner of the specified subdivision lies in Kansas.'
          ELSEIF (STATUS.EQ.-51)THEN
           WRITE (*,*)
     &          'Implied corner for footage measurement is ambiguous.'
           WRITE(*,*)
     &             'Location could be due north of SE or SW corner.'
          ELSEIF(STATUS.EQ.-52)THEN
           WRITE(*,*)
     &          'Implied corner for footage measurement is ambiguous.'
           WRITE(*,*)
     &             'Location could be due south of NE or NW corner.'
          ELSEIF(STATUS.EQ.-53)THEN
           WRITE(*,*)
     &          'Implied corner for footage measurement is ambiguous.'
           WRITE(*,*)
     &             'Location could be due east of SW or NW corner.'
          ELSEIF(STATUS.EQ.-54)THEN
           WRITE(*,*)
     &          'Implied corner for footage measurement is ambiguous.'
            WRITE(*,*)
     &             'Location could be due west of NE or SE corner.'
          ELSEIF (STATUS.EQ.-99) THEN
                WRITE (*,*) 'Nonstandard section can not be processed'
           ENDIF
         ENDIF
c        DO 208 I = 1,4
c         SUBD(I) = BLANK2
208      CONTINUE
         slack=0
         WRITE(*,*) '   ** ** L ** E ** O ** 3 ** **'
         GOTO 140
C ******************************************************************
C
```

```
C ***********  CONVERSIONS  ******************************************
C       *******  from  UTM  to GEOGRAPHIC or LEGAL  *************
C       *******  from  GEOGRAPHIC  to UTM or LEGAL  *************
C
C
C ********************************************************************
C
C
       ELSEIF (SOURCE.EQ.2.or.source.eq.3) THEN
c
c
        if(result.eq.1)then
239      continue
         WRITE (*,*)' '
         WRITE(*,*)'OUTPUT OPTIONS'
         WRITE(*,*)'  -2 = Stop program'
         WRITE(*,*)'  -1 = reset all options'
         WRITE(*,*)
     &            '   0 = Subdivide up to 4 levels, give closest point'
         WRITE(*,*)'   1 = Give footages from closest surveyed corner'
         write(*,*)'   2 = Give footages from selected corner'
         WRITE (*,*) ' '
         WRITE (*,*) 'Enter Option Number (-2 to 2): '
         READ (*,*,ERR=239) OPTION2
         if (OPTION2.eq.0)then
238       continue
          write(*,*)'Give maximum levels of subdivision (0-4): '
          read (*,*,err=238) maxlvl
          if(maxlvl.lt.0.or.maxlvl.gt.4)goto 238
         endif
         if (OPTION2.eq.2)then
237        continue
           write(*,*)'Enter corner for all footages (NE,NW,SW,SE): '
           read (*,*,err=237)cornrgl
           if(cornrgl.ne.'NE'.and.cornrgl.ne.'NW'.and.cornrgl.ne.'SW'.
     &      and.cornrgl.ne.'SE'.and.cornrgl.ne.'ne'.and.cornrgl.ne.'nw'.
     &      and.cornrgl.ne.'sw'.and.cornrgl.ne.'se'.and.cornrgl.ne.'A'.
     &      and.cornrgl.ne.'a'.and.cornrgl.ne.'B'.and.cornrgl.ne.'b'.
     &      and.cornrgl.ne.'C'.and.cornrgl.ne.'c'.and.cornrgl.ne.'D'.
     &      and.cornrgl.ne.'d') goto 237
         endif
         IF (OPTION2.EQ.-1) GOTO 10
         IF (OPTION2.eq.-2) STOP
         if (OPTION.lt.-2.or.OPTION.gt.2) goto 239
        endif
c
```

```
          if(result.eq.3)then
951        continue
           write(*,*)' '
           write(*,*)'Output OPTIONs for UTM coordinates:'
           write(*,*)
     &       ' -1 = Reset source OPTION'
           write(*,*)
     &       '  0 = find X, Y, UTM zone, scale factor for each location'
           write(*,*)
     &       '  1 = find X, Y, UTM zone for each location'
           write(*,*)
     &       '  2 = find X, Y, scale factor relative to selected UTM zone'
           write(*,*)
     &       '  3 = find X,Y relative to selected UTM zone'
           write(*,*)'Select UTM output format: '
           read(*,*,err=51)utmform
           if(utmform.lt.-1.or.utmform.gt.3)goto 951
952        continue
           zone=0
           if(utmform.gt.1)then
             write(*,*)'Select the UTM zone common to the most'
             write(*,*)'  significant portion of your data:'
             write(*,*)'      zone 15 -- 90-96 deg west longitude'
             write(*,*)'      zone 14 -- 96-102 deg west longitude'
             write(*,*)'      zone 13 -- 102-108 deg west longitude'
             write(*,*)'Specify UTM zone [13 CO, 14 W KS , 15 E KS]: '
             read(*,*,err=952)zone
             if(zone.lt.13.or.zone.gt.15)goto 952
           endif
           goto 300
          endif
c
C
C
c          ******************************
C
```

```
          IF (BATCH.EQ.1) THEN
C
C
C         ************************  BATCH  **********************
C
C
C       **  Decimal degree input is required for batch OPTION
c       When source is geographic, OPTION = 0 implies decimal degrees
c
          OPTION = 0
C
C               **  Read, convert, and write for the batch OPTION
C
C
c       First set formats for legal or utm output
c       If output is geographic (from utm)
c        then, format will be decimal degrees
c
c       In batch mode, set legal output sequence format option
c
          if(result.eq.1)then
2142       CONTINUE
C
           WRITE(*,*)'OUTPUT FORMAT OPTIONS'
           WRITE(*,*)'          -- for Public Land Survey locations'
           WRITE(*,*)' -1 = Reset all OPTIONs'
           WRITE(*,*)
     &      '  1 = From largest to smallest location designation:'
           WRITE(*,*)
     &      '        TSHP, RANGE, E/W, SECT, SUBD(4) large to small,'
           WRITE(*,*)'          CORNER,FOOTAGE N/S, FOOTAGE E/W'
           WRITE(*,*)
     &      '     FORMAT(2I3,1X,A1,I3,4(1X,A2),1X,A2,2F8.0)'
C
           WRITE(*,*)'  2 = From smallest to largest designation:'
           WRITE(*,*)'       FOOTAGE N/S, FOOTAGE E/W, CORNER,'
           WRITE(*,*)
     &       '          SUBD(4) small to large, SECT, TSHP, RANGE, E/W,'
           WRITE(*,*)
     &       '     FORMAT(2F8.0,1X,A2,4(1X,A2),3I3,1X,A1)'
           WRITE(*,*)' '
           WRITE(*,*)'ENTER PREFERRED FORMAT OPTION:  '
C
           READ(*,*,ERR=2142)PLSSMAT
           IF((PLSSMAT.LT.1.AND.PLSSMAT.NE.-1).
     &         OR.PLSSMAT.GT.2)GOTO 2142
           IF(PLSSMAT.eq.-1)goto 50
240        continue
C
           WRITE (*,*) 'Enter 0=OMIT or 1=INCLUDE the Status code'
           WRITE (*,*) '  and Non-standard Section code? '
           READ (*,*,ERR=240) INXTRA
           IF (INXTRA .EQ. 1) THEN
            XTRA = .TRUE.
           ELSE
            XTRA = .FALSE.
           ENDIF
          endif
```

```
c
c       Begin conversions, first from utm to geographic
c
          RECORD = 0
250       continue
c         write(*,*)'Source = ',source
c
c          When source is UTM:
c
          if(source.eq.3)then
           read(uni,1251,end=260) utmx,utmy,zone
           if (TRACE) then
            write(*,*)'UTM values: ',utmx,utmy,zone,utmdir,utmform
           endif
1251       format(2f12.2,1x,i2)
           call utmsub(zone,lambda,phi,utmx,utmy,utmk,utmdir,utmform)
           if (TRACE) then
            write(*,*)'GEO values: ',lambda,phi
           endif
           plon=lambda
           plat=phi
           IF(result.eq.2)then
            WRITE (UNO,'(1X,F11.5,F9.5)') PLON,PLAT
            goto 250
           ENDIF
           goto 1250
```

```
      elseif(source.eq.2)then
       READ (UNI,*,END=260) PLON, PLAT
       IF(result.eq.3)then
        lambda=plon
        phi=plat
        if (TRACE) then
         write(*,*)'zone = ',zone,'  longitude = ',
     &         lambda,'  latitude = ',phi
        endif
        call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &         utmdir,utmform)
        sfactor=utmk
        plon=utmx
        plat=utmy
        if (TRACE) then
         write(*,*)'utmx = ',utmx,'  utmy = ',utmy
        endif
C
        IF (XTRA) THEN
         if(utmform.eq.1.or.utmform.eq.3)then
          WRITE (UNO,'(2f12.2,1x,i2,3x,2I4,2X,I5)')
     &           PLON,PLAT,zone,STATUS,NONSTD,SFLAG(SECT)
         else
          WRITE (UNO,'(2f12.2,1x,i2,1x,f6.4,,3x,2I4,2X,I5)')
     &           PLON,PLAT,zone,sfactor,STATUS,NONSTD,SFLAG(SECT)
         endif
        ELSE
         if(utmform.eq.1.or.utmform.eq.3)then
          WRITE (UNO,'(2f12.2,1x,i2)') PLON,PLAT,zone
         else
          WRITE (UNO,'(2f12.2,1x,i2,1x,f6.4)')
     &          PLON,PLAT,zone,sfactor
         endif
        ENDIF
        goto 250
       ENDIF
      endif
1250  continue
      RECORD = RECORD + 1
      if(record.eq.500)write(*,*)'Record = 500'
      if(record.eq.1000)write(*,*)'Record = 1000'
      if(record.eq.1500)write(*,*)'Record = 1500'
      if(record.eq.2000)write(*,*)'Record = 2000'
      if(record.eq.2500)write(*,*)'Record = 2500'
      if(record.eq.3000)write(*,*)'Record = 3000'
      if(record.eq.3500)write(*,*)'Record = 3500'
      if(record.eq.4000)write(*,*)'Record = 4000'
      if(record.eq.4500)write(*,*)'Record = 4500'
      if(record.eq.5000)write(*,*)'Record = 5000'
      IF (EKO) THEN
       WRITE(*,'('' Input record'',I4,5X,2F12.5)')
     &       RECORD,PLON,PLAT
       IF (TRACE) WRITE (UNO,'('' Record'',I4,5X,2F12.5)')
     &          RECORD, PLON, PLAT
      ENDIF
```

```
      slack=0
      slackn=0
      slackw=0
      slacknw=0
      CALL LEO3CVT (TSHIP, RANGE, EW, SECT, OPTION, SUBD,
     &           PLON, PLAT, UNL, STATUS, SOURCE)
C
      IF(SLACKN.EQ.1.OR.SLACKW.EQ.1.OR.SLACKNW.EQ.1)SLACK=1
      IF(NONSTD.EQ.1.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &           RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
       NONSTD=0
       IF(STLINE.EQ.1)NONSTD=NONSTD+2
       IF(RIVER.EQ.1)NONSTD=NONSTD+4
       IF(RESERVE.EQ.1)NONSTD=NONSTD+8
       IF(TREATY.EQ.1)NONSTD=NONSTD+16
       IF(SLACK.EQ.1)NONSTD=NONSTD+32
      ELSEIF(NONSTD.EQ.0.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &           RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
       IF(STLINE.EQ.1)NONSTD=NONSTD+2
       IF(RIVER.EQ.1)NONSTD=NONSTD+4
       IF(RESERVE.EQ.1)NONSTD=NONSTD+8
       IF(TREATY.EQ.1)NONSTD=NONSTD+16
       IF(SLACK.EQ.1)NONSTD=NONSTD+32
      ENDIF
      IF (XTRA) THEN
       if(OPTION2.eq.0)then
        if(PLSSMAT.EQ.1)THEN
         if(status.ge.0)then
          WRITE(UNO,'(2I3,1X,A1,I3,5(1X,A2),16X,2X,2I4,2X,I5)')
     &          TSHIP,RANGE,EW,SECT,SUBD,CORNR,
     &          STATUS,NONSTD,SFLAG(SECT)
         else
          write(uno,'(44x,2I4,2X,I5)')status,nonstd,sflag(sect)
         endif
        elseif(PLSSMAT.EQ.2)THEN
         sub1=SUBD(4)
         sub2=SUBD(3)
         sub3=SUBD(2)
         sub4=SUBD(1)
         SUBD(1)=sub1
         SUBD(2)=sub2
         SUBD(3)=sub3
         SUBD(4)=sub4
C
         if(status.ge.0)then
          WRITE(UNO,'(16X,5(1X,A2),3I3,1X,A1,2X,2I4,2X,I5)')CORNR,
     &          SUBD,SECT,TSHIP,RANGE,EW,STATUS,NONSTD,SFLAG(SECT)
         else
          write(uno,'(44x,2I4,2x,i5)')status,nonstd,sflag(sect)
         endif
        endif
```

```
      elseif(OPTION2.eq.1.or.OPTION2.eq.2)then
       if(PLSSMAT.EQ.1)THEN
        if(status.ge.0)then
         WRITE(UNO,'(2I3,1X,A1,I3,5(1X,A2),2f8.0,2X,2I4,2X,I5)')
&             TSHIP,RANGE,EW,SECT,SUBD,CORNR,FTNS,FTEW,
&            STATUS,NONSTD,SFLAG(SECT)
        else
         write(uno,'(44x,2I4,2X,I5)')status,nonstd,sflag(sect)
        endif
       elseif(PLSSMAT.EQ.2)THEN
        sub1=SUBD(4)
        sub2=SUBD(3)
        sub3=SUBD(2)
        sub4=SUBD(1)
        SUBD(1)=sub1
        SUBD(2)=sub2
        SUBD(3)=sub3
        SUBD(4)=sub4
        if(status.ge.0)then
         WRITE(UNO,'(2F8.0,5(1X,A2),3I3,1X,A1,2X,2I4,2X,I5)')
&             FTNS,FTEW,CORNR,SUBD,SECT,TSHIP,RANGE,EW,
&            STATUS,NONSTD,SFLAG(SECT)
        else
         write(uno,'(44x,2I4,2X,I5)')status,nonstd,sflag(sect)
        endif
       endif
      endif
```

```
      ELSE
       if(OPTION2.eq.0)then
        if(PLSSMAT.EQ.1)THEN
         if(status.ge.0)then
          WRITE (UNO,'(2I3,1X,A1,I3,5(1X,A2))') TSHIP,
     &          RANGE, EW, SECT, SUBD, CORNR
         else
          write(uno,'(44x,i4)')status
         endif
        elseif(PLSSMAT.EQ.2)THEN
         sub1=SUBD(4)
         sub2=SUBD(3)
         sub3=SUBD(2)
         sub4=SUBD(1)
         SUBD(1)=sub1
         SUBD(2)=sub2
         SUBD(3)=sub3
         SUBD(4)=sub4
         if(status.ge.0)then
          WRITE (UNO,'(16X,5(1X,A2),3I3,1X,A1)') CORNR, SUBD,
     &          SECT, TSHIP, RANGE, EW
         else
          write(uno,'(44x,i4)')status
         endif
        endif
       elseif(OPTION2.eq.1.or.OPTION2.eq.2)then
        if(PLSSMAT.EQ.1)THEN
         if(status.ge.0)then
          WRITE(UNO,'(2I3,1X,A1,I3,5(1X,A2),2f8.0)') TSHIP,
     &          RANGE, EW, SECT, SUBD, CORNR, FTNS, FTEW
         else
          write(uno,'(44x,i4)')status
         endif
        elseif(PLSSMAT.EQ.2)THEN
         sub1=SUBD(4)
         sub2=SUBD(3)
         sub3=SUBD(2)
         sub4=SUBD(1)
         SUBD(1)=sub1
         SUBD(2)=sub2
         SUBD(3)=sub3
         SUBD(4)=sub4
         if(status.ge.0)then
          WRITE(UNO,'(2f8.0,5(1X,A2),3I3,1X,A1)') FTNS, FTEW,
     &          CORNR, SUBD, SECT, TSHIP, RANGE, EW
         else
          write(uno,'(44x,i4)')status
         endif
        endif
       endif
      ENDIF
      GOTO 250
260   continue
      WRITE (*,*) '** Reference Conversion Completed **'
      WRITE (*,*)' '
      STOP
     ENDIF
C
```

```
c        BATCH.NE.1  --  therefore, conversion is interactive
C
C
C *****  INTERACTIVE CONVERSIONS FROM UTM OR GEOGRAPHIC  **************
C
C       *******  from  UTM  to GEOGRAPHIC or LEGAL  *************
C       *******  from  GEOGRAPHIC  to UTM or LEGAL  *************
C
C
298      continue
c
c          When source is UTM:
c
         if(source.eq.3)then
299       continue
          WRITE (*,*)' '
          WRITE (*,*)'UTM COORDINATE INPUT FORMAT'
          WRITE (*,*)'  -2 = Stop program'
          WRITE (*,*)'  -1 = Reset all options'
          WRITE (*,*)
     &      '   1 = x coordinate (meters), y coordinate (meters), zone'
          write(*,*)'           zone 15 -- 90-96 deg west longitude'
          write(*,*)'           zone 14 -- 96-102 deg west longitude'
          write(*,*)'           zone 13 -- 102-108 deg west longitude'
          WRITE(*,*)'Enter format option: '
          READ (*,*,ERR=299)OPTION
          if (OPTION.eq.-2) stop
          IF (OPTION.eq.-1) GOTO 10
          IF (OPTION.GT.1.or.OPTION.lt.-2.or.OPTION.EQ.0) GOTO 299
          IF (OPTION.EQ.1) THEN
           WRITE (*,*)'Enter x coordinate, y coordinate, utm zone'
           read(*,*,ERR=299) utmx,utmy,zone
          ENDIF
          call utmsub(zone,lambda,phi,utmx,utmy,utmk,utmdir,utmform)
          plon=lambda
          plat=phi
C
          WRITE (*,*) ' '
          WRITE (*,'('' LON, LAT = '',2F12.5)') PLON, PLAT
C
c        Finished if desired result is geographic
c
          IF(result.eq.2) goto 299
c
```

```
c       When desired result is PLSS reset OPTION for decimal degrees
c
          OPTION=0
          goto 351
c
c         Begin result = PLSS
c
         elseif(source.eq.2)then
300       continue
          WRITE (*,*)' '
          WRITE (*,*)'GEOGRAPHIC COORDINATE INPUT FORMAT'
          WRITE (*,*)'  -2 = Stop program'
          WRITE (*,*)'  -1 = Reset all options'
          WRITE (*,*)'   0 = Decimal Degrees'
          WRITE (*,*)'   1 = Deg-Min-Sec'
          WRITE(*,*)'Enter format option: '
          READ (*,*,ERR=300)OPTION
          if (OPTION.eq.-2) stop
          IF (OPTION.eq.-1) GOTO 10
          IF (OPTION.GT.1.or.OPTION.lt.-2) GOTO 300
          IF (OPTION.EQ.1) THEN
330        WRITE (*,*)'Enter Longitude (D,M,S): '
           READ (*,*,ERR=330) (LONX(1,J), J = 1,3)
340        WRITE (*,*)'Enter Latitude (D,M,S): '
           READ (*,*,ERR=340) (LATX(1,J), J = 1,3)
           if(result.eq.3)then
            plon=lonx(1,1)+lonx(1,2)/60.0+lonx(1,3)/3600.0
            plat=latx(1,1)+latx(1,2)/60.0+latx(1,3)/3600.0
           endif
          ELSE
C          OPTION = 0
350        WRITE (*,*)'Enter Longitude, Latitude in decimal degrees: '
           READ (*,*,ERR=350)PLON,PLAT
          ENDIF
         endif
C
c       When desired result is UTM
c
C
         if(result.eq.3)then
          lambda=plon
          phi=plat
          call utmsub(zone,lambda,phi,utmx,utmy,utmk,
     &         utmdir,utmform)
          sfactor=utmk
C
C               **  Single Point in UTM coordinates
C
          if(utmform.eq.1.or.utmform.eq.3)then
           WRITE(*,'('' X,Y,zone = '',2f12.2,1x,i2)')utmx,utmy,zone
          else
           WRITE (*,'('' X,Y,zone,scale-factor = '',
     &           2f12.2,1x,i2,1x,f6.4)')
     &           utmx,utmy,zone,sfactor
          endif
         endif
c
c
```

```
c       When desired result is PLSS
C
351      continue
         slack=0
         slackn=0
         slackw=0
         slacknw=0
         CALL LEO3CVT (TSHIP,RANGE,EW,SECT,OPTION,SUBD,PLON,PLAT,
     &         UNL, STATUS, SOURCE)
C
         IF(SLACKN.EQ.1.OR.SLACKW.EQ.1.OR.SLACKNW.EQ.1)SLACK=1
         IF(NONSTD.EQ.1.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &      RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
          NONSTD=0
          IF(STLINE.EQ.1)NONSTD=NONSTD+2
          IF(RIVER.EQ.1)NONSTD=NONSTD+4
          IF(RESERVE.EQ.1)NONSTD=NONSTD+8
          IF(TREATY.EQ.1)NONSTD=NONSTD+16
          IF(SLACK.EQ.1)NONSTD=NONSTD+32
         ELSEIF(NONSTD.EQ.0.AND.(STLINE.EQ.1.OR.RIVER.EQ.1.OR.
     &      RESERVE.EQ.1.OR.TREATY.EQ.1.OR.SLACK.EQ.1))THEN
          IF(STLINE.EQ.1)NONSTD=NONSTD+2
          IF(RIVER.EQ.1)NONSTD=NONSTD+4
          IF(RESERVE.EQ.1)NONSTD=NONSTD+8
          IF(TREATY.EQ.1)NONSTD=NONSTD+16
          IF(SLACK.EQ.1)NONSTD=NONSTD+32
         ENDIF
```

```
WRITE (*,*)' '
WRITE (*,*)'   STATUS = ',STATUS,'   (NONSTD = ', NONSTD, ')'
WRITE (*,*)'   SECTION FLAG = ',sflag(sect)
WRITE (*,*)' '
if(status.ge.0)then
 WRITE (*,*)'Township  =  ', TSHIP, BLANK2,'S'
 WRITE (*,*)'Range     =  ', RANGE, BLANK2, EW
 WRITE (*,*)'Section   =  ',SECT
 if(OPTION2.eq.0)then
  WRITE (*,*)' '
  WRITE (*,*)'Subdivisions are (largest to smallest):  '
  WRITE (*,*)'     ',(SUBD(I), BLANK2, I=1,4)
  WRITE (*,*)' '
  WRITE (*,*)'Closest corner, mid-side or center = ', CORNR
 endif
 if(OPTION2.eq.1)then
  write (*,*)' '
  if(status.eq.1)then
   write (*,*)'Closest surveyed section corner = ',cornr
  elseif(status.eq.10)then
   write (*,*)'Closest corner (protracted) = ',cornr
  endif
  write (*,*)'N-S footage from ',cornr,' corner = ',ftns
  write (*,*)'E-W footage from ',cornr,' corner = ',ftew
 endif
 if(OPTION2.eq.2)then
  write (*,*)' '
  if(status.eq.1)then
   write (*,*)'Selected section corner = ',cornr
  elseif(status.eq.10)then
  write (*,*)'Selected corner (protracted) = ',cornr
  endif
  write (*,*)'N-S footage from ',cornr,' corner = ',ftns
  write (*,*)'E-W footage from ',cornr,' corner = ',ftew
 endif
 WRITE (*,*)' '
```

```
      else
       IF (STATUS.EQ.-10) THEN
        WRITE (*,*) 'Invalid Longitude for LEO3/KGS'
       ELSEIF (STATUS.EQ.-11) THEN
        WRITE (*,*) 'Invalid Latitude for LEO3/KGS'
       ELSEIF (STATUS.EQ.-12) THEN
        WRITE (*,*) 'Invalid value for OPT (OPTION)'
       ELSEIF (STATUS.EQ.-13) THEN
        WRITE (*,*) 'Database (LEO3BASE) I/O error'
       ELSEIF (STATUS.EQ.-14) THEN
        WRITE (*,*) 'Thrashing error on the Township level'
       ELSEIF (STATUS.EQ.-15) THEN
        WRITE (*,*) 'Thrashing error on the Section level'
       ELSEIF (STATUS.EQ.-19) THEN
        WRITE (*,*) 'Township record has no usable sections'
       ELSEIF (STATUS.EQ.-20) THEN
        WRITE (*,*) 'Point-in-Polygon Error'
       ELSEIF (STATUS.EQ.-21) THEN
        WRITE (*,*)
&          'At least one township corner missing from ',cornr,'side.'
       ELSEIF (STATUS.EQ.-22) THEN
        WRITE (*,*) 'Two township corners missing'
       ELSEIF (STATUS.EQ.-23) THEN
        WRITE (*,*) 'Three township corners missing'
       ELSEIF (STATUS.EQ.-24) THEN
        WRITE (*,*) 'Four township corners missing'
       ELSEIF (STATUS.EQ.-27) THEN
        WRITE (*,*) 'Illegal record # in township search'
       ELSEIF (STATUS.EQ.-28) THEN
        WRITE (*,*) 'Failed search within incomplete township'
       ELSEIF (STATUS.EQ.-31) THEN
        WRITE (*,*)
&          'One or more section corner protracted or now in Missouri'
       ELSEIF (STATUS.EQ.-37) THEN
        WRITE (*,*)
&             'The requested point lies outside the state.'
        write (*,*)
&       'Only one corner of the specified subdivision lies in Kansas.'
       ELSEIF (STATUS.EQ.-99) THEN
             WRITE (*,*) 'Nonstandard section can not be processed'
       ENDIF
      endif
      GOTO 298
     ELSEIF(SOURCE.EQ.0)THEN
      STOP
     ENDIF
     GOTO 10
     END
```

```
C (modified 1 June 1996 by drc and rjs)
C
C* * * * * * * * * * * * * *    L E O 3 C V T    *
C
C       Conversion between Legal and Geographic reference in Kansas
C
        SUBROUTINE LEO3CVT (TSHP,RANG,EW,SECT,OPT,SUBD,PLON,PLAT,
     &   LUN,STAT,SORCE)
C
C
C*********  DECLARATIONS **
C
C
        IMPLICIT NONE
C
C***********  CHARACTER **
C
        CHARACTER EW*1,SUBD*2(4),CORNR*2,PMIN*2,CMIN*2,BLANK2*2,
     &   cornrgl*2
C
C***********  DOUBLE PRECISION **
C
        DOUBLE PRECISION SOUTH,NORTH,CLON,EAST,WEST,XB,YB,BASE,
     &   SCALE,CORN(9),DBLON,DBLAT,XDBL,YDBL
C
C***********  INTEGER **
C
        INTEGER*2 TSHP,RANG,SECT,STAT,SORCE,LUN,TREC,I,J,SROW,
     &   SCOL,N,S,W,E,VZT(2380),VZTD(2380),RTOP,LEVEL,TROW,TCOL,
     &   SEARCH(6,6),OUTFLAG,OPT,TFLAG,SFLAG(36),LEGAL,GEOGR,ORDER,
     &   NUM,OFF,NSUB,NONSTD,SCOR(4),SURV,maxlvl,OPTION2,
     &   STLINE,RIVER,RESERVE,TREATY,SLACKN,SLACKW,SLACKNW,
     &   delrow,hshur40,hshut40,hshur1c,hshut1c
C
C*************  LOGICAL **
C
        LOGICAL TRCE/.FALSE./,CENTER/.FALSE./,FOURCRNR/.FALSE./,trace
C
C***********  REAL **
C
        REAL PLON,PLAT,LON(7,7),LAT(7,7),LONNE,LATNE,LONNW,LATNW,
     &   LONSW,LATSW,LONSE,LATSE,CTRLON,CTRLAT,DLON,DLAT,XLAT,XLON,
     &   LATOMN,LATOMX,LATIMN,LATIMX,LONOMN,LONOMX,LONIMN,LONIMX,
     &   WSTLON(7),DELON(7),DELAT,BASLAT,FTNS,FTEW,LONX,LATX,
     &   LONDEG,LONMIN,LONSEC,LATDEG,LATMIN,LATSEC,
     &   XSW,YSW,XSE,YSE,XNE,YNE,XNW,YNW,LENGNS,LENGEW,
     &   LONINT,LATINT,LONW,LATW,LONE,LATE,LONS,LATS,LONN,LATN,
     &   LONOFF,LATOFF,ALPHA,BETA,GAMMA,OMEGA,
     &   DMIN,DCMIN,DNE,DN,DNW,DW,DSW,DS,DSE,DE,DCTR,
     &   XN,XW,XS,XE,XCTR,YN,YW,YS,YE,YCTR,XP,YP,
     &   AVLATN,AVLATS,DLATSEC,DLATP,DLONPFN,DLONPFS,RATDDLL,
     &   lon40,lat40,lon100,lat100
C
```

```
C******  COMMON **
C
        common/leotest/trace
        COMMON/LEOXTR/LONX(4,3),LATX(4,3),FTNS,FTEW,NONSTD,CORNR,
     &          maxlvl,OPTION2,cornrgl
        COMMON/LEOMAP/SOUTH,NORTH,CLON,EAST,WEST,XB,YB,BASE,SCALE
        COMMON/LEOFLG/TFLAG,SFLAG,SCOR
        COMMON/LEOTYPE/STLINE,RIVER,RESERVE,TREATY,SLACKN,SLACKW,SLACKNW
        common/LEOUPS/LONNE,LATNE,LONNW,LATNW,LONSW,LATSW,
     &          LONSE,LATSE,CTRLON,CTRLAT,level,lon,lat
C
C******  DATA **
C
        DATA BASLAT/40.0021/,DELAT/0.0145378/,VZT/2380*0/,VZTD/2380*0/,
     &   WSTLON/-102.19857,-102.1787,-102.14777,-102.12151,-102.09101,
     &   -102.07046,-102.0443/,
     &   DELON/.0186716,.0186243,.0185299,.0184235,.0183046,.0182329,
     &   .0182083/, SCALE/12.D0/
        BLANK2='   '
c
C       Along the state border, a fudge factor of 40 feet may be used
C       to avoid discarding results due to digitizing error.
c
        lon40=0.00014
        lat40=0.00011
        hshur40=0
        hshut40=0
        lon100=0.000343
        lat100=0.00028
        hshur1c=0
        hshut1c=0
c
        if(trace)trce=.true.
C
C
C  *****     *****     *****     *****     *****     *****     *****
C
```

```
C                    Variables:
C                    ----------
C
C       TSHP = Township number, 1-35
C       RANG = RANG number, 1-25 if EW='E', 1-43 if EW='W'
C       EW = RANG direction from central meridian 'E' or 'e'= East,
C            W' or 'w'= West
C       SECT = Section number, 1-36, =0 to specify the township
C              as the area for corner/side midpoint selection or as
C              the four corner locations for the 4-corner option
C       OPT=Specifies method of locating point in the target area
C              within the section (see Subdivision)
C
C             When SORCE=1, the option implied by the OPT value is:
C
C               0=select the center of the area defined by the four
C                    corners of the smallest area specified by SUBD
C                    (may use CORNR to select closest significant point)
C               1=select the corner/side midpoint specified in SUBD
C                    of a selected section (SECT) or township (SECT=0)
C               2=apply given footages from implied section corner
C                    (or return section side lengths when footages=0)
C               3=return the locations of the four corners of the
C                    specified section or the township (SECT=0)
C               4=select the center (as in 0), but in degrees,
C                    minutes,and seconds of lon/lat,the DMS option
C                    (may use CORNR to select closest significant point)
C               5=(as in 1) select the specified corner or side
C                    midpoint of a section, but return in DMS form
C               6=apply footages from the implied section or township
C                    corner, but return in DMS form in LONX (1,1-3)
C                    and LATX (1,1-3)
C               7=return the section or township four corners,
C                    but in the DMS form
C
C       NOTE: For OPT=0-2, the geographic location is returned via PLON
C             and PLAT with a positive longitude. All other options use
C             LONX and LATX to return 1 or 4 values, (all in DMS except
C             the basic 4-corners OPTION=3), with positive longitude(s)
C
C       NOTE: When SORCE=2,the OPTION implied by the OPT value is
C
C               0=Input is in decimal degrees (in PLON < 0, PLAT)
C               1=Input is in the DMS form (in LONX > 0, LATX)
C
```

```
C       SUBD=specify 1-4 subdivisions of the section,each applied to
C       the result of the previous ones.  Subdivisions are defined as
C       selection of a half or quarter of the area being divided,and
C       are specified by letters representing points of the compass:
C
C       1/4 Area....    NE or Ab....Northeast  (b=blank)
C                  NW or Bb....Northwest
C                  SW or Cb....Southwest
C                  SE or Db....Southeast
C                        bb....Unused
C       1/2 Area..... Nb or N2......North
C                  Sb or S2....South
C                  Eb or E2....East
C                  Wb or W2....West
C                        bb....Unused
C
C        SUBD(1)=The largest (1/n section) subdivision,and the one
C                  that is applied first (to the whole section).
C        SUBD(2)=The second largest (1/n-1/n section) subdivision.
C        SUBD(3)=The second smallest (1/n-1/n-1/n section) subdivision
C        SUBD(4)=The smallest (by area 1/n-1/n-1/n-1/n section) and the
C               last subdivision to be applied to the result of the
C               preceeding three subdivisions in SUBD(1-3)
C               (where each 1/n represents either 1/2 or 1/4)
C
C       PLON = Longitude of Point for or from conversion (positive)
C       PLAT = Latitude of Point
C       LUN = Unit number of (previously opened) LEO3BASE
```

```
C       STAT = code indicating success or failure of the requested
C              operation.  The codes are:
C
C                1=SUCCESS
C               10=Warning: Footage is measured from a protracted
C                  corner.  No PLSS corners are available for this
C                  section if OPTION searched for closest corner.
C               16=Warning: N-S footage exceeds average distance
C                  between north and south section corners.
C               17=Warning: E-W footage exceeds average distance
C                  between east and west section corners.
C               34=Part of the requested subdivision lies outside the
C                  state.  Information is provided for the portion of
C                  the subdivision in Kansas.
C               35=Part of the requested subdivision lies outside the
C                  state. The requested point in that subdivision is
C                  in Kansas.
C               36=Part of the requested subdivision lies outside the
C                  state. The requested point in that subdivision also
C                  lies outside Kansas. Information is provided for
C                  the corresponding point in the partial subdivision.
C               41=Footage is measured from protracted (NE)
C                 section corner.
C               42=Footage is measured from protracted (NW)
C                 section corner.
C               43=Footage is measured from protracted (SW)
C                 section corner.
C               44=Footage is measured from protracted (SE)
C                 section corner.
C               45=Combination of STAT=16 and STAT=41
C               46=Combination of STAT=17 and STAT=41
C               47=Combination of STAT=16 and STAT=42
C               48=Combination of STAT=17 and STAT=42
C               49=Combination of STAT=16 and STAT=43
C               50=Combination of STAT=17 and STAT=43
C               51=Combination of STAT=16 and STAT=44
C               52=Combination of STAT=17 and STAT=44
```

```
C                -1=Township number not permitted
C                -2=Range number not permitted
C                -3=RANG direction (EW) not 'E' or 'W'
C                -4=Section number not permitted
C                -5=Illegal/unrecognized subdivision spec  #1
C                -6=    "        "             "       "    #2
C                -7=    "        "             "       "    #3
C                -8=    "        "             "       "    #4
C                -9=Illegal conversion direction (SORCE) value
C               -10=Invalid Longitude for LEO/KGS
C               -11=Invalid Latitude for LEO/KGS
C               -12=Invalid value for OPT
C               -13=Database (LEOBASE) I/O error
C               -14=Thrashing error on the Township level
C               -15=Thrashing error on the Section level
C               -16=Illegal footages from explicit corner
C               -17=Error in degrees,minutes,or seconds value
C               -18=Missing SUBD value for OPT=1 or 5
C               -19=Township record has no usable sections
C               -20=Point-in-Polygon Error
C               -21=One township corner missing from the Kansas PLSS
C               -22=Two township corners missing from the Kansas PLSS
C               -23=Three township corners missing from the Kansas PLSS
C               -24=All four township corners missing from the Kansas
C                   PLSS
C               -25=Illegal or unrecognized CORNER specification
C               -26=Option is not allowed for townships
C               -27=Illegal record # in township search
C               -28=Failed search within incomplete township
C               -31=One or more section corner protracted or now in
C                   Missouri
C               -32=Section is not within the Kansas PLSS or no longer
C                   in Kansas
C               -33=Requested subdivision is entirely outside the state
C               -37=The requested point lies outside the state.
C                   Only one corner of the specified subdivision lies
C                   in Kansas.
C               -51=Implied corner of origin for footage measurement is
C                   ambiguous.  Location could be due north of SE or SW
C                   corner.
C               -52=Implied corner of origin for footage measurement is
C                   ambiguous.  Location could be due south of NE or NW
C                   corner.
C               -53=Implied corner of origin for footage measurement is
C                   ambiguous.  Location could be due east of SW or NW
C                   corner.
C               -54=Implied corner of origin for footage measurement is
C                   ambiguous.  Location could be due west of NE or SE
C                   corner.
C               -99=Special section requires non-standard subdivision
C
```

```
C       SORCE = SORCE, indicates direction of the conversion:
C                 1=Legal, Convert From Legal to Geographic
C                 2=Geographic, Convert From Geographic to Legal
c                 3=UTM, Universal Transverse Mercator (x,y) in meters
C
C       LONX = has longitudes in degrees, minutes, seconds for the
C       LATX = DMS OPTION, if SORCE=2, the geographic location is
C              stored in LONX and  LATX with degrees of longitude in
C              LONX(1,1) and minutes and seconds in LONX(1,2-3).  The
C              corresponding degrees,minutes,seconds of latitude are
C              stored in LATX(1,1-3).  When SORCE=1 (Legal),the
C              converted output location is stored in the same manner,
C              as well as in PLON and PLAT.  For the 4 corners OPTION,
C              the 4 corner locations are stored similarly with the NW
C              corner in LON/LATX(1,1-3), the NE corner in (2,*)
C              the SW in (3,*) and the SE in (4,*).
C       FTNS = Footage applied to the north if > 0, south (if < 0)
C       FTEW = Footage applied to the east if > 0, west if < 0
C         NOTE: footages are applied from the section or township
C               corner implied by the signs of the footage values,
C               unless a corner is identified explicitly by CORNR.
C                   Implicit:  FTNS<0 and FTEW<0 imply CORNR=NE
C                        FTNS<0 and FTEW>0 imply CORNR=NW
C                        FTNS>0 and FTEW>0 imply CORNR=SW
C                        FTNS>0 and FTEW<0 imply CORNR=SE
C                   Explicit:  If the section is not laid out on
C                        true NS-EW lines, then (as an example)
C                        some locations in the section may lie
C                        to the north and east of the SE corner.
C                        To calculate the geographic coordinates
C                        of such a point, set:
C                           FTNS>0 and FTEW>0, and
C                           CORNR=SE (to over-ride implicit SW)
C         NOTE: when both footages are 0,the lengths of the section
C               sides (in feet) are returned in LONX (1-4,1)
C
```

```
C       NONSTD = 2, if STLINE = 1
C              = 4, if RIVER = 1
C              = 6, if STLINE = 1, and RIVER = 1
C              = 8, if RESERVE = 1
C              = 10, if STLINE = 1, and RESERVE = 1
C              = 12, if RIVER = 1, and RESERVE = 1
C              = 14, if STLINE = 1, and RIVER = 1, and RESERVE = 1
C              = 16, if TREATY = 1
C              = 18, if STLINE = 1, and TREATY = 1
C              = 20, if RIVER = 1, and TREATY = 1
C              = 22, if STLINE = 1, and RIVER = 1, and TREATY = 1
C              = 24, if RESERVE = 1, and TREATY = 1
C              = 26, if STLINE = 1, and RESERVE = 1, and TREATY = 1
C              = 28, if RIVER = 1, and RESERVE = 1, and TREATY = 1
C              = 30, if STLINE = 1, and RIVER = 1, and RESERVE = 1,
C    &              and TREATY = 1
C              = 32, if SLACKN = 1, or SLACKW = 1, or SLACKNW =1
C              = 34, if STLINE = 1 & SLACK
C              = 36, if RIVER = 1 & SLACK
C              = 38, if STLINE = 1, and RIVER = 1 & SLACK
C              = 40, if RESERVE = 1 & SLACK
C              = 42, if STLINE = 1, and RESERVE = 1 & SLACK
C              = 44, if RIVER = 1, and RESERVE = 1 & SLACK
C              = 46, if STLINE=1, and RIVER=1, and RESERVE=1 & SLACK
C              = 48, if TREATY = 1 & SLACK
C              = 50, if STLINE = 1, and TREATY = 1 & SLACK
C              = 52, if RIVER = 1, and TREATY = 1 & SLACK
C              = 54, if STLINE=1, and RIVER=1, and TREATY=1 & SLACK
C              = 56, if RESERVE = 1, and TREATY = 1 & SLACK
C              = 58, if STLINE=1, and RESERVE=1, and TREATY=1 & SLACK
C              = 60, if RIVER=1, and RESERVE=1, and TREATY=1 & SLACK
C              = 62, if STLINE = 1, and RIVER = 1, and RESERVE = 1,
C    &              and TREATY = 1 & SLACK
C             = NONSTD+100, if one or more corners of the section
C             must be redefined from the values given in LEO3BASE.
C             = NONSTD+200, if the section involved in either
C             conversion is in some way irregular, requiring that
C             subdivision of the section be performed in a non-std
C             manner (i.e. other than the standard four equal parts
C             subd.)
C
C             = 0,if the section is subdivided in the standard fashion
C
C       CORNR = If SOURC=1 (Legal), CORNR specifies, in the same form
C             as the SUBD values, one of nine points in the target
C             area defined by the legal description,including the
C             subdivisions in SUBD.  The nine points are the four
C             corners,the side midpoints (S,N...) and the center.
C             When SORCE=2 (Geographic), CORNR is set to indicate
C             the closest one of the nine points within the fourth
C             subdivision to the point described in PLON,PLAT
C
```

```
C       TFLAG = township flag, = 36 if all 36 sections have 4 defined
C          corners, = 0 if no sections have 4 defined corners, or = n,
C          0< n <36, n = number of sections with 4 defined corners
C
C
C       SFLAG = Section flag
C            For sections with 4 defined corners in LEO3BASE which are
C            sufficient for subdivided by standard procedures:
C              SFLAG = four digit code (>0) with each digit having
C              a value of 1, 2, 3, or 4.  The digits reflect the
C              source status of each of the four corners as follows:
C                 1 = corner surveyed in PLSS, with location in KS.
C                 2 = corner not surveyed in Kansas PLSS. LEO3BASE
C                     uses protracted location inside Kansas borders.
C                 3 = corner not surveyed in Kansas PLSS. LEO3BASE
C                     uses protracted location outside Kansas borders.
C                 4 = corner surveyed in Kansas PLSS, with location
C                     now in Missouri due to shift in state boundary
C                     associated with shift in location of Missouri
C                     River.
C              The digits refer, in order (left to right), to the NE,
C              NW, SW, and SE corner locations (similar to the USGS
C              use of 'a-b-c-d' with 'NE-NW-SW-SE.'  Therefore, SFLAG
C              for sections which LEO3 will subdivide by standerd
C              procedures will have non-sero digits with values from
C              1 to 4 and at least one digit with a value of 1 or 2.
C              For example:
C              Section flag   Meaning
C                  1111 all corners surveyed and in Kansas
C                  1311 the NW corner location is protracted
C                         into Missouri.
C                  1121 the SW corner location is protracted
C                         into excluded areas in Kansas.
C                  4111 the NE corner is a surveyed location
C                         of the Kansas Public Land Survey
C                         System which is now in Missouri.
C
C            For partial sections falling along the Missouri, Oklahoma
C            or Missouri borders (other than the Missouri River):
C              SFLAG = 5000 + normal code
C              On the Missouri line, 3113 becomes 8113.
C              The section at the SE corner of Kansas has SFLAG=8133.
C              On the Oklahoma line, 1133 becomes 6133.
C              The section at the SW corner of Kansas has SFLAG=6333.
C              On the Colorado line, 1331 becomes 6331.
C
C            For partial sections resulting from areas excluded from the
C            PLSS by water (major rivers):
C              SFLAG = 500 + normal code
C
C            For partial sections resulting from areas excluded from the
C            PLSS by milirary reserves (e.g., Ft. Riley and Ft.
C            Leavenworth) or prior land grants (e.g., the Half-Breed
C            Lands north of the Kansas River between Topeka and
C            Lawrence, or the Republican River Bridge Company Lands on
C            the south side of the Ft. Riley Reserve):
C              SFLAG = 50 + normal code
C
```

```
C            For sections in which the PLSS was completed in two stages
C            as the result of lands originally excluded from the PLSS
C            (primarily as Indian lands with a few land grants):
C              SFLAG = 5 + normal code
C                 In most cases these sections are complete
C                 (i.e., they are approximately 640 acres - one
C                 square mile).  Due to the separate surveys on
C                 either side of old Indian boundaries, these
C                 sections have a mixture of standard and
C                 nonstandard subdivisions and may require care
C                 in the interpretation of legal descriptions
C                 of locations within the sections.  Footage
C                 measurements would generally be preferable.
C
C            For sections which do not exist because the area where they
C            would be located falls completely within lands excluded
C            from the Kansas Public Land Survey System):
C              SFLAG = 0, or -n as defined below (when some form of
C                 processing or comment is deemed appropriate)
C
C            For sections which require some nonstandard treatment for
C            accurate subdivision:
C              SFLAG = -n, where n is the normal code for SFLAG.
C                 When SFLAG<0 the system calls FEDX, a subroutine
C                 for special handling.  Within FEDX, the process
C                 is directed to a statement number which matches
C                 the township record number in LEO3BASE of the
C                 township which contains that section.  The
C                 township record number is based on Township,
C                 Range, and Range direction (E or W) as follows:
C                 If range direction is east:
C                       TRECB = (TSHIP - 1) * 68  +  RANGE + 43
C                 If range direction is west:
C                       TRECB = (TSHIP - 1) * 68  +  44 - RANGE
C                 Routines for any sections which require special
C                 handling within a township will be found in FEDX
C                 after the statement number matching that
C                 township's record number in LEO3BASE.
C
C
C          SCOR = corner flag
C                 If the absolute value of SFLAG = abcd (non-zero), then:
C                    SCOR(1) = a, or SCOR(1) = a-5 if a>5
C                    SCOR(2) = b, or SCOR(2) = b-5 if b>5
C                    SCOR(3) = c, or SCOR(3) = c-5 if c>5
C                    SCOR(4) = d, or SCOR(4) = d-5 if d>5
C                 The value of SCOR reflects the source status of each
C                 section corner as described for SFLAG.  This source
C                 status is based on the corner's representation on USGS
C                 7.5' quadrangle maps or, where that is unclear, on
C                 information obtained from county surveyor's or the
C                 original plats of the PLSS maintained by the Kansas
C                 Historical Society in Topeka, Kansas.
C
```

```
C       XC,YC = projected locations of the 49 corners of a township
C          in the same order as the longitudes and latitudes in LON,LAT
C    (T/S)ROW = Row number of the Township or Section within a Township
C          based on an artificial grid.  For Townships the grid is 35
C          rows numbered from the North and 68 columns numbered from
C          the west.  For Sections the grid is 7 by 7 representing the
C          49 corner points, Row 1 at the North, Column 1 at the West
C    (T/S)COL = associated column number of the township or section
C       LONNE = Longitude of NE corner of section or subdivision
C       LATNE = Latitude of same
C       LONSE = Longitude of SE corner of same
C       LATSE = Latitude of same
C       LONSW = Longitude of SW corner
C       LATSW = Latitude of same
C       LONNW = Longitude of NW corner
C       LATNW = Latitude of same
C       OUTFLAG = Code like a clipping code indicates if a point is in
C          or out and where if out of a rectangle
C       LONOFF = Offset from section corner, in longitude, calculated
C          as the ratio of the NS footage to the average length of the
C          north and south sides,multiplied by the change in longitude
C          from the east to the west along those sides but scaled
C       LATOFF = The latitude equivalent of LONOFF
C       LONOMN = Minimum Longitude of Outside rectangle
C       LATOMN =     "     Latitude of same
C       LONOMX = Maximum Longitude of same
C       LATOMX =     "       Latitude of same
C       LONIMN = Minimum Longitude of Inside rectangle
C       LATIMN =     "     Latitude of same
C       LONIMX = Maximum Longitude of same
C       LATIMX =     "       Latitude of same
C       DLON = Length of side of triangle solved to find XLON,the
C        Longitude of an intersection point
C       DLAT = Same but in latitude
C       XLON = Longitude of the intersection point of a vertical ray
C        passing through THE point with a side of the quadrilateral
C       XLAT = Latitude of ray intersection through point with side
C
C
C******* Check the input values for errors for both conversion types
C
        LEGAL=1
        GEOGR=2
        NONSTD=0
        STAT=0
        CENTER=.FALSE.
        FOURCRNR=.FALSE.
        IF (OPT.EQ.3 .OR. OPT.EQ.7) THEN
              FOURCRNR = .TRUE.
        ENDIF
        STLINE=0
        RIVER=0
        RESERVE=0
        TREATY=0
        SLACKN=0
        SLACKW=0
        SLACKNW=0
c
```

```
c******* Standardize corner and side notation
c
        IF(CORNR.EQ.'NE'.OR.CORNR.EQ.'ne'.OR.
     &             CORNR.EQ.'A '.OR.CORNR.EQ.'a ')THEN
              CORNR='NE'
           ELSEIF(CORNR.EQ.'NW'.OR.CORNR.EQ.'nw'.OR.
     &             CORNR.EQ.'B '.OR.CORNR.EQ.'b ')THEN
              CORNR='NW'
           ELSEIF(CORNR.EQ.'SW'.OR.CORNR.EQ.'sw')then
              CORNR='SW'
           elseif(CORNR.EQ.'C '.OR.CORNR.EQ.'c ')THEN
              if(opt.eq.1.or.opt.eq.5)then
                 CORNR='SW'
              else
                 cornr='C'
              endif
           ELSEIF(CORNR.EQ.'SE'.OR.CORNR.EQ.'se'.OR.
     &             CORNR.EQ.'D '.OR.CORNR.EQ.'d ')THEN
              CORNR='SE'
           ELSEIF (CORNR.EQ.'S ' .OR. CORNR.EQ.'s ') THEN
              CORNR='S '
           ELSEIF (CORNR.EQ.'E ' .OR. CORNR.EQ.'e ') THEN
              CORNR='E '
           ELSEIF (CORNR.EQ.'N ' .OR. CORNR.EQ.'n ') THEN
              CORNR='N '
           ELSEIF (CORNR.EQ.'W ' .OR. CORNR.EQ.'w ') THEN
              CORNR='W '
           ELSEIF (CORNR.EQ.'CR'.OR.CORNR.EQ.'CC'.OR.
     &             CORNR.EQ.'cr'.OR.CORNR.EQ.'cc')THEN
              CORNR='C'
           ENDIF
           IF(CORNRGL.EQ.'NE'.OR.CORNRGL.EQ.'ne'.OR.
     &             CORNRGL.EQ.'A '.OR.CORNRGL.EQ.'a ')THEN
              CORNRGL='NE'
           ELSEIF(CORNRGL.EQ.'NW'.OR.CORNRGL.EQ.'nw'.OR.
     &             CORNRGL.EQ.'B '.OR.CORNRGL.EQ.'b ')THEN
              CORNRGL='NW'
           ELSEIF(CORNRGL.EQ.'SW'.OR.CORNRGL.EQ.'sw'.OR.
     &             CORNRGL.EQ.'C '.OR.CORNRGL.EQ.'c ')THEN
              CORNRGL='SW'
           ELSEIF(CORNRGL.EQ.'SE'.OR.CORNRGL.EQ.'se'.OR.
     &             CORNRGL.EQ.'D '.OR.CORNRGL.EQ.'d ')THEN
              CORNRGL='SE'
           ENDIF
C
```

```
C*********************************************************************
C* * * * * * * * * * * * * * * * * * * * * * * * * CHECK LEGAL INPUT VALUES
C*********************************************************************
C
        IF (SORCE.EQ.LEGAL) THEN
C
C****   Initialize output variables to safe values and
C*      check for errors in Legal input parameters
C
           PLON=0.
           PLAT=0.
           LONX(1,1)=0.
           LATX(1,1)=0.
           IF (TSHP.LT.1 .OR. TSHP.GT.35) THEN
              STAT=-1
              RETURN
           ENDIF
           IF (EW.EQ.'E' .OR. EW.EQ.'e') THEN
              IF (RANG.LT.1 .OR. RANG.GT.25) THEN
                 STAT=-2
                 RETURN
              ENDIF
              EW='E'
           ELSEIF (EW.EQ.'W' .OR. EW.EQ.'w') THEN
              IF (RANG.LT.1 .OR. RANG.GT.43) THEN
                 STAT=-2
                 RETURN
              ENDIF
              EW='W'
           ELSE
              STAT=-3
              RETURN
           ENDIF
           IF (OPT.LT.0 .OR. OPT.GT.7) THEN
              STAT=-12
              RETURN
           ENDIF
           IF (SECT.LT.0 .OR. SECT.GT.36) THEN
              STAT=-4
              RETURN
           ENDIF
           IF (SECT.GE.1 .AND. SECT.LE.5) SLACKN=1
           IF (SECT.EQ.6)SLACKNW=1
           IF (SECT.EQ.7.OR.SECT.EQ.18.OR.SECT.EQ.19.
     &          OR.SECT.EQ.30.OR.SECT.EQ.31) SLACKW=1
C
C*  For corner/side OPTION check for spec in SUBD(1) or CORNR
C*  If CORNR is blank but SUBD is not, copy SUBD to CORNR
C
           IF (OPT.EQ.1 .OR. OPT.EQ.5) THEN
              IF (CORNR.EQ.'  ' .AND. SUBD(1).NE.'  ') CORNR=SUBD(1)
              IF (CORNR.EQ.'  ') FOURCRNR=.TRUE.
               if(fourcrnr)write(*,*)'fourcrnr=.TRUE.'
C
```

```
C* * * * * * * * * * * * * * * * * * CHECK SUBDIVISION SPECS (OPT = 0/4)
C
          ELSEIF (OPT.EQ.0 .OR. OPT.EQ.4) THEN
C
C*  For subdivision options, check SUBD
C*  values and set ORDER to guide the subdivision process
C
C
C* * * * * * * * * * * * * * * * * * * * * * CHECK AND CODE SUBDIVISIONS
C
C
C*  Number of subdivisions used initially asssumed to be 4, process
C*  control is initially zero, and unusual center specs for CORNR are
C*  reset to blank.  Process control via ORDER is created by adding
C*  one digit in each of 0 to 4 decimal positions in ORDER, with each
C*  digit uniquely representing one of eight possible subdivision specs
C
             NSUB=4
             ORDER=0
             IF (CORNR(1:1).EQ.' ') CORNR = CORNR(2:2) // CORNR(1:1)
             IF (CORNR.EQ.'CC' .OR. CORNR.EQ.'cc') CENTER=.TRUE.
             IF (CORNR.EQ.'CR' .OR. CORNR.EQ.'cr') CENTER=.TRUE.
             IF (CORNR.EQ.'C ' .OR. CORNR.EQ.'c ') CENTER=.TRUE.
             IF (CORNR.EQ.'  ') FOURCRNR=.TRUE.
             DO 50 I=4,1,-1
                IF(SUBD(I)(1:1).EQ.' ') THEN
                   SUBD(I)=SUBD(I)(2:2) // SUBD(I)(1:1)
                ENDIF
                IF (SUBD(I).EQ.'D ' .OR. SUBD(I).EQ.'SE') THEN
                   SUBD(I)='SE'
                   ORDER=10*ORDER+1
                ELSEIF (SUBD(I).EQ.'d ' .OR. SUBD(I).EQ.'se') THEN
                   SUBD(I)='SE'
                   ORDER=10*ORDER+1
                ELSEIF (SUBD(I).EQ.'A ' .OR. SUBD(I).EQ.'NE') THEN
                   SUBD(I)='NE'
                   ORDER=10*ORDER+2
                ELSEIF (SUBD(I).EQ.'a ' .OR. SUBD(I).EQ.'ne') THEN
                   SUBD(I)='NE'
                   ORDER=10*ORDER+2
                ELSEIF (SUBD(I).EQ.'B ' .OR. SUBD(I).EQ.'NW') THEN
                   SUBD(I)='NW'
                   ORDER=10*ORDER+3
                ELSEIF (SUBD(I).EQ.'b ' .OR. SUBD(I).EQ.'nw') THEN
                   SUBD(I)='NW'
                   ORDER=10 * ORDER + 3
                ELSEIF (SUBD(I).EQ.'C ' .OR. SUBD(I).EQ.'SW') THEN
                   SUBD(I)='SW'
                   ORDER=10 * ORDER + 4
                ELSEIF (SUBD(I).EQ.'c ' .OR. SUBD(I).EQ.'sw') THEN
                   SUBD(I)='SW'
                   ORDER=10 * ORDER + 4
                ELSEIF (SUBD(I).EQ.'N ' .OR. SUBD(I).EQ.'n ') THEN
                   SUBD(I)='N '
                   ORDER=10 * ORDER + 5
                ELSEIF (SUBD(I).EQ.'W ' .OR. SUBD(I).EQ.'w ') THEN
                   SUBD(I)='W'
                   ORDER=10 * ORDER + 6
```

```
                ELSEIF (SUBD(I).EQ.'S ' .OR. SUBD(I).EQ.'s ') THEN
                   SUBD(I)='S'
                   ORDER=10 * ORDER + 7
                ELSEIF (SUBD(I).EQ.'E ' .OR. SUBD(I).EQ.'e ') THEN
                   SUBD(I)='E'
                   ORDER=10 * ORDER + 8
                ELSEIF (SUBD(I).EQ.'  ' .AND. ORDER.EQ.0) THEN
                   NSUB=I-1
                   IF (I.LT.1) NSUB=1
                ELSE
                   STAT = -4 - I
                   RETURN
                ENDIF
50           CONTINUE
          ENDIF
C
```

```
C * * * * * * * * * * * * * * * * * * * * CHECK GEOGRAPHIC INPUT VALUES
C*
C**  Now check the arguments for Geographic to Legal (SORCE=2)
C*
C
        ELSEIF (SORCE.EQ.GEOGR.or.sorce.eq.3) THEN
           TSHP=0
           RANG=0
           SECT=0
           EW='  '
           SUBD(1)='  '
           SUBD(2)='  '
           SUBD(3)='  '
           SUBD(4)='  '
           CORNR='  '
C
C*  If OPT=1,convert DMS values in LONX(1) and LATX(1)
C*   to PLON,PLAT after checking the dms values for errors
C
           IF (OPT.EQ.1) THEN
              LONDEG=ABS(LONX(1,1))
              LONMIN=ABS(LONX(1,2))
              LONSEC=ABS(LONX(1,3))
              LATDEG=LATX(1,1)
              LATMIN=LATX(1,2)
              LATSEC=LATX(1,3)
              IF (LONDEG.LT.94. .OR. LONDEG.GT.102.) THEN
                 STAT=-17
                 RETURN
              ELSEIF (LONMIN.GT.59. .OR. LONSEC.GT.59.) THEN
                 STAT=-17
                 RETURN
              ELSEIF (LONMIN.LT.0. .OR. LONSEC.LT.0.) THEN
                 STAT=-17
                 RETURN
              ELSEIF (LATDEG.LT.36. .OR. LATDEG.GT.40.) THEN
                 STAT=-17
                 RETURN
              ELSEIF (LATMIN.GT.59. .OR. LATMIN.LT.0.) THEN
                 STAT=-17
                 RETURN
              ELSEIF (LATSEC.GT.59. .OR. LATSEC.LT.0.) THEN
                 STAT=-17
                 RETURN
              ENDIF
              PLON=LONDEG+LONMIN/60.+LONSEC/3600.
              PLAT=LATDEG+LATMIN/60.+LATSEC/3600.
           ENDIF
C
```

```
C*  Make sure that the input longitudes are set negative for
C*  internal processing (output values will be positive),and
C*  also check the long/lat extremes for Kansas
C
           IF (PLON.GT.0.) PLON=-1.*PLON
           IF (PLON.LT.-102.05 .OR. PLON.GT.-94.55) THEN
              STAT=-10
              RETURN
           ENDIF
           IF (PLAT.LT.36.95 .OR. PLAT.GT.40.05) THEN
              STAT=-11
              RETURN
           ENDIF
        ELSE
           STAT=-9
           RETURN
        ENDIF
C
C*
```

```
C       **  First handle the case SORCE=1. SORCE information
C       **  is Legal (Township/RANG/Section etc.) and the desired
C       **  output is Geographic (Longitude,Latitude).
C*
C
C*********************************************************************
C * * * * * * * * * * * * * * * * * * * * LEGAL TO GEOGRAPHIC CONVERSION
C*********************************************************************
C
C
C * * * * * * * * * FIND CORNER LOCATIONS OF SECTION/TSHP
C
C*  Calculate the record number and read the Township record
C
        IF (SORCE.EQ.LEGAL) THEN
            TREC=(TSHP-1)*68
            IF (EW.EQ.'E') THEN
                TREC=TREC+43+RANG
            ELSE
                TREC=TREC+44-RANG
            ENDIF
c               write(*,*)'LEO3BASE record number = ',trec
            READ(LUN,REC=TREC,ERR=1999)
     &          ((LON(I,J),LAT(I,J),J=1,7),I=1,7), TFLAG, SFLAG
C
C*  Check for empty township
C
            IF (TFLAG.EQ.0) THEN
                STAT=-19
                RETURN
            ENDIF
            IF (SECT.NE.0) THEN
C
C*          Check for missing section corners
C
                IF (SFLAG(SECT).EQ.0) THEN
                   STAT=-32
                   RETURN
C
```

```
C*  Calculate corner flags
C
               ELSE
                    if(sflag(sect).lt.0)then
                   call fedx
     &            (TSHP,RANG,EW,SECT,OPT,SUBD,STAT,SORCE,trec)
                    endif
                  ALPHA=ABS(SFLAG(SECT))
                  SCOR(1)=INT(ALPHA/1000)
                  BETA=ALPHA-1000*SCOR(1)
                  SCOR(2)=INT(BETA/100)
                  GAMMA=BETA-100*SCOR(2)
                  SCOR(3)=INT(GAMMA/10)
                  OMEGA=GAMMA-10*SCOR(3)
                  SCOR(4)=OMEGA
C
                  IF(SCOR(1).GT.5)STLINE=1
                  IF(SCOR(2).GT.5)RIVER=1
                  IF(SCOR(3).GT.5)RESERVE=1
                  IF(SCOR(4).GT.5)TREATY=1
C
                  do 101 i=1,4
                        IF(SCOR(i).GT.5)SCOR(i)=SCOR(i)-5
101               continue
               ENDIF
C
C*  Get the Row and Column of the NW corner of desired section
C
               SROW=(SECT-1)/6+1
               SCOL=MOD (INT (SECT),6)
               IF (SCOL.EQ.0) SCOL=6
               IF (MOD (INT (SROW),2).EQ.1) SCOL=7-SCOL
          ELSE
C
C*  Check for missing township corners (SECT=0)
C
             STAT=0
C
C*  Check for footage or subdivision options which are not
C*  allowed when the township has been selected
C
             IF (MOD (INT (OPT),2).EQ.0) THEN
                STAT=-26
                RETURN
             ENDIF
C
C*  Set SROW=SCOL=1 for the township (NW corner)
C
             SROW=1
             SCOL=1
          ENDIF
C
```

```
C************************************
C * * * * * * * * * * * * * * * * * * FOUR CORNERS OPTION (OPT=3/7)
C************************************
C*
C**  Do the Four Corners option first (OPT=3,7)
C*
C
           IF (OPT.EQ.3 .OR. OPT.EQ.7) THEN
              DO 100 I=1,3,2
                 LONX(I,1)=ABS(LON(SROW,SCOL))
                 LATX(I,1)=LAT(SROW,SCOL)
                 IF (SECT.NE.0) THEN
                    SCOL=SCOL+1
                 ELSE
                    SCOL=7
                 ENDIF
                 LONX(I+1,1)=ABS(LON(SROW,SCOL))
                 LATX(I+1,1)=LAT(SROW,SCOL)
                 IF (SECT.NE.0) THEN
                    SROW=SROW+1
                    SCOL=SCOL-1
                 ELSE
                    SROW=7
                    SCOL=1
                 ENDIF
100           CONTINUE
              if(abs(sflag(sect)).gt.6000)stat=34
C
C
C*  Go set STAT value and exit
C
              IF (OPT.LT.4) GOTO 575
              GOTO 550
           ENDIF
C
C*****************************
C * * * * * * * * * * * * * * * * EXTRACT SECTION/TSHP CORNER LOCATIONS
C* * * * * * * * * * * * * * * * * FOR ALL OTHER OPTIONS (0/4, 1/5, 2/6)
C*****************************
C
C*******  Get the corner locations for all options except 4 corners
C
           OFF=1
           IF (SECT.EQ.0) OFF=6
           LONNW=LON(SROW,SCOL)
           LATNW=LAT(SROW,SCOL)
           LONSW=LON(SROW+OFF,SCOL)
           LATSW=LAT(SROW+OFF,SCOL)
           LONSE=LON(SROW+OFF,SCOL+OFF)
           LATSE=LAT(SROW+OFF,SCOL+OFF)
           LONNE=LON(SROW,SCOL+OFF)
           LATNE=LAT(SROW,SCOL+OFF)
           CTRLON=(LONNW+LONSW+LONSE+LONNE)/4.
           CTRLAT=(LATNW+LATSW+LATSE+LATNE)/4.
C
```

```
          IF (OPT.EQ.1 .OR. OPT.EQ.5) THEN
C
C*  Go to apply CORNR, optional DMS conversion, and to set STAT
C
             GOTO 510
          ENDIF
C
C***********************************************
C * * * * * * * * * * * * * * * * * * * * * * * FOOTAGES (OPT=2/6)
C***********************************************
C
C*
C*  Do Footages for Legal to Geographic conversions here
C*
C
          IF (OPT.EQ.2 .OR. OPT.EQ.6) THEN
C
C*  First,set the projection area to be the minimum
C*  bounding rectangle about the section corners and
C*  initialize the projection software with a fixed MP
C*  projection at scale=12 to produce units equal to
C*  feet on the earth's surface.
C
C
C* * * * * * * * * * * * * * * * * * PROJECT CORNERS TO FIND SIDE LENGTHS
C
C
              SOUTH=AMIN1 (LATSW,LATSE)
              NORTH=AMAX1 (LATNW,LATNE)
              WEST=AMIN1 (LONSW,LONNW)
              EAST=AMAX1 (LONSE,LONNE)
C
C*  Initialize the (MP) projection
C
              CALL LEO3SETP (CORN)
C
C*  Project the four section corners (Footages are applied only
C*  from the section corners)
C
              DBLON=DBLE (LONSW)
              DBLAT=DBLE (LATSW)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XSW=SNGL (XDBL)
              YSW=SNGL (YDBL)
              DBLON=DBLE (LONSE)
              DBLAT=DBLE (LATSE)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XSE=SNGL (XDBL)
              YSE=SNGL (YDBL)
              DBLON=DBLE (LONNE)
              DBLAT=DBLE (LATNE)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XNE=SNGL (XDBL)
              YNE=SNGL (YDBL)
              DBLON=DBLE (LONNW)
              DBLAT=DBLE (LATNW)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XNW=SNGL (XDBL)
```

```
                YNW=SNGL (YDBL)
      C
```

```
C*******  If footages are (approx.) equal to zero,the call is to
C*        get the lengths of all four sides of the section,measured
C*        in feet on the ground (projected units when done as it is
C*        for the footage application).  These are calculated and
C*        stored in LONX(1-4,1) in the order N,E,W,and S.  Then
C*******  go to set the value of STAT and NONSTD and exit.
C
            IF (ABS(FTNS).LT.1. .AND. ABS(FTEW).LT.1.) THEN
              LONX(1,1)=SQRT((XNE-XNW)**2+(YNE-YNW)**2)
              LONX(2,1)=SQRT((XNE-XSE) ** 2+(YNE-YSE) ** 2 )
              LONX(3,1)=SQRT((XNW-XSW) ** 2+(YNW-YSW) ** 2 )
              LONX(4,1)=SQRT((XSE-XSW) ** 2+(YSE-YSW) ** 2 )
              GOTO 575
            ENDIF
C
C******  Calculate the NS and EW ratios of footage to length after
C*       calculating the appropriate side lengths which are
C*       determined by the footage directions that are implied
C*       by the signs of the footages (+=N,E and -=S,W)
C*      which implies the corner for application if not given
C******         explicitly.
C
C* * * * * * * * * * * * * * * * * APPLY SIDE LENGTH TO FOOTAGE RATIOS TO
C * * * * * * * * * * * * * * * * * LONG/LAT CHANGES TO FIND POINT LOCATION
C
C
C*  LENGNS and LENGEW are the averaged side lengths of the
C*  section. Divide the footages by them to get the ratios.
C*  (Make sure the footages don't exceed the lengths)
C*  (Side lengths are measured as changes in x or y only)
C
            LENGNS=((YNE-YSE)+(YNW-YSW))/2.
            IF (ABS(FTNS).GT.LENGNS) STAT=16
            LENGEW=((XNE-XNW)+(XSE-XSW))/2.
            IF (ABS(FTEW).GT.LENGEW) STAT=17
c
c
C
C
```

```
C*  The long/lat offsets are calculated.
C
C
C*  Apply offsets from the appropriate corner
C
             IF (CORNR.EQ.BLANK2)THEN
               IF(FTNS.GT.0)THEN
                  IF(FTEW.GT.0)CORNR='SW'
                  IF(FTEW.LT.0)CORNR='SE'
                  IF(FTEW.EQ.0)THEN
                  STAT=-51
                  RETURN
                  ENDIF
               ELSEIF(FTNS.LT.0)THEN
                  IF(FTEW.GT.0)CORNR='NW'
                  IF(FTEW.LT.0)CORNR='NE'
                  IF(FTEW.EQ.0)THEN
                  STAT=-52
                  RETURN
                  ENDIF
               ELSE
                  IF(FTEW.GT.0)THEN
                  STAT=-53
                  RETURN
                  ELSE
                  STAT=-54
                  RETURN
                  ENDIF
               ENDIF
             ENDIF
             IF(CORNR.EQ.'NE')THEN
               GOTO 431
             ELSEIF(CORNR.EQ.'NW')THEN
               GOTO 432
             ELSEIF(CORNR.EQ.'SW')THEN
               GOTO 433
             ELSEIF(CORNR.EQ.'SE')THEN
               GOTO 434
             ELSE
C              no acceptable corner specified for footage location
               STAT=-25
               RETURN
             ENDIF
430          CONTINUE
```

```
433             continue
c               footage is measured from section's SW corner
                if(ftns.lt.0.and.ftew.lt.0)then
                STAT=-16
                return
                endif
                 SURV=SCOR(3)
                 IF(SURV.GT.5)SURV=SURV-5
                 IF(SURV.NE.1 .AND. SURV.NE.4) THEN
                IF(STAT.NE.16.AND.STAT.NE.17)THEN
                      STAT=43
                ELSE
                      STAT=STAT-11+44
                ENDIF
                 ENDIF
                LATOFF=FTNS*(LATNE-LATSE)/(YNE-YSE)
                PLAT=LATSW+LATOFF
                DLONPFN=(LONNE-LONNW)/(XNE-XNW)
                DLONPFS=(LONSE-LONSW)/(XSE-XSW)
                AVLATN=(LATNE+LATNW)/2
                AVLATS=(LATSE+LATSW)/2
                DLATSEC=(AVLATN-AVLATS)
                RATDDLL=(DLONPFN-DLONPFS)/DLATSEC
                DLATP=PLAT-AVLATS
                LONOFF=FTEW*(DLONPFN+(RATDDLL*DLATP))
                PLON=LONSW+LONOFF
                goto 435
434             continue
c               footage is measured from section's SE corner
                if(ftns.lt.0.and.ftew.gt.0)then
                STAT=-16
                return
                endif
                 SURV=SCOR(4)
                 IF(SURV.GT.5)SURV=SURV-5
                 IF(SURV.NE.1 .AND. SURV.NE.4) THEN
                IF(STAT.NE.16.AND.STAT.NE.17)THEN
                      STAT=44
                ELSE
                      STAT=STAT-9+44
                ENDIF
                 ENDIF
                LATOFF=FTNS*(LATNE-LATSE)/(YNE-YSE)
                PLAT=LATSE+LATOFF
                DLONPFN=(LONNE-LONNW)/(XNE-XNW)
                DLONPFS=(LONSE-LONSW)/(XSE-XSW)
                AVLATN=(LATNE+LATNW)/2
                AVLATS=(LATSE+LATSW)/2
                DLATSEC=(AVLATN-AVLATS)
                RATDDLL=(DLONPFN-DLONPFS)/DLATSEC
                DLATP=PLAT-AVLATS
                LONOFF=FTEW*(DLONPFN+(RATDDLL*DLATP))
                PLON=LONSE+LONOFF
                goto 435
```

```
432               continue
c                 footage is measured from section's NW corner
                  if(ftns.gt.0.and.ftew.lt.0)then
                  STAT=-16
                  return
                  endif
                   SURV=SCOR(2)
                   IF(SURV.GT.5)SURV=SURV-5
                   IF(SURV.NE.1 .AND. SURV.NE.4) THEN
                  IF(STAT.NE.16.AND.STAT.NE.17)THEN
                        STAT=42
                  ELSE
                        STAT=STAT-13+44
                  ENDIF
                   ENDIF
                  LATOFF=FTNS*(LATNE-LATSE)/(YNE-YSE)
                  PLAT=LATNW+LATOFF
                  DLONPFN=(LONNE-LONNW)/(XNE-XNW)
                  DLONPFS=(LONSE-LONSW)/(XSE-XSW)
                  AVLATN=(LATNE+LATNW)/2
                  AVLATS=(LATSE+LATSW)/2
                  DLATSEC=(AVLATN-AVLATS)
                  RATDDLL=(DLONPFN-DLONPFS)/DLATSEC
                  DLATP=PLAT-AVLATN
                  LONOFF=FTEW*(DLONPFN+(RATDDLL*DLATP))
                  PLON=LONNW+LONOFF
                  goto 435
431               continue
c                 footage is measured from section's NE corner
                  if(ftns.gt.0.and.ftew.gt.0)then
                  STAT=-16
                  return
                  endif
                  SURV=SCOR(1)
                  IF(SURV.GT.5)SURV=SURV-5
                  IF(SURV.NE.1 .AND. SURV.NE.4) THEN
                  IF(STAT.NE.16.AND.STAT.NE.17)THEN
                        STAT=41
                  ELSE
                        STAT=STAT-15+44
                  ENDIF
                  ENDIF
                  LATOFF=FTNS*(LATNE-LATSE)/(YNE-YSE)
                  PLAT=LATNE+LATOFF
                  DLONPFN=(LONNE-LONNW)/(XNE-XNW)
                  DLONPFS=(LONSE-LONSW)/(XSE-XSW)
                  AVLATN=(LATNE+LATNW)/2
                  AVLATS=(LATSE+LATSW)/2
                  DLATSEC=(AVLATN-AVLATS)
                  RATDDLL=(DLONPFN-DLONPFS)/DLATSEC
                  DLATP=PLAT-AVLATN
                  LONOFF=FTEW*(DLONPFN+(RATDDLL*DLATP))
                  PLON=LONNE+LONOFF
```

```
435            continue
c       write(*,*)'degrees longitude per foot, at north side: ',dlonpfn
c       write(*,*)'feet per degree longitude = ',1/dlonpfn
c       write(*,*)'degrees longitude per foot, at south side: ',dlonpfs
c       write(*,*)'feet per degree longitude = ',1/dlonpfs
C
C*  Go to optional DMS conversion and to set STAT
C
              IF (OPT.EQ.2) GOTO 575
              GOTO 550
           ENDIF
C
C
C*****************************
C * * * * * * * * * * * * * * SECTION & SUBAREA SUBDIVISION (OPT=0/4)
C*****************************
C
C*
C**  Perform up to four (standard) subdivisions
C*
C
C*  Do nothing if indicated, else perform NSUB subdivisions, starting
C*  with SUBD(1), each applied to the area created by the previous ones,
C*  and the type of each indicated by the corresponding code digit in
C*  ORDER.  SUBD(1) with the least significant digit, and so on
C
        IF (abs(SFLAG(SECT)).GT.6000)THEN
              CALL STATE_EDGE(SFLAG(SECT),LONSW,LATSW,
     &           LONSE,LATSE,LONNW,LATNW,LONNE,LATNE)
              CTRLON=(LONNW+LONSW+LONSE+LONNE)/4.
              CTRLAT=(LATNW+LATSW+LATSE+LATNE)/4.
        ENDIF
           IF (ORDER.EQ.0) GOTO 510
           DO 500 I=1,NSUB
              NUM = MOD ( INT (ORDER), 10)
C
C       **  Remove the digit for this subdivision
C
              ORDER = ORDER / 10
              IF (NUM.EQ.0) GOTO 510
C
```

```
C*********************************************
C * * * * * * * * * * * * * * * * * * * * * * * QUARTER-AREA SUBDIVISION
C*********************************************
C
C       **  First the 1/4 Areas
C
             IF (NUM.EQ.1) THEN
C
C       **  South East
C
                 LONSW=(LONSW+LONSE)/2.
                 LATSW=(LATSW+LATSE)/2.
                 LONNE=(LONNE+LONSE)/2.
                 LATNE=(LATNE+LATSE)/2.
                 LONNW=CTRLON
                 LATNW=CTRLAT
              ELSEIF (NUM.EQ.2) THEN
C
C       **  North East
C
                 LONSE=(LONNE+LONSE)/2.
                 LATSE=(LATNE+LATSE)/2.
                 LONNW=(LONNW+LONNE)/2.
                 LATNW=(LATNW+LATNE)/2.
                 LONSW=CTRLON
                 LATSW=CTRLAT
              ELSEIF (NUM.EQ.3) THEN
C
C       **  North West
C
                 LONSW=(LONSW+LONNW)/2.
                 LATSW=(LATSW+LATNW)/2.
                 LONNE=(LONNW+LONNE)/2.
                 LATNE=(LATNW+LATNE)/2.
                 LONSE=CTRLON
                 LATSE=CTRLAT
              ELSEIF (NUM.EQ.4) THEN
C
C       **  South West
C
                 LONNW=(LONNW+LONSW)/2.
                 LATNW=(LATNW+LATSW)/2.
                 LONSE=(LONSW+LONSE)/2.
                 LATSE=(LATSW+LATSE)/2.
                 LONNE=CTRLON
                 LATNE=CTRLAT
C
```

```
C************************************************
C * * * * * * * * * * * * * * * * * * * * * * * * HALF-AREA SUBDIVISION
C************************************************
C
C       **  Now for the 1/2 Area subdivisions
C
             ELSEIF (NUM.EQ.5) THEN
C
C       **  North
C
                LONSW=(LONNW+LONSW)/2.
                LATSW=(LATNW+LATSW)/2.
                LONSE=(LONSE+LONNE)/2.
                LATSE=(LATSE+LATNE)/2.
              ELSEIF (NUM.EQ.6) THEN
C
C       **  West
C
                LONNE=(LONNE+LONNW)/2.
                LATNE=(LATNE+LATNW)/2.
                LONSE=(LONSE+LONSW)/2.
                LATSE=(LATSE+LATSW)/2.
              ELSEIF (NUM.EQ.7) THEN
C
C       **  South
C
                LONNW=(LONNW+LONSW)/2.
                LATNW=(LATNW+LATSW)/2.
                LONNE=(LONNE+LONSE)/2.
                LATNE=(LATNE+LATSE)/2.
              ELSEIF (NUM.EQ.8) THEN
C
C       **  East
C
                LONSW=(LONSW+LONSE)/2.
                LATSW=(LATSW+LATSE)/2.
                LONNW=(LONNW+LONNE)/2.
                LATNW=(LATNW+LATNE)/2.
              ELSE
                STAT=-4-I
                RETURN
              ENDIF
C
C       **  Calculate the new center
C
              CTRLON=(LONNW+LONNE+LONSE+LONSW)/4.
              CTRLAT=(LATNW+LATNE+LATSE+LATSW)/4.
500         CONTINUE
C
C
```

```
C*****************************************
C * * * * * * * * * * * * * * * * * * * * * APPLICATION OF CORNR SPEC
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * (OPT = 0/4, 1/5)
C*****************************************
C
C
C       *  If CORNR option is used,select the indicated point of the
C       *  nine significant points in the target area, or if blank
C       *  use the corners of the area as the selected point(s)
C
510        IF (CENTER) THEN
              PLON=CTRLON
              PLAT=CTRLAT
C
C
C * * * * * * * * * * * * * * * * * * * * * * * * * FOUR CORNERS OF A SUBAREA
C                    (OPT = 0/4 OR 1/5 WITH CORNR = '   ')
C
C       **  If CORNR spec is blank, put the corner locations of the
C       **  target area into the LEOXTR variables (LONX, LATX)
C
           ELSEIF (FOURCRNR) THEN
                LONX(1,1)=ABS(LONNW)
                LONX(2,1)=ABS(LONNE)
                LONX(3,1)=ABS(LONSW)
                LONX(4,1)=ABS(LONSE)
                LATX(1,1)=LATNW
                LATX(2,1)=LATNE
                LATX(3,1)=LATSW
                LATX(4,1)=LATSE
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * APPLYING THE CORNR SPEC
C
           ELSEIF (CORNR.EQ.'NE') THEN
              PLON=LONNE
              PLAT=LATNE
           ELSEIF (CORNR.EQ.'NW') THEN
              PLON=LONNW
              PLAT=LATNW
           ELSEIF (CORNR.EQ.'SW') THEN
              PLON=LONSW
              PLAT=LATSW
           ELSEIF (CORNR.EQ.'SE') THEN
              PLON=LONSE
              PLAT=LATSE
           ELSEIF (CORNR.EQ.'S ') THEN
              if(sect.eq.0.and.(sflag(31).eq.0.or.sflag(36).eq.0))then
                stat=-21
                return
              endif
              PLON=(LONSW+LONSE)/2.
              PLAT=(LATSW+LATSE)/2.
```

```
            ELSEIF (CORNR.EQ.'E ') THEN
               if(sect.eq.0.and.(sflag(1).eq.0.or.sflag(36).eq.0))then
                 stat=-21
                 return
               endif
               PLON=(LONSE+LONNE)/2.
               PLAT=(LATSE+LATNE)/2.
            ELSEIF (CORNR.EQ.'N ') THEN
               if(sect.eq.0.and.(sflag(1).eq.0.or.sflag(6).eq.0))then
                 stat=-21
                 return
               endif
               PLON=(LONNW+LONNE)/2.
               PLAT=(LATNW+LATNE)/2.
            ELSEIF (CORNR.EQ.'W ') THEN
               if(sect.eq.0.and.(sflag(31).eq.0.or.sflag(6).eq.0))then
                 stat=-21
                 return
               endif
               CORNR='W '
               PLON=(LONNW+LONSW)/2.
               PLAT=(LATNW+LATSW)/2.
            ELSE
               STAT=-25
               RETURN
            ENDIF
C
         IF(abs(SFLAG(SECT)).GT.6000)THEN
                 CALL  CHECK_STATE_EDGE(STAT,LONSW,LATSW,LONSE,LATSE,
     &              LONNW,LATNW,LONNE,LATNE,PLON,PLAT,CORNR)
                 WRITE(*,*)'STATUS = ',STAT
                 if(stat.lt.0)return
C                write(*,*)'lonnw = ',lonnw
C                write(*,*)'lonsw = ',lonsw
C
C * * * * * * * * * * * * * * * * * * * * * * * FOUR CORNERS OF A SUBAREA
C                      (OPT = 0/4 OR 1/5 WITH CORNR = '   ')
C
C          **  If CORNR spec is blank, put the corner locations of the
C          **  target area into the LEOXTR variables (LONX, LATX)
C
                 if(stat.eq.34)then
                  LONX(1,1)=ABS(LONNW)
                  LONX(2,1)=ABS(LONNE)
                  LONX(3,1)=ABS(LONSW)
                  LONX(4,1)=ABS(LONSE)
                  LATX(1,1)=LATNW
                  LATX(2,1)=LATNE
                  LATX(3,1)=LATSW
                  LATX(4,1)=LATSE
                 endif
                 IF(STAT.EQ.36)THEN
C
```

```
C * * * * * * * *  APPLYING THE CORNR SPEC TO PARTIAL SUBDIVISION
C
                  IF (CENTER) THEN
                    CTRLON=(LONNW+LONNE+LONSE+LONSW)/4.
                    CTRLAT=(LATNW+LATNE+LATSE+LATSW)/4.
                    PLON=CTRLON
                    PLAT=CTRLAT
                  ELSEIF (CORNR.EQ.'NE') THEN
                     PLON=LONNE
                     PLAT=LATNE
                  ELSEIF (CORNR.EQ.'NW') THEN
                     PLON=LONNW
                     PLAT=LATNW
                  ELSEIF (CORNR.EQ.'SW') THEN
                     PLON=LONSW
                     PLAT=LATSW
                  ELSEIF (CORNR.EQ.'SE') THEN
                     PLON=LONSE
                     PLAT=LATSE
                  ELSEIF (CORNR.EQ.'S ') THEN
                     PLON=(LONSW+LONSE)/2.
                     PLAT=(LATSW+LATSE)/2.
                  ELSEIF (CORNR.EQ.'E ') THEN
                     PLON=(LONSE+LONNE)/2.
                     PLAT=(LATSE+LATNE)/2.
                  ELSEIF (CORNR.EQ.'N ') THEN
                     PLON=(LONNW+LONNE)/2.
                     PLAT=(LATNW+LATNE)/2.
                  ELSEIF (CORNR.EQ.'W ') THEN
                     PLON=(LONNW+LONSW)/2.
                     PLAT=(LATNW+LATSW)/2.
                  ELSE
                     STAT=-25
                     RETURN
                  ENDIF
               endif
C
       ENDIF
C
C       *  Point longitude and latitude have been identified,now
C       *  translate to DMS if required,set STAT value and exit
C
C
C
```

```
C * * * * * * * * * * * * * * * * * * * * * DMS CONVERSION * * * * * *
C                                        FOR 4 CORNERS AND SINGLE POINTS
C
C ********************          *****************          ************
C   FIRST FOR 4 CORNERS FROM OPT = 7 AND CORNR='   ' WITH OPT = 4 & 5
C ********************          *****************          ************
C*
C
550       IF (OPT.LT.4) GOTO 575
           IF (FOURCRNR) THEN
        write(*,*)'FOURCRNR = TRUE'
              DO 200 I=1,4
        write(*,*)'i = ',i
C
C*  Convert the 4 longitudes in degrees in LONX(1-4,1) into
C*  degrees (1-4,1),minutes (1-4,2),and seconds (1-4,3).
C*  Start by setting the longitudes to positive values since
C*  all degree-minute-second values must be positive.
C
                 LONX(I,1)=ABS(LONX(I,1))
        write(*,*)'lonx(i,1) = ',lonx(i,1)
                 LONX(I,2)=60.*(LONX(I,1)-INT(LONX(I,1)))
        write(*,*)'lonx(i,2) = ',lonx(i,2)
                 LONX(I,1)=INT(LONX(I,1))
        write(*,*)'lonx(i,1) = ',lonx(i,1)
                 LONX(I,3)=60.*(LONX(I,2)-INT(LONX(I,2)))
        write(*,*)'lonx(i,3) = ',lonx(i,3)
                 LONX(I,2)=INT(LONX(I,2))
        write(*,*)'lonx(i,2) = ',lonx(i,2)
                 IF (LONX(I,3).GE.60.) THEN
                    LONX(I,3)=LONX(I,3)-60.
        write(*,*)'lonx(i,3) = ',lonx(i,3)
                    LONX(I,2)=LONX(I,2)+1.
        write(*,*)'lonx(i,2) = ',lonx(i,2)
                 ENDIF
                 IF (LONX(I,2).GE.60.) THEN
                    LONX(I,2)=LONX(I,2)-60.
        write(*,*)'lonx(i,2) = ',lonx(i,2)
                    LONX(I,1)=LONX(I,1)+1.
        write(*,*)'lonx(i,1) = ',lonx(i,1)
                 ENDIF
C
C*  Convert the 4 latitudes in degrees in LATX(1-4,1) into
C*  degrees (1-4,1),minutes (1-4,2),and seconds (1-4,3).
C
                 LATX(I,2)=60.*(LATX(I,1)-INT(LATX(I,1)))
                 LATX(I,1)=INT (LATX(I,1))
                 LATX(I,3)=60.*(LATX(I,2)-INT(LATX(I,2)))
                 LATX(I,2)=INT (LATX(I,2))
                 IF (LATX(I,3).GE.60.) THEN
                    LATX(I,3)=LATX(I,3)-60.
                    LATX(I,2)=LATX(I,2)+1.
                 ENDIF
                 IF (LATX(I,2).GE.60.) THEN
                    LATX(I,2)=LATX(I,2)-60.
                    LATX(I,1)=LATX(I,1)+1.
                 ENDIF
200            CONTINUE
```

```
C
C       *  Now convert the Single Point options (OPT = 4,5) to DMS
C
          ELSE
             LONX(1,1)=ABS(PLON)
             LONX(1,2)=60.*(LONX(1,1)-INT (LONX(1,1)))
             LONX(1,1)=INT (LONX(1,1))
             LONX(1,3)=60.*(LONX(1,2)-INT (LONX(1,2)))
             LONX(1,2)=INT (LONX(1,2))
C
             LATX(1,1)=INT (PLAT)
             LATX(1,2)=60.*(PLAT-LATX(1,1))
             LATX(1,3)=60.*(LATX(1,2)-INT (LATX(1,2)))
             LATX(1,2)=INT (LATX(1,2))
C
C       *  Check for and correct overflow errors
C
             IF (LONX(1,3).GE.60.) THEN
               LONX(1,3)=LONX(1,3)-60.
               LONX(1,2)=LONX(1,2)+1.
             ENDIF
             IF (LONX(1,2).GE.60.) THEN
               LONX(1,2)=LONX(1,2)-60.
               LONX(1,1)=LONX(1,1)+1.
             ENDIF
             IF (LATX(1,3).GE.60.) THEN
                LATX(1,3)=LATX(1,3)-60.
                LATX(1,2)=LATX(1,2)+1.
             ENDIF
             IF (LATX(1,2).GE.60.) THEN
                LATX(1,2)=LATX(1,2)-60.
                LATX(1,1)=LATX(1,1)+1.
             ENDIF
          ENDIF
C
C*********************************************
C* * * * * * * * * * * * * * * * * * * * * * SUCCESS, SET STAT=1
C*********************************************
C
C       *  Success,set STAT=1.  If this is in a section that is
C       *  marked as standard (SFLAG > 0),NONSTD=0
C
575     CONTINUE
        IF(abs(SFLAG(SECT)).GT.6000)THEN
               if(stat.gt.1)GOTO 576
               if(stat.lt.0)return
        ENDIF
        STAT=1
576     CONTINUE
          IF (SFLAG(SECT).GT.0) NONSTD=0
C
C       *  Make certain the longitude is positive and return
C
          IF (OPT.LT.3) PLON=ABS(PLON)
          RETURN
C       Conclude conditional -- IF(SORCE.EQ.LEGAL) -- at beginning of
C       conversion from Legal to Geographic reference system
        ENDIF
```

```
C*
C      **  Now for the second case, SORCE=2,in which the SORCE
C      **  information is Geographic (Longitude,Latitude) and the
C      **  calling routine wants the Legal (Township/RANG/Section)
C      **  description to be returned
C
C**
C*********************************************************************
C* * * * * * * * * * * * * * * * * * * * * GEOGRAPHIC TO LEGAL CONVERSION
C*********************************************************************
C**
C
C       *  First make an educated guess on the Township.  The RANG
C       *  number here is absolute with R43W being 1 and R25E is 68
C
        TROW=(BASLAT-PLAT)/DELAT
        TROW=TROW+1
        I=(TROW-1)/30+1
        TCOL=(PLON-WSTLON(I))/DELON(I)
        TCOL=TCOL+1
        TSHP=(TROW-1)/6+1
        RANG=(TCOL-1)/6+1
        IF(RANG.GT.68)RANG=68
C
C       * Calculate the record number for the Township, check if it has
C       * already been accessed ("thrashing") and then read the record
C
        RTOP=1
600     TREC=(TSHP-1)*68+RANG
        if(trce)write(*,*)'record = ',trec
        IF (TREC.LT.0 .OR. TREC.GT.2380) THEN
           STAT = -27
           GOTO 1500
        ENDIF
        if(trce)write(*,*)'prior visits to record ',trec,' = ',vzt(trec)
        IF (VZT(TREC).LE.2) THEN
           READ(LUN,REC=TREC,ERR=1999)
     &      ((LON(I,J),LAT(I,J),J=1,7),I=1,7), TFLAG, SFLAG
           IF (TRCE) WRITE(*,*)
     &    '**  TSHP, Pseudo-RANG = ',TSHP,RANG,' township flag = ',tflag
           VZT(TREC)=VZT(TREC)+1
           VZTD(RTOP)=TREC
           RTOP=RTOP+1
C
C*  Check township for lack of data
C
           IF (TFLAG.EQ.0) THEN
              if(rang.gt.43)then
                tshp=tshp-1
                rang=rang-1
              else
                tshp=tshp+1
              endif
              goto 600
           ENDIF
C
```

```
C* Now find the two boxes for outer and inner limits of township
C
          lonomn=lon(1,1)
          lonimn=lon(1,1)
          do 601 i=2,7
             if(lon(i,1).lt.lonomn)lonomn=lon(i,1)
             if(lon(i,1).gt.lonimn)lonimn=lon(i,1)
601       continue
          latomx=lat(1,1)
          latimx=lat(1,1)
          do 602 i=2,7
             if(lat(1,i).gt.latomx)latomx=lat(1,i)
             if(lat(1,i).lt.latimx)latimx=lat(1,i)
602       continue
          lonomx=lon(1,7)
          lonimx=lon(1,7)
          do 603 i=2,7
             if(lon(i,7).gt.lonomx)lonomx=lon(i,7)
             if(lon(i,7).lt.lonimx)lonimx=lon(i,7)
603       continue
          latomn=lat(7,1)
          latimn=lat(7,1)
          do 604 i=1,7
             if(lat(7,i).lt.latomn)latomn=lat(7,i)
             if(lat(7,i).gt.latimn)latimn=lat(7,i)
604       continue
C

C       *  If this is 2nd visit, skip TSHP analysis - find section
C

          IF (VZT(TREC).GT.1) THEN
             GOTO 700
          ENDIF
C
C       *  If some corners are missing, go do section search
C
          IF (TFLAG.GT.0.AND.TFLAG.LT.36) THEN
             IF (TRCE) write(*,*)'**  partially full township  **'
             GOTO 700
          ENDIF
       ELSE
          IF (TFLAG.EQ.0) then
             STAT=-19
             GOTO 1500
          ELSE
C
C       *  Thrashing has occurred in the Township search
C
             STAT=-14
             GOTO 1500
          ENDIF
       ENDIF
C
C
```

```
C*****************************************************
C * * * * * * * * * * * * * * * * * * * * * * * * * * TOWNSHIP SEARCH
C*****************************************************
C
C
C       *  The first searches will be to locate the point as being
C       *  easily outside or inside the Township.  For this,two
C       *  boxes are used. The first is the outer box,defined as
C       *  the minimum bounding rectangle defined by the minimum
C       *  and maximum coordinates of the 4 corners.  The other box
C       *  is the inner box,defined by the coordinates not used
C       *  for the outer box.  If the point is outside the outer
C       *  box,then it is not in the Township.  Likewise,if it is
C       *  inside the inner box,it is inside the Township.
C***
C*************************************
C * * * * * * * * * * * * * * * * * * * * SEARCH INTERIOR AND EXTERIOR
C * * * * * * * * * * * * * * * * * * * * * RECTANGLES (EASY SEARCH)
C*************************************
C
C
C       **  Look for point outside the outer box.  If found,reset
C       **  TSHP and/or RANG to search the next township
C
        OUTFLAG=0
        IF (PLON.GT.LONOMX) THEN
           OUTFLAG=1
c               point is out to the east
           RANG=RANG+1
        ELSEIF (PLON.LT.LONOMN) THEN
           OUTFLAG=1
c               point is out to the west
           RANG=RANG-1
        ENDIF
        IF (PLAT.GT.LATOMX) THEN
           OUTFLAG=1
c               point is out to the north
           TSHP=TSHP-1
        ELSEIF (PLAT.LT.LATOMN) THEN
           OUTFLAG=1
c               point is out to the south
           TSHP=TSHP+1
        ENDIF
        if(TRCE .AND. outflag.ne.0)then
           write(*,*)'++ outerbox at: ',lonomn,latomn,lonomx,latomx
           write(*,*)'    plon, plat = ',plon,plat
           write(*,*)'** outside t-box, new TSHP,pseudo-RANG= ',
     &          TSHP,RANG,' **'
        endif
        IF (OUTFLAG.NE.0) GOTO 600
C
```

```
C       *  Check the inner box for points easily inside.  When found,
C       *  go to search for the section within the township
C
C       *  OUTFLAG=0
C
        IF (PLON.LT.LONIMN) THEN
           OUTFLAG=1
        ELSEIF (PLON.GT.LONIMX) THEN
           OUTFLAG=4
        ENDIF
        IF (PLAT.LT.LATIMN) THEN
           OUTFLAG=OUTFLAG+2
        ELSEIF (PLAT.GT.LATIMX) THEN
           OUTFLAG=OUTFLAG+8
        ENDIF
        IF (OUTFLAG.LT.1) THEN
           IF (TRCE) write (*,*) '**  inside TSHP box  **'
           GOTO 700
        ENDIF
C
C*
C       **  These points were not eliminated by the easy method,so
C       **  check the point against the four sides as follows.  If the
C       **  point is outside the endpoints of the side indicated by
C       **  the above evaluation,then send it off to another Township.
C       **  Otherwise,find the intersection point of a vertical or
C       **  horizontal line through the point with the side.  If this
C       **  intersection point is inside the point,then the point is
C       **  outside the Township.
C
```

```
C****************************************
C * * * * * * * * * * * * * * * * * * * * *SEARCH BETWEEN INTERIOR AND
C* * * * * * * * * * * * * * * * * * * * * EXTERIOR RECTANGLES (COMPLEX)
C****************************************
C
        IF (TRCE) write (*,*) '**  fancy TSHP tests  **'
        if (trce) write (*,*) '* outflag = ',outflag
        IF (OUTFLAG.GT.7) THEN
C
C       **  Is the point east or west of the north side?
C
           OUTFLAG=OUTFLAG-8
           IF (PLON.LT.LON(1,1)) THEN
              TSHP=TSHP-1
              RANG=RANG-1
              GOTO 600
           ELSEIF (PLON.GT.LON(1,7)) THEN
              TSHP=TSHP-1
              RANG=RANG+1
              GOTO 600
           ENDIF
C
C       **  Find intersection of vertical through point with north side
C             and test for N/S of north side
C
           IF(PLON.LT.LON(1,2))THEN
                I=1
           ELSEIF(PLON.LT.LON(1,3))THEN
                I=2
           ELSEIF(PLON.LT.LON(1,4))THEN
                I=3
           ELSEIF(PLON.LT.LON(1,5))THEN
                I=4
           ELSEIF(PLON.LT.LON(1,6))THEN
                I=5
           ELSEIF(PLON.LE.LON(1,7))THEN
                I=6
           ENDIF
           J=I+1
           DLAT=(LAT(1,J)-LAT(1,I))*(PLON-LON(1,I))
           DLAT=DLAT/(LON(1,J)-LON(1,I))
           XLAT=LAT(1,I)+DLAT
           IF (PLAT.GT.XLAT) THEN
              TSHP=TSHP-1
              GOTO 600
           ELSE
              GOTO 700
           ENDIF
        ENDIF
```

```
      IF (OUTFLAG.GT.3) THEN
C
C       **  Now for the east side tests
C
          OUTFLAG=OUTFLAG-4
C
C       **  Is the point above or below the east side?
C
          IF (PLAT.GT.LAT(1,7)) THEN
             TSHP=TSHP-1
             RANG=RANG+1
             GOTO 600
          ELSEIF (PLAT.LT.LAT(7,7)) THEN
             TSHP=TSHP+1
             RANG=RANG+1
             GOTO 600
          ENDIF
C
C       **  No, find intersection of horizontal through point with
C             east side and test E/W of east
C
          IF(PLAT.GT.LAT(2,7))THEN
             I=1
          ELSEIF(PLAT.GT.LAT(3,7))THEN
             I=2
          ELSEIF(PLAT.GT.LAT(4,7))THEN
             I=3
          ELSEIF(PLAT.GT.LAT(5,7))THEN
             I=4
          ELSEIF(PLAT.GT.LAT(6,7))THEN
             I=5
          ELSEIF(PLAT.GE.LAT(7,7))THEN
             I=6
          ENDIF
          J=I+1
          DLON=(LON(J,7)-LON(I,7))*(PLAT-LAT(I,7))
          DLON=DLON/(LAT(J,7)-LAT(I,7))
          XLON=LON(I,7)+DLON
          IF (PLON.GT.XLON) THEN
             RANG=RANG+1
             GOTO 600
          ELSE
             GOTO 700
          ENDIF
      ENDIF
```

```
      IF (OUTFLAG.GT.1) THEN
C
C       **  Now testing for the south side
C
          OUTFLAG=OUTFLAG-2
C
C       **  Is point east or west of the south side?
C
          IF (PLON.LT.LON(7,1)) THEN
             TSHP=TSHP+1
             RANG=RANG-1
             GOTO 600
          ELSEIF (PLON.GT.LON(7,7)) THEN
             TSHP=TSHP+1
             RANG=RANG+1
             GOTO 600
          ENDIF
C
C       **  Find intersection of vertical through point with south
C             side and test N/S of south side
C
          IF(PLON.LT.LON(7,2))THEN
               I=1
          ELSEIF(PLON.LT.LON(7,3))THEN
               I=2
          ELSEIF(PLON.LT.LON(7,4))THEN
               I=3
          ELSEIF(PLON.LT.LON(7,5))THEN
               I=4
          ELSEIF(PLON.LT.LON(7,6))THEN
               I=5
          ELSEIF(PLON.LE.LON(7,7))THEN
               I=6
          ENDIF
          J=I+1
          DLAT=(LAT(7,j)-LAT(7,i))*(PLON-LON(7,i))
          DLAT=DLAT/(LON(7,j)-LON(7,i))
          XLAT=LAT(7,1)+DLAT
          IF (PLAT.LT.XLAT) THEN
             TSHP=TSHP+1
             GOTO 600
          ELSE
             GOTO 700
          ENDIF
      ENDIF
```

```
      IF (OUTFLAG.GT.0) THEN
C
C     **  Test the West side
C
         OUTFLAG=0
C
C     **  Is point north or south of the west side?
C
         IF (PLAT.GT.LAT(1,1)) THEN
            TSHP=TSHP-1
            RANG=RANG-1
            GOTO 600
         ELSEIF (PLAT.LT.LAT(7,1)) THEN
            TSHP=TSHP+1
            RANG=RANG-1
            GOTO 600
         ENDIF
C
C     **  Find intersection of horizontal through point with west
C            side and check E/W of west side
C
         IF(PLAT.GT.LAT(2,1))THEN
            I=1
         ELSEIF(PLAT.GT.LAT(3,1))THEN
            I=2
         ELSEIF(PLAT.GT.LAT(4,1))THEN
            I=3
         ELSEIF(PLAT.GT.LAT(5,1))THEN
            I=4
         ELSEIF(PLAT.GT.LAT(6,1))THEN
            I=5
         ELSEIF(PLAT.GE.LAT(7,1))THEN
            I=6
         ENDIF
         J=I+1
         DLON=(LON(J,1)-LON(I,1))*(PLAT-LAT(I,1))
         DLON=DLON/(LAT(J,1)-LAT(I,1))
         XLON=LON(I,1)+DLON
         IF (PLON.LT.XLON) THEN
            RANG=RANG-1
            GOTO 600
         ENDIF
      ENDIF
C
C
```

```
C************************************
C * * * * * * * * * * * * * * * * * * TOWNSHIP FOUND, SEARCH FOR SECTION
C************************************
C
C***
C       The point is (probably) inside this township, at least
C       according to the four township corners.  The actual
C       section boundaries may not always agree.  Anyway,now
C       the most likely section is selected according to world
C       knowledge of the size of sections in various parts of
C       the state.  The section is searched in the thorough
C       manner as the townships were.  If the point is found to
C       lie outside the section,a new target section is tried.
C       In this process,it is possible to move to a different
C       target township.  When the correct section is found,
C       the subdivisions are generated for the complete legal
C       description and the task is complete.
C***
C
C       *  Zero the search mask (SEARCH) to indicate that no sections
C       *  have been searched for this township.  As they
C       *  are searched, the value is set to 1 to prevent thrashing.
C
700     DO 725 I=1,6
           DO 725 J=1,6
              SEARCH(I,J)=0
725     CONTINUE
```

```
        IF (TFLAG.EQ.36) THEN
C
C       **  Now to select the initial section for the search
C
C       **  For full townships calculate the row and column
C       **  of the NW corner of the most likely section,
C       **  based on the spatial relationships of the corners
C
           if(trce)write(*,*)'latomx = ',latomx
           if(trce)write(*,*)'latomn = ',latomn
           if(trce)write(*,*)'PLAT = ',plat
                OUTFLAG=0
                IF (PLON.GT.LONOMX) THEN
                   OUTFLAG=1
c                  point is out to the east
                   RANG=RANG+1
                ELSEIF (PLON.LT.LONOMN) THEN
                   OUTFLAG=1
c                  point is out to the west
                   RANG=RANG-1
                ENDIF
                IF (PLAT.GT.LATOMX) THEN
                   OUTFLAG=1
c                  point is out to the north
                   TSHP=TSHP-1
                ELSEIF (PLAT.LT.LATOMN) THEN
                   OUTFLAG=1
c                  point is out to the south
                   TSHP=TSHP+1
                ENDIF
                if(TRCE .AND. outflag.ne.0)then
                   write(*,*)'** outside t-box, new TSHP,pseudo-RANG= ',
     &                 TSHP,RANG,' **'
                endif
                IF (OUTFLAG.NE.0) GOTO 600
           SROW=6*(LATomx-PLAT)/(LATomx-LATomn)
           SCOL=6*(PLON-LONomn)/(LONomx-LONomn)
C
c          SROW=6*(LAT(1,1)-PLAT)/(LAT(1,1)-LAT(7,1))
c          SCOL=6*(PLON-LON(1,1))/(LON(1,7)-LON(1,1))
           SROW=SROW+1
           SCOL=SCOL+1
           IF (TRCE) write(*,*)'** initial srow, scol = ',srow,scol
        ELSE
C
C       **  For townships with fewer than 36 complete sections
C       **  all sections will be checked if necessary.  Start
C       **  in the NW corner (section 6) and search sections
C       **  that have 4 corners until a section contains the
C       **  point.  If none does, end with an error.
C
           SROW = 1
           SCOL = 1
           GOTO 775
        ENDIF
C
```

```
C       *  Make sure the section is in this township.  This is more
C       *  important when recycling for a neighboring section.  (If
C       *  necessary,go search the neighboring township.)
C
750    OUTFLAG=0
       IF (SROW.LT.1) THEN
          if(tflag.lt.36)then
               srow=1
                 IF (TRCE) write(*,*)'search(',srow,scol,') = ',
     &             search(srow,scol)
               if(search(srow,scol).ge.1)then
                  goto 751
               endif
C
C              *  Calculate the section number from row and column
C              *  Check for missing corner values via the section flag
C
               IF (MOD (INT (SROW),2).EQ.0) THEN
                  SECT=6*(SROW-1)+SCOL
               ELSE
                  SECT=6*SROW+1-SCOL
               ENDIF
               IF (TRCE) write(*,*)'-- section = ',sect,
     &             '   sflag = ',sflag(sect)
               if(sflag(sect).eq.0)then
                  goto 751
               endif
               goto 775
          endif
751       continue
          OUTFLAG=1
          TSHP=TSHP-1
       ELSEIF (SROW.GT.6) THEN
          if(tflag.lt.36)then
               srow=6
                 IF (TRCE) write(*,*)'search(',srow,scol,') = ',
     &             search(srow,scol)
               if(search(srow,scol).ge.1)then
                  goto 752
               endif
C              *  Calculate the section number from row and column
C              *  Check for missing corner values via the section flag
               IF (MOD (INT (SROW),2).EQ.0) THEN
                  SECT=6*(SROW-1)+SCOL
               ELSE
                  SECT=6*SROW+1-SCOL
               ENDIF
               IF (TRCE) write(*,*)'-- section = ',sect,
     &             '   sflag = ',sflag(sect)
               if(sflag(sect).eq.0)then
                  goto 752
               endif
               goto 775
          endif
752       continue
          OUTFLAG=1
          TSHP=TSHP+1
       ENDIF
```

```
        IF (SCOL.LT.1) THEN
           if(tflag.lt.36)then
                scol=1
                  IF (TRCE) write(*,*)'search(',srow,scol,') = ',
     &             search(srow,scol)
                if(search(srow,scol).ge.1)then
                   goto 753
                endif
C
C               *  Calculate the section number from row and column
C               *  Check for missing corner values via the section flag
C
                IF (MOD (INT (SROW),2).EQ.0) THEN
                   SECT=6*(SROW-1)+SCOL
                ELSE
                   SECT=6*SROW+1-SCOL
                ENDIF
                IF (TRCE) write(*,*)'-- section = ',sect,
     &             '   sflag = ',sflag(sect)
                if(sflag(sect).eq.0)then
                   goto 753
                endif
                goto 775
           endif
753        continue
           OUTFLAG=1
           RANG=RANG-1
        ELSEIF (SCOL.GT.6) THEN
           if(tflag.lt.36)then
                scol=6
                if(search(srow,scol).ge.1)goto 754
C
C               *  Calculate the section number from row and column
C               *  Check for missing corner values via the section flag
C
                IF (MOD (INT (SROW),2).EQ.0) THEN
                   SECT=6*(SROW-1)+SCOL
                ELSE
                   SECT=6*SROW+1-SCOL
                ENDIF
                IF (TRCE) write(*,*)'-- section = ',sect,
     &             '   sflag = ',sflag(sect)
                if(sflag(sect).eq.0)then
                   goto 754
                endif
                goto 775
           endif
754        continue
           OUTFLAG=1
           RANG=RANG+1
        ENDIF
```

```
      IF (OUTFLAG.GT.0) THEN
         IF (TRCE) write(*,*)'range horseshoe (40ft) = ',hshur40
         if(rang.lt.1.or.rang.gt.68)then
              if(hshur40.eq.0)then
                 hshur40=1
                 if(rang.lt.1)then
                 plon=plon+lon40
                 rang=1
                 else
                 plon=plon-lon40
                 rang=68
                 endif
              elseif(hshur40.eq.1)then
                 hshur1c=1
                 if(rang.lt.1)then
                 plon=plon+lon100-lon40
                 rang=1
                 else
                 plon=plon-lon100+lon40
                 rang=68
                 endif
              else
                 stat=-10
                 return
              endif
         endif
         if(tshp.lt.1.or.tshp.gt.35)then
              if(hshut40.eq.0)then
                 hshut40=1
                 if(tshp.lt.1)then
                 plat=plat-lat40
                 tshp=1
                 else
                 plat=plat+lat40
                 tshp=35
                 endif
              elseif(hshut40.eq.1)then
                 hshut1c=1
                 if(tshp.lt.1)then
                 plat=plat-lat100+lat40
                 tshp=1
                 else
                 plat=plat+lat100-lat40
                 tshp=35
                 endif
              else
                 stat=-11
                 return
              endif
         endif
         OUTFLAG=0
         GOTO 600
      ENDIF
C
```

```
C       *  Set pointers to corners of section. If section has not
C       *  been searched,check for unknown values in LEO3BASE.
C
775     N=SROW
        S=SROW+1
        W=SCOL
        E=SCOL+1
C
C       **  If section has already been searched-a thrashing error.
C       **  Otherwise,mark the section as having been searched.
C
        IF (TFLAG.EQ.36 .AND. SEARCH(N,W).NE.0) THEN
           STAT=-15
           GOTO 1500
        ELSE
           SEARCH(N,W)=SEARCH(N,W)+1
        ENDIF
C
C       *  Calculate the section number from row and column
C       *  Check for missing corner values via the section flag
C
        IF (MOD (INT (SROW),2).EQ.0) THEN
           SECT=6*(SROW-1)+SCOL
        ELSE
           SECT=6*SROW+1-SCOL
        ENDIF
        IF (TRCE) write(*,*)
     &          '## section = ',sect,'   sflag = ',sflag(sect)
        IF (TRCE) WRITE (*,*) '* * *  section = ', sect
C
C       *  For partial townships, just proceed through the sections
C
        IF (TFLAG.GT.0.and.TFLAG.lt.36) THEN
           if(search(n,w).gt.1)then
                IF (SFLAG(SECT).eq.0) THEN
                 STAT=-32
                 GOTO 1500
                ENDIF
           endif
C
18              CONTINUE
C
C       *  Sections with 4 corners may be searched
C
           IF (SFLAG(SECT).GT.1110) then
                GOTO 780
           elseif (SFLAG(sect).lt.-1110)then
C
C       *   ** more program changes required here when fully implemented
C
                goto 780
           endif
C
```

```
C       *  If section flag is in (-1110,1110), Try the next section
C
           SCOL = SCOL + 1
           IF (SCOL.GT.6) THEN
              SCOL = 1
              SROW = SROW + 1
              IF (SROW.GT.6) THEN
                if(hshut40.eq.0)goto 750
                 STAT = -28
c                write(*,*)'status = -28, section flag = ',sflag(sect)
                 GOTO 1500
              ENDIF
           ENDIF
           GOTO 775
        ENDIF
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * SIMPLE SECTION SEARCH
C
C*********  Create Inner and Outer boxes for simple cases
C
780     continue
        delrow=0
        IF (LON(N,W).LT.LON(S,W)) THEN
           LONOMN=LON(N,W)
           LONIMN=LON(S,W)
        ELSE
           LONOMN=LON(S,W)
           LONIMN=LON(N,W)
        ENDIF
        IF (LAT(N,W).GT.LAT(N,E)) THEN
           LATOMX=LAT(N,W)
           LATIMX=LAT(N,E)
        ELSE
           LATOMX=LAT(N,E)
           LATIMX=LAT(N,W)
        ENDIF
        IF (LON(N,E).GT.LON(S,E)) THEN
           LONOMX=LON(N,E)
           LONIMX=LON(S,E)
        ELSE
           LONOMX=LON(S,E)
           LONIMX=LON(N,E)
        ENDIF
        IF (LAT(S,E).LT.LAT(S,W)) THEN
           LATOMN=LAT(S,E)
           LATIMN=LAT(S,W)
        ELSE
           LATOMN=LAT(S,W)
           LATIMN=LAT(S,E)
        ENDIF
C
```

```
C       *  Look for point outside the outer box.  If found, reset
C       *  the SROW and/or SCOL pointers for the next search.
C
        OUTFLAG=0
        IF (PLON.GT.LONOMX) THEN
           OUTFLAG=1
           if(TRCE) write(*,*)
     &          '*    outside section box column, scol = ',scol,' *'
           SCOL=SCOL+1
        ELSEIF (PLON.LT.LONOMN) THEN
           OUTFLAG=1
           if(TRCE) write(*,*)
     &          '**   outside section box column, scol = ',scol,' *'
           SCOL=SCOL-1
        ENDIF
        IF (PLAT.GT.LATOMX) THEN
           OUTFLAG=1
           if(TRCE) write(*,*)
     &          '***  outside section box row,    srow = ',srow,' *'
           SROW=SROW-1
           delrow=-1
        ELSEIF (PLAT.LT.LATOMN) THEN
           OUTFLAG=1
           if(TRCE) write(*,*)
     &          '**** outside section box row,    srow = ',srow,' *'
           SROW=SROW+1
           delrow=1
        ENDIF
C
C       *  If point is out (OUTFLAG=1),go for next section
C
        IF (OUTFLAG.NE.0) THEN
           if(TRCE) write(*,*)
     &          '**********          next srow, scol= ',srow,scol,' *'
           GOTO 750
        ENDIF
C
C       **  Check the inner box for points easily inside
C
        OUTFLAG=0
        IF (PLON.LT.LONIMN) THEN
           OUTFLAG=1
        ELSEIF (PLON.GT.LONIMX) THEN
           OUTFLAG=4
        ENDIF
        IF (PLAT.LT.LATIMN) THEN
           OUTFLAG=OUTFLAG+2
        ELSEIF (PLAT.GT.LATIMX) THEN
           OUTFLAG=OUTFLAG+8
        ENDIF

        IF (OUTFLAG.LT.1) THEN
           if (TRCE) write (*,*) '* * *  inside section box  * * *'
           GOTO 800
        ENDIF
C
```

```
C*****************************************
C * * * * * * * * * * * * * * * * * * * * (COMPLEX) SEARCH BETWEEN
C* * * * * * * * * * * * * * * * * * * * * * INNER AND OUTER BOXES
C*****************************************
C*
C       *  These points were not eliminated by the easy method,so
C       *  check the point against the four sides as follows.  If the
C       *  point is outside the endpoints of the side indicated by
C       *  the above evaluation,then send it off to another Township.
C       *  Otherwise,find the intersection point of a vertical or
C       *  horizontal line through the point with the side.  If this
C       *  intersection point is inside the point,then the point is
C       *  outside the Township.
C*
        IF (TRCE) WRITE (*,*)
     &           '* * fancy section tests, outflag= ',outflag,' * *'
        IF (OUTFLAG.GT.7) THEN
C
C       **  Here to test those above the north side of inner box
C       *****  If west of west end or east of east end of the north
C              side then shift to the appropriate diagonal section
C
           OUTFLAG=OUTFLAG-8
           IF (PLON.LT.LON(N,W)) THEN
              SROW=SROW-1
              SCOL=SCOL-1
                if(plat.lt.lat(n,w))srow=srow+1
              GOTO 750
           ELSEIF (PLON.GT.LON(N,E)) THEN
              SROW=SROW-1
              SCOL=SCOL+1
                if(plat.lt.lat(n,e))srow=srow+1
              GOTO 750
           ENDIF
C
C       *  Find intersection of vertical through point with north side
C
           DLAT=(LAT(N,E)-LAT(N,W))*(PLON-LON(N,W))
           DLAT=DLAT/(LON(N,E)-LON(N,W))
           XLAT=LAT(N,W)+DLAT
           if(trce)write(*,*)'north latitude = ', xlat
C
C       *  NOTE:  If there is a correction line along this side of
C                 the section, errors could result from use of only
C                 the four corners of one section at a time.  If the
C                 corners from sections on both sides of the line are
C                 not in perfect alignment there will be slivers
C                 of land which appear to be in both sections or in
C                 neither section.  The result could be erroneous
C                 assignment or thrashing without solution.
C
           IF (PLAT.GT.XLAT) THEN
              SROW=SROW-1
              GOTO 750
           ELSE
              GOTO 800
           ENDIF
        ENDIF
```

```
      IF (OUTFLAG.GT.3) THEN
C
C       **  Now for the east side tests
C
          OUTFLAG=OUTFLAG-4
C
C       *  Is the point above or below the side?
C
          IF (PLAT.GT.LAT(N,E)) THEN
             SROW=SROW-1
             SCOL=SCOL+1
              if(plon.lt.lon(n,e))scol=scol-1
             GOTO 750
          ELSEIF (PLAT.LT.LAT(S,E)) THEN
             SROW=SROW+1
             SCOL=SCOL+1
              if(plon.lt.lon(s,e))scol=scol-1
             GOTO 750
          ENDIF
C
C       *  No, find the intersection with horizontal
C
          DLON=(LON(N,E)-LON(S,E))*(PLAT-LAT(S,E))
          DLON=DLON/(LAT(N,E)-LAT(S,E))
          XLON=LON(S,E)+DLON
          IF (PLON.GT.XLON) THEN
             SCOL=SCOL+1
             GOTO 750
          ELSE
             GOTO 800
          ENDIF
      ENDIF
```

```
        IF (OUTFLAG.GT.1) THEN
C
C       **  Now testing for the south side
C
           OUTFLAG=OUTFLAG-2
C
C       *  Is the point outside the south side?
C
           IF (PLON.LT.LON(S,W)) THEN
              SROW=SROW+1
              SCOL=SCOL-1
                if(plat.gt.lat(s,w))srow=srow-1
              GOTO 750
           ELSEIF (PLON.GT.LON(S,E)) THEN
              SROW=SROW+1
              SCOL=SCOL+1
                if(plat.gt.lat(s,e))srow=srow-1
              GOTO 750
           ENDIF
C
C       *  Calculate intersection point with vertical and test
C
           DLAT=(LAT(S,E)-LAT(S,W))*(PLON-LON(S,W))
           DLAT=DLAT/(LON(S,E)-LON(S,W))
           XLAT=LAT(S,W)+DLAT
           if(trce)write(*,*)'south latitude = ',xlat
           IF (PLAT.LT.XLAT) THEN
              SROW=SROW+1
              GOTO 750
           ELSE
              GOTO 800
           ENDIF
        ENDIF
```

```
        IF (OUTFLAG.GT.0) THEN
C
C       **  Test the West side
C
           OUTFLAG=0
C
C       *  Is point outside the segment?
C
           IF (PLAT.GT.LAT(N,W)) THEN
              SROW=SROW-1
              SCOL=SCOL-1
                if(plon.gt.lon(n,w))scol=scol+1
              GOTO 750
           ELSEIF (PLAT.LT.LAT(S,W)) THEN
              SROW=SROW+1
              SCOL=SCOL-1
                if(plon.gt.lon(s,w))scol=scol+1
              GOTO 750
           ENDIF
C
C       *  Find horizontal intersection and check
C
           DLON=(LON(N,W)-LON(S,W))*(PLAT-LAT(S,W))
           DLON=DLON/(LAT(N,W)-LAT(S,W))
           XLON=LON(S,W)+DLON
           IF (PLON.LT.XLON) THEN
              SCOL=SCOL-1
              GOTO 750
           ENDIF
        ENDIF
C
C
C************************************
C * * * * * * * * * * * * * * * * * * * * POINT FOUND - SET SUBDIVISIONS
C* * * * * * * * * * * * * * * * * * * * AND CLOSEST OF 9 POINTS (CORNR)
C************************************
C
C
C       The point is inside this section!!!
C
C       Apply up to four subdivisions and exit.
C***
C
C       **  Get coordinates of corners for subdivision
C
800     LONSW=LON(S,W)
        LATSW=LAT(S,W)
        LONSE=LON(S,E)
        LATSE=LAT(S,E)
        LONNE=LON(N,E)
        LATNE=LAT(N,E)
        LONNW=LON(N,W)
        LATNW=LAT(N,W)
        IF (TRCE) WRITE (*,*) '* * *  point inside section !!  * * *'
C
```

```
C*************
C       *  Subdivide each rectangle according to the quarter that
C       *  the point lies in (NW,SW,SE,NE) once for each of the
C       *  four subdivisions that are set along the way (SUBD).
C*************
C
       LEVEL=1
C
C       *  The program has determined that the given geographic
C       *  coordinates are within the polygon defined by the section
C       *  corner locations in LEO3BASE.  for partial sections along
C       *  the Missouri, Oklahoma and Colorado borders, correct
C       *  evaluation of subdivision location for legal reference
C       *  requires the construction of false corners which describe
C       *  a full section including the portion in Kansas and a portion
C       *  extending into the adjoining state.
C       *  In the case of footage descriptions, corners of partial
C       *  sections where section lines terminate on the Missouri,
C       *  Oklahoma or Colorado borders (excluding partial sections
C       *  along the Missouri River) are used as corners of origin
C       *  for the footage measurements.
C
       if(OPTION2.eq.0)then
           IF (abs(SFLAG(SECT)).GT.6000)THEN
              CALL STATE_EDGE(SFLAG(SECT),LONSW,LATSW,
     &            LONSE,LATSE,LONNW,LATNW,LONNE,LATNE)
           ENDIF
       endif
C
C*  Find side midpoint and center locations
C
1000   LONN=(LONNW+LONNE)/2.
       LATN=(LATNW+LATNE)/2.
       LONE=(LONNE+LONSE)/2.
       LATE=(LATNE+LATSE)/2.
       LONS=(LONSE+LONSW)/2.
       LATS=(LATSE+LATSW)/2.
       LONW=(LONSW+LONNW)/2.
       LATW=(LATSW+LATNW)/2.
       CTRLON=(LONSW+LONSE+LONNE+LONNW)/4.
       CTRLAT=(LATSW+LATSE+LATNE+LATNW)/4.
       if(OPTION2.eq.1.or.OPTION2.eq.2.OR.LEVEL.GT.maxlvl)goto 1100
C
```

```
C       *  Closest Point Option:
C
C       *  To see where point is,find longitude of intersection of
C       *  horizontal line through point with line through north and
C       *  south midpoints and latitude of intersection of vertical
C       *  line through point with line through east & west midpoints
C
        LONINT=LONN-(LONN-LONS)*(LATN-PLAT)/(LATN-LATS)
        LATINT=LATW+(LATE-LATW)*(PLON-LONW)/(LONE-LONW)
C
C       *  Set the 1/4-area subdivision for the current level (LEVEL)
C
        IF (PLON.GT.LONINT) THEN
           IF (PLAT.GT.LATINT) THEN
              SUBD(LEVEL)='NE'
              LONSW=CTRLON
              LATSW=CTRLAT
              LONNW=LONN
              LATNW=LATN
              LONSE=LONE
              LATSE=LATE
           ELSE
              SUBD(LEVEL)='SE'
              LONSW=LONS
              LATSW=LATS
              LONNW=CTRLON
              LATNW=CTRLAT
              LONNE=LONE
              LATNE=LATE
           ENDIF
        ELSEIF (PLAT.GT.LATINT) THEN
           SUBD(LEVEL)='NW'
           LONNE=LONN
           LATNE=LATN
           LONSE=CTRLON
           LATSE=CTRLAT
           LONSW=LONW
           LATSW=LATW
        ELSE
           SUBD(LEVEL)='SW'
           LONNE=CTRLON
           LATNE=CTRLAT
           LONSE=LONS
           LATSE=LATS
           LONNW=LONW
           LATNW=LATW
        ENDIF
C
C       *  Cycle until maxlvl (up to 4) subdivisions are set
C
        IF (LEVEL.LT.5) THEN
           LEVEL=LEVEL+1
           GOTO 1000
        ENDIF
C
```

```
C*************
C       **  Set CORNR value to indicate closest of nine points
C*************
C
1100    CONTINUE
C
C*  Check for protracted corners: footage OPTION should return
C*  measurements from nearest SURVEYED corner (not protracted).
C
                IF (SFLAG(SECT).EQ.0) THEN
                   STAT=-32
                   RETURN
                ELSE
                   ALPHA=ABS(SFLAG(SECT))
                   SCOR(1)=INT(ALPHA/1000)
                   BETA=ALPHA-1000*SCOR(1)
                   SCOR(2)=INT(BETA/100)
                   GAMMA=BETA-100*SCOR(2)
                   SCOR(3)=INT(GAMMA/10)
                   OMEGA=GAMMA-10*SCOR(3)
                   SCOR(4)=OMEGA
C
                   IF(SCOR(1).GT.5)STLINE=1
                   IF(SCOR(2).GT.5)RIVER=1
                   IF(SCOR(3).GT.5)RESERVE=1
                   IF(SCOR(4).GT.5)TREATY=1
C
                   do 1101 i=1,4
                         IF(SCOR(i).GT.5)SCOR(i)=SCOR(i)-5
1101               continue
                ENDIF
C       write(*,*)'corner flags = ',scor
c
c*  For OPTION = footage from nearest surveyed corner, consider
c*  termination of section lines at state border (other than
c*  along the Missouri River) to be surveyed corners.
c
                if(abs(sflag(sect)).gt.6000)then
                   do 1104 i=1,4
                         if(scor(i).eq.3)scor(i)=1
1104               continue
                endif
c
c*  determine if there are no surveyed corners available for footage
c*  OPTION, in which case footage to nearest protracted corner will
c*  be returned.
c
                surv=0
                do 1105 i=1,4
                   if(scor(i).eq.1.or.scor(i).eq.4)surv=1
1105            continue
C
C
```

```
C* * * * * PROJECT CORNERS, CENTER AND MID-SIDES TO FIND CLOSEST POINT
C
C
              SOUTH=AMIN1 (LATSW,LATSE)
              NORTH=AMAX1 (LATNW,LATNE)
              WEST=AMIN1 (LONSW,LONNW)
              EAST=AMAX1 (LONSE,LONNE)
C
C*  Initialize the (MP) projection
C
              CALL LEO3SETP (CORN)
C
C*  Project point, then four corners, center and mid-sides.
C*  Calculate distance from point to each section location.
C*  Determine closest point in subdivision and footage to nearest
C*  section corner
C
              DBLON=DBLE (PLON)
              DBLAT=DBLE (PLAT)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XP=SNGL (XDBL)
              YP=SNGL (YDBL)
C
              DBLON=DBLE (LONNE)
              DBLAT=DBLE (LATNE)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XNE=SNGL (XDBL)
              YNE=SNGL (YDBL)
C
                DNE=SQRT((XNE-XP)**2+(YNE-YP)**2)
                DMIN=DNE
c       write(*,*)'dne = ',dne
                PMIN='NE'
                if(surv.eq.0.or.scor(1).eq.1.or.scor(1).eq.4)then
                   DCMIN=DNE
                   CMIN='NE'
                else
                   DCMIN=15000
                endif
C
              DBLON=DBLE (LONN)
              DBLAT=DBLE (LATN)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XN=SNGL (XDBL)
              YN=SNGL (YDBL)
C
                DN=SQRT((XN-XP)**2+(YN-YP)**2)
c       write(*,*)'dn = ',dn
                IF(DN.LT.DMIN)then
                   DMIN=DN
                   PMIN='N '
                ENDIF
C
              DBLON=DBLE (LONNW)
              DBLAT=DBLE (LATNW)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XNW=SNGL (XDBL)
              YNW=SNGL (YDBL)
```

```
C
              DNW=SQRT((XNW-XP)**2+(YNW-YP)**2)
c       write(*,*)'dnw = ',dnw
              IF(DNW.LT.DMIN)then
                 DMIN=DNW
                 PMIN='NW'
              ENDIF
              if(surv.eq.0.or.scor(2).eq.1.or.scor(2).eq.4)then
                 IF(DNW.LT.DCMIN)THEN
                       DCMIN=DNW
                       CMIN='NW'
                 ENDIF
              endif
C
            DBLON=DBLE (LONW)
            DBLAT=DBLE (LATW)
            CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
            XW=SNGL (XDBL)
            YW=SNGL (YDBL)
C
              DW=SQRT((XW-XP)**2+(YW-YP)**2)
c       write(*,*)'dw = ',dw
              IF(DW.LT.DMIN)then
                 DMIN=DW
                 PMIN='W '
              ENDIF
C
            DBLON=DBLE (LONSW)
            DBLAT=DBLE (LATSW)
            CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
            XSW=SNGL (XDBL)
            YSW=SNGL (YDBL)
C
              DSW=SQRT((XSW-XP)**2+(YSW-YP)**2)
c       write(*,*)'dsw = ',dsw
              IF(DSW.LT.DMIN)then
                 DMIN=DSW
                 PMIN='SW'
              ENDIF
              if(surv.eq.0.or.scor(3).eq.1.or.scor(3).eq.4)then
                 IF(DSW.LT.DCMIN)THEN
                       DCMIN=DSW
                       CMIN='SW'
                 ENDIF
              endif
C
            DBLON=DBLE (LONS)
            DBLAT=DBLE (LATS)
            CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
            XS=SNGL (XDBL)
            YS=SNGL (YDBL)
C
              DS=SQRT((XS-XP)**2+(YS-YP)**2)
c       write(*,*)'ds = ',ds
              IF(DS.LT.DMIN)then
                 DMIN=DS
                 PMIN='S '
              ENDIF
```

```
C
              DBLON=DBLE (LONSE)
              DBLAT=DBLE (LATSE)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XSE=SNGL (XDBL)
              YSE=SNGL (YDBL)
C
                DSE=SQRT((XSE-XP)**2+(YSE-YP)**2)
c       write(*,*)'dse = ',dse
                IF(DSE.LT.DMIN)then
                   DMIN=DSE
                   PMIN='SE'
                ENDIF
                if(surv.eq.0.or.scor(4).eq.1.or.scor(4).eq.4)then
                   IF(DSE.LT.DCMIN)THEN
                         DCMIN=DSE
                         CMIN='SE'
                   ENDIF
                endif
C
              DBLON=DBLE (LONE)
              DBLAT=DBLE (LATE)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XE=SNGL (XDBL)
              YE=SNGL (YDBL)
C
                DE=SQRT((XE-XP)**2+(YE-YP)**2)
c       write(*,*)'de = ',de
                IF(DE.LT.DMIN)then
                   DMIN=DE
                   PMIN='E '
                ENDIF
C
              DBLON=DBLE (CTRLON)
              DBLAT=DBLE (CTRLAT)
              CALL LEO3PRO (DBLON,DBLAT,XDBL,YDBL)
              XCTR=SNGL (XDBL)
              YCTR=SNGL (YDBL)
C
                DCTR=SQRT((XCTR-XP)**2+(YCTR-YP)**2)
c       write(*,*)'dctr = ',dctr
                IF(DCTR.LT.DMIN)then
                   DMIN=DCTR
                   PMIN='C '
                ENDIF
C
C*******
C
```

```
      if(OPTION2.eq.0)then
              cornr=pmin
      elseif(OPTION2.eq.1)then
              cornr=cmin
              if(cornr.eq.'NE')then
                 cornr='NE'
                 FTNS=(YP-YNE)
                 FTEW=(XP-XNE)
              elseif(cornr.eq.'NW')then
                 cornr='NW'
                 FTNS=(YP-YNW)
                 FTEW=(XP-XNW)
              elseif(cornr.eq.'SW')then
                 cornr='SW'
                 FTNS=(YP-YSW)
                 FTEW=(XP-XSW)
              elseif(cornr.eq.'SE')then
                 cornr='SE'
                 FTNS=(YP-YSE)
                 FTEW=(XP-XSE)
              endif
      elseif(OPTION2.eq.2)then
              cornr=cornrgl
              if(cornr.eq.'NE')then
                 cornr='NE'
                 FTNS=(YP-YNE)
                 FTEW=(XP-XNE)
              elseif(cornr.eq.'NW')then
                 cornr='NW'
                 FTNS=(YP-YNW)
                 FTEW=(XP-XNW)
              elseif(cornr.eq.'SW')then
                 cornr='SW'
                 FTNS=(YP-YSW)
                 FTEW=(XP-XSW)
              elseif(cornr.eq.'SE')then
                 cornr='SE'
                 FTNS=(YP-YSE)
                 FTEW=(XP-XSE)
              endif
      endif
C
C
```

```
C                  ********************          ***
C***************  S U C C E S S ! ! !  **********
C                  ********************          ***
        STAT=1
        IF (SFLAG(SECT).GT.0) NONSTD=0
        if(hshut1c.eq.1)then
                NONSTD=900
                if(tshp.eq.1)ftns=ftns+100
                if(tshp.eq.35)ftns=ftns-100
        elseif(hshut40.eq.1)then
                NONSTD=940
                if(tshp.eq.1)ftns=ftns+40
                if(tshp.eq.35)ftns=ftns-40
        endif
        if(hshur1c.eq.1)then
                NONSTD=800
                if(rang.eq.68)ftew=ftew+100
                if(rang.eq.1)ftew=ftew-100
        elseif(hshur40.eq.1)then
                NONSTD=840
                if(rang.eq.68)ftew=ftew+40
                if(rang.eq.1)ftew=ftew-40
        endif
        if(OPTION2.eq.1.and.surv.eq.0)stat=10
        if(OPTION2.eq.2.and.
     &       (cornr.eq.'NE'.and.scor(1).ne.1.and.scor(1).ne.4))stat=10
        if(OPTION2.eq.2.and.
     &       (cornr.eq.'NW'.and.scor(2).ne.1.and.scor(2).ne.4))stat=10
        if(OPTION2.eq.2.and.
     &       (cornr.eq.'SW'.and.scor(3).ne.1.and.scor(3).ne.4))stat=10
        if(OPTION2.eq.2.and.
     &       (cornr.eq.'SE'.and.scor(4).ne.1.and.scor(4).ne.4))stat=10
C
C        *  Nothing left to do but set the RANG indicators to the
C        *  expected E and W values,and reset thrashing monitor
C        *  arrays to zero for any subsequent calls
C
        IF (RANG.LT.44) THEN
           RANG=44-RANG
           EW='W'
        ELSE
           RANG=RANG-43
           EW='E'
        ENDIF
C        *  Reset the thrashing monitors if necessary
1500    IF (RTOP.GT.1) THEN
           DO 1600 I=1,RTOP-1
              VZT(VZTD(I))=0
              VZTD(I)=0
1600       CONTINUE
           RTOP=1
        ENDIF
        RETURN
C        *  For ERR= errors, set error flag and RETURN
1999    STAT=-13
        IF (SORCE.EQ.2) GOTO 1500
        RETURN
        END
```

```
C
      SUBROUTINE STATE_EDGE(ICODE,XLL,YLL,XLR,YLR,XUL,YUL,XUR,YUR)
C
C           ICODE is the section flag SFLAG(SECT)
C
      IMPLICIT NONE
      INTEGER*2 ICODE,abs
      REAL XLL,YLL,XLR,YLR,XUL,YUL,XUR,YUR
      REAL*8 XC,YC
      REAL*8 AM, BM, ECC, EC2, PIE
      REAL*8 XCR, YCR
      REAL*8 DEGRAD
      REAL*8 LADJ
C     REAL*8 WEST,EAST,SOUTH,NORTH
      REAL*8 CORN(9)
      REAL*8 XLLP,YLLP,XLRP,YLRP
      REAL*8 XULP,YULP,XURP,YURP
      REAL*8 DLX,DUX,DLY,DRY
      REAL*8 XIN(2),YIN(2),XOT(2),YOT(2),ELEN(2),XIP(2),YIP(2)
      INTEGER I
      REAL*8 ETEST
      REAL*8 XMILE, YMILE
      REAL*8 XP, YP, R, XDIF, YDIF
C
      REAL*8 SOUTHD,NORTHD,CLON,EASTD,WESTD,XB,YB,BASE,SCALE
      COMMON /LEOMAP/ SOUTHD,NORTHD,CLON,EASTD,WESTD,XB,YB,BASE,SCALE
C
      REAL*8 XXLL,YYLL,XXLR,YYLR,XXUL,YYUL,XXUR,YYUR
      REAL*8 LLON, LLAT
      INTEGER ECODE
      COMMON /SAVE_SEC/ XXLL,YYLL,XXLR,YYLR,XXUL,YYUL,XXUR,YYUR,
     X                  LLON, LLAT, ECODE
c
C
c       ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c     LLAT = length of 1 degree latitude (ft/deg)
c     LLON = length of 1 degree longitude (ft/deg)
c
c
c     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
      if(icode.lt.0)then
            abs=1
            icode=-icode
      endif
C
C...  SET PROJECTION PARAMETERS
C
      AM = 251110472.8
      BM = 250259213.
      PIE = 3.14159265
      DEGRAD = PIE / 180.0
      EC2 = (AM ** 2 - BM ** 2)/AM ** 2
      ECC = DSQRT (EC2)
      SCALE = 12.D0
      ETEST = 100.0
C
```

```
C...  SAVE SECTION CORNERS
C
      ECODE = ICODE
      XXLL = XLL
      YYLL = YLL
      XXLR = XLR
      YYLR = YLR
      XXUL = XUL
      YYUL = YUL
      XXUR = XUR
      YYUR = YUR
C
C...  CALCULATE THE CENTER OF THE SECTION FROM THE DATABASE
C
      XC=(XLL+XLR+XUL+XUR)/4.0
      YC=(YLL+YLR+YUL+YUR)/4.0
C     WRITE (*,*) 'CENTER = ',ICODE,XC,YC
C
C...  CALCULATE THE LENGTH OF 1 DEG OF LATITUDE AND LONGITUDE
C
      XCR = XC * DEGRAD
      YCR = YC * DEGRAD
      LADJ = PIE/180.0
      LLAT = LADJ*AM/12.0
      LLON = LADJ*(AM/12.0)*DCOS(YCR)
c     write(*,*)'LLAT = ',llat,' ft/deg lat'
c     write(*,*)'LLON = ',llon,' ft/deg lon'
C
C...  CALCULATE MAP LIMITS -- APPROX. 1.5 MILES FROM CENTER OF SECTION
C
      WESTD = XC - 8000.0/LLON
      EASTD = XC + 8000.0/LLON
      SOUTHD = YC - 8000.0/LLAT
      NORTHD = YC + 8000.0/LLAT
c     write (*,*) 'W-E = ',WESTD,EASTD
c     write (*,*) 'S-N = ',SOUTHD,NORTHD
      CALL LEO3SETP(CORN)
C
C...  PROJECT CORNERS OF SECTION FROM DATABASE
C
      CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XLLP,YLLP)
c     write(*,*)'LL ',XLLP,YLLP
      CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XLRP,YLRP)
c     write(*,*)'LR ',XLRP,YLRP
      CALL LEO3PRO(DBLE(XUL),DBLE(YUL),XULP,YULP)
c     write(*,*)'UL ',XULP,YULP
      CALL LEO3PRO(DBLE(XUR),DBLE(YUR),XURP,YURP)
c     write(*,*)'UR ',XURP,YURP
C
C...  CALCULATE LENGTH OF EACH EDGE OF SECTION
C
      DLX = DSQRT( (XLLP-XLRP)**2 + (YLLP-YLRP)**2)
      DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
c     write (*,*) 'DIST X = (dlx,dux)',DLX,DUX
      DLY = DSQRT( (XLLP-XULP)**2 + (YLLP-YULP)**2)
      DRY = DSQRT( (XLRP-XURP)**2 + (YLRP-YURP)**2)
c     write (*,*) 'DIST Y = (dly,dry)',DLY,DRY
C
```

```
C...  ADJUST EAST EDGE OF STATE (ALONG KANSAS-MISSOURI BORDER)
C
      IF ((ICODE .GE. 8113) .AND. (ICODE .LE. 8213) .AND.
     X                              (ICODE .NE. 8133)) THEN
          XIN(1)=XLL
          YIN(1)=YLL
          XOT(1)=XLR
          YOT(1)=YLR
          ELEN(1)=DLX
          XIN(2)=XUL
          YIN(2)=YUL
          XOT(2)=XUR
          YOT(2)=YUR
          ELEN(2)=DUX
          XIP(1)=XLLP
          YIP(1)=YLLP
          XIP(2)=XULP
          YIP(2)=YULP
          DO 100 I=1,2
C            WRITE (*,*) 'EDGE LEN = ',ELEN(I),XOT(I)-XIN(I)
             IF (ELEN(I) .LT. ETEST) THEN
              XMILE = 5000.0/LLON
              XOT(I)=XIN(I) + XMILE
              YOT(I)=YIN(I)
              CALL LEO3PRO(XOT(I),YOT(I),XP,YP)
              ELEN(I) = DSQRT((XIP(I)-XP)**2 + (YIP(I)-YP)**2)
             ENDIF
             IF (ELEN(I) .LT. 5280.0) THEN
              R = 5280.0/ELEN(I)
              XDIF = XOT(I) - XIN(I)
              YDIF = YOT(I) - YIN(I)
              XOT(I) = XIN(I) + XDIF*R
              YOT(I) = YIN(I) + YDIF*R
             ENDIF
100       CONTINUE
          XLR = XOT(1)
          YLR = YOT(1)
          XUR = XOT(2)
          YUR = YOT(2)
C         CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XLRP,YLRP)
C         CALL LEO3PRO(DBLE(XUR),DBLE(YUR),XURP,YURP)
C         DLX = DSQRT( (XLLP-XLRP)**2 + (YLLP-YLRP)**2)
C         DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C         WRITE (*,*) 'MISS UR = ',XUR,YUR,DUX,XOT(2)-XIN(2)
C         WRITE (*,*) 'MISS LR = ',XLR,YLR,DLX,XOT(1)-XIN(1)
        ENDIF
C
```

```
C...  ADJUST SOUTHEAST CORNER OF STATE
C
      IF (ICODE .EQ. 8133) THEN
C        WRITE (*,*) 'TOP EDGE LEN = ',DUX,XUR-XUL
         IF (DUX .LT. ETEST) THEN
            XMILE = 5000.0/LLON
            XUR=XUL + XMILE
            YUR=YUL
            CALL LEO3PRO(DBLE(XUR),DBLE(YUR),XP,YP)
            DUX = DSQRT((XULP-XP)**2 + (YULP-YP)**2)
          ENDIF
         IF (DUX .LT. 5280.0) THEN
            R = 5280.0/DUX
            XDIF = XUR - XUL
            YDIF = YUR - YUL
            XUR = XUL + XDIF*R
            YUR = YUL + YDIF*R
          ENDIF
         CALL LEO3PRO(DBLE(XUR),DBLE(YUR),XURP,YURP)
         DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C        WRITE (*,*) 'OM CORNER UR = ',XUR,YUR,DUX,XUR-XUL
C
C        WRITE (*,*) 'WEST EDGE LEN = ',DLY,YLL-YUL
         IF (DLY .LT. ETEST) THEN
            YMILE = 5000.0/LLAT
            XLL=XUL
            YLL=YUL - YMILE
            CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XP,YP)
            DLY = DSQRT((XULP-XP)**2 + (YULP-YP)**2)
          ENDIF
         IF (DLY .LT. 5280.0) THEN
            R = 5280.0/DLY
            XDIF = XLL - XUL
            YDIF = YLL - YUL
            XLL = XUL + XDIF*R
            YLL = YUL + YDIF*R
          ENDIF
C        CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XLLP,YLLP)
C        DLY = DSQRT( (XULP-XLLP)**2 + (YULP-YLLP)**2)
C        WRITE (*,*) 'OM CORNER LL = ',XLL,YLL,DLY,YUR-YLL
         XLR = XUR
         YLR = YLL
C        CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XLRP,YLRP)
C        DLX = DSQRT( (XLLP-XLRP)**2 + (YLLP-YLRP)**2)
C        DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C           WRITE (*,*) 'OM DIST X = ',DLX,DUX
C        DLY = DSQRT( (XLLP-XULP)**2 + (YLLP-YULP)**2)
C        DRY = DSQRT( (XLRP-XURP)**2 + (YLRP-YURP)**2)
C        WRITE (*,*) 'OM DIST Y = ',DLY,DRY
        ENDIF
C
```

```
C...  ADJUST SOUTH EDGE OF STATE (ALONG KANSAS-OKLAHOMA BORDER)
C
      IF ((ICODE .GE. 6113) .AND. (ICODE .LE. 6133)) THEN
         XIN(1)=XUL
         YIN(1)=YUL
         XOT(1)=XLL
         YOT(1)=YLL
         ELEN(1)=DLY
         XIN(2)=XUR
         YIN(2)=YUR
         XOT(2)=XLR
         YOT(2)=YLR
         ELEN(2)=DRY
         XIP(1)=XULP
         YIP(1)=YULP
         XIP(2)=XURP
         YIP(2)=YURP
         DO 110 I=1,2
C           WRITE (*,*) 'EDGE LEN = ',ELEN(I),YOT(I)-YIN(I)
            IF (ELEN(I) .LT. ETEST) THEN
             YMILE = 5000.0/LLAT
             XOT(I)=XIN(I)
             YOT(I)=YIN(I) - YMILE
             CALL LEO3PRO(XOT(I),YOT(I),XP,YP)
             ELEN(I) = DSQRT((XIP(I)-XP)**2 + (YIP(I)-YP)**2)
            ENDIF
            IF (ELEN(I) .LT. 5280.0) THEN
             R = 5280.0/ELEN(I)
             XDIF = XOT(I) - XIN(I)
             YDIF = YOT(I) - YIN(I)
             XOT(I) = XIN(I) + XDIF*R
             YOT(I) = YIN(I) + YDIF*R
            ENDIF
110      CONTINUE
         XLL = XOT(1)
         YLL = YOT(1)
         XLR = XOT(2)
         YLR = YOT(2)
C        CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XLLP,YLLP)
C        CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XLRP,YLRP)
C        DLY = DSQRT( (XLLP-XULP)**2 + (YLLP-YULP)**2)
C        DRY = DSQRT( (XLRP-XURP)**2 + (YLRP-YURP)**2)
C        WRITE (*,*) 'OKLA LL = ',XLL,YLL,DLY,YOT(1)-YIN(1)
C        WRITE (*,*) 'OKLA LR = ',XLR,YLR,DRY,YOT(2)-YIN(2)
        ENDIF
C
```

```
C...  ADJUST SOUTHWEST CORNER OF STATE
C
      IF (ICODE .EQ. 6333) THEN
C        WRITE (*,*) 'TOP EDGE LEN = ',DUX,XUL-XUR
         IF (DUX .LT. ETEST) THEN
            XMILE = 5000.0/LLON
            XUL=XUR + XMILE
            YUL=YUR
            CALL LEO3PRO(DBLE(XUL),DBLE(YUL),XP,YP)
            DUX = DSQRT((XULP-XP)**2 + (YULP-YP)**2)
          ENDIF
         IF (DUX .LT. 5280.0) THEN
            R = 5280.0/DUX
            XDIF = XUL - XUR
            YDIF = YUL - YUR
            XUL = XUR + XDIF*R
            YUL = YUR + YDIF*R
          ENDIF
         CALL LEO3PRO(DBLE(XUL),DBLE(YUL),XULP,YULP)
         DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C        WRITE (*,*) 'CO CORNER UL = ',XUL,YUL,DUX,XUL-XUR
C
C        WRITE (*,*) 'EAST EDGE LEN = ',DRY,YLR-YUR
         IF (DRY .LT. ETEST) THEN
            YMILE = 5000.0/LLAT
            XLR=XUR
            YLR=YUR - YMILE
            CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XP,YP)
            DRY = DSQRT((XURP-XP)**2 + (YURP-YP)**2)
          ENDIF
         IF (DRY .LT. 5280.0) THEN
            R = 5280.0/DRY
            XDIF = XLR - XUR
            YDIF = YLR - YUR
            XLR = XUR + XDIF*R
            YLR = YUR + YDIF*R
          ENDIF
C        CALL LEO3PRO(DBLE(XLR),DBLE(YLR),XLRP,YLRP)
C        DRY = DSQRT( (XURP-XLRP)**2 + (YURP-YLRP)**2)
C        WRITE (*,*) 'CO CORNER LR = ',XLR,YLR,DRY,YUR-YLR
         XLL = XUL
         YLL = YLR
C        CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XLLP,YLLP)
C        DLX = DSQRT( (XLLP-XLRP)**2 + (YLLP-YLRP)**2)
C        DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C           WRITE (*,*) 'CO DIST X = ',DLX,DUX
C        DLY = DSQRT( (XLLP-XULP)**2 + (YLLP-YULP)**2)
C        DRY = DSQRT( (XLRP-XURP)**2 + (YLRP-YURP)**2)
C        WRITE (*,*) 'CO DIST Y = ',DLY,DRY
        ENDIF
C
```

```
C...  ADJUST WEST EDGE OF STATE (ALONG KANSAS-COLORADO BORDER)
C
      IF (((ICODE .GE. 6331) .AND. (ICODE .LE. 6332))
     X                         .or.icode.eq.7331) THEN
          XIN(1)=XLR
          YIN(1)=YLR
          XOT(1)=XLL
          YOT(1)=YLL
          ELEN(1)=DLX
          XIN(2)=XUR
          YIN(2)=YUR
          XOT(2)=XUL
          YOT(2)=YUL
          ELEN(2)=DUX
          XIP(1)=XLRP
          YIP(1)=YLRP
          XIP(2)=XURP
          YIP(2)=YURP
          DO 120 I=1,2
c            write (*,*) 'EDGE LEN = ',ELEN(I),XOT(I)-XIN(I)
             IF (ELEN(I) .LT. ETEST) THEN
              XMILE = 5000.0/LLON
              XOT(I)=XIN(I) - XMILE
              YOT(I)=YIN(I)
              CALL LEO3PRO(XOT(I),YOT(I),XP,YP)
              ELEN(I) = DSQRT((XIP(I)-XP)**2 + (YIP(I)-YP)**2)
c            write (*,*) 'EDGE LEN = ',ELEN(I)
             ENDIF
             IF (ELEN(I) .LT. 5280.0) THEN
              R = 5280.0/ELEN(I)
              XDIF = XOT(I) - XIN(I)
              YDIF = YOT(I) - YIN(I)
              XOT(I) = XIN(I) + XDIF*R
              YOT(I) = YIN(I) + YDIF*R
             ENDIF
120       CONTINUE
          XLL = XOT(1)
          YLL = YOT(1)
          XUL = XOT(2)
          YUL = YOT(2)
C         CALL LEO3PRO(DBLE(XLL),DBLE(YLL),XLLP,YLLP)
C         CALL LEO3PRO(DBLE(XUL),DBLE(YUL),XULP,YULP)
C         DLX = DSQRT( (XLLP-XLRP)**2 + (YLLP-YLRP)**2)
C         DUX = DSQRT( (XULP-XURP)**2 + (YULP-YURP)**2)
C         WRITE (*,*) 'COLO UL = ',XUL,YUL,DUX,XOT(2)-XIN(2)
C         WRITE (*,*) 'COLO LL = ',XLL,YLL,DLX,XOT(1)-XIN(1)
        ENDIF
C
      if(abs.eq.1)icode=-icode
      RETURN
      END
C
```

```
      SUBROUTINE CHECK_STATE_EDGE(RCODE,XLL,YLL,XLR,YLR,
     X                   XUL,YUL,XUR,YUR,XCP,YCP,CORNR)
C
      IMPLICIT NONE
      INTEGER*2 RCODE
C
C           RCODE is the STATUS code
C
      REAL XLL,YLL,XLR,YLR,XUL,YUL,XUR,YUR,XCP,YCP
      CHARACTER*2 CORNR
      INTEGER I,ICHK(5),ICHKS(5)
      REAL*8 X,Y,XI,YI
      REAL*8 XX(5),YY(5)
      INTEGER SOUT,souts,sin
      REAL*8 DIV
      INTEGER NP
C
      REAL*8 XXLL,YYLL,XXLR,YYLR,XXUL,YYUL,XXUR,YYUR
      REAL*8 LLON, LLAT
      INTEGER ICODE,abs
      COMMON /SAVE_SEC/ XXLL,YYLL,XXLR,YYLR,XXUL,YYUL,XXUR,YYUR,
     X                    LLON, LLAT, ICODE
c
      if(icode.lt.0)then
            abs=1
            icode=-icode
      endif
C
C---- SAVE CORNERS AND POINT INTO ARRAYS
C
      XX(1)=XUR
      YY(1)=YUR
      XX(2)=XUL
      YY(2)=YUL
      XX(3)=XLL
      YY(3)=YLL
      XX(4)=XLR
      YY(4)=YLR
      IF (CORNR .NE. '  ') THEN
         XX(5)=XCP
         YY(5)=YCP
         NP=5
        ELSE
         NP=4
        ENDIF
C
C---- CHECK EACH POINT TO SEE IF INSIDE ORIGINAL SECTION
C     IF INSIDE ICHK() = ZERO, IF OUTSIDE WILL BE GREATER THAN 0
C
      DO 10 I=1,5
        ICHK(I)=0
        ICHKS(I)=0
        X=XX(I)
        Y=YY(I)
C
```

```
C           Test relative to Oklahoma line
C
        IF((ICODE.EQ.6133).OR.(ICODE.EQ.6233).OR.(ICODE.EQ.6333).OR.
     &              (ICODE.EQ.7133).OR.(ICODE.EQ.8133))THEN
           DIV = XXLR-XXLL
c          write(*,*)div*llon
           IF (DABS(DIV) .GT. 10.0/LLON) THEN
              YI=(X-XXLL)*(YYLR-YYLL)/DIV + YYLL
           ELSE
              YI=(YYLR+YYLL)/2.0
           ENDIF
           IF (Y .LT. YI) then
                 IF((ICODE.EQ.6333).OR.(ICODE.EQ.8133))then
                       ICHKS(I)=ICHKS(I)+1
                 else
                       ICHK(I)=ICHK(I)+1
                 ENDIF
           ENDIF
        ENDIF
C
C     **********
C
C     The following is appropriate for test of subdivisions of
C     sections 1-6 on the north side of townships
C
C       DIV = XXUR-XXUL
C       IF (DABS(DIV) .GT. 10.0/LLON) THEN
C          YI=(X-XXUL)*(YYUR-YYUL)/DIV + YYUL
C         ELSE
C          YI=(YYUR+YYUL)/2.0
C         ENDIF
C       IF (Y .GT. YI) ICHK(I)=ICHK(I)+1
C
C     **********
C
C           Test relative to Colorado line
C
        IF((ICODE.EQ.6331).OR.(ICODE.EQ.6332).OR.(ICODE.EQ.6333).OR.
     &              (ICODE.EQ.7331))THEN
           DIV = YYUL-YYLL
           IF (DABS(DIV) .GT. 10.0/LLAT) THEN
              XI=(Y-YYLL)*(XXUL-XXLL)/DIV + XXLL
           ELSE
              XI=(XXUL+XXLL)/2.0
           ENDIF
           IF (X .LT. XI) ICHK(I)=ICHK(I)+1
        ENDIF
C
```

```
C           Test relative to Missouri line
C
        IF((ICODE.EQ.8113).OR.(ICODE.EQ.8123).OR.(ICODE.EQ.8133).OR.
     &              (ICODE.EQ.8213))THEN
            DIV = YYUR-YYLR
c           write(*,*)div*llat
            IF (DABS(DIV) .GT. 10.0/LLAT) THEN
               XI=(Y-YYLR)*(XXUR-XXLR)/DIV + XXLR
            ELSE
               XI=(XXUR+XXLR)/2.0
            ENDIF
            IF (X .GT. XI) ICHK(I)=ICHK(I)+1
        ENDIF
10     continue
C      write(*,*)'ICHK = ',ichk
C      if((icode.eq.8133).or.(icode.eq.6333))then
C           write(*,*)'ICHKS = ',ichks
C      endif
C
C...   FIND NUMBER OF CORNERS OF SUB-DIVISION THAT ARE OUTSIDE STATE
C
       SOUT=0
       souts=0
       sin=0
       DO 20 I=1,4
         IF (ICHK(I) .GT. 0) SOUT=SOUT+1
         IF (ICHKS(I) .GT. 0) souts=souts+1
         if(ichks(i).eq.0 .and. ichk(i).eq.0) sin=sin+1
20     continue
       if(icode.eq.8133)then
        write(*,*)
      X  'Corners of subdivision outside KS across MO line = ',sout
        write(*,*)
      X  'Corners of subdivision outside KS across OK line = ',souts
        write(*,*)'Corners of subdivision in Kansas = ',sin
       elseif(icode.eq.6333)then
        write(*,*)
      X  'Corners of subdivision outside KS across CO line = ',sout
        write(*,*)
      X  'Corners of subdivision outside KS across OK line = ',souts
        write(*,*)'Corners of subdivision in Kansas = ',sin
       else
        write(*,*)'Corners of subdivision outside Kansas = ',sout
       endif
C
```

```
C...  CHECK FOR EASY CASES - EVERYTHING IN OR OUT
C
      IF (CORNR .EQ. '   ') THEN
         RCODE = 1
         IF ((SOUT .EQ. 0).AND.(SOUTS.EQ.0)) RETURN
         RCODE = -33
         IF ((SOUT .EQ. 4).OR.(SOUTS.EQ.4)) RETURN
         RCODE = 34
      ELSE
         RCODE = 1
         IF (SOUT.EQ.0 .AND. SOUTS.EQ.0 .AND. ICHK(5).EQ.0 .AND.
     X      ICHKS(5).EQ.0) RETURN
         RCODE = -33
         IF ((SOUT.EQ.4 .OR. SOUTS.EQ.4) .AND.
     X      (ICHK(5).GE.1 .OR. ICHKS(5).GE.1))  RETURN
         IF ((SOUT.GE.1 .OR. SOUTS.GE.1) .AND. ICHK(5).EQ.0 .AND.
     X      ICHKS(5).EQ.0) THEN
            RCODE = 35
            RETURN
         ENDIF
         IF (((SOUT.EQ.1 . OR. SOUT.EQ.2) .AND. ICHK(5).GE.1) .OR.
     &            ((SOUTS.EQ.1 . OR. SOUTS.EQ.2) .AND. ICHKS(5).GE.1))
     X      RCODE = 36
         IF ((SOUT.EQ.3 .AND. ICHK(5).GE.1) .OR.
     X      (SOUTS.EQ.3 .AND. ICHKS(5).GE.1))THEN
            RCODE = -37
            RETURN
         ENDIF
        ENDIF
C
C...  CASES WHERE SUB-DIVISION IS PARTIALLY IN STATE
C
C...  ADJUST SOUTH EDGE OF STATE (ALONG KANSAS-OKLAHOMA BORDER)
C
      IF ((ICODE .GE. 6113) .AND. (ICODE .LE. 6133)) THEN
         IF (ICHK(3) .GE. 1) THEN
            CALL INTERSECT(XXLL,YYLL,XXLR,YYLR,XLL,YLL,XUL,YUL, XI,YI)
            XLL=XI
            YLL=YI
          ENDIF
         IF (ICHK(4) .GE. 1) THEN
            CALL INTERSECT(XXLL,YYLL,XXLR,YYLR,XLR,YLR,XUR,YUR, XI,YI)
            XLR=XI
            YLR=YI
          ENDIF
      ENDIF
      IF (icode.eq.8133.or.icode.eq.6333) THEN
         IF (ICHKS(3) .GE. 1) THEN
            CALL INTERSECT(XXLL,YYLL,XXLR,YYLR,XLL,YLL,XUL,YUL, XI,YI)
            XLL=XI
            YLL=YI
          ENDIF
         IF (ICHKS(4) .GE. 1) THEN
            CALL INTERSECT(XXLL,YYLL,XXLR,YYLR,XLR,YLR,XUR,YUR, XI,YI)
            XLR=XI
            YLR=YI
          ENDIF
      ENDIF
```

```
C
C...  ADJUST EAST EDGE OF STATE (ALONG KANSAS-MISSOURI BORDER)
C
      IF ((ICODE .GE. 8113) .AND. (ICODE .LE. 8213) .OR.
     X                              (ICODE .EQ. 8133)) THEN
          IF (ICHK(1) .GE. 1) THEN
             CALL INTERSECT(XXLR,YYLR,XXUR,YYUR,XUL,YUL,XUR,YUR, XI,YI)
             XUR=XI
             YUR=YI
           ENDIF
          IF (ICHK(4) .GE. 1) THEN
             CALL INTERSECT(XXLR,YYLR,XXUR,YYUR,XLL,YLL,XLR,YLR, XI,YI)
             XLR=XI
             YLR=YI
           ENDIF
       ENDIF
C
C
C...  ADJUST WEST EDGE OF STATE (ALONG KANSAS-COLORADO BORDER)
C
C     write(*,*)'Section flag = ',icode
      IF ((ICODE .eq. 6331) .or. (ICODE .eq. 6332)
     X      .or. (icode .eq. 7332) .OR. (ICODE.EQ.6333)) THEN
          IF (ICHK(2) .GE. 1) THEN
             CALL INTERSECT(XXLL,YYLL,XXUL,YYUL,XUL,YUL,XUR,YUR, XI,YI)
             XUL=XI
             YUL=YI
           ENDIF
          IF (ICHK(3) .GE. 1) THEN
             CALL INTERSECT(XXLL,YYLL,XXUL,YYUL,XLL,YLL,XLR,YLR, XI,YI)
             XLL=XI
             YLL=YI
          ENDIF
       ENDIF
C
      if(abs.eq.1)icode=-icode
      RETURN
      END
C
```

```
C.... ROUTINE TO CALCULATE THE INTERSECTION POINT OF TWO LINES
C
C     CALCULATE THE INTERSECTION POINT OF TWO LINES
C       (X1,Y1 X2,Y2) AND (X3,Y3  X4,Y4)

      SUBROUTINE INTERSECT(X1,Y1,X2,Y2,X3,Y3,X4,Y4,XI,YI)
      IMPLICIT NONE
      REAL*8 X1,Y1,X2,Y2,XI,YI
      REAL X3,Y3,X4,Y4
      REAL*8 D11,D12
      REAL*8 D21,D22,d31
      REAL*8 DD
C
c
      d11=x2-x1
      d12=y2-y1
      d21=x4-x3
      d22=y4-y3
      d31=y3-y1
      dd=d12*d21-d22*d11
      xi=(d11*d21*d31-x3*d22*d11+x1*d12*d21)/dd
      if(d11.gt.d21)then
            yi=(xi-x1)*d12/d11+y1
      else
            yi=(xi-x3)*d22/d21+y3
      endif
c     write(*,*)'intersect'
c     write(*,*)'xi = ',xi
c     write(*,*)'yi = ',yi
      RETURN
      END
c
c
c     Modifications:
c     960802 -- state_edge and check_state_edge
c           modified to use absolute value of section flag
```

```
C
C       * * *     L E O 3 S E T P     * * *
C
C       * SCALE  =  MAP SCALE FACTOR
C       * CORN   =  STORAGE FOR PROJECTED CORNERS
C       * NPRO   =  -1 FOR MODIFIED POLYCONIC
C       MOD (INT(SEC),6)
        SUBROUTINE LEO3SETP (CORN)
        IMPLICIT NONE
        DOUBLE PRECISION AP,BP,CP,DP,EP,AM,BM,ECC,PIE,SOUTHD,NORTHD,
     &   CLON,EASTD,WESTD,XB,YB,BASE,SCALE,SOUTH,NORTH,EAST,WEST
        DOUBLE PRECISION AN, AN2, AN3, AN4, AN5, RADB, DELTA, CORN(9)
        COMMON /LEOPRO/ AP, BP, CP, DP, EP
        COMMON/LEOMAP/SOUTHD,NORTHD,CLON,EASTD,WESTD,XB,YB,BASE,SCALE
        COMMON /LEOEAR/ AM, BM, ECC, PIE
        AM = 251110472.8
        BM = 250259213.
        PIE = 3.14159265
        ECC = DSQRT ((AM ** 2 - BM ** 2)/AM ** 2)
        SCALE = 12.D0
        SOUTH = SOUTHD * 2. * PIE / 360.
        NORTH = NORTHD * 2. * PIE / 360.
        EAST = EASTD * 2. * PIE / 360.
        WEST = WESTD * 2. * PIE / 360.
        CLON = (EAST + WEST)/2.
C
C       * INITIALIZE FOR THE MODIFIED
C       * POLYCONIC PROJECTION
C
C       * SOUTH,NORTH LATITUDES AT BOTTOM
C       *       AND TOP OF OUTPUT MAP
C       * XB,YB COORDINATES OF BASE POINT
C       * SCALE IS MAP SCALING FACTOR
C       * BASE IS THE UNSCALED DISTANCE
C       *       FROM THE EQUATOR TO THE BASE
C       * AP,BP,CP,DP,EP ARE USED TO FIND
C       *       THE DISTANCES FROM THE EQUATOR
C       * SOUTH, NORTH, WEST, AND EAST MUST BE
C       *       SET PRIOR TO CALL
C
        AN = (AM - BM) / (AM + BM)
        AN2 = AN ** 2
        AN3 = AN ** 3
        AN4 = AN ** 4
        AN5 = AN ** 5
C
        AP = 1. - AN + (5./4.) * (AN2-AN3) + (81./64.) * (AN4-AN5)
        AP = AM * AP
        BP = AN - AN2 + (7. / 8.) * (AN3 - AN4) + (55. / 64.) * AN5
        BP = 1.5 * AM * BP
        CP = (15. / 16.) * AM * (AN2 - AN3 + (3. / 4.) * (AN4 - AN5))
        DP = (35. / 48.) * AM * (AN3 - AN4 + (11. / 16.) * AN5)
        EP = (315. / 512.) * AM * (AN4 - AN5)
C
```

```
C       * FIND BASE COORDINATES (XB,YB)
C       * BASED ON NORMALIZED ASSUMPTION
C       * WITH SW CORNER AT 1.0,1.0
C       * AND SE CORNER AT YSE =  1.0
C
        BASE = AP * SOUTH - BP * DSIN(2.*SOUTH) + CP * DSIN(4.*SOUTH)
        BASE = BASE - DP * DSIN (6. * SOUTH) + EP * DSIN (8. * SOUTH)
        RADB = 1. - (ECC ** 2) * (DSIN (SOUTH) ** 2)
        RADB = (AM / SCALE) / DSQRT (RADB)
        RADB = (RADB * DCOS (SOUTH)) / DSIN (SOUTH)
        DELTA = (CLON - WEST) * DSIN (SOUTH)
        XB = 1.0 + RADB * DSIN (DELTA)
        YB = 1.0 - RADB * (1. - DCOS (DELTA))
C
C       * SET CORNER POINTS BY PROJECTION
C
        CALL LEO3PRO (WESTD, SOUTHD, CORN(1), CORN(2))
        CALL LEO3PRO (EASTD, SOUTHD, CORN(3), CORN(4))
        CALL LEO3PRO (EASTD, NORTHD, CORN(5), CORN(6))
        CALL LEO3PRO (WESTD, NORTHD, CORN(7), CORN(8))
        CORN(9) = YB
        RETURN
        END
```

```
C
C       * * *     L E O 3 P R O     * * *
C
C       * MODIFIED POLYCONIC PROJECTION
C
C       * GIVEN LATITUDE AND LONGITUDE
C       * FOR A POINT, PROJECT IT TO
C       * X,Y COORDINATES BASED ON
C       * PRE-SET POLYCONIC PARAMETERS
C
        SUBROUTINE LEO3PRO (PLOND, PLATD, XP, YP)
        IMPLICIT NONE
        DOUBLE PRECISION PLON,PLAT,XP,YP,RADC,YDISP,YC,DELON,DELTA,
     &      XDEL, YDEL, PLOND, PLATD
        DOUBLE PRECISION SOUTHD,NORTHD,CLON,EASTD,WESTD,XB,YB,BASE,
     &      SCALE, AM, BM, ECC, PIE, AP, BP, CP, DP, EP
        COMMON /LEOEAR/ AM, BM, ECC, PIE
        COMMON/LEOMAP/SOUTHD,NORTHD,CLON,EASTD,WESTD,XB,YB,BASE,SCALE
        COMMON /LEOPRO/ AP, BP, CP, DP, EP
        PLON = 2. * PIE * PLOND / 360.
        PLAT = 2. * PIE * PLATD / 360.
        RADC = 1. - (ECC ** 2) * (DSIN (PLAT) ** 2)
        RADC = (AM / SCALE) / DSQRT (RADC)
        RADC = (RADC * DCOS (PLAT)) / DSIN (PLAT)
        YDISP = AP * PLAT - BP * DSIN (2.*PLAT) + CP * DSIN(4. * PLAT)
        YDISP = YDISP - DP * DSIN (6. * PLAT) + EP * DSIN (8. * PLAT)
        YDISP = (YDISP - BASE) / SCALE
        YC = YB + YDISP
        DELON = DABS (CLON - PLON)
        DELTA = DELON * DSIN (PLAT)
        XDEL = RADC * DSIN (DELTA)
        IF (CLON.GT.PLON) XDEL = -XDEL
        YDEL = RADC * (1. - DCOS (DELTA))
        XP = XB + XDEL
        YP = YC + YDEL
        RETURN
        END
```

```
C
C....   ROUTINE TO PROVIDE FOR SPECIAL HANDLING OF NONSTANDARD SECTIONS
C
C
C            For sections which require some nonstandard treatment for
C            accurate subdivision:
C               SFLAG = -n, where n is the normal code for SFLAG.
C                  When SFLAG<0 the system calls FEDX, a subroutine
C                  for special handling.  Within FEDX, the process
C                  is directed to a statement number which matches
C                  the township record number in LEO3BASE of the
C                  township which contains that section.  The
C                  township record number is based on Township,
C                  Range, and Range direction (E or W) as follows:
C                  If range direction is east:
C                        TRECB = (TSHIP - 1) * 68  +  RANGE + 43
C                  If range direction is west:
C                        TRECB = (TSHIP - 1) * 68  +  44 - RANGE
C                  Routines for any sections which require special
C                  handling within a township will be found in FEDX
C                  after the statement number matching that
C                  township's record number in LEO3BASE.
C
C
      SUBROUTINE FEDX(TSHP,RANG,EW,SECT,OPT,SUBD,STAT,SORCE,trec)
c
      implicit none
c
      character ew*1,SUBD*2(4),cornr*2,blank2*2,cornrgl*2
c
      integer*2 tshp,rang,sect,opt,stat,sorce,trec,level,maxlvl,
     &  tflag,sflag(36),scor(4),nonstd,OPTION2
c
      real lonx,latx,LONNE,LATNE,LONNW,LATNW, LONSW,LATSW,
     &   LONSE,LATSE,CTRLON,CTRLAT,ftns,ftew,lon(7,7),lat(7,7)
C
      COMMON/LEOXTR/LONX(4,3),LATX(4,3),FTNS,FTEW,NONSTD,CORNR,
     &        maxlvl,OPTION2,cornrgl
      COMMON/LEOFLG/TFLAG,SFLAG,SCOR
      common/LEOUPS/LONNE,LATNE,LONNW,LATNW,LONSW,LATSW,
     &        LONSE,LATSE,CTRLON,CTRLAT,level,lon,lat
      blank2='  '
C
```

```
C******* Array assignments for corner coordinates LON(i,j),LAT(i,j)
C
C (i,j)
C        j=1    j=2    j=3    j=4    j=5    j=6    j=7
C
C  i=1  (1,1)  (1,2)  (1,3)  (1,4)  (1,5)  (1,6)  (1,7)
C
C            6      5      4      3      2      1
C
C  i=2  (2,1)  (2,2)  (2,3)  (2,4)  (2,5)  (2,6)  (2,7)
C
C            7      8      9      10     11     12
C
C  i=3  (3,1)  (3,2)  (3,3)  (3,4)  (3,5)  (3,6)  (3,7)
C
C            18     17     16     15     14     13
C
C  i=4  (4,1)  (4,2)  (4,3)  (4,4)  (4,5)  (4,6)  (4,7)
C
C            19     20     21     22     23     24
C
C  i=5  (5,1)  (5,2)  (5,3)  (5,4)  (5,5)  (5,6)  (5,7)
C
C            30     29     28     27     26     25
C
C  i=6  (6,1)  (6,2)  (6,3)  (6,4)  (6,5)  (6,6)  (6,7)
C
C            31     32     33     34     35     36
C
C  i=7  (7,1)  (7,2)  (7,3)  (7,4)  (7,5)  (7,6)  (7,7)
C
C
C*******
C
        if(tshp.eq.8.and.rang.eq.22.and.ew.eq.'E')goto 541
        if(tshp.eq.8.and.rang.eq.23.and.ew.eq.'E')goto 542
        if(tshp.eq.10.and.rang.eq.23.and.ew.eq.'E')goto 678
        if(tshp.eq.11.and.rang.eq.15.and.ew.eq.'E')goto 738
        if(tshp.eq.11.and.rang.eq.16.and.ew.eq.'E')goto 739
        if(tshp.eq.11.and.rang.eq.17.and.ew.eq.'E')goto 740
        if(tshp.eq.11.and.rang.eq.18.and.ew.eq.'E')goto 741
        if(tshp.eq.11.and.rang.eq.23.and.ew.eq.'E')goto 746
        if(tshp.eq.11.and.rang.eq.24.and.ew.eq.'E')goto 747
        if(tshp.eq.11.and.rang.eq.25.and.ew.eq.'E')goto 748
        if(tshp.eq.12.and.rang.eq.20.and.ew.eq.'E')goto 811
        if(tshp.eq.12.and.rang.eq.21.and.ew.eq.'E')goto 812
        if(tshp.eq.12.and.rang.eq.23.and.ew.eq.'E')goto 814
        IF(tshp.eq.35.and.rang.eq.25.and.ew.eq.'E')goto 2380
        return
C
C           TREC=(TSHP-1)*68
C           IF (EW.EQ.'E') THEN
C              TREC=TREC+43+RANG
C           ELSE
C              TREC=TREC+44-RANG
C           ENDIF
C
541     CONTINUE
```

```
C       8S-22E
        IF(SECT.EQ.12.OR.SECT.EQ.11)THEN
C               These sections are not in the Kansas PLSS due to
C               area excluded for Ft. Leavenworth military reserve.
        ENDIF
        IF(SECT.EQ.1.OR.SECT.EQ.2)THEN
C               These sections are not in the Kansas PLSS due to
C               area excluded for Ft. Leavenworth military reserve
C               and change in state boundary location along Missouri
C               River
        ENDIF
        RETURN
542     CONTINUE
C       8S-23E
        IF(SECT.EQ.6.OR.SECT.EQ.7)THEN
C               These sections are not in the Kansas PLSS due to
C               area excluded for Ft. Leavenworth military reserve.
        ENDIF
        RETURN
678     CONTINUE
C       10S-23E
        IF(SECT.EQ.3)THEN
                  LON(2,5)=-94.82574
                  LAT(2,5)= 39.20122
        ENDIF
        IF(SECT.EQ.3.OR.SECT.EQ.4)THEN
                  LON(2,4)=-94.84435
                  LAT(2,4)= 39.20121
        ENDIF
        IF(SECT.EQ.4.OR.SECT.EQ.5)THEN
                  LON(2,3)=-94.86288
                  LAT(2,3)= 39.20117
        ENDIF
        RETURN
738     CONTINUE
C       11S-15E
c       Section lines are offset as follows:
c               30 - W side
        IF(SECT.EQ.22.OR.SECT.EQ.23)THEN
C                 LON(4,5)=-
C                 LAT(4,5)=
        ENDIF
        IF(SECT.EQ.25)THEN
C                 LON(5,7)=-
C                 LAT(5,7)=
        ENDIF
        RETURN
```

```
739     CONTINUE
c       11S-16E
c       Section lines are offset as follows:
c               14 - W & S sides
c               15 - E
c               23 - N
        IF(SECT.EQ.20.OR.SECT.EQ.21)THEN
C                  LON(4,3)=-
C                  LAT(4,3)=
        ENDIF
        IF(SECT.EQ.21.OR.SECT.EQ.22)THEN
C                  LON(4,4)=-
C                  LAT(4,4)=
        ENDIF
        IF(SECT.EQ.25.OR.SECT.EQ.26)THEN
c                  LON(5,6)=-
c                  LAT(5,6)=
        ENDIF
        IF(SECT.EQ.30)THEN
C                  LON(5,1)=-
C                  LAT(5,1)=
        ENDIF
        RETURN
740     CONTINUE
c       11S-17E
        IF(SECT.EQ.27.OR.SECT.EQ.28)THEN
c                  LON(5,4)=-
c                  LAT(5,4)=
        ENDIF
        IF(SECT.EQ.28.OR.SECT.EQ.29)THEN
c                  LON(5,3)=-
c                  LAT(5,3)=
        ENDIF
        IF(SECT.EQ.29.OR.SECT.EQ.30)THEN
c                  LON(5,2)=-
c                  LAT(5,2)=
        ENDIF
        RETURN
741     CONTINUE
C       11S-18E
        IF(SECT.EQ.27.OR.SECT.EQ.28)THEN
c                  LON(5,4)=-
c                  LAT(5,4)=
        ENDIF
        RETURN
746     CONTINUE
C       11S-23
        IF(SECT.EQ.28.OR.SECT.EQ.29)THEN
                   LON(6,3)=-94.87198
                   LAT(6,3)= 39.05793
                   nonstd=nonstd+100
        ENDIF
        RETURN
```

```
747     CONTINUE
C       11S-24
C       Section lines have offsets as follows:
C               14 - N side
C               22 - E
C               23 - N & W
C               28 - W
C               29 - E
C               31 - E
C               32 - W
        RETURN
748     CONTINUE
C       11S-25
C       Section lines have offsets as follows:
C               17 - S side
C               20 - N
        IF(SECT.EQ.11)THEN
                  LON(2,6)=-94.60690
                  LAT(2,6)= 39.11682
                  nonstd=nonstd+100
        ENDIF
        RETURN
811     CONTINUE
C       12S-20E
C       Section lines have offsets as follows:
C               25 - W side
C               26 - E & W
C               27 - E
        RETURN
812     CONTINUE
C       12S-21E
C       Section lines have offsets as follows:
C               28 - S side
C               33 - N
        RETURN
814     CONTINUE
C       12S-23E
        IF(SECT.EQ.8.OR.SECT.EQ.5)THEN
                  LON(2,2)=-94.89028
                  LAT(2,2)= 39.02944
                  nonstd=nonstd+100
        ENDIF
2380    CONTINUE
C       35S-25E
C       IF(SECT.EQ.7.OR.SECT.EQ.8.OR.SECT.EQ.17.OR.SECT.EQ.18)THEN
C               A land grant extends through these sections, requiring
C               LEO3 to use protracted section corners for the
C               following:
C                  LON(3,2)
C                  LAT(3,2)
C                  LON(4,2)
C                  LAT(4,2)
C       ENDIF
        RETURN
        END
```

```
c
c     utmsub.f --  program for UTM / (Lon,lat) conversion
      subroutine utmsub(zone,lambda,phi,x,y,k,utmdir,utmform)
c
C     This subroutine is based on formulas presented in:
C
C     Snyder, John P., 1987, Map Projections - A Working Manual, U.S.
C          Geological Survey Professional Paper 1395, United States
C          Government Printing Office, Washington, D.C., 383 pages.
C
C     specifically, pages 57-58, 60-64 and 269-271.
c
c**   DECLARATIONS
c
      implicit none
      double precision pie,a0,esquare,eprimsq,n,t,c,a,m,x,y,k
      double precision lambda,lambda0,phi,phi0,k0,m0
      double precision e1,mu,phi1,c1,t1,n1,r1,d
      integer*2 utmdir,zone,utmform
      logical trace
      common/leotest/trace
c
1     continue
c     write(*,*)'utmsub input values:'
c     write(*,*)'zone = ',zone
c     write(*,*)'x = ',x
c     write(*,*)'y = ',y
c     write(*,*)'utmdir = ',utmdir
c     write(*,*)'utmform = ',utmform
c     write(*,*)'lambda = ',lambda
c     write(*,*)'phi = ',phi
c     write(*,*)'k = ',k
c
c
c**   parameters
c
      pie = 3.14159265
      a0 = 6378206.4
      esquare = 0.00676866
      k0=0.9996
      m0=0
c     phi0 = 0
c     phi0=phi0*pie/180
c
c      direction is FROM utm if utmdir = 0
      if(utmdir.eq.0)goto 100
c
c*****      input value
c           phi = latitude, lambda = longitude
C
c*****************************************************************
c
c     the following section involves conversion TO utm
c
c*****************************************************************
c
      if(lambda.gt.0)lambda=-1.0*lambda
      if(utmform.lt.2)then
```

```
        zone = int((180+lambda)/6)+1
        if (TRACE) then
            write(*,*)'zone = ',zone,'  longitude = ',
     &                  lambda,'  latitude = ',phi
        endif
       endif
       lambda0 = -1*((180-zone*6)+3)
       if (TRACE) then
             write(*,*)'lambda0 = ',lambda0
       endif
cc
c**    convert phi, lambda, phi0, lambda0 to radians
c
       phi=phi*pie/180
       lambda=lambda*pie/180
c      phi0=phi0*pie/180
       lambda0=lambda0*pie/180
             if (TRACE) then
                   write(*,*)'lambda0 = ',lambda0
             endif
c
c      Clark 1866 ellipsoid
c
c
c**    (8-12)
c
       eprimsq = esquare/(1-esquare)
             if (TRACE) then
                   write(*,*)'eprimesq = ',eprimsq
             endif
c
c**    (4-20)
c
       n = a0/dsqrt(1-esquare*(dsin(phi))**2)
             if (TRACE) then
                   write(*,*)'N = ',n
             endif
c
c**    (8-13)
c
       t = (dtan(phi)*dtan(phi))
             if (TRACE) then
                   write(*,*)'T = ',t
             endif
c
c**    (8-14)
c
       c = eprimsq*(dcos(phi))**2
             if (TRACE) then
                   write(*,*)'C = ',c
             endif
c
c**    (8-15)
c
       a = (lambda-lambda0)*dcos(phi)
             if (TRACE) then
                   write(*,*)'A = ',a
             endif
```

```
c
c**   (3-22)
c
      m = 111132.0894*phi*180/pie-16216.94*dsin(2*phi)+
     &            17.21*dsin(4*phi)-0.02*dsin(6*phi)
            if (TRACE) then
                  write(*,*)'M = ',m
            endif
c
c**   (8-9)
c
      x = 500000.0+k0*n*(a+(1-t+c)*a**3/6+
     &            (5-18*t+t**2+72*c-58*eprimsq)*a**5/120)
            if (TRACE) then
                  write(*,*)'x = ',x
            endif
c
c**   (8-10)
c
      y = k0*(m-m0+n*dtan(phi)*(a**2/2+(5-t+9*c+4*c**2)*a**4/24+
     &            (61-58*t+t**2+600*c-330*eprimsq)*a**6/720))
            if (TRACE) then
                  write(*,*)'y = ',y
            endif
c
c**   (8-11)
c
      k = k0*(1+(1+c)*a**2/2+(5-4*t+42*c+13*c**2-28*eprimsq)*
     &            a**4/24+(61-148*t+16*t**2)*a**6/720)
            if (TRACE) then
                  write(*,*)'k = ',k
            endif
      goto 999
c*********************************************************************
c
c     the following section involvs conversion FROM utm
c
c*********************************************************************
100   continue
c
      lambda0 = -1*(180-(zone*6)+3)
            if (TRACE) then
                  write(*,*)'lambda0 = ',lambda0
            endif
      lambda0=lambda0*pie/180
c*****      input values
c
c     x=627106.46739009
c     y=4484124.4376708
c
      x=x-500000.0
c
c**   (8-12)
c
      eprimsq = esquare/(1-esquare)
c     write(*,*)'eprimesq = ',eprimsq
c
c**   (8-20)
```

```
c
      m=m0+y/k0
c     write(*,*)'M = ',m
c
c**   (3-24)
c
      e1=(1-dsqrt(1-esquare))/(1+(dsqrt(1-esquare)))
c     write(*,*)'e1 = ',e1
c
c**   (7-19)
c
      mu=m/(a0*(1-esquare/4-3*esquare**2/64-5*esquare**3/256))
c     write(*,*)'mu = ',mu
c
c**   (3-26)
c
      phi1=mu+(3*e1/2-27*e1**3/32)*dsin(2*mu)+(21*e1**2/16-
     &            55*e1**4/32)*dsin(4*mu)+(151*e1**3/96)*sin(6*mu)
c     write(*,*)'phi1 = ',phi1
c
c**   (8-21)
c
      c1=eprimsq*(dcos(phi1))**2
c     write(*,*)'c1 = ',c1
c
c**   (8-22)
c
      t1=(dtan(phi1))**2
c     write(*,*)'t1 = ',t1
c
c**   (8-23)
c
      n1=a0/dsqrt(1-esquare*(dsin(phi1))**2)
c     write(*,*)'n1 = ',n1
c
c**   (8-24)
c
      r1=a0*(1-esquare)/dsqrt((1-esquare*dsin(phi1)**2)**3)
c     write(*,*)'r1 = ',r1
c
c**   (8-25)
c
      d=x/(n1*k0)
c     write(*,*)'d = ',d
c
c**   (8-17)
c
      phi=(phi1-(n1*dtan(phi1)/r1)*(d**2/2-
     &            (5+3*t1+10*c1-4*c1**2-9*eprimsq)*d**4/24+
     &            (61+90*t1+298*c1+45*t1**2-252*eprimsq-3*c1**2)
     &            *d**6/720))*180/pie
c     write(*,1001)'   phi = ',phi
c
c**   (8-18)
c
      lambda=(lambda0+(d-(1+2*t1+c1)*d**3/6+(5-2*c1+28*t1-3*c1**2+
     &            8*eprimsq+24*t1**2)*d**5/120)/dcos(phi1))*180/pie
c     write(*,1001)'lambda = ',lambda
```

```
c
1001  format(a9,f12.6)
c
999   continue
      return
      end
```