APPENDIX B

PROGRAMS SOURCES LISTING

B.1 VAX Fortran

```
      PROGRAM PBVAX
C
C     LOGICAL FLAGS:
C        NOTF=LINE NOT FOUND              LEND=AT END OF LINE
C        PRFL=PROFILE PLOT                STOP=STOP PLOTTING
C
C     INTEGER FLAGS:
C        IERR=ERROR CODE                  JJ=REC'D IN BUF
C        LINE= LINE  PLOTTED              II=BUF WORD PTR
C        LASTR=LAST REC'D OF FILE         LASTL=LAST LINE
C
C     ERROR CODES:
C        0=NO ERROR                       1=ACTION PT FLG ERR
C        2=LINE EXCEEDS LAST              3=EMPTY LINE
C
C     RBUF CONTAINS CURRENT DIGITAL FILE RECORD.
C
C     SUBROUTINES USED: PBTMT.
C                       PBTIO.
C                       PBTRNF.
C                       PBSRH.
C                       PBPLTL.
C                       RFIL.
C ******
C SUBROUTINE PBTMT() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
C PURPOSE: ASK THE USER TO ENTER THE MAP SCALE TO BE PLOTTED. IN
C          ADITION, FOR   PROFILES, IT ASK FOR THE PROFILE STEP-
C          OVER AMOUNT  AND THE ELEVATION EXAGERATION   DESIRED.
C          THE MODEL TO TABLE SCALE IS CALCULATED AND  DISPLAYED
C          TO THE OPERATOR.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL PBTMT(STOP,PRFL)
C
C     WHERE:  STOP - LOGICAL, RETURNED AS  TRUE IF USER  ENTER A
C             CONTROL Z AS A RESPONSE.
C             PRFL- IS SET TO BE TRUE BY CALLER (PB) IF THE FILE
C             TO BE PLOTTED IS A PROFILE FILE.
C ******
C SUBROUTINE PBTIO() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
```

```
C PURPOSE: ASK THE USER TO ENTER INFORMATION WHICH GOVERNS THE
C          LOCATION AND  ORIENTATION OF THE  PLOT ON THE PLOT-
C          TING TABLE.  THE LOCATION OF MODEL  ORIGIN RELATIVE
C          TO THE TABLE INDEX IS CALCULATED AND DISPLAYED.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL PBTIO(STOP)
C
C     WHERE:  STOP - LOGICAL, RETURNED AS TRUE IF USER ENTER A
C             CNTRL Z AS A RESPONSE.
C
C ******
C SUBROUTINE PBTRNF() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
C PURPOSE: USES THE  PARAMETER DETERMINED BY  PBTIO AND  PBTMT,
C          ALONG WITH XMBAR AND YMBAR FROM THE DIGITAL RECORDING
C          FILE TO COMPUTE THE TABLES COORDINATES CORRESPONDING
C          TO THE INPUT MODEL COORDINATES.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL PBTRNF(X,Y)
C
C     WHERE:  X,Y AT ENTRY ARE THE MODEL COORDINATES AND AT EXIST
C             ARE TABLE COORDINATES Y.
C
C ******
C SUBROUTINE PBSRH() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
C PURPOSE: SEARCHES THE DIGITAL RECORDING FILE  FOR THE START OF
C          THE NEXT LINE TO BE PLOTTED. KEEPS A POINTER WHERE OF
C          WHERE THE LINE  IS FOUND, AND STARTS THE NEXT  SEARCH
C          FROM THIS POINT.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL PBSRH
C
C ******
C SUBROUTINE PBPLTL() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
C PURPOSE: CONTROLS THE PLOTTING OF A LINE. FOLLOWING THE LINE
C          THROUGH THE FILE AND CORRECTLY HANDLING PEN UPS AND
C          DELETED POINTS, AND MONITORING THE TERMINAL FOR THE
C          ENTRY  OF A CNTRL Z. THE PEN  IS PUT UP AND  RETURN
C          MADE WHEN THE LINE ENDS.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL PBPLTL
C
```

```
C ******
C SUBROUTINE RFIL() - DEFENSE MAPPING AGENCY, WASHINGTON D.C.
C
C PURPOSE: READS A  RECORD  FROM THE FILE  AND CHECKS  THE LINE
C          POINTERS FOR CONSISTENCY.  INCONSISTENSY IS REPORTED
C          TO THE USER. FOR SERIOUS ERRORS SUCH AS REQUESTING A
C          RECORD  OUTSIDE THE  FILE, THE  CALLING  PROGRAM  IS
C          NOTIFIED VIA A FLAG.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL RFIL(J,LERR)
C
C     WHERE:  J NUMBER OF RECORD TO BE READ.
C             LERR SET TRUE IS SERIOUS PROBLEM WITH RECORD.
C ******
C     LANGUAGE:    VAX FORTRAN version 5.0.
C     DATE:        15 July 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      COMMON       SGM
      COMMON       /TINTOR/COST,SINT,TXOFF,TYOFF
      COMMON         /MTRFRM/SCX,SCY,PINDX,PRSTEP,XMIN,XMAX,YMIN,YMAX
      COMMON       /FLAGL/NOTF,PLOTC,PRFL,LEND,ALL,STOP,PLOT1
      COMMON       /POINT/RBUF,XCOORD,YCOORD,XORG,YORG,EORG
      COMMON       /FLAGI/IERR,LINE,LASTR,LASTL,JJ,II
      BYTE         BBUF(512)
      INTEGER*2    IBUF(256)
      DIMENSION    RBUF(128)
      EQUIVALENCE (BBUF,IBUF,RBUF)
      COMMON       /WFDNEW/LINETP,IOSB,KK,IDPB
      COMMON       /WFD2/LINPTP,PARALE,MTAPE,IPPASS,IMTFST
      LOGICAL      PARALE,MTAPE
      BYTE         NAME(80),YES,BBYTE
      LOGICAL*1    NOTF,PLOTC,PRFL,LEND,STOP,ALL,PLOT1
      DIMENSION    IOSB(2),IDPB(6)
      DATA         YES,IZERO/'Y',0/
      DATA          NAME(1),NAME(2),NAME(3),NAME(4)/'M','S','0',':'/
      DATA         PARALE,NOTF,PLOTC,PRFL,LEND,ALL,STOP/7*.FALSE./
      DATA         IERR,LINE,LASTR,LASTL/4*0/
      DATA         XORG,YORG,EORG/3*0.0/
      DATA         SCX,SCY,COST,SINT/3*1.0,0.0/
C
    1 WRITE (5,5)      !READY?
    5 FORMAT (/'$* ',' READY TO CONVERT FILE?, Y OR N : ')
C
C     CHECK IF FILE IS READY TO CONVERT.
C
      READ (5,140,END=560,ERR=1)BBYTE
      IF (BBYTE .NE. YES) GO TO 560   !EXIT
    3 WRITE(5,1000)
 1000 FORMAT(/'$* FILE ON MAG TAPE ? (Y OR N) :')
```

```
C
C     CHECK IF FILE IS ON MAG TAPE.
C
      READ(5,140) BBYTE
      MTAPE=.FALSE.
      IF(BBYTE.EQ.YES) MTAPE=.TRUE.
 1020 WRITE(5,1030)
 1030 FORMAT(/'$* IS FILE READY ? (Y OR N) : ')
C
C     CHECK IF FILE IS READY.
C
      READ(5,140,ERR=1020,END=560) BBYTE
      IF(BBYTE.NE.YES) GO TO 1020
C
C     SET UP FOR QIO.
C
      IDPB(2)=1                !BYTES IN NAME
C
C     PROMPT FILE NAME, 80 BYTE MAX.
C
   15 WRITE (5,25)
   25 FORMAT (//,'$*  ENTER NAME OF RECORDING FILE : ')
      READ (5,30,END=999,ERR=15) NAME
   30 FORMAT (80A1)
      DO 35 L=1,80             !ASCII NULL IS FILE END
          IF (NAME(L) .EQ. ' ') GO TO 40
   35 M=L
   40 NAME(L+1)=0              !FILE END FOR OPEN CALL
C
C     OPEN DIGITAL RECORDING FILE.
C
      IF(.NOT.MTAPE) OPEN (UNIT=2,NAME=NAME,TYPE='OLD',
     +ACCESS='DIRECT',ERR=900,READONLY,RECORDSIZE=128)
      OPEN (UNIT=2,NAME=NAME,TYPE='OLD',ERR=900,
     +READONLY,BLOCKSIZE=512)
      WRITE (5,55) (NAME(M),M=1,L)    !ADVANCE TO TOP OF PAGE
   55 FORMAT (//'    ',' DIGITAL RECORDING FILE  ',32A1)
      IF(.NOT.MTAPE) GO TO 57
C
C     READ IN THE FIRST 7 RECORDS OF THE DIGITAL RECORDING FILE.
C
      DO 56 I=1,7
   56 READ(2,1111,ERR=800,END=800)
      READ(2,1111,ERR=800,END=800) BBUF
 1111 FORMAT(128A1,128A1,128A1,128A1)
      GO TO 58
  800 WRITE(5,810)
  810 FORMAT(/' *** BAD MAG TAPE FILE ***'/)
      CLOSE(UNIT=2)
      GO TO 515
   57 READ (2'8) BBUF          ! READ REC'D 8
   58 CONTINUE
```

```
C
C     ASSIGN THE VALUES FOR XMIN, XMAX, YMIN, YMAX.
C
      XMIN=RBUF(2)
      XMAX=RBUF(3)
      YMIN=RBUF(4)
      YMAX=RBUF(5)
C
C     SHOW USER THE VALUES FOR XMIN, XMAX, YMIN, YMAX, SECTION
C     SIZE AND THE LAST LINE USED IN FILE.
C
           WRITE (5,70) XMIN,XMAX,YMIN,YMAX,RBUF(6),IBUF(14)  !LIST FILE
   70 FORMAT (5X,' XMIN ',16X,F15.3/5X,' XMAX ',16X,F15.3
     +          /5X,' YMIN ',16X,F15.3/5X,' YMAX ',16X,F15.3
     +          /5X,' SSIZ ',16X,F15.3
     +          /5X,' LAST LINE',I7)
C
C     CHECK TYPE OF FILE.
C
      IF(IBUF(13).EQ.IZERO) GO TO 90    !CONTOUR
      WRITE(5,80)
   80 FORMAT(5X,' PROFILE LINES'/)
      PRFL=.TRUE.
      GO TO 105
   90 WRITE(5,100)
  100 FORMAT(5X,' CONTOUR LINES'/)
      PRFL=.FALSE.
  105 CONTINUE
C
C     ASSIGN THE VALUES OF LAST LINE AND LAST RECORD.
C
  115 LASTL=IBUF(14)                    !LAST LINE
      LASTR=IBUF(1)-1                   !LAST RECORD
      CLOSE(UNIT=2)
      IF(MTAPE) GO TO 155
 1233 READ(2'LASTR,ERR=1234)
      GO TO 155
 1234 WRITE(5,1235) LASTR
 1235 FORMAT(' CAN''T READ LAST RECORD --- ',I6/
     +'$ENTER FILE SIZE : ')
      READ(5,1236) LASTR
 1236 FORMAT(I6)
      GO TO 1233
  120 WRITE (5,130)                     !PLOT THIS FILE?
  130 FORMAT ('$* ',' PLOT CURRENT FILE?, Y OR N : ')
      READ (5,140,END=999,ERR=120)BBYTE
  140 FORMAT (A1)
  141 IF(BBYTE .NE. YES) GO TO 999
  155 XORG=0.0    !FOR PROFILING
C
C     CHECK MODEL/TABLE TRANSFORM.
C
```

```
  180 CALL PBTMT(STOP,PRFL)
C
C     CHECK TABLE ORIENTATION.
C
      CALL PBTIO(STOP,PRFL)
  167 WRITE(5,168)
  168 FORMAT(/'$*   PLOT LINES (Y OR N) : ')
      READ(5,140,END=999,ERR=167) BBYTE
      IF(BBYTE.NE.YES) GO TO 900
C
C     GET TO LINES ON MAG TAPE.
C
      IF(.NOT.MTAPE) GO TO 169
      OPEN (UNIT=2, NAME=NAME, TYPE='OLD',ERR=900,
     +READONLY, BLOCKSIZE=512)
C
C     GO TO DATA RECORDS.
C
      DO 1160 I=1,29
 1160 READ(2,1111)
 1170 READ(2,1111) BBUF
C
C     CHECK IF IT IS A RECORD OF SYMBOLS.
C
      IF(IBUF(6).EQ.-1) GO TO 1170
      LINPTP=0
  169 WRITE (5,170)
  170 FORMAT ('$* ',' READ ALL LINES?, Y OR N : ')
      READ (5,140,END=999,ERR=155)BBYTE
      IF (BBYTE .EQ. YES) ALL=.TRUE.          !ALL LINES
      IF (BBYTE .NE. YES) ALL=.FALSE.
  171 CONTINUE
      LINETP=0
      IF (ALL .EQ. .TRUE.) GO TO 400          !PLOT ALL
C
C     SELECT A LINE MODE.
C
  200 WRITE (5,215)
  215 FORMAT(/'$*   INPUT LINE TYPE TO PLOT : ')
      READ (5,220,END=376) LINETP
      IF(.NOT.MTAPE) GO TO 225
      IF(LINPTP.EQ.0) GO TO 225
      WRITE(5,227)LINETP,LINPTP
  227 FORMAT(' DOES LINE TYPE ',I4,' APPEAR IN THE CATALOG ',
          +'BEFORE LINE TYPE ',I4,'(LAST PLOTTED)'/'IF SO THEN THE MAG',
     +'TAPE WILL BE REWOUND'/'$ENTER ''Y'' , ''N'' ,OR ''R'' ',
     +'( R WILL RE-ASK THE LINE TYPE) : ')
      READ(5,140,ERR=200,END=999) BBYTE
      IF(BBYTE.EQ.'N') GO TO 225
      IF(BBYTE.EQ.'R') GO TO 200
      DO 2160 I=1,52
 2160 READ(2,1111)
```

```
 2170 READ(2,1111) BBUF
      IF(IBUF(6).EQ.-1) GO TO 2170
      LINPTP=0
      LINNM=1
  225 IF(LINETP.LT.101) GO TO 900
  220 FORMAT (I6)
      GO TO 400                          !SELECT NEXT OPERATION
C
C     ENTER HERE FOR EOF ON KEYBOARD READ.
C
  376 CALL CLOSE ( 5 )
      GO TO 900
  400 CONTINUE
      IF(MTAPE) GO TO 403
      WRITE(5,173)
  173 FORMAT('$*  LINE NUMBER TO START WITH : ')
      READ(5,220,ERR=171,END=376) LINNM
      IF(LINNM.GE.1) GO TO 410
  403 LINNM=1
  410 KK=30
      WRITE (5,405)
  405 FORMAT (/5X,' TO ABORT PLOT LOOP, TYPE CTRL/Z'/)
      IL=0
C
C     ALLOW ABORT WITH QIO READ VIRTUAL BLK.
C
  420 CALL PBSRH                 !SEARCH FOR LINE
      IF (NOTF .EQ. .FALSE.) GO TO 425
      IF(LINETP.NE.0)  WRITE(5,422) IL,LINETP
      IF(LINETP.EQ.0 ) WRITE(5,423) IL
  422 FORMAT(3X,I6,' TYPE ',I6,' LINES PLOTTED')
  423 FORMAT(3X,I6,' LINES PLOTTED')
      GO TO 475
  425 IF(LINE.LT.LINNM) LRSTRT=KK
      PLOTC=.FALSE.
      IF (LINE.GE.LINNM) CALL PBPLTL           !PLOT A LINE
      IERR=0
C
C     MONITOR CTRL/Z INPUT.
C
      IF (IOSB(1) .EQ. 246) GO TO 475  !A CTRL/Z
           IF(.NOT.PLOT1.AND. LINE.GE.LINNM) WRITE (5,1265) LINE,LINPTP
 1265 FORMAT('+     ',' LINE',I6,' HAS NO POINTS  TYPE = ',I6)
      IF(PLOTC.EQ. .FALSE.) GO TO 420
      IL=IL+1
  455 WRITE (5,265) LINE,LINPTP        !TYPE LINE PLOTTED
  265 FORMAT('+     ',' LINE',I6,' PLOTTED  TYPE = ',I6)
  465 XORG=XORG+PINDX          !FOR PROFILING
      GO TO 420
  475 CONTINUE
      IF(LINETP.NE.0) GO TO 200
      GO TO 120                !TO MODE SELECT
```

```
C
C     FILE OPEN ERROR.
C
  900 WRITE (5,910)
  910 FORMAT (5X,' FILE OPEN ERROR')
      GO TO 15
C
C     TERMINATE PLOTTING ?.
C
  999 CLOSE(UNIT=2)
      CALL CLOSE ( 5 )
      IERR=0
C
C     GET ANOTHER FILE ?.
C
  515 WRITE (5,525)
  525 FORMAT ('$* ',' GET ANOTHER FILE?, Y OR N : ')
      READ (5,140,END=999,ERR=515) BBYTE
      IF (BBYTE .EQ. YES) GO TO 3     !GET FILE NAME
      CALL CLOSE(3)                   !CLOSE TT0: ASSIGNMENT
  560 CONTINUE
      STOP 'PB EXITING'
      END
C ***************************************************************
      SUBROUTINE PBTMT(STOP,PRFL)
      COMMON      /MTRFRM/SCX,SCY,PINDX,PRSTEP
      COMMON      /TINTOR/COST,SINT,TXOFF,TYOFF,ROT,XMBAR,YMBAR
      COMMON      SGM,ALPHAP
      COMMON      /WFD2/LINPTP,PARALE,MTAPE
      LOGICAL     PARALE,MTAPE
      BYTE        YES,BBYTE
      LOGICAL*1   STOP,PRFL
      DATA        YES/'Y'/
   25 FORMAT (A1)
C
C     DISPLAY TRANSFORM.
C
   35 CONTINUE
   50 CONTINUE
C
C     ENTER TRANSFORM PARAMETERS.
C
      OPEN (UNIT=2,NAME=NAME,TYPE='OLD',READONLY,
     +BLOCKSIZE=512)
  110 WRITE (5,165)
  165 FORMAT (/'$* ',' ENTER MAP SCALE : ')
      READ (5,180,END=999,ERR=110) SCLMAP
  180 FORMAT (F12.0)
      IF(.NOT.MTAPE) GO TO 90
      DO 60 I=1,3
   60 READ(2,4)
C
```

```
C     READ IN VALUES OF MODEL COORDINATES OF MID- BASE POINT AND
C     APROXIMATE AIRBASE TO GROUND ANGLE.
C
           READ(2,4)(DUM,I=1,103),SGM,(DUM,I=1,5),XMBAR,YMBAR,DUM,ALPHAP
C
C     READ X MODEL DISTANCE BETWEEN PROFILES.
C
      READ(2,4) (DUM,I=1,71),PRSTEP
    4 FORMAT(128A4)
      WRITE(5,*) SGM,XMBAR,YMBAR,ALPHAP
      GO TO 91
     90 READ(2'4)(DUM,I=1,103),SGM,(DUM,I=1,5),XMBAR,YMBAR,DUM,ALPHAP
      READ(2'5) (DUM,I=1,71),PRSTEP
   91 SGM=33.95257
      XMBAR=25.97437
      YMBAR=47.11736
      ALPHAP=0.000
      SCY=1000./(SGM*SCLMAP)
      SCX=SCY          !ASSUME CONTOURS
      PINDX=0
      IF (PRFL .EQ. .FALSE.) GO TO 150
      WRITE (5,135)
  135 FORMAT (5X,' IF INDEX=0, PRFL ORIGIN IS FIRST POINT'
     +          /5X,' IF INDEX NOT 0, ORIGIN IS TABLE ORIGIN'/
     + 5X,'AND PROFILES ARE OFFSET BY ''INDEX'''/
     + 5X,'MM ON PLOTTING TABLE')
  138 WRITE (5,140)
  140 FORMAT ('$ * ',' ENTER INDEX VALUE (MM) : ')
      READ (5,260,END=999,ERR=138) PINDXP
  260 FORMAT (F15.0)
      PINDX=PINDXP/SCY         !MAKE INTO MM IN MODEL
  200 WRITE (5,215)
  215 FORMAT ('$* ',' ENTER ELEVATION EXAGGERATION : ')
      READ (5,180,END=999,ERR=200) SCE
      SCX=SCY*SCE
      PINDX=PINDXP/SCX
  150 CONTINUE
      WRITE (5,65) SCLMAP,SCY
   65 FORMAT(' MAP SCALE = ',G15.7/
     +   ' MODEL TO TABLE SCALE = ',G15.7)
      CLOSE (UNIT=2)
      IF(PRFL.EQ. .TRUE.) WRITE(5,66) SCE,PINDXP
   66 FORMAT(' PROFILE ELEVATION EXAGERATION = ',G15.5/
     +   ' INDEX BETWEEN PROFILES = ',G15.5/)
  290 WRITE (5,300)
  300 FORMAT ('$* ',' CORRECT AN ENTRY?, Y OR N : ')
      READ(5,25,END=999,ERR=290) BBYTE
      IF(BBYTE.NE. YES) GO TO 998
      GO TO 110
  999 STOP=.TRUE.
      CALL CLOSE ( 5 )
      RETURN
```

```
  998 STOP=.FALSE.
      RETURN
      END
C ***************************************************************
      SUBROUTINE PBTIO(STOP,PRFL)
      COMMON       /TINTOR/COST,SINT,TXOFF,TYOFF,ROT,XMBAR,YMBAR
      COMMON       /MTRFRM/SCX,SCY,PINDX
      COMMON       SGM,ALPHAP
      COMMON       /WFD2/LINPTP,PARALE,MTAPE
      LOGICAL      PARALE,MTAPE
      BYTE         YES,BBYTE
      LOGICAL*1    STOP,PRFL
      DATA         YES/1HY/
      WRITE (5,50)ROT,TXOFF,TYOFF
   50 FORMAT(/5X,'TABLE ROT',9X,G15.5/5X,' ABS TX
     +  OFFSET(MM)',5X,F15.3
     +  /5X,' ABS TY OFFSET(MM)',5X,F15.3)
   91 WRITE(5,92) ALPHAP
   92 FORMAT(' *  USE ''ALPHAP = ',F7.2,' DEGREES'/
     +  '$*   AS TABLE ROTATION  (Y OR N) : ')
      READ(5,100,END=999,ERR=91) BBYTE
      IDID=0
      IF(BBYTE.NE.YES) GO TO 65
      IDID=1
      ROT=-ALPHAP
   65 COST=COS(ROT*0.0174532925)
      SINT=SIN(-ROT*0.0174532925)
      TX=0
      TY=0
      IF(PRFL) TXOFF=0
      IF(PRFL) TYOFF=0
      CALL PBTRNF(TX,TY)
      IF(PRFL)TXOFF=-TX
      IF(PRFL)TYOFF=-TY
      IF(.NOT.(PRFL)) WRITE(5,70) TX,TY
   70 FORMAT(' MODEL CENTER IS AT  X = ',F10.0,'  Y = ',F10.0/
     +  ' WITH RESPECT TO THE TABLE INDEX POINT'/)
      WRITE (5,80)
   80 FORMAT('$* ',' CHANGE TABLE ORIENTATION VALUES? Y OR N : ')
      READ (5,100,END=999,ERR=65)BBYTE
  100 FORMAT(A1)
      IF(BBYTE .NE. YES)GO TO 998
C
C     ENTER VALUES.
C
      IF(IDID.EQ.1) GO TO 170
  120 WRITE (5,135)
  135 FORMAT('$* ',' ENTER ROTATION (DEG) : ')
      READ (5,150,END=999,ERR=120)ROT
  150 FORMAT(F9.0)
      COST=COS(ROT*0.0174532925)
      SINT=SIN(-ROT*0.0174532925)
```

```
  170 IF(PRFL) GO TO 240

      WRITE (5,190)
  190 FORMAT('$* ',' ENTER ABS TABLE X OFFSET (MM) : ')
      READ (5,200,END=999,ERR=170)TXOFF
  200 FORMAT(F15.0)
  220 WRITE (5,230)
  230 FORMAT ('$* ',' ENTER ABS TABLE Y OFFSET (MM) : ')
      READ (5,200,END=999,ERR=220)TYOFF
  240 WRITE(5,50) ROT,TXOFF,TYOFF
  250 WRITE (5,270)
  270 FORMAT ('$*   CORRECT AN ENTRY?, Y OR N : ')
      READ(5,100,END=999,ERR=250) BBYTE
      IF(BBYTE.EQ.YES) GO TO 91
      GO TO 998
  999 STOP=.TRUE.
      CALL CLOSE ( 5 )
      RETURN
  998 STOP=.FALSE.
      RETURN
      END
C *****************************************************************
      SUBROUTINE PBTRNF(TX,TY)
      COMMON       /TINTOR/COST,SINT,TXOFF,TYOFF,ROT,XMBAR,YMBAR
      COMMON       /MTRFRM/SCX,SCY,PINDX
      WRITE(5,*) 'PBTRNF'
      X=(TX+XMBAR)*SCX      !SAME AS SCE FOR PRFL
      Y=(TY+YMBAR)*SCY      !SAME AS SCXY FOR PRFL
      TX=TXOFF+X*COST+Y*SINT
      TY=TYOFF-X*SINT+Y*COST
  999 RETURN
      END
C *****************************************************************
      SUBROUTINE PBSRH
      COMMON       /FLAGL/NOTF,PLOTC,PRFL,LEND,ALL,STOP
      COMMON       /POINT/RBUF,XCOORD,YCOORD,XORG,YORG,EORG
      COMMON       /FLAGI/IERR,LINE,LASTR,LASTL,JJ,II
      BYTE         BBUF(512)
      INTEGER*2    IBUF(256)
      DIMENSION    RBUF(128)
      EQUIVALENCE (BBUF,IBUF,RBUF)
      COMMON       /WFDNEW/LINETP,IOSB(2),KK
      COMMON       /WFD2/LINPTP,PARALE,MTAPE
      LOGICAL      PARALE,MTAPE
      LOGICAL*1    NOTF,PLOTC,PRFL,LEND,ALL,STOP
      LEND=.FALSE.
      NOTF=.TRUE.
      IF(MTAPE) GO TO 100
C
C     SEQUENTIAL RECORD SEARCH FOR BACKWARD PTR -1.
C
      IF(KK.GT.LASTR) GO TO 50
```

```
      DO 20 JJ=KK,LASTR
          READ (2'JJ,ERR=50) IBUF          !DIGITAL FILE
          IF(IBUF(6).LE.0) GO TO 20        !SYMBOL RECORD
          IF (IBUF(2) .NE. -1) GO TO 20
          IF(LINETP.EQ.0) GO TO 30
          IF (IBUF(6) .EQ. LINETP) GO TO 30
   20 CONTINUE
      GO TO 40
   30 NOTF=.FALSE.
      LINE=IBUF(7)
      LINPTP=IBUF(6)
      IF (IBUF(1) .EQ. -1) LEND=.TRUE.
      WRITE (5,*) LINE,LINPTP,IBUF(1),LINETP
   40 KK=JJ+1
   50 RETURN
  100 CONTINUE
      IF(LINPTP.EQ.0) GOTO 115
      IF(LINETP.NE.0 .AND. LINETP.NE.LINPTP) GO TO 115
  110 READ(2,1111,END=40,ERR=40)IBUF
 1111 FORMAT(128A2,128A2)
      IF(IBUF(2).NE.-1) GO TO 110      !NOT LINE START SO IGNORE
  115 CONTINUE
      IF(LINETP.EQ.0) GO TO 30
  120 IF(IBUF(6).NE.LINETP .AND. LINETP.EQ.LINPTP) GO TO 40
      IF(IBUF(6).EQ.LINETP) GO TO 30
      GO TO 110
      END
C ***************************************************************
      SUBROUTINE PBPLTL
      COMMON       /FLAGL/NOTF,PLOTC,PRFL,LEND,ALL,STOP,PLOT1
      COMMON       /POINT/RBUF,XCOORD,YCOORD,XORG,YORG,EORG
      COMMON       /FLAGI/IERR,LINE,LASTR,LASTL,JJ,II
      COMMON       /WFDNEW/LINETP,IOSB(2),KK,IDPB(6)
      COMMON       /WFD2/LINPTP,PARALE,MTAPE,IPPASS,IMTFST
      LOGICAL*1    NOTF,PLOTC,PRFL,LEND,ALL,STOP,PLOT1
      LOGICAL      LERR,PARALE,MTAPE
      REAL         XCORDS,YCORDS,ECORDS,XCOORD,YCOORD,ECOORD
      BYTE         BBUF(512)
      INTEGER*2    IBUF(256)
      DIMENSION    RBUF(128)
      EQUIVALENCE (BBUF,IBUF,RBUF)
      JOLD=JJ          !REMEMBER WHERE THE LINE STARTS
      JJOLD=-1
   25 PLOTC=.FALSE.            !LINE COMPLETE
      WRITE(5,*)
      PLOT1=.FALSE.            !A POINT PLOTTED
      IPEN=0                   !PEN UP FOR PT 1
      PARALE=.FALSE.   !NOT DOING
      IPPASS=0         !A
      IMTFST=0         !PARALLEL LINE
C
C     GO TO 1ST PT WITH PEN UP.
```

```
C
      II=11                        !INTEGER WORD PTR
      M=6                          !REAL WORD PTR
      GO TO 150
   85 XCOORD=RBUF(M)
      YCOORD=RBUF(M+1)
      ECOORD=RBUF(M+2)
      IF (XCOORD .GE. 9E9) WRITE (3,*) '9.000005E+09'
      IF (XCOORD .LT. 9E9) GO TO 250  !A POINT
C
C     ACTION PROCESSING.
C
  110 IF(ECOORD.GE.0) GO TO 150
      II=II+6
      M=M+3
C
C     TRY NEXT POINT.
C
  150 II=II+6
      M=M+3
      IF (II .LT. IBUF(5)) GO TO 85   !NEXT PT
C
C     END OF BUFFER.
C
  180 IF (LEND .EQ. .FALSE.) GO TO 210 !GET NEXT REC'D
      IERR=3                       !NO PTS IN LINE
      GO TO 950                    !ERROR TEST
C
C     GET NEXT REC'D.
C
  210 J=IBUF(1)
      CALL RFIL(J,LERR)
      IF(LERR) GO TO 770
      JJOLD=JJ
      JOLD=J
      JJ=J
      IF (IBUF(1) .EQ. -1) LEND=.TRUE.
  225 GO TO 25                     !RESTART
C
C     GO TO PT 1, PEN UP.
C
  250 IPEN=0
      PLOT1=.TRUE.
      XCORDS=XCOORD
      YCORDS=YCOORD
      ECORDS=ECOORD
      OPEN (UNIT=3,NAME='YOYETO.999',TYPE='NEW')
      WRITE (3,*) XCORDS,YCORDS,ECORDS
  251 IF (II .LT. IBUF(5)) GOTO 150
C
C     DO PARALLEL LINES IF NECESSARY.
C
```

```
      PARALE=.TRUE.
      WRITE (5,*)
      IMTFST=1
      WRITE (5,*) LERR
      IF(LERR) GO TO 770
      LEND=.FALSE.
      JJ=JJOLD                  !MAKE RFIL CHECKS WORK
      IPPASS=1                  !MARK SECOND PASS
      CALL RFIL(JOLD,LERR)      !BACK TO LINE BEGINNING
      IF(LERR) GO TO 770
      JJ=JOLD
      IF(LERR) GO TO 770
      LEND=.FALSE.
      JJ=JJOLD
      IPPASS=2                  !MARK THIRD PASS
      CALL RFIL(JOLD,LERR)      !BACK TO LINE BEGINNING
      WRITE (5,*) LERR
      IF(LERR) GO TO 770
      JJ=JOLD
      XCOORD=XCORDS
      YCOORD=YCORDS
      ECOORD=ECORDS
      IPEN=0
      CALL PBPT(IPEN,PRFL)      !PLOT PT
      II=II+6                   !PT TO NEXT X
  600 CONTINUE
      PLOT1=.TRUE.
  650 N=II
      IXEND=IBUF(5)-5           !LAST X COORD
  680 IF (II .GT. IXEND) GO TO 780    !REC'D END
      M=II/2+1                  !REAL PTR
      IF (RBUF(M) .LT. 9E9) GO TO 720 !PLOT
C
C     ACTION PT.
C
      IF(RBUF(M+2).EQ.0) IPEN=3
      IF(RBUF(M+2).LT.0) II=II+6
      GO TO 750
  720 CALL PBPT(IPEN,PRFL)      !PLOT PT
  750 II=II+6
      GO TO 680
C
C     LINE END TEST.
C
  770 LEND=.TRUE.
  780 IF (LEND .EQ. .TRUE.) GO TO 930 !LINE END
  810 J=IBUF(1)                 !GET NEXT REC'D
      IF(LERR) GO TO 770
      JJ=J
      II=17                     !1ST X COORD
      IF (IBUF(1) .EQ. -1) LEND=.TRUE.
      GO TO 650
```

```
C
C     EXIT TESTS.
C
  930 IF (PLOT1 .EQ. .FALSE.) GO TO 950        !ERROR
      PLOTC=.TRUE.
      IPEN=3            !PEN UP AT LAST POINT ON LINE
  950 CONTINUE                  !NO PTS, OR ERROR
      PARALE=.FALSE.    !NOT DOING
      IPPASS=0          !A
      IMTFST=0          !PARALELL LINE
      RETURN
      END
C ***************************************************************
      SUBROUTINE PBPT(IPEN,PRFL)
           COMMON    /MTRFRM/SCX,SCY,PINDX,PRSTEP,XMIN,XMAX,YMIN,YMAX
      COMMON       /POINT/RBUF,XCOORD,YCOORD,XORG,YORG,EORG
      COMMON       /FLAGI/IERR,LINE,LASTR,LASTL,JJ,II
      BYTE         BBUF(512)
      DIMENSION    RBUF(128)
      DIMENSION    IBUF(256)
      EQUIVALENCE (BBUF,IBUF,RBUF)
      DIMENSION    IX(2),IY(2),IE(2)
      EQUIVALENCE (XCOORD,IX),(YCOORD,IY),(ECOORD,IE)
      LOGICAL*1    PRFL
      LOGICAL      PARALE
      COMMON       /WFD2/LINPTP,PARALE
      DATA         IFRSTP/0/
      DATA         OLX,OLY/9E8,9E8/
      WRITE (5,*)
      DO 10 K=1,2
          IX(K)=IBUF(II+K-1)       !XCOORD
          IY(K)=IBUF(II+2+K-1)     !YCOORD
          IE(K)=IBUF(II+4+K-1)     !ECOORD
          WRITE (5,*) II+K-1,K,IX(K),IY(K)
   10 CONTINUE
      WRITE (5,*) PRFL
   12 IF (PRFL .EQ. 0) GO TO 400         !CONTOUR
C
C     PROFILING.
C
      IF (IPEN .NE. 0) GO TO 200                !CONTINUE LINE
   30 IF (PINDX .EQ. 0.0) GO TO 40
      YORG=YORG+(YCOORD-Y1ST)
      IF(IFRSTP.EQ.0) YORG=0.0
      IF(IFRSTP.EQ.0) IFRSTP=1
      GO TO 60
   40 XORG=XCOORD               !POINT 1
      YORG=YCOORD
   60 X1ST=XCOORD
      Y1ST=YCOORD
      E1ST=ECOORD
C
```

```
C     MOVE TO PRFL ORIGIN.
C
  100 X=XORG
      Y=YORG
C
C     PEN 2 FOR EVERY TENTH PROFILE.
C
      RX=(XCOORD-XMIN)/PRSTEP +.5
      IRX=RX
   20 MIRX=MOD(IRX,10)
      IF(MIRX.GE.10) GO TO 20
      CALL PBTRNF(X,Y)          !TABLE COORD TRANS
      IPEN=1                       !1ST PT PLOTTED
      GO TO 999                    !EXIT
C
C     CONTINUE PRFL LINE.
C
C     XYDIST & ELEV ARE OFFSETS FROM ORIGIN.
C
  200 ELEV=XORG+(ECOORD-E1ST)
      XSGR=(XCOORD-X1ST)*(XCOORD-X1ST)
      YSGR=(YCOORD-Y1ST)*(YCOORD-Y1ST)
      XYDIST=YORG+SQRT(XSGR+YSGR)
      CALL PBTRNF(ELEV,XYDIST)          !TABLE COORD TRANS
      XCOORD=ELEV
      YCOORD=XYDIST
           IF(ABS(OLX-ELEV).LT..3 .AND. ABS(OLY-XYDIST).LT..3) GO TO 999
      OLX=ELEV
      OLY=XYDIST
      IPEN=1                            !AT LEAST 1 PLOTTED
      GO TO 999                         !EXIT
C
C     CONTOUR, IPEN=PEN UP/DOWN FLAG.
C
  400 CALL PBTRNF(XCOORD,YCOORD)        !TABLE COORD TRANS
  999 CONTINUE
      RETURN
      END
C ***************************************************************
      SUBROUTINE RFIL(J,LERR)
      COMMON       /FLAGL/NOTF,PLOTC,PRFL,LEND,ALL,STOP
      COMMON       /POINT/RBUF,XCOORD,YCOORD,XORG,YORG,EORG
      COMMON       /FLAGI/IERR,LINE,LASTR,LASTL,JJ,II
      BYTE         BBUF(512)
      DIMENSION    RBUF(128)
      INTEGER*2    IBUF(256)
      EQUIVALENCE (BBUF,IBUF,RBUF)
      COMMON       /WFDNEW/LINETP,IOSB(2),KK,IDPB(6)
      COMMON       /WFD2/LINPTP,PARALE,MTAPE,IPPASS,IMTFST
      LOGICAL      PARALE,MTAPE
      LOGICAL*1    NOTF,PLOTC,PRFL,LEND,ALL,STOP,PLOT1
      LOGICAL      LERR
```

```
      BYTE        BUF(80)
      DIMENSION   IPRL(6)
      DATA        IOWBT/"500/
      DATA        IFRT/0/
      IF(IOSB(1).EQ.246) GO TO 770            !CTL Z CHECK
      LERR=.FALSE.
      IF(MTAPE) GO TO 800
      IPRL(2)=80
      IPRL(3)="40
   10 CONTINUE
      IF(IBUF(1).LE.LASTR .AND. IBUF(2).LE.LASTR .AND.
     +  IBUF(3).LE.LASTR .AND. IBUF(4).LE.LASTR) GO TO 200
      ENCODE(80,1237,BUF) JJ
 1237 FORMAT(' PB - REC ',I6,' POINTS OUTSIDE FILE')
      IF(J.GT.LASTR) LERR=.TRUE.
      IF(LERR) RETURN
  200 CONTINUE
      READ(2'J,ERR=770)BBUF
      IF(IBUF(2).EQ.JJ) GO TO 210
      ENCODE(80,205,BUF) J,JJ
  205 FORMAT(' PB - REC ',I6,' DOES NOT POINT BACK TO REC ',I6)
  210 CONTINUE
      IF(IBUF(6).EQ.LINPTP) GO TO 213
      ENCODE(80,1234,BUF) LINPTP,IBUF(6),J,JJ
 1234 FORMAT(' PB -  LN TYPE FROM ',I5,' TO ',I5,
     +' AT REC = ',I5,' FROM REC = ',I5)
      IF(IBUF(6).LE.0) GO TO 770
  213 IF(IBUF(7).EQ.LINE) GO TO 215
      ENCODE(80,1235,BUF) LINE,IBUF(7),J,JJ
 1235 FORMAT(' PB -  LN # FROM ',I5,' TO ',I5,
     +' AT REC = ',I5,' FROM REC = ',I5)
  215 IF(IBUF(5).LE.256) GO TO 220
      ENCODE(80,1236,BUF) J,JJ
 1236 FORMAT(' PB - LASWRD GT THAN 256 AT REC ',I5,
     +'  FROM REC = ',I5)
      LERR=.TRUE.
  220 CONTINUE
      RETURN
  770 LERR=.TRUE.
      RETURN
  800 CONTINUE
      IF(PARALE.AND. (IPPASS.NE.0)) GO TO 900
  820 READ(2,1111,ERR=770,END=770) BBUF
 1111 FORMAT(128A1,128A1,128A1,128A1)
      IF(.NOT.(PARALE)) GO TO 220
      IMTFST=IMTFST+1
      IRD=1
      GO TO 220
  900 CONTINUE
      IRD=IRD+1
      IF(IRD.EQ.(IMTFST+1)) IRD=1
      GO TO 220
```

```
      END
C ************************************************************
```

B.2 Lahey Fortran

B.2.1 Listing of Program PSCPREIN

```
      PROGRAM PSCPREIN
C
C     This program is for converting from a IGMXXXX.DAT format written
C     by PDP11 computer to the PCGiant "PREP.IN" format. Note that it
C     have been assigned 0 for the radial lens distorsion, tangential
C     lens distorsion and atmospheric refraction line. They should be
C     corrected in the PCGiant run.
C
C     DATA FORMS FOR THE IMPUT FILE: PREP.IN (GPA ASSOCIATES).
C
C     OPTIONS/FOCAL LENGTH & IMAGE WEIGHTS LINE: FORMAT (2I,6F,A8)
C        3 4 5 6 OR 8 <=> 3- 4- 5- 6- OR 8- PARAMETER TRANSFORMATION
C         1 <=> CREATE MODEL.IN FILE, 0 DO NOT.
C        CALIBRATED FOCAL LENGTH OF LENS (WITH PROPER SIGN) (IN MM).
C         STANDARD DEVIATION OF IMAGE POINTS X-COMPONENT (IN MM).
C         STANDARD DEVIATION OF IMAGE POINTS Y-COMPONENT (IN MM).
C         X-COORDINATE PRINCIPAL POINT OFFSET.
C         Y-COORDINATE PRINCIPAL POINT OFFSET.
C         UNITS CONVERSION FACTOR TO MILLIMETERS.
C         CAMERA ID (A8).
C
C     CALIBRATED FIDUCIAL COORDINATES          FORMAT (I,2F).
C     ENDING LINE.                             FORMAT (0 0 0).
C
C     LENS DISTORSION LINES:
C         RADIAL LENS DISTORSION FUNCTION:
C            COEFFICIENTS FK0, FK1, FK2, FK3  FORMAT (4D OR 0 0 0 0)
C         DECENT LENS DISTORSION FUNCTION:
C             COEFFICIENTS FJ1, FJ2, PHI       FORMAT (3D OR 0 0 0)
C
C     ATMOSPHERIC REFRACTION LINE
C        ATMOSPHERIC REFRACTION # OF ENTRIES  FORMAT (I OR 0)  < 50
C
C     IF PREVIOUS LINE HAD A NUMBER GREATER THAN ZERO:
C         ATMOSPHERIC REFRACTION DATA IN TABLE  FORMAT (2F)
C
C     MEASURED DATA SET:              REPEAT FOR EACH FRAME MEASURED:
C     *    FRAME IDENTIFICATION IN                FORMAT (A)
C     *    OBSERVED FIDUCIAL COORDINATES IN       FORMAT (I,2F) <2000
C     *    ENDING LINE                            FORMAT (0 0 0)
C     *    OBSERVED PLATE COORDINATES IN          FORMAT (A,2F)  < 99
```

```
C     *   ENDING LINE                                FORMAT (0 0 0)
C     END OF JOB LINE   (8 ASTERISKS)                FORMAT (********)
C
C     SUBROUTINE USED: MERGE.
C
C ******
C SUBROUTINE MERGE() - PROYECTO ESPOL - UNO.
C
C PURPOSE: MEASURED FRAME DATA PER FLIGHTLINE.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL MERGE
C ******
C     LANGUAGE:    Lahey Fortran Version 5.1.
C     DATE:        28 January 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      IMPLICIT NONE
      EXTERNAL SYSTEM
      INTEGER    I, PARTRA, MOFIL, ENDLIN, DUMMY, ATMRE, NOPL
      REAL       CFL, STDX, STDY, PPOX, PPOY,  UCTM
      REAL       RFSX(8), RFSY(8)
      CHARACTER  IEND*8, CAMID*5, F2*8
      DATA IEND  /'********'/
      WRITE (*, 10)
  10  FORMAT (////////37X, 'PSCPREIN'//27X, 'A PC Utility',
     . /18X,  "for the Conversion of an PDP11 'IGMxxxx.DAT'"
     . /23x,  "file to a PC Giant 'xxxx.IN' file. "
     . //20X, '(c) Proyecto ESPOL - UNO'////////)
      PAUSE
C
C     CREATE FILE PRE.IN.
C
      F2='PREP.IN'
      OPEN(2,f2,STATUS='NEW')
C
C     SET UP VALUES FOR OPTIONS/FOCAL LENGHT & IMAGE WEIGHT LINES.
C
      PARTRA =  3
      MOFIL  =  1
      STDX   =  0.010
      STDY   =  0.010
      CFL    =  152.350
      PPOX   = -0.010
      PPOY   = -0.010
      UCTM   =  0
      CAMID  =  'CAMID'
C
C     WRITE OUT OPTIONS/FOCAL LENGHT & IMAGE WEIGHT LINES.
C
            WRITE(2, '(2I2, 5F10.3, I2, A8)')PARTRA, MOFIL, CFL, STDX, STDY,
```

```
     .PPOX, PPOY, UCTM, CAMID
C
C      SET UP VALUES FOR CALIBRATED FIDUTIAL COORDINATES.
C
       RFSX(1) = -112.001
       RFSY(1) = -0.005
       RFSX(2) = -110.000
       RFSY(2) = -110.010
       RFSX(3) =  0.000
       RFSY(3) = -112.009
       RFSX(4) =  109.996
       RFSY(4) = -109.997
       RFSX(5) =  112.001
       RFSY(5) =  0.005
       RFSX(6) =  109.996
       RFSY(6) =  110.003
       RFSX(7) =  0.000
       RFSY(7) =  112.005
       RFSX(8) = -110.002
       RFSY(8) =  109.995
       ENDLIN  =  0
       DUMMY   =  0
       ATMRE   =  0
C
C      WRITE OUT CALIBRATED FIDUTIAL COORDINATES.
C
       DO I=1, 8, 1
           WRITE (2, '(I4, 2F10.3)') I, RFSX(I), RFSY(I)
       ENDDO
C
C      WRITE OUT ENDING LINE.
C
       WRITE (2, '(3I2)') ENDLIN, ENDLIN, ENDLIN
C

C      WRITE OUT LENS DISTORSION LINE.
C
       WRITE (2, '(4I2)') DUMMY, DUMMY, DUMMY, DUMMY
       WRITE (2, '(3I2)') DUMMY, DUMMY, DUMMY
C
C      WRITE OUT ATMOSPHERIC REFRACTION LINE.
C
       WRITE (2, '(I2)') ATMRE
       CALL SYSTEM ('CLS')
       WRITE (*, 15)
    15  FORMAT (////' ENTER THE NUMBER OF PHOTOS FOR THIS FLIGHTLINE:')
       READ (*, *)   NOPL
       DO I=1, NOPL, 1
C
C          READ FRAME MEASUREMENTS FOR THE FLITGHLINE AND WRITE OUT TO
C           PREP.IN.
C
```

```
          CALL MERGE
        ENDDO
C
C       WRITE OUT END OF JOB LINE.
C
        WRITE (2, '(A8)') IEND
        CALL SYSTEM ('CLS')
        WRITE (*,'(////////)')
        WRITE (*, *) ' PHOTOS PROCESSED & WRITTEN TO FILE ',  F2
        END
C     ---------------------------------------------------------------
        SUBROUTINE MERGE
        INTEGER    I,J,K, ENDLIN
        REAL       PHONY(650), X1(40), Y1(40), X2(40), Y2(40)
        CHARACTER  ID(40)*6, PHOTOID*4, FNAME*8, F1*12, IDA(650)*6
        CALL SYSTEM ('CLS')
        ENDLIN = 0
        WRITE(*, 20)
     20 FORMAT (//////////'   ENTER IGMXXXX (PART OF INPUT FILENAME TO
       .LEFT OF DECIMAL): ')
        READ(*, *)FNAME
            DO I=8, 1, -1
                IF(FNAME(I:I).NE.' ')EXIT
            ENDDO
        PHOTOID=FNAME(4:I)
        F1=FNAME(1:I)//'.DAT'
C
C       OPEN FILE IGMXXXX.DAT.
C
        OPEN(1,F1,STATUS='OLD')
        F1=FNAME(4:I)//'.IN'
        DO I=1, 47, 1
            READ(1, *)
        ENDDO
C
C       READ OBSERVED FIDUCIAL: X COORDINATE.
C
        DO I=48, 55, 1
            READ(1, *) IDA(I), PHONY(I)
            J=I-47
            X1(J)=PHONY(I)
        ENDDO
        DO I=56, 65, 1
            READ(1, *)
        ENDDO
C
C       READ OBSERVED FIDUCIAL: Y COORDINATE.
C
        DO I=66, 73, 1
            READ(1, *) IDA(I), PHONY(I)
            J=I-65
            Y1(J)=PHONY(I)
```

```
      ENDDO
      DO I=74, 292, 1
          READ (1, *)
      ENDDO
C
C     READ OBSERVED XPLATE COORDINATES.
C
      DO I=293, 324, 1
          READ(1, *) IDA(I), PHONY(I)
          IF (PHONY(I)>999) THEN
              GOTO 30
          ENDIF
          J=I-292
          X2(J)=PHONY(I)
          ID(J)=IDA(I)
      ENDDO
   30 R=I+1
      DO J=R, 324, 1
          READ (1, *)
      ENDDO
C
C     READ OBSERVED YPLATE COORDINATES.
C
      DO J=325, I+31, 1
          READ(1, *) IDA(J), PHONY(J)
          IF (PHONY(J)<999) THEN
              K=J-324
              Y2(K)=PHONY(J)
          ENDIF
      ENDDO
      WRITE (2, '(A4)')PHOTOID
      DO J=1, 8, 1
C
C         WRITE OUT OBSERVED FIDUCIAL COORDINATES.
C
          WRITE (2, '(A12, 2F14.3)') PHOTOID//'00', X1(J), Y1(J)
      ENDDO
C
C     WRITE OUT ENDING LINE.
C
      WRITE (2, '(3I2)') ENDLIN, ENDLIN, ENDLIN
      CALL SYSTEM ('CLS')
C
C     READ THE LABEL FOR EACH OBSERVED PLATE COORDINATE.
C
      DO J=1, I-293, 1
                  WRITE (*, "(' ENTER THE LABEL FOR THE POINT:', A6)") ID(J)
          READ (*, *) ID(J)
      ENDDO
C
C     WRITE OUT OBSERVED PLATE COORDINATES.
C
```

```
      DO J=1, I-293, 1
          WRITE (2, '(A12, 2F14.3)') ID(J), X2(J), Y2(J)
      ENDDO
C
C     WRITE OUT ENDING LINE.
C
      WRITE (2, '(3I2)') ENDLIN, ENDLIN, ENDLIN
      CLOSE (1)
      I = 0
      J = 0
      K = 0
      END
```

B.2.2 Listing of Program DSCMODEL

```
      PROGRAM dscMODEL
C
C     This program is for creating the models for compilation on the
C     DSC's the data is taken from the output of a GIANT run and the
C     parameter file MODEL.DAT.  If you NEED to change a value  from
C     MODEL.DAT file, you can do it by editing and renaming this file
C     before running this program.
C
C     FILE MANAGEMENT:
C
C     LOGICAL NAME       FILE
C         1             gxxxx.dat
C         2             model.in
C         3             obj.out
C         4             img.dat
C         5             model.dat
C
C     SUBROUTINES USED: BEEP.
C                       MODEL.
C                       OBJ.
C                       MOHEAD.
C                       MOTAIL.
C ******
C SUBROUTINE BEEP() - PROYECTO ESPOL - UNO.
C
C PURPOSE:
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL BEEP
C ******
C SUBROUTINE MODEL() - PROYECTO ESPOL - UNO.
C
C PURPOSE: EXTRACT INFORMATION FROM THE MODEL.IN FILE.
```

```
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL MODEL
C ******
C SUBROUTINE OBJ - PROYECTO ESPOL - UNO.
C
C PURPOSE : SEARCH AND WRITE INFORMATION OF GROUND AND PLATE COORDI-
C           NATES.
C
C FORTRAN CALLING PROCEDURE :
C
C      CALL OBJ (
C ******
C SUBROUTINE MOHEAD() - PROYECTO ESPOL - UNO.
C
C PURPOSE :  WRITES HEADER FOR MODEL FILE.
C
C FORTRAN CALLING PROCEDURE :
C
C     CALL MOHEAD
C ******
C SUBROUTINE MOTAIL() - PROYECTO ESPOL - UNO.
C
C PURPOSE : WRITES TRAILER FOR MODEL FILES.
C
C FORTRAN CALLING PROCEDURE :
C
C     CALL MOTAIL
C ******
C     LANGUAGE:    Lahey Fortran Version 5.1.
C     DATE:        26 December 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      IMPLICIT NONE
      EXTERNAL SYSTEM
      CHARACTER*64  CFILE1
      CHARACTER*8   F2
      CHARACTER*1   CYN
      WRITE (*, 10)
   10 FORMAT (////////35X, 'DSCModel'//33X, 'A PC Utility',
     . /14X,  "for the Creation of the model files 'Gxxxx.DAT'from"
          . /14x, "a PC Giant run and a parameter file called MODEL.DAT"
     . /15x,  "these files are ready for compilation on the DSC."
     . //31X, 'Proyecto ESPOL - UNO'////////)
      CALL BEEP
      PAUSE
      F2='MODEL.IN'
      OPEN(2,F2,STATUS='OLD')
      CALL SYSTEM ('CLS')
  100 CONTINUE
      WRITE(*,*)'ENTER MODEL FILE NAME IF DIFERENT THAN MODEL.DAT:'
```

```
      READ(*,2100)CFILE1
 2100 FORMAT(A64)
      WRITE(*,*)'OPENING:',CFILE1
      CALL MODEL
C
C     ASK IF USER WANTS TO RUN PROGRAM AGAIN.
C
      WRITE(*,*)'PROCESS ANOTHER FILE Y/N?'
      READ(*,3100)CYN
 3100 FORMAT(A1)
      IF(CYN.EQ.'Y' .OR. CYN.EQ.'y') GOTO 100
C
C     OTHERWISE JUST QUIT.
C
      WRITE(*,*)'PROGRAM DONE.'
      END
C     ------------------------------------------------------------
      SUBROUTINE BEEP
      WRITE (*, *) CHAR(7)
      END
C     ------------------------------------------------------------
      SUBROUTINE MODEL()
      IMPLICIT NONE
      EXTERNAL SYSTEM
      INTEGER    POINTS, I
      CHARACTER  CFILE2*12, PHOTOX*8, PHOTOY*8
  510 CONTINUE
      READ(2, *, END=8100) PHOTOX, PHOTOY, POINTS
          DO I=8, 1, -1
              IF(PHOTOX(I:I).NE.' ')EXIT
          ENDDO
      CFILE2='G'//PHOTOX(1:I)//'.DAT'
      OPEN(1,FILE=CFILE2,STATUS='NEW')
      WRITE(*,*)'OPENING:',CFILE2
      WRITE(*,*)'MODEL=',CFILE2
      WRITE(*,*)'PROCESSING...'
C
C     ---START PARSE LIST
C
C     CALL MODEL.IN FOR INFORMATION ABOUT THE MODELS.
C
      CALL OBJ(PHOTOX, PHOTOY, POINTS)
C
C     ---END OF PARSE LIST
C
      GOTO 510
 8100 CONTINUE
C
C     DONE READING FILE.
C     FINISH UP.
C     CLOSE UP FILES.
C
```

```
      CLOSE(2)
      CALL SYSTEM ('CLS')
      WRITE (*,'(////////)')
           WRITE (*, *)'THE MODELS WERE CREATED, DO NOT FORGET TO KERMIT'
      WRITE (*, *)'THEM OVER TO A VAX, GOOD LUCK IN COMPILING'
      WRITE (*,'(///////)')
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE OBJ(PHOTOX, PHOTOY, POINTS)
      INTEGER      POINTS
      INTEGER    K, M, J, I, L
      REAL       CORX, CORY, CORZ, IMGCORX, IMGCORY
      REAL       GCORX(32),GCORY(32),GCORZ(32)
           REAL     IMGCORXA(32), IMGCORYA(32), IMGCORXB(32), IMGCORYB(32)
      CHARACTER*8 PHOTOX, PHOTOY
      CHARACTER*8 POINTID, LABEL, F3, CFILE3,IDIMG, PHOTOIMG
      CHARACTER   IEND*8
      DATA IEND   /'********'/
      CHARACTER   MOPAR(650)*6
      CHARACTER*6 MPAR
C
C     LOOP FOR POINTS IN THE MODEL.
C
      K = 1
      J = 1
      L = 1
      DO I=1, POINTS, 1
          READ(2, *) POINTID
C
C         SEARCH FOR GROUND COORDINATE POINTS IN OBJ.OUT.
C
          F3='OBJ.OUT'
          OPEN(3,FILE=f3,STATUS='OLD')
 1000     CONTINUE
          READ(3, *)LABEL, CORX, CORY, CORZ
          IF(LABEL.EQ.IEND)GOTO 50
          IF(LABEL.EQ.POINTID)THEN
                   GCORX(K) = CORX
                   GCORY(K) = CORY
                   GCORZ(K) = CORZ
                   K = K+1
                   GOTO 50
          ENDIF
          GOTO 1000
 50       CONTINUE
          CLOSE(3)
C
C         SEARCH FOR IMAGE COORDINATES IN IMG.DAT.
C         FIRST IN PHOTO A.
C
          CFILE3='IMG.DAT'
```

```
            OPEN(4, FILE=CFILE3,STATUS='OLD')
 2500       CONTINUE
            READ(4,*,END=3000)PHOTOIMG
            IF (PHOTOIMG.EQ.PHOTOX) THEN
                DO M=1, 32, 1
                     IF (PHOTOIMG.EQ.IEND) THEN
                          GOTO 70
                     ENDIF
                     READ(4, *)IDIMG, IMGCORX, IMGCORY
                     IF (IDIMG.EQ.POINTID) THEN
                          IMGCORXA(J) = IMGCORX
                          IMGCORYA(J) = IMGCORY
                          J=J+1
                          GOTO 3000
                     ENDIF
                ENDDO
            ENDIF
   70       CONTINUE
            GOTO 2500
 3000       CONTINUE
            CLOSE(4)
 C
 C          THEN IN PHOTO B.
 C
            OPEN(4, FILE=CFILE3,STATUS='OLD')
 3500       CONTINUE
            READ(4,*,END=4000)PHOTOIMG
            IF (PHOTOIMG.EQ.PHOTOY) THEN
                DO M=1, 32, 1
                     IF (PHOTOIMG.EQ.IEND) THEN
                          GOTO 90
                     ENDIF
                     READ(4, *)IDIMG, IMGCORX, IMGCORY
                     IF (IDIMG.EQ.POINTID) THEN
                          IMGCORXB(L) = IMGCORX
                          IMGCORYB(L) = IMGCORY
                          L=L+1
                          GOTO 4000
                     ENDIF
                ENDDO
            ENDIF
     90     CONTINUE
            GOTO 3500
  4000      CONTINUE
            CLOSE(4)
        ENDDO
 C
 C      FILL UP THE REST OF THE ARRAYS UP TO 32.
 C
        DO I=K, 32, 1
            GCORX(I)     = 0.90000005E+10
            GCORY(I)     = 0.90000005E+10
```

```
            GCORZ(I)     = 0.90000005E+10
            IMGCORXA(I) = 0.90000005E+10
            IMGCORYA(I) = 0.90000005E+10
            IMGCORXB(I) = 0.90000005E+10
            IMGCORYB(I) = 0.90000005E+10
        ENDDO
        I=1
C
C       READ FIRST FIELD FROM MODEL.DAT.
C
        OPEN(5,FILE='MODEL.DAT',STATUS='OLD')
 103    CONTINUE
        READ(5, 105, END=110)MPAR
 105    FORMAT(A6)
        MOPAR(I)=MPAR
        I=I+1
        GOTO 103
 110    CONTINUE
        CLOSE(5)
C
C       PUT HEADER INFORMATION FROM MODEL.DAT.
C
        CALL MOHEAD
C
C       WRITE GROUND COORDINATE POINTS IN GXXXX.DAT
C
        WRITE(*,*)'WRITING GROUND COORDINATES:', POINTS
        DO I=1, 32, 1
            WRITE(1, 4100)MOPAR(I+132),GCORX(I)
        ENDDO
        DO I=1, 32, 1
            WRITE(1, 4100)MOPAR(I+164),GCORY(I)
        ENDDO
        DO I=1, 32, 1
            WRITE(1, 4100)MOPAR(I+196), GCORZ(I)
        ENDDO
C
C      WRITE PHOTO A X PHOTO COORDS FOR 32 CONTROL POINTS TO GXXXX.DAT.
C
        WRITE(*,*)'WRITING IMAGE COORDINATES: ', POINTS
        DO I=1, 32, 1
            WRITE(1, 4100)MOPAR(I+228),IMGCORXA(I)
        ENDDO
C
C      WRITE PHOTO A Y PHOTO COORDS FOR 32 CONTROL POINTS TO GXXXX.DAT.
C
        DO I=1, 32, 1
            WRITE(1, 4100)MOPAR(I+260),IMGCORYA(I)
        ENDDO
C
C      WRITE PHOTO B X PHOTO COORDS FOR 32 CONTROL POINTS TO GXXXX.DAT.
C
```

```
      DO I=1, 32, 1
          WRITE(1, 4100)MOPAR(I+292),IMGCORXB(I)
      ENDDO
C
C     WRITE PHOTO B Y PHOTO COORDS FOR 32 CONTROL POINTS TO GXXXX.DAT.
C
      DO I=1, 32, 1
          WRITE(1, 4100)MOPAR(I+324),IMGCORYB(I)
      ENDDO
4100  FORMAT(A6, E15.8)
C
C     PUT END OF FILE INFORMATION IN MODEL FILE.
C
      CALL MOTAIL
      CLOSE(5)
      CALL BEEP
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE MOHEAD()
      CHARACTER*6   FIELD
      REAL          VALUE
C
C     HEADER SECTION.
C
      OPEN(5, FILE='MODEL.DAT', STATUS='OLD')
      WRITE(*, *)'WRITING HEADER INFORMATION'
      DO I=1, 132, 1
          READ(5, 4200)FIELD, VALUE
          WRITE(1, 4200)FIELD, VALUE
      ENDDO
4200  FORMAT(A6, E15.8)
      DO I=133, 356, 1
          READ(5, *)FIELD, VALUE
      ENDDO
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE MOTAIL()
      CHARACTER*6   FIELD
      REAL          VALUE
      WRITE(*, *)'WRITING TRAILER INFORMATION'
      DO I=357, 650, 1
          READ(5, 4300)FIELD, VALUE
          WRITE(1, 4300)FIELD, VALUE
      ENDDO
4300  FORMAT(A6, E15.8)
      RETURN
      END
```

B.2.3 Listing of Program DSCDXF

```
      PROGRAM DSCDXF
C
C     This program reads a digital recording file data file
C     MAPXXXX.DAT and generates two files: an ASCII file and
C     a DXF file.
C
C     SUBROUTINES USED: CFORMAT.
C                       MERGE.
C                       DXHEAD.
C                       DXSETL.
C                       DXPL3.
C                       DXTAIL.
C ******
C SUBROUTINE CFORMAT() - PROYECTO ESPOL - UNO.
C
C PURPOSE: CONVERTS AN ASCII FILE FROM DSC TO A FILE READY FOR
C          AUTOCAD.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL CFORMAT
C ******
C SUBROUTINE MERGE() - PROYECTO ESPOL - UNO.
C
C PURPOSE:
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL MERGE
C ******
C SUBROUTINE DXHEAD() - PROYECTO ESPOL - UNO.
C
C PURPOSE: WRITES HEADER FOR DXF FILE.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL DXHEAD
C ******
C SUBROUTINE DXSETL - PROYECTO ESPOL - UNO.
C
C PURPOSE: SETS CURRENT LAYER.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL DXSETL(CLX)
C
C          WHERE:
C          CLX
C ******
C SUBROUTINE DXPL3() - PROYECTO ESPOL - UNO.
C
C PURPOSE: POLYLINE 3D.
```

```
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL DXPL3(XA,YA,ZA,NP,IOP)
C
C         WHERE:
C         XA,YA,ZA   ARRAYS FOR X,Y,Z COORDINATES FOR POLYLINE.
C         NP         NUMBER OF POINTS IN ARRAYS XA,YA.
C         IOP        =0 FOR OPEN POLYLINE.
C                    =1 FOR CLOSED POLYLINE (1ST AND LAST POINTS
C                       CONNECTED).
C ******
C SUBROUTINE DXTAIL() - PROYECTO ESPOL - UNO.
C
C PURPOSE: WRITES TRAILER FOR DXF FILE.
C
C FORTRAN CALLING PROCEDURE:
C
C     CALL DXTAIL
C ******
C     LANGUAGE:    Lahey Fortran Version 5.1.
C     DATE:        13 September 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      EXTERNAL SYSTEM
      CHARACTER*64 CFILE1,CFILE2
      CHARACTER*80 CTEXT
      CHARACTER*8  CITEM
      CHARACTER*8  CLX
      CHARACTER*1  CYN
      DIMENSION    XA(500),YA(500),ZA(500)
      WRITE (*, 10)
   10 FORMAT (////////37X, 'PSCASCII'//27X, 'A PC Utility',
     . /18X,  "for the Conversion of an PDP11 'DIGDATxx.DAT'"
     . /19x,  "file to a AutoCad DXF file 'xxxx.DXF' file. "
     . //20X, '(c) Proyecto ESPOL - UNO'////////)
      PAUSE
      CALL SYSTEM ('CLS')
      CALL CFORMAT
  100 CONTINUE
      WRITE(*,*)'ENTER INPUT FILE NAME IF DIFERENT THAN DXF.DAT:'
      READ(*,2100)CFILE1
 2100 FORMAT(A64)
      WRITE(*,*)'OPENING:',CFILE1
      OPEN(5,FILE=CFILE1,STATUS='OLD')
      WRITE(*,*)'ENTER OUTPUT FILE NAME XXXX.DXF:'
      READ(*,2105)CFILE2
 2105 FORMAT(A64)
      WRITE(*,*)'OPENING:',CFILE2
      OPEN(7,FILE=CFILE2,STATUS='NEW')
      WRITE(*,*)'PROCESSING...'
C
```

```
C     PUT HEADER INFORMATION OUT TO DXF FILE.
C
      CALL DXHEAD
C
C     READ FILE UNTIL EOF REACHED.
C
  510 CONTINUE
      READ(5,2200,END=8100)CITEM
 2200 FORMAT(A8)
      WRITE(*,*)'CITEM=',CITEM
C
C     ---START PARSE LIST.
C
C     =====2D ROUTINES LAYER 0.
C
C     ---LINE 2D.
C
      IF(CITEM.EQ.'DXLINE') THEN
          WRITE(*,*)' LINE2D'
          READ(5,*)X1,Y1,X2,Y2
          CALL DXLINE(X1,Y1,X2,Y2)
          GOTO 600
      END IF

C
C     ---TEXT 2D.
C
      IF(CITEM.EQ.'DXTEXT') THEN
          WRITE(*,*)' TEXT2D'
          READ(5,*)X1,Y1,HEIGHT,ANGLE
          READ(5,2300)CTEXT
 2300     FORMAT(A80)
          CALL DXTEXT(X1,Y1,HEIGHT,ANGLE,CTEXT)
          GOTO 600
      END IF
C
C     ---ARC 2D.
C
      IF(CITEM.EQ.'DXAR2') THEN
          WRITE(*,*)' ARC2D'
          READ(5,*)X1,Y1,RADIUS,ABEG,AEND
          CALL DXAR2(X1,Y1,RADIUS,ABEG,AEND)
          GOTO 600
      END IF
C
C     ---CIRCLE 2D.
C
      IF(CITEM.EQ.'DXCI2') THEN
          WRITE(*,*)' CIR2D'
          READ(5,*)X1,Y1,RADIUS
          CALL DXCI2(X1,Y1,RADIUS)
          GOTO 600
```

```
      END IF
C
C     ---POINT 2D.
C
      IF(CITEM.EQ.'DXPN2') THEN
          WRITE(*,*)' POINT2D'
          READ(5,*)X1,Y1
          CALL DXPN2(X1,Y1)
          GOTO 600
      ENDIF
C
C     ---TRACE 2D.
C
      IF(CITEM.EQ.'DXTR2') THEN
          WRITE(*,*)' TRACE 2D'
          READ(5,*)XA(1),YA(1)
          READ(5,*)XA(2),YA(2)
          READ(5,*)XA(3),YA(3)
          READ(5,*)XA(4),YA(4)
          CALL DXTR2(XA,YA)
          GOTO 600
      END IF
C
C     =====2D ROUTINES WITH LAYER SPECIFIED.
C
C     ---LINE 2D.
C
      IF(CITEM.EQ.'DXLN2L') THEN
          WRITE(*,*)' LINE2D'
          READ(5,*)X1,Y1,X2,Y2
          CALL DXLN2L(X1,Y1,X2,Y2)
          GOTO 600
      END IF
C
C     ---TEXT 2D.
C
      IF(CITEM.EQ.'DXTX2L') THEN
          WRITE(*,*)' TEXT2D'
          READ(5,*)X1,Y1,HEIGHT,ANGLE
          READ(5,2310)CTEXT
 2310     FORMAT(A80)
          CALL DXTX2L(X1,Y1,HEIGHT,ANGLE,CTEXT)
          GOTO 600
      END IF
C
C     ---ARC 2D.
C
      IF(CITEM.EQ.'DXAR2L') THEN
          WRITE(*,*)' ARC2D'
          READ(5,*)X1,Y1,RADIUS,ABEG,AEND
          CALL DXAR2L(X1,Y1,RADIUS,ABEG,AEND)
          GOTO 600
```

```
      END IF
C
C     ---CIRCLE 2D.
C
      IF(CITEM.EQ.'DXCI2L') THEN
          WRITE(*,*)' CIR2D'
          READ(5,*)X1,Y1,RADIUS
          CALL DXCI2L(X1,Y1,RADIUS)
          GOTO 600
      END IF
C
C     ---POINT 2D.
C
      IF(CITEM.EQ.'DXPN2L') THEN
          WRITE(*,*)' POINT2D'
          READ(5,*)X1,Y1
          CALL DXPN2L(X1,Y1)
          GOTO 600
      ENDIF
C
C     ---2D TRACE.
C
      IF(CITEM.EQ.'DXTR2L') THEN
          WRITE(*,*)' TRACE 2D'
          READ(5,*)XA(1),YA(1)
          READ(5,*)XA(2),YA(2)
          READ(5,*)XA(3),YA(3)
          READ(5,*)XA(4),YA(4)
          CALL DXTR2L(XA,YA)
          GOTO 600
      END IF
C
C     ---SET LAYER.
C
      IF(CITEM.EQ.'DXSETL') THEN
          WRITE(*,*)' SET LAYER'
          READ(5,2320)CLX
 2320     FORMAT(A8)
          WRITE(*,*)'CLX=',CLX
          CALL DXSETL(CLX)
          GOTO 600
      ENDIF
C
C     =====3D ROUTINES LAYER 0.
C
C     ---LINE 3D.
C
      IF(CITEM.EQ.'DXLN3') THEN
          WRITE(*,*)' LINE3D'
          READ(5,*)X1,Y1,Z1,X2,Y2,Z2
          CALL DXLN3(X1,Y1,Z1,X2,Y2,Z2)
          GOTO 600
```

```
      END IF
C
C     ---TEXT 3D.
C
      IF(CITEM.EQ.'DXTX3') THEN
          WRITE(*,*)' TEXT3D'
          READ(5,*)X1,Y1,Z1,HEIGHT,ANGLE
          READ(5,2330)CTEXT
 2330     FORMAT(A80)
          READ(5,*)XUCS,YUCS,ZUCS
          CALL DXTX3(X1,Y1,Z1,HEIGHT,ANGLE,CTEXT,XUCS,YUCS,ZUCS)
          GOTO 600
      END IF
C
C     ---ARC 3D.
C
      IF(CITEM.EQ.'DXAR3') THEN
          WRITE(*,*)' ARC3D'
          READ(5,*)X1,Y1,Z1,RADIUS,ABEG,AEND,XUCS,YUCS,ZUCS
          CALL DXAR3(X1,Y1,Z1,RADIUS,ABEG,AEND,XUCS,YUCS,ZUCS)
          GOTO 600
      END IF
C
C     ---CIRCLE 3D.
C
      IF(CITEM.EQ.'DXCI3') THEN
          WRITE(*,*)' CIR3D'
          READ(5,*)X1,Y1,Z1,RADIUS,XUCS,YUCS,ZUCS
          CALL DXCI3(X1,Y1,Z1,RADIUS,XUCS,YUCS,ZUCS)
          GOTO 600
      END IF
C
C     ---POINT 3D.
C
      IF(CITEM.EQ.'DXPN3') THEN
          WRITE(*,*)' POINT3D'
          READ(5,*)X1,Y1,Z1
          CALL DXPN3(X1,Y1,Z1)
          GOTO 600
      ENDIF
C
C     ---POLYLINE 2D.
C
      IF(CITEM.EQ.'DXPL2') THEN
          WRITE(*,*)' POLYLINE 2D '
          READ(5,*)NP,IOP
          DO 700 I=1,NP
              READ(5,*)XA(I),YA(I)
  700     CONTINUE
          CALL DXPL2(XA,YA,NP,IOP)
          GOTO 600
      END IF
```

```
C
C     ---POLYLINE 3D.
C
      IF(CITEM.EQ.'DXPL3') THEN
          WRITE(*,*)' POLYLINE 3D '
          READ(5,*)NP,IOP
          DO 710 I=1,NP
              READ(5,*)XA(I),YA(I),ZA(I)
  710     CONTINUE
          CALL DXPL3(XA,YA,ZA,NP,IOP)
          GOTO 600
      END IF
C
C     ---3D FACE.
C
      IF(CITEM.EQ.'DXFA3') THEN
          WRITE(*,*)' 3D FACE'
          READ(5,*)XA(1),YA(1),ZA(1)
          READ(5,*)XA(2),YA(2),ZA(2)
          READ(5,*)XA(3),YA(3),ZA(3)
          READ(5,*)XA(4),YA(4),ZA(4)
          CALL DXFA3(XA,YA,ZA)
          GOTO 600
      END IF
C
C     ---LINE 3D REL.9.
C
      IF(CITEM.EQ.'DX9L3') THEN
          WRITE(*,*)' LINE3D R9'
          READ(5,*)X1,Y1,Z1,X2,Y2,Z2
          CALL DX9L3(X1,Y1,Z1,X2,Y2,Z2)
          GOTO 600
      END IF
C
C     =====3D ROUTINES WITH SPECIFIED LAYER.
C
C     ---LINE 3D.
C
      IF(CITEM.EQ.'DXLN3L') THEN
          WRITE(*,*)' LINE3D'
          READ(5,*)X1,Y1,Z1,X2,Y2,Z2
          CALL DXLN3L(X1,Y1,Z1,X2,Y2,Z2)
          GOTO 600
      END IF
C
C     ---TEXT 3D.
C
      IF(CITEM.EQ.'DXTX3L') THEN
          WRITE(*,*)' TEXT3D'
          READ(5,*)X1,Y1,Z1,HEIGHT,ANGLE
          READ(5,2340)CTEXT
 2340     FORMAT(A80)
```

```
            READ(5,*)XUCS,YUCS,ZUCS
            CALL DXTX3L(X1,Y1,Z1,HEIGHT,ANGLE,CTEXT,XUCS,YUCS,ZUCS)
            GOTO 600
        END IF
C
C       ---ARC 3D.
C
        IF(CITEM.EQ.'DXAR3L') THEN
            WRITE(*,*)' ARC3D'
            READ(5,*)X1,Y1,Z1,RADIUS,ABEG,AEND,XUCS,YUCS,ZUCS
            CALL DXAR3L(X1,Y1,Z1,RADIUS,ABEG,AEND,XUCS,YUCS,ZUCS)
            GOTO 600
        END IF
C
C       ---CIRCLE 3D.
C
        IF(CITEM.EQ.'DXCI3L') THEN
            WRITE(*,*)' CIR3D'
            READ(5,*)X1,Y1,Z1,RADIUS,XUCS,YUCS,ZUCS
            CALL DXCI3L(X1,Y1,Z1,RADIUS,XUCS,YUCS,ZUCS)
            GOTO 600
        END IF
C
C       ---POINT 3D.
C
        IF(CITEM.EQ.'DXPN3L') THEN
            WRITE(*,*)' POINT3D'
            READ(5,*)X1,Y1,Z1
            CALL DXPN3L(X1,Y1,Z1)
            GOTO 600
        ENDIF
C
C       ---POLYLINE 2D.
C
        IF(CITEM.EQ.'DXPL2L') THEN
            WRITE(*,*)' POLYLINE 2D '
            READ(5,*)NP,IOP
            DO 720 I=1,NP
                READ(5,*)XA(I),YA(I)
  720       CONTINUE
            CALL DXPL2L(XA,YA,NP,IOP)
            GOTO 600
        END IF
C
C       ---POLYLINE 3D.
C
        IF(CITEM.EQ.'DXPL3L') THEN
            WRITE(*,*)' POLYLINE 3D '
            READ(5,*)NP,IOP
            DO 730 I=1,NP
                READ(5,*)XA(I),YA(I),ZA(I)
  730       CONTINUE
```

```
            CALL DXPL3L(XA,YA,ZA,NP,IOP)
            GOTO 600
        END IF
C
C       ---3D FACE.
C
        IF(CITEM.EQ.'DXFA3L') THEN
            WRITE(*,*)' 3D FACE'
            READ(5,*)XA(1),YA(1),ZA(1)
            READ(5,*)XA(2),YA(2),ZA(2)
            READ(5,*)XA(3),YA(3),ZA(3)
            READ(5,*)XA(4),YA(4),ZA(4)
            CALL DXFA3L(XA,YA,ZA)
            GOTO 600
        END IF
C
C       ---LINE 3D REL.9.
C
        IF(CITEM.EQ.'DX9L3L') THEN
            WRITE(*,*)' LINE3D R9'
            READ(5,*)X1,Y1,Z1,X2,Y2,Z2
            CALL DX9L3L(X1,Y1,Z1,X2,Y2,Z2)
            GOTO 600
        END IF
C
C       ---END OF PARSE LIST.
C
  600 CONTINUE
        GOTO 510
C
C       DONE READING FILE.
C
 8100 CONTINUE
C
C       FINISH UP.
C       PUT END OF FILE INFORMATION IN DXF FILE.
C
        CALL DXTAIL
C
C       CLOSE UP FILES.
C
        CLOSE(5)
        CLOSE(7)
C
C       ASK IF USER WANTS TO RUN PROGRAM AGAIN.
C
        WRITE(*,*)'PROCESS ANOTHER FILE Y/N?'
        READ(*,3100)CYN
 3100 FORMAT(A1)
        IF(CYN.EQ.'Y' .OR. CYN.EQ.'y') GOTO 100
C
C       OTHERWISE JUST QUIT.
```

```
C
      WRITE(*,*)'PROGRAM DONE.'
      END
C     ------------------------------------------------------------
      SUBROUTINE CFORMAT()
      IMPLICIT NONE
      EXTERNAL SYSTEM
      CHARACTER  F2*8
      F2='DXF.DAT'
      OPEN(2,F2,STATUS='NEW')
      WRITE(2, *) 'PREPROCESED FILE PREVIOUS TO AUTOCAD'
      WRITE(2, *) 'DXSETL'
      WRITE(2, *) 'MANZANAS'
      CALL SYSTEM ('CLS')
      CALL MERGE
      CALL SYSTEM ('CLS')
      WRITE (*,'(////////)')
           WRITE (*, *)'THE POLYGONS WERE PROCESSED & WRITTEN TO FILE',F2
      WRITE (*,'(///////)')
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE MERGE()
      INTEGER I, W, J, R, N, M, H(1200)
      REAL CORX(1200), CORY(1200), CORZ(1200)
      CHARACTER  MODELID*2, FNAME*8, F1*12
      CALL SYSTEM ('CLS')
      WRITE(*, 20)
   20 FORMAT (//////////'  ENTER DIGDATXX (PART OF INPUT FILENAME TO
     . LEFT OF DECIMAL): ')
      READ(*, *)FNAME
          DO I=8, 1, -1
              IF(FNAME(I:I).NE.' ')EXIT
          ENDDO
      MODELID=FNAME(7:I)
      F1=FNAME(1:I)//'.DAT'
      OPEN(1,F1,STATUS='OLD')
      F1=FNAME(7:I)//'.DAT'
      J=0
      M=0
      N=1
      DO I=1, 1200, 1
          READ(1, *, END=55) CORX(I), CORY(I), CORZ(I)
          IF (CORX(I).EQ.9.000005E+9) THEN
              IF (CORX(I-1).EQ.9.000005E+9) THEN
              GOTO 50
              ENDIF
          H(N)=J
          N=N+1
          J=1
          GOTO 50
          ENDIF
```

```
      J=J+1
50    CONTINUE
      M=M+1
      ENDDO
55    CONTINUE
      H(N)=J
      W=1
      WRITE(2, *) 'DXPL3L'
      WRITE(2, *) H(W), ', ', '0'
      DO I=1, M-1, 1+R
          R=0
          IF (CORX(I).EQ.9.000005E+9) THEN
              IF (CORX(I-1).EQ.9.000005E+09) THEN
              GOTO 60
              ENDIF
          W=W+1
          WRITE(2, *) 'DXPL3L'
          WRITE(2, *) H(W), ', ', '0'
          R=1
          ENDIF
          WRITE(2, *) CORX(I+R),',', CORY(I+R),',', CORZ(I+R)
60        CONTINUE
      ENDDO
      CLOSE(1)
      CLOSE(2)
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE DXHEAD()
C
C     HEADER SECTION.
C
      WRITE(7,5110)
 5110 FORMAT('  0')
      WRITE(7,5120)
 5120 FORMAT('SECTION')
      WRITE(7,5130)
 5130 FORMAT('  2')
      WRITE(7,5140)
 5140 FORMAT('ENTITIES')
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE DXSETL(CLX)
      COMMON/DXFLAYR/CLAYER
      CHARACTER*8 CLAYER
      CHARACTER*8 CLX
      CLAYER=CLX
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE DXPL3(XA,YA,ZA,NP,IOP)
```

```
      DIMENSION XA(*),YA(*),ZA(*)
      IOP2=IOP+8
      IOP3=32
      WRITE(7,5010)
 5010 FORMAT('  0',/,'POLYLINE')
      WRITE(7,5020)
 5020 FORMAT('  8',/,'0')
      WRITE(7,5030)
 5030 FORMAT(' 66',/,'     1' )
      WRITE(7,5040)IOP2
 5040 FORMAT(' 70',/,I6)
      DO 100 ,I=1,NP
          WRITE(7,5050)
 5050     FORMAT('  0',/,'VERTEX')
          WRITE(7,5060)
 5060     FORMAT('  8',/,'0')
          WRITE(7,5070)XA(I),YA(I),ZA(I)
 5070     FORMAT(' 10',/,E12.6,/,' 20',/,E12.6,/,' 30',/,E12.6)
          WRITE(7,5075)IOP3
 5075     FORMAT(' 70',/,I6)
  100 CONTINUE
      WRITE(7,5080)
 5080 FORMAT('  0',/,'SEQEND')
      WRITE(7,5090)
 5090 FORMAT('  8',/,'0')
      RETURN
      END
C     ------------------------------------------------------------
      SUBROUTINE DXTAIL()
      WRITE(7,5110)
 5110 FORMAT('  0',/,'ENDSEC',/,'  0',/,'EOF')
      RETURN
      END
```

B.2.3 Listing of Program DSCASCII

```
      PROGRAM DSCASCII
C
C     This program reads a digital recording file data file
C     digxxxx.dat and generates the ascii fiel MANZxxxx.dat
C
      EXTERNAL     SYSTEM
      CHARACTER*1  CYN,
C
C
      WRITE (*, 10)
   10 FORMAT (////////35X, 'PSCASCII'//33X, 'A PC Utility',
     . /18X,  "for the Conversion of an PDP11 'DIGxxxx.DAT'"
     . /19x,  "file to a Ascii file 'MANZxxxx.DAT' file. "
```

```
     . //31X, 'Proyecto ESPOL - UNO'////////)
       CALL BEEP
       PAUSE
  100  CONTINUE
       CALL SYSTEM ('CLS')
       CALL CFORMAT
C
C      ASK IF USER WANTS TO RUN PROGRAM AGAIN.
C
       WRITE(*,*)'PROCESS ANOTHER FILE Y/N?'
       READ(*,3100)CYN
 3100  FORMAT(A1)
       IF(CYN.EQ.'Y' .OR. CYN.EQ.'y') GOTO 100
C
C      OTHERWISE JUST QUIT.
C
       WRITE(*,*)'PROGRAM DONE.'
       END
C      ------------------------------------------------------------
       SUBROUTINE CFORMAT()
C
C      CONVERTS AN ASCII FILE FROM DSC TO A FILE READY FOR AUTOCAD.
C
       IMPLICIT     NONE
       EXTERNAL     SYSTEM
       CHARACTER*8  F2, NAMELY
C
       F2='MANZ.DAT'
       OPEN(2,F2,STATUS='NEW')
       WRITE(2, *) 'PREPROCESED FILE PREVIOUS TO MAPIX'
       WRITE(2, *) 'DXSETL'
       WRITE (*, 21)
   21  FORMAT (//////////'      ENTER THE NAME OF THIS LAYER: ')
       READ(*, 33)NAMELY      !NAME OF THE LAYER
   33  FORMAT(A8)
       WRITE(2, *) NAMELY
       CALL SYSTEM ('CLS')
       CALL MERGE
       CALL SYSTEM ('CLS')
       WRITE (*,'(///////)')
            WRITE (*, *)'THE POLYGONS WERE PROCESSED & WRITTEN TO FILE', F2
       WRITE (*,'(//////)')
       RETURN
       END
C      ------------------------------------------------------------
       SUBROUTINE MERGE()
C
C      COMPUTE GROUND COORDINATES OUT OF THE MODEL COORDINATES.
C
       INTEGER I, W, J, R, N, M, H(9050),L(9050), S
       REAL CORX(9050), CORY(9050), CORZ(9050)
       REAL VALUE(650)
```

```
      CHARACTER*6 LABEL(650)
      REAL*8 CORXA(9050), CORYA(9050), CORZA(9050)
      REAL*8 XLA, XLB, YLA, YLB
      REAL*8 XLMA, YLMA
      REAL*8 XM, YM, ZM
      REAL*8 XG, YG, ZG
      CHARACTER  MODELID*2, FNAME*8, F1*12, F3*12
      CALL SYSTEM ('CLS')
      WRITE(*, 20)
   20 FORMAT (//////////'   ENTER MAPXXXX (PART OF INPUT FILENAME TO
     . LEFT OF DECIMAL): ')
      READ(*, *)FNAME
          DO I=8, 1, -1
              IF(FNAME(I:I).NE.' ')EXIT
          ENDDO
      MODELID=FNAME(7:I)
      F1=FNAME(1:I)//'.DAT'
      OPEN(1,F1,STATUS='OLD')
      F1=FNAME(7:I)//'.DAT'
      F3='MOD'//FNAME(4:7)//'.DAT'
      OPEN(3,F3,STATUS='OLD')
C
C     READ IN THE VALUES FOR STATIC ORIENTATION ELEMENTS FOR PHOTO A,
C     B AND THE MODEL COORDINATES OF ESP. STATIONS A AND B.
C
      DO S=1, 650, 1
          READ(3, 65)LABEL(S), VALUE(S)
      ENDDO
   65 FORMAT(A6, E15.8)
      CLOSE(3)
C
      CALL SYSTEM ('CLS')
      XLA  = VALUE(526)
      YLA  = VALUE(527)
      XLB  = VALUE(532)
      YLB  = VALUE(533)
      XLMA = VALUE(538)
      YLMA = VALUE(539)
C
      J=0
      M=0
      N=1
      DO I=1, 9050, 1
      READ(1, *, END=55) CORX(I), CORY(I), CORZ(I)
      IF (CORX(I).EQ.9.000005E+09) THEN
          IF (CORX(I-1).EQ.9.000005E+09) THEN
              GOTO 50
          ENDIF
          H(N)=J
          IF (CORX(I-1).EQ.CORX(I-J+1)) THEN
          L(N)=1
          ENDIF
```

```
            N=N+1
            J=1
            GOTO 50
        ENDIF
        J=J+1
50      CONTINUE
        M=M+1
        ENDDO
55      CONTINUE
        H(N)=J
        W=1
        WRITE(2, *) 'DXPL3L'
        WRITE(2, *) H(W), ', ', L(W), ', ',  W
        DO I=1, M-1, 1+R
        R=0
        IF (CORX(I).EQ.9.000005E+09) THEN
            IF (CORX(I-1).EQ.9.000005E+09) THEN
                GOTO 60
            ENDIF
            W=W+1
            WRITE(2, *) 'DXPL3L'
            WRITE(2, *) H(W), ', ', L(W), ', ', W
            R=1
        ENDIF
C
C       COMPUTE THE X GROUND COORDINATE.
C
        CORXA(I+R)=(((YLA-YLB)*(XLMA*XLMA+YLMA*YLMA)+XLMA*(YLMA*
       .(XLA-XLB)-XLMA*(YLA-YLB)))*(CORX(I+R)-XLMA)+YLMA*(YLMA*
       .(XLA-XLB)-XLMA*(YLA-YLB))*(CORY(I+R)-YLMA)+2*YLMA*(XLMA*XLMA+
       .YLMA*YLMA)*(XLA+490000))/(2*YLMA*(XLMA*XLMA+YLMA*YLMA))
C
C       COMPUTE THE Y GROUND COORDINATE
C
        CORYA(I+R)=(YLMA*((XLMA*XLMA+YLMA*YLMA)*(YLA+YLB+4000000)-
       .(YLMA*(XLA-XLB)-XLMA*(YLA-YLB))*CORX(I+R))+((YLA-YLB)
       .*(XLMA*XLMA+YLMA*YLMA)+XLMA*(YLMA*(XLA-XLB)-XLMA*(YLA-YLB)))
       .*CORY(I+R))/(2*YLMA*(XLMA*XLMA+YLMA*YLMA))
C
C       COMPUTE THE Z GROUND COORDINATE.
C
        CORZA(I+R)=CORZ(I+R)*SQRT(((YLA-YLB)*(XLMA*XLMA+YLMA*YLMA)+XLMA*
       .(YLMA*(XLA-XLB)-XLMA*(YLA-YLB)))*((YLA-YLB)*(XLMA*XLMA+YLMA*
       .YLMA)+XLMA*(YLMA*(XLA-XLB)-XLMA*(YLA-YLB)))+YLMA*YLMA*(YLMA
       .*(XLA-XLB)-XLMA*(YLA-YLB))*(YLMA*(XLA-XLB)-XLMA*(YLA-YLB)))/(2*
       .YLMA*(XLMA*XLMA+YLMA*YLMA))
        WRITE(2, *) CORXA(I+R),',', CORYA(I+R),',', CORZA(I+R)
60      CONTINUE
        ENDDO
        CLOSE(1)
        CLOSE(2)
        RETURN
```

```
      END
C     ------------------------------------------------------------------
      SUBROUTINE BEEP
      WRITE (*, *) CHAR(7)
      END
```

B.2.5 Listing of Program MATRIX

```
      PROGRAM ANGLE_DETERM_ROT_MATRIX
      IMPLICIT REAL*8 (A-Z)
      R=45/DATAN(1)
      OPEN(1, 'ANG.IN')
      OPEN(2, 'MATRIX.OUT')
1     READ(1, *, END=9) A, H, S    !  (YXZ)
C
C     LEFT, UP, CAM AXIS.
C
      SA=DSIN(A/R)
      CA=DCOS(A/R)
      SH=DSIN(H/R)
      CH=DCOS(H/R)
      SS=-DSIN(S/R)
      CS=DCOS(S/R)
      X11= CS*CA+SS*SH*SA
      X21=-SS*CA+CS*SH*SA
      K=DATAN2(-X21, X11)*R
      X32= CH*CA
      X33=-SH
      O=DATAN2(X32, X33)*R
      X31= CH*SA
      P=DASIN(X31)*R
      M11=COS(P)*COS(K)
      M12=SIN(O)*SIN(P)*COS(K)+COS(O)*SIN(K)
      M13=-COS(O)*SIN(P)*COS(K)+SIN(O)*SIN(K)
      M21=-COS(P)*SIN(K)
      M22=-SIN(O)*SIN(P)*SIN(K)+COS(O)*COS(K)
      M23=COS(O)*SIN(O)*SIN(K)+SIN(O)*COS(K)
      M31=SIN(O)
      M32=-SIN(O)*COS(P)
      M33=COS(O)*COS(P)
      WRITE(2, '(3F10.4)')O, P, K   !  (XYZ)
      WRITE(2, '(3F12.4)')M11, M12, M13
      WRITE(2, '(3F12.4)')M21, M22, M23
      WRITE(2, '(3F12.4)')M31, M32, M33
      GO TO 1
9     CONTINUE
      END
```

B.3 Microsoft Fortran

B.3.1 Listing of Program DSCMAPIX

```
      program DSCMAPIX
C
C
C     This program reads an dsc ascii file output of pb4vax.for
C     and generates Mapix's related files.
C
C     Logical file names:
C         TEXT     Name of the Mapix related files.
C         TEXT2    Ascii file from pb4vax.for.
C         TEXT3    TEXT without extension.
C
C     Size of DHEDER is at least 4 times the size of record.
C    Size of RECORD is ( Dbase [1] + Size of each field + size of all
C                    the agis fields [5]).
C     Size of DHEDERC is (1 [dbase] + Number of Fields + 5[agis]) *
C                     32.
C
C     SUBROUTINES USED: CONVIC.
C                       AFILE2.
C                       VEC_FILE.
C                       VEC_SET.
C                       LJUST.
C                       VV_ADD.
C                       VO_ADD.
C                       RJUST.
C                       VO_CALC.
C ******
C     LANGUAGE:    Microsoft Fortran version 5.1.
C     DATE:        31 October 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      REAL*8     DHEDER8(1)
      REAL*8     R8GEO(32),  XMIN, XMAX, YMIN, YMAX,
     *           R8BUF(2050)
      REAL*8     ISTX, ISTY, STRTX, STRTY
      REAL*4     DHEDER4(1)
      INTEGER*4  VEC_FILE,   VEC_SET,  I4GEO(64), VO_ADD, VV_ADD
      INTEGER*4  DHEDER(1284), PATH(20), VO_CALC
      INTEGER*2  DHEDER2(1), HANDL3, DATE(3)
      INTEGER*1  DHEDER1(1)
      CHARACTER*704 DHEDERC
      CHARACTER*321 RECORD
      CHARACTER*64  VPATH,   FILIST(3)
      CHARACTER*80  TEXT, TEXT2, TEXT3
      EQUIVALENCE  ( R8GEO,   I4GEO )
```

```
      EQUIVALENCE  ( DHEDER,    DHEDER1    )
      EQUIVALENCE  ( DHEDER,    DHEDER2    )
      EQUIVALENCE  ( DHEDER,    DHEDER4    )
      EQUIVALENCE  ( DHEDER,    DHEDER8    )
      EQUIVALENCE  ( DHEDER,    DHEDERC    )
      WRITE (*, 12)
12    FORMAT (////////33X, 'DSCMAPIX'//27X, 'A PC  Utility',
     * /23X,  'for the Conversion of an ascii file PLANIMETRY'
     * /24x,  'file to a Mapix2 related files. '
     * //28X, 'Proyecto ESPOL - UNO'////////)
      PAUSE
C
C     MIN AND MAXES FOR VECTOR OBJECT FILES.
C
      XMIN =  99999999.0D0
      YMIN =  99999999.0D0
      XMAX = -99999999.0D0
      YMAX = -99999999.0D0
C
C     RETRIEVE INPUT FILE NAME.
C
      WRITE( *, * ) 'ENTER THE FILE TO ACCESS : '
      READ( *, '(A80)' ) TEXT2
C
C     RETRIEVE OUTPUT FILE NAME.
C
      WRITE( *, * ) 'ENTER THE NAME OF THE FILE TO CREATE : '
      READ( *, '(A64 )' ) TEXT
      LT = LENTEX ( TEXT )
      DO 5 I = LT, 1, -1
         IF ( TEXT(I:I) .EQ. '.' ) THEN
         LT = LT - 1
         GO TO 10
         END IF
    5 CONTINUE
   10 VPATH = TEXT(1:LT)
      TEXT3 = VPATH
      LOC = INDEX ( TEXT3, ' ' )
C
C     CREATE DATABASE FILE.
C
      TEXT3(LOC:) = '.DBF'
      CALL CONVCI ( TEXT3, PATH, 80 )
      CALL AFILE2 ( HANDL3, 11, 0, 0, PATH, ISTAT )
C
C     GET INFORMATION ABOUT MONTH,  YEAR,  DAY.
C
      CALL GETDAT ( DATE(1), DATE(2), DATE(3) )
      DO 102 LOOP1 = 1, 1284
          DHEDER(LOOP1) = 0
  102 CONTINUE
      DHEDER1(1) = 3
```

```
      DHEDER1(2) = MOD ( DATE(1), 100 )
      DHEDER1(3) = DATE(2)
      DHEDER1(4) = DATE(3)
      DHEDER (2) = 0
      DHEDER2(5) = 705
      DHEDER2(6) = 321
C
C     START OF THE USER DEFINED FIELDS.
C     NOTE THAT THE INDEX WIDTH IS THE NUMBER OF CHARACTERS IN THE
C    FIELD NAME,  AND THAT THE FIRST FIELD START IN THE POSITION 33.
C     FOR EVERY FIELD ADDED,  YOU HAVE TO ADD 32 TO THE START OF THE
C     INDEX.
C     DO NOT FORGET TO PUT THE AGIS FIELDS [5] AT THE END.
C
      DHEDERC(  33:38  ) = 'CODIGO'
      DHEDERC(  65:69  ) = 'TRACT'
      DHEDERC(  97:103 ) = 'TIPO_LN'
      DHEDERC( 129:135 ) = 'L_NORTE'
      DHEDERC( 161:167 ) = 'TIPO_LE'
      DHEDERC( 193:198 ) = 'L_ESTE'
      DHEDERC( 225:231 ) = 'TIPO_LS'
      DHEDERC( 257:261 ) = 'L_SUR'
      DHEDERC( 289:295 ) = 'TIPO_LO'
      DHEDERC( 321:327 ) = 'L_OESTE'
      DHEDERC( 353:360 ) = 'GRAD_DEF'
      DHEDERC( 385:387 ) = 'USO'
      DHEDERC( 417:422 ) = 'SECTOR'
      DHEDERC( 449:452 ) = 'FOTO'
      DHEDERC( 481:487 ) = 'SISTEMA'
      DHEDERC( 513:521 ) = 'PARROQUIA'
C
C     THIS FIELD IS GOING TO BE FILLED UP WITH THE OBJECT NUMBER.
C     ( THE LINE BELOW )
C
      DHEDERC( 545:553 ) = '_AGIS_00_'
      DHEDERC( 577:586 ) = '_AGIS_00_X'
      DHEDERC( 609:617 ) = 'AGIS_00_A'
      DHEDERC( 641:649 ) = 'AGIS_00_O'
      DHEDERC( 673:681 ) = 'AGIS_00_I'
C
C     ADDITIONAL FIELDS CAN BE ADDED HERE.
C
      DHEDERC(  44:44  ) = 'C'
      DHEDERC(  76:76  ) = 'C'
      DHEDERC( 108:108 ) = 'C'
      DHEDERC( 140:140 ) = 'C'
      DHEDERC( 172:172 ) = 'C'
      DHEDERC( 204:204 ) = 'C'
      DHEDERC( 236:236 ) = 'C'
      DHEDERC( 268:268 ) = 'C'
      DHEDERC( 300:300 ) = 'C'
      DHEDERC( 332:332 ) = 'C'
```

```
      DHEDERC( 364:364 ) = 'C'
      DHEDERC( 396:396 ) = 'C'
      DHEDERC( 428:428 ) = 'C'
      DHEDERC( 460:460 ) = 'N'
      DHEDERC( 492:492 ) = 'C'
      DHEDERC( 524:524 ) = 'C'
      DHEDERC( 556:556 ) = 'C'
      DHEDERC( 588:588 ) = 'C'
      DHEDERC( 620:620 ) = 'N'
      DHEDERC( 652:652 ) = 'N'
      DHEDERC( 684:684 ) = 'N'
C
      DHEDER1(  49 ) =  8
      DHEDER1(  81 ) =  3
      DHEDER1( 113 ) =  3
      DHEDER1( 145 ) = 35
      DHEDER1( 177 ) =  3
      DHEDER1( 209 ) = 35
      DHEDER1( 241 ) =  3
      DHEDER1( 273 ) = 35
      DHEDER1( 305 ) =  3
      DHEDER1( 337 ) = 35
      DHEDER1( 369 ) =  1
      DHEDER1( 401 ) =  5
      DHEDER1( 433 ) = 40
      DHEDER1( 465 ) =  4
      DHEDER1( 497 ) =  3
      DHEDER1( 529 ) = 30
      DHEDER1( 561 ) = 16
      DHEDER1( 593 ) =  4
      DHEDER1( 625 ) = 18
      DHEDER1( 657 ) = 18
      DHEDER1( 689 ) = 18
C
      DHEDER1( 50) = 0
      DHEDER1( 82) = 0
      DHEDER1(114) = 0
      DHEDER1(146) = 0
      DHEDER1(178) = 0
      DHEDER1(210) = 0
      DHEDER1(242) = 0
      DHEDER1(274) = 0
      DHEDER1(306) = 0
      DHEDER1(338) = 0
      DHEDER1(370) = 0
      DHEDER1(402) = 0
      DHEDER1(434) = 0
      DHEDER1(466) = 0
      DHEDER1(498) = 0
      DHEDER1(530) = 0
      DHEDER1(562) = 0
      DHEDER1(594) = 0
```

```
      DHEDER1(626) = 2
      DHEDER1(658) = 2
      DHEDER1(690) = 2
C
      DHEDER1(705) = 13
C
C     WRITE OUT BACK HEADER RECORD.
C
      CALL AFILE2 ( HANDL3, 4, 0, 705, DHEDER, ISTAT )
      LOC9 = 0
      LOC8 = 705
      TEXT3 = ' '
C
C     THIS INFORMATION IS IMPORTANT IF YOUR FILES ARE IN
C     LAT/LON GRID BASED ON WGS84 DATUM.
C
      TEXT = 'MERCATOR'
      CALL CONVCI ( TEXT, I4GEO(10), 36 )
      TEXT = '2.9 '
      CALL CONVCI ( TEXT, I4GEO(20), 4 )
      I4GEO(21) = 2
      I4GEO(24) = 1803
      R8GEO(14) = -79.879211111111100D0
      R8GEO(15) = -2.192525000D0
      R8GEO(30) = 500000.0D0
      R8GEO(31) = 2242320.5100D0
      I4GEO(63) = 47
      ISTAT = VEC_FILE ( VPATH, R8GEO, XMIN, XMAX, YMIN, YMAX )
      ISTAT = VEC_SET ( FILIST, VPATH )
C
C     START THE CODE PROCESSING ( OPEN THE INPUT FILE).
C
      OPEN ( 2, FILE = TEXT2, ACCESS = 'SEQUENTIAL',
     *       FORM = 'FORMATTED' )
C
C     LOOP THROUGH VERTICES.
C
      IFIRST = 1
      ITIEMPO = 0
      LOC1=14
   51 FORMAT(D25.9, A1, D25.8)
   50 CONTINUE
C
C     THIS LOOP IS BASED ON A CERTAIN FILE FORMAT STRUCTURE.
C
      READ ( 2, '(A80)', END = 100 ) TEXT3
      IF ( TEXT3  .NE. 'DXPL3L' ) GO TO 50
      READ ( 2, '(A80)', END = 100 ) TEXT3
      LOC = INDEX ( TEXT3, ',' )
      LOC2 = LOC - 1
      LOC1 = LOC1 - 10
      CALL LJUST ( TEXT3(LOC1:LOC2) )
```

```
      READ ( TEXT3(LOC1:LOC2), '(I10)' ) NUMV
      LOC1 = LOC + 1
      LOC2 = LOC1 + 12
      CALL LJUST ( TEXT3(LOC1:LOC2) )
      READ( TEXT3(LOC1:LOC2), '(I13)' ) ICLFL
      IX  = 1
      IY  = 2
      IVS = 1
      IFIRST = 1
C
C     LOOP FOR NUMBER OF VERTICIES.
C
      DO 1000 I = 1, NUMV
          READ ( 2, '(A80)', END = 100 ) TEXT3
C
C         THIS FILLS UP THE VERTICE STRING BUFFER.
C
          READ( TEXT3(1:25),  '(F25.9)' ) R8BUF(IX)
          READ( TEXT3(27:51), '(F25.9)' ) R8BUF(IY)
          IF ( IFIRST .EQ. 1 ) THEN
              STRTX = R8BUF(IX)
              STRTY = R8BUF(IY)
              IFIRST = 0
          END IF
          IX = IX + 2
          IY = IY + 2
          IVS = IVS + 1
C
C         IF THIS VALUE GETS ABOVE 1025 THEN THAT MEANS WE ARE
C         AT THE MAXIMUM NUMBER OF VERTICIES FOR A STRING.
C
          IF ( IVS .EQ. 1025 ) THEN
              ISTX = R8BUF(IX-2)
              ISTY = R8BUF(IY-2)
              ISTAT = VV_ADD ( FILIST(2), 1024, R8BUF )
              ISTAT2 = VO_ADD(1, FILIST(1), 0, 1, 1, ISTAT, 1024 )
              R8BUF(1) = ISTX
              R8BUF(2) = ISTY
              IX = 3
              IY = 4
              IVS = 2
          ENDIF
 1000 CONTINUE
      IFIRST = 1
      IF ( ICLFL .EQ. 1 ) ITYPE = 3
      IF ( ICLFL .EQ. 0 ) ITYPE = 2
      IF ( ITYPE .EQ. 3 ) THEN
          IF ( ( R8BUF(IX-2) .NE. STRTX ) .AND.
     *         ( R8BUF(IY-2) .NE. STRTY ) ) THEN
              R8BUF(IX) = STRTX
              R8BUF(IY) = STRTY
              IX = IX + 2
```

```
                IY = IY + 2
                IVS = IVS + 1
                IF ( IVS .EQ. 1025 ) THEN
                    ISTX = R8BUF(IX-2)
                    ISTY = R8BUF(IY-2)
                    ISTAT = VV_ADD ( FILIST(2), 1024, R8BUF )
                                ISTAT2 = VO_ADD(1, FILIST(1), 0, 1, 1, ISTAT, 1024)
                    R8BUF(1) = ISTX
                    R8BUF(2) = ISTY
                    IX = 3
                    IY = 4
                    IVS = 2
                ENDIF
                NUMV = NUMV + 1
            END IF
        END IF
        ISTAT = VV_ADD ( FILIST(2), NUMV, R8BUF )
        ISTAT2 = VO_ADD( 1, FILIST(1), 0, ITYPE, 1, ISTAT, NUMV )
        ISTAT3 = VO_ADD( 4, FILIST(1), 0, ITYPE, 1, ISTAT, NUMV )
        ITIEMPO = ITIEMPO + 1
        WRITE( *, * ) 'OBJECT NUMBER COLLECTED : ', ITIEMPO
C
C       FILL OUT DATA BASE RECORD.
C
        RECORD = ' '
C
C       THE LINE OF CODE BELOW WILL PUT ONE(1) INTO THE BLOCK
C       NUMBER FIELD BLOCKNO.
C
        IBLKNO = 44390000 + ITIEMPO
        WRITE ( RECORD(  2:9 ), '(I8)' ) IBLKNO
        CALL RJUST ( RECORD( 2:9 ) )
        WRITE ( RECORD( 248:263 ), '(I16)' ) ITIEMPO
        CALL RJUST ( RECORD( 248:263 ) )
        RECORD(268:285)  = '              0.00'
        RECORD(286:303)  = '              0.00'
        RECORD(304:321)  = '              0.00'
        CALL AFILE2( HANDL3, 4, LOC8, 321, RECORD, ISTAT )
        DHEDER(2) = DHEDER(2) + 1
        LOC8 = LOC8 + 321
        GOTO 50
  100 CLOSE ( 2 )
C
C       WRITE OUT BACK HEADER RECORD.
C
        CALL AFILE2 ( HANDL3, 4, 0, 705, DHEDER, ISTAT )
C
C       END OF DATABASE WRITES.
C
        DHEDER1(1) = 26
        LOC1 = 705 + DHEDER(2) * 321
C
```

```
C     WRITE OUT END OF FILE MARK.
C
      CALL AFILE2 ( HANDL3, 4, LOC1, 1, DHEDER1, ISTAT )
C
C     SUBROUTINE MUST BE CALLED TO UPDATE AREAS.
C
      ISTAT = VO_CALC ( FILIST, R8BUF, 16400 )
      END
```

B.3.2 Listing of Program DSCPOINT

```
      program DSCPOINT
C
C     This program reads an dsc ascii file output of pb4vax.for
C     and generates Mapix's related files (HEIGHTS).
C
C     Logical file names:
C         TEXT     Name of the Mapix related files
C         TEXT2    Ascii file from pb4vax.for.
C         TEXT3    TEXT without extension.
C
C     Size of DHEDER is at least 4 times the size of record.
C    Size of RECORD is ( Dbase [1] + Size of each field + size of all
C          the agis fields [5]).
C     Size of DHEDERC is (1 [dbase] + Number of Fields + 5[agis]) *
C                    32.
C
C     SUBROUTINES USED: CONVIC.
C                       AFILE2.
C                       VEC_FILE.
C                       VEC_SET.
C                       LJUST.
C                       VV_ADD.
C                       VO_ADD.
C                       RJUST.
C                       VO_CALC.
C ******
C     LANGUAGE:    Microsoft Fortran version 5.1.
C     DATE:        07 November 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      REAL*8     DHEDER8(1)
      REAL*8     R8GEO(32),  XMIN, XMAX, YMIN, YMAX,
     *           R8BUF(2050)
      REAL*8     ISTX, ISTY, STRTX, STRTY
      REAL*4     DHEDER4(1)
      INTEGER*4  VEC_FILE,   VEC_SET,  I4GEO(64), VO_ADD, VV_ADD
      INTEGER*4  DHEDER(832), PATH(20), VO_CALC
      INTEGER*2  DHEDER2(1), HANDL3, DATE(3)
```

```
      INTEGER*1  DHEDER1(1)
      CHARACTER*288 DHEDERC
      CHARACTER*150 RECORD
      CHARACTER*64  VPATH,   FILIST(3)
      CHARACTER*80  TEXT, TEXT2, TEXT3
      EQUIVALENCE  ( R8GEO,   I4GEO )
      EQUIVALENCE  ( DHEDER,    DHEDER1 )
      EQUIVALENCE  ( DHEDER,    DHEDER2 )
      EQUIVALENCE  ( DHEDER,    DHEDER4 )
      EQUIVALENCE  ( DHEDER,    DHEDER8 )
      EQUIVALENCE  ( DHEDER,    DHEDERC )
      WRITE (*, 12)
12    FORMAT (////////33X, 'DSCPOINT'//27X, 'A PC  Utility',
     * /23X,  'for the Conversion of an ascii file (HEIGTS)'
     * /24x,  'file to a Mapix2 related files. '
     * //28X, 'Proyecto ESPOL - UNO'////////)
      PAUSE
C
C     MIN AND MAXES FOR VECTOR OBJECT FILES.
C
      XMIN =  99999999.0D0
      YMIN =  99999999.0D0
      XMAX = -99999999.0D0
      YMAX = -99999999.0D0
C
C     RETRIEVE INPUT FILE NAME.
C
      WRITE( *, * ) 'ENTER THE FILE TO ACCESS : '
      READ( *, '(A80)' ) TEXT2
C
C     RETRIEVE OUTPUT FILE NAME.
C
      WRITE( *, * ) 'ENTER THE NAME OF THE FILE TO CREATE : '
      READ( *, '(A64)' ) TEXT
      LT = LENTEX ( TEXT )
      DO 5 I = LT, 1, -1
          IF ( TEXT(I:I) .EQ. '.' ) THEN
              LT = I - 1
              GO TO 10
          END IF
    5 CONTINUE
   10 VPATH = TEXT(1:LT)
      TEXT3 = VPATH
      LOC = INDEX ( TEXT3, ' ' )
C
C     CREATE DATABASE FILE.
C
      TEXT3(LOC:) = '.DBF'
      CALL CONVCI ( TEXT3, PATH, 80 )
      CALL AFILE2 ( HANDL3, 11, 0, 0, PATH, ISTAT )
      CALL GETDAT ( DATE(1), DATE(2), DATE(3) )
      DO 102 LOOP1 = 1, 444
```

```
          DHEDER(LOOP1) = 0
  102 CONTINUE
      DHEDER1(1) = 3
      DHEDER1(2) = MOD ( DATE(1), 100 )
      DHEDER1(3) = DATE(2)
      DHEDER1(4) = DATE(3)
      DHEDER (2) = 0
      DHEDER2(5) = 289
      DHEDER2(6) = 150
C
      DHEDERC( 33:40 ) = 'NORTHING'
      DHEDERC( 65:71 ) = 'EASTING'
      DHEDERC( 97:105) = 'ELEVATION'
      DHEDERC(129:137) = '_AGIS_00_'
      DHEDERC(161:170) = '_AGIS_00_X'
      DHEDERC(193:201) = 'AGIS_00_A'
      DHEDERC(225:233) = 'AGIS_00_O'
      DHEDERC(257:265) = 'AGIS_00_I'
C
C     ADDITIONAL FIELDS CAN BE ADDED HERE.
C
      DHEDERC( 44:44 ) = 'N'
      DHEDERC( 76:76 ) = 'N'
      DHEDERC(108:108) = 'N'
      DHEDERC(140:140) = 'C'
      DHEDERC(172:172) = 'C'
      DHEDERC(204:204) = 'N'
      DHEDERC(236:236) = 'N'
      DHEDERC(268:268) = 'N'
C
      DHEDER1( 49) = 25
      DHEDER1( 81) = 25
      DHEDER1(113) = 25
      DHEDER1(145) = 16
      DHEDER1(177) =  4
      DHEDER1(209) = 18
      DHEDER1(241) = 18
      DHEDER1(273) = 18
C
      DHEDER1( 50) = 9
      DHEDER1( 82) = 9
      DHEDER1(114) = 9
      DHEDER1(146) = 0
      DHEDER1(178) = 0
      DHEDER1(210) = 2
      DHEDER1(242) = 2
      DHEDER1(274) = 2
C
      DHEDER1(289) = 13
C
C     WRITE OUT BACK HEADER RECORD.
C
```

```
      CALL AFILE2 ( HANDL3, 4, 0, 289, DHEDER, ISTAT )
      LOC9 = 0
      LOC8 = 289
      TEXT3 = ' '
C
C     THIS INFORMATION IS IMPORTANT IF YOUR FILES ARE IN
C     LAT/LON GRID BASED ON WGS84 DATUM.
C
      TEXT = 'MERCATOR'
      CALL CONVCI ( TEXT, I4GEO(10), 36 )
      TEXT = '2.9 '
      CALL CONVCI ( TEXT, I4GEO(20), 4 )
      I4GEO(21) = 2
      I4GEO(24) = 1803
      R8GEO(14) = -79.879211111111100D0
      R8GEO(15) = -2.192525000D0
      R8GEO(30) = 500000.0D0
      R8GEO(31) = 2242320.5100D0
      I4GEO(63) = 47
      ISTAT = VEC_FILE ( VPATH, R8GEO, XMIN, XMAX, YMIN, YMAX )
      ISTAT = VEC_SET ( FILIST, VPATH )
C
C     START THE CODE PROCESSING ( SOME WAY WILL BE NEEDED TO OPEN THE
C     INPUT FILE.
C
      OPEN ( 2, FILE = TEXT2, ACCESS = 'SEQUENTIAL',
     *       FORM = 'FORMATTED' )
C
C     LOOP THROUGH VERTICIES.
C
      IFIRST = 1
      ITIEMPO = 0
   51 FORMAT(D25.9, A1, D25.8)
   50 CONTINUE
C
C     THIS LOOP IS BASED ON A CERTAIN FILE FORMAT STRUCTURE
C     YOURS WILL PROBABLY BE DIFFERENT.
C
      READ ( 2, '(A80)', END = 100 ) TEXT3
      IF ( TEXT3  .NE. 'DXPL3L' ) GO TO 50
      READ ( 2, '(A80)', END = 100 ) TEXT3
      LOC = INDEX ( TEXT3, ',' )
      LOC2 = LOC - 1
      LOC1 = LOC1 - 10
      CALL LJUST ( TEXT3(LOC1:LOC2) )
      READ ( TEXT3(LOC1:LOC2), '(I10)' ) NUMV
      LOC1 = LOC + 1
      LOC2 = LOC1 + 4
      CALL LJUST ( TEXT3(LOC1:LOC2) )
      READ( TEXT3(LOC1:LOC2), '(I5)' ) ICLFL
      IX  = 1
      IY  = 2
```

```
      IVS = 1
      IFIRST = 1
C
C     LOOP FOR NUMBER OF VERTICIES.
C
      DO 1000 I = 1, NUMV
          READ ( 2, '(A80)', END = 100 ) TEXT3
C
C         THIS FILLS UP THE VERTICE STRING BUFFER.
C
          READ( TEXT3(1:25),  '(F25.9)' ) R8BUF(1)
          READ( TEXT3(27:51), '(F25.9)' ) R8BUF(2)
          READ( TEXT3(53:77), '(F25.9)' ) R8BUF(3)
          ITYPE = 1
          ISTAT = VV_ADD ( FILIST(2), 1, R8BUF )
          ISTAT2 = VO_ADD( 1, FILIST(1), 0, ITYPE, 1, ISTAT, 1 )
          ISTAT3 = VO_ADD( 4, FILIST(1), 0, ITYPE, 1, ISTAT, 1 )
          ITIEMPO = ITIEMPO + 1
          WRITE( *, * ) 'OBJECT NUMBER COLLECTED : ', ITIEMPO
C
C         FILL OUT DATA BASE RECORD.
C
          RECORD = ' '
C
C         PLACE ELEVATION INTO ITS PROPER FIELD.
C
          WRITE ( RECORD(  2:26 ), '(F25.9)' ) R8BUF(2)
          CALL RJUST ( RECORD( 2:26 ) )
          WRITE ( RECORD(  27:51 ), '(F25.9)' ) R8BUF(1)
          CALL RJUST ( RECORD( 27:51 ) )
          WRITE ( RECORD(  52:76 ), '(F25.9)' ) R8BUF(3)
          CALL RJUST ( RECORD( 52:76 ) )
          WRITE ( RECORD( 77:92 ), '(I16)' ) ITIEMPO
          CALL RJUST ( RECORD( 77:92 ) )
          RECORD(97:114)   = '              0.00'
          RECORD(115:132)  = '              0.00'
          RECORD(133:150)  = '              0.00'
          CALL AFILE2( HANDL3, 4, LOC8, 150, RECORD, ISTAT )
          DHEDER(2) = DHEDER(2) + 1
          LOC8 = LOC8 + 150
 1000 CONTINUE
      GOTO 50
  100 CLOSE ( 2 )
C
C     WRITE OUT BACK HEADER RECORD.
C
      CALL AFILE2 ( HANDL3, 4, 0, 289, DHEDER, ISTAT )
C
C     END OF DATABASE WRITES.
C
      DHEDER1(1) = 26
      LOC1 = 289 + DHEDER(2) * 150
```

```
C
C     WRITE OUT END OF FILE MARK.
C
      CALL AFILE2 ( HANDL3, 4, LOC1, 1, DHEDER1, ISTAT )
C
C     SUBROUTINE MUST BE CALLED TO UPDATE AREAS.
C
      ISTAT = VO_CALC ( FILIST, R8BUF, 16400 )
      END
```

B.3.3 Listing of Program SHFTVEC

```
      PROGRAM SHFTVEC
C
C     THIS PROGRAM TO SHIFT LAT/LON VECTOR FILES IN ON DATUM TO
C     ANOTHER DATUM.
C
C     SUBROUTINES USED: BEEP.
C                       UPRCSA.
C                       CONVIC.
C                       AFILE2.
C                       VEC_FILE.
C                       VEC_SET.
C                       VO_CALC.
C ******
C     LANGUAGE:    Microsoft Fortran version 5.1.
C     DATE:        19 August 1994.
C     PROJECT:     ESPOL-UNO.
C ******
C
C     DECLARED VARIABLES.
C
      REAL*8    WORK(2050), DX, DY, DZ, DS, RX, RY, RZ, ORGX,
     *          ORGY, ORGZ, ELIPHT, DEC, TRANBF(1024)
      REAL*8    XMIN, XMAX, YMIN, YMAX
      INTEGER*4 SECFLG, DSHIFT, JMODEL, JDIREC, JUNIT1,
     *          JUNIT2, DATM1, DATM2, ISTBF(4)
      INTEGER*4 STRBF(64), STRBF2(64), STRBF3(64), GEO(64),
     *          STRBUF(64)
      INTEGER*4 VECTNM(20), LABLNM(20), VO_CALC
      INTEGER*4 VEC_SET,   VEC_FILE
      INTEGER*2 VOHAN1, VTHAN1, VLHAN1
      INTEGER*2 VOHAN2, VTHAN2, VLHAN2
      INTEGER*2 HANDL1, HANDL2, HANDL3
      CHARACTER*80      TEXT
      CHARACTER*64      LABL1,        LABL2,        OBJCT1,
     *                  OBJCT2,       VRTEX1,       VRTEX2
      CHARACTER*64      STRNAM(3)
```

```
C
C     DATUM SHIFT CODE.
C
    1 FORMAT(F26.15)
      JDIREC = 1
      JUNIT1 = 2
      JUNIT2 = 1
      WRITE( *, * ) 'ENTER TRANSFORMATION MODEL '
      WRITE( *, * ) ' 1 - BURSA MODEL '
      WRITE( *, * ) ' 2 - MOLODENSKY MODEL '
      READ( *, '(I4)' ) JMODEL
      WRITE( *, * )
      WRITE( *, * )
      IF ( JMODEL .EQ. 2 ) THEN
          WRITE( *, * ) 'ENTER ORIGIN X VALUE : '
          READ( *, 1) ORGX
          WRITE( *, * )
          WRITE( *, * ) 'ENTER ORIGIN Y VALUE : '
          READ( *, 1) ORGY
          WRITE( *, * )
          WRITE( *, * ) 'ENTER ORIGIN Z VALUE : '
          READ( *, 1) ORGZ
          ORGX     = DEC(ORGX)
          ORGY     = DEC(ORGY)
          ORGZ     = DEC(ORGZ)
      END IF
      WRITE( *, * )
      WRITE( *, * )
      WRITE( *, * ) 'CURRENT DATUM SELECTIONS'
      WRITE( *, * )
      WRITE( *, * )'PSAD56 - 7     WGS84 - 3    NAD27 - 1'
      WRITE( *, * )
      WRITE( *, * ) 'ENTER FIRST SYSTEM DATUM SELECTION :'
      READ( *, '(I4)') DATM1
      WRITE( *, * )
      WRITE( *, * )
      WRITE( *, * )
      WRITE( *, * ) 'CURRENT DATUM SELECTIONS'
      WRITE( *, * )
      WRITE( *, * ) 'PSAD56 - 7     WGS84 - 3    NAD27 - 1'
      WRITE( *, * )
      WRITE( *, * ) 'ENTER SECOND SYSTEM DATUM SELECTION :'
      READ( *, '(I4)') DATM2
      WRITE( *, * )
      WRITE( *, * )
      WRITE( *, * ) 'ENTER DX :'
      READ( *, 1 ) DX
      WRITE( *, * )
      WRITE( *, * ) 'ENTER DY :'
      READ( *, 1 ) DY
      WRITE( *, * )
      WRITE( *, * ) 'ENTER DZ :'
```

```
      READ( *, 1 ) DZ
      WRITE( *, * )
      WRITE( *, * ) 'ENTER EPSILON_X : '
      READ( *, 1 ) RX
      WRITE( *, * )
      WRITE( *, * ) 'ENTER PSI_Y   : '
      READ( *, 1 ) RX
      WRITE( *, * )
      WRITE( *, * ) 'ENTER OMEGA_Z :'
      READ( *, 1 ) RZ
      WRITE( *, * )
      WRITE( *, * ) 'ENTER DS :'
      READ( *, 1 ) DS
      WRITE( *, * )
      WRITE( *, * ) 'ENTER 1 FOR SEC OR 0 FOR RAD :'
      READ( *, '(I4)' ) SECFLG
      WRITE( *, * )
      WRITE( *, * ) 'ENTER AVERAGE HEIGHT ABOVE ELLIPSOID :'
      READ( *, 1 ) ELIPHT
      WRITE( *, * )
      WRITE( *, * )
C
C     GET VECTOR FILE NAME TO SHIFT.
C
    5 WRITE( *, * ) 'ENTER THE VECTOR FILE TO SHIFT :'
      WRITE( *, * ) 'ENTER ! TO EXIT :'
      READ ( *, '(A64)' ) OBJCT1
      CALL UPRCSA ( OBJCT1, 64 )
      IF ( OBJCT1(1:1) .EQ. '!' ) GO TO 9999
      LT    = INDEX ( OBJCT1, '.' )
      IF ( LT .EQ. 0 ) THEN
          LT = INDEX ( OBJCT1, ' ' )
          OBJCT1(LT:) = '.VOB'
      ELSE
          IF ( OBJCT1(LT+1:LT+3) .NE. 'VOB' ) THEN
              WRITE( *, * )
              WRITE( *, * ) 'INVALID FILE NAME'
              WRITE( *, * )
              WRITE( *, * )
              CALL BEEP
              GO TO 5
          END IF
      END IF
C
C     CHECK IF FILE EXIST.
C
      CALL CONVCI ( OBJCT1, STRBF, 64 )
      CALL AFILE2 ( VOHAN1, 13, 0, 0, STRBF, ISTBF )
      IF ( ISTBF(1) .LT. 0 ) THEN
          WRITE( *, * )
          WRITE( *, * ) 'INVALID FILE NAME'
          WRITE( *, * )
```

```
            WRITE( *, * )
            CALL BEEP
            GO TO 5
          END IF
C
C       GET VECTOR FILE NAME TO SHIFT.
C
        6 WRITE( *, * ) 'ENTER THE VECTOR FILE TO CREATE :'
          WRITE( *, * ) 'ENTER ! TO EXIT :'
          READ ( *, '(A64)' ) OBJCT2
          CALL UPRCSA ( OBJCT2, 64 )
          IF ( OBJCT2(1:1) .EQ. '!' ) GO TO 9999
          LT    = INDEX ( OBJCT2, '.' )
          IF ( LT .EQ. 0 ) THEN
              LT = INDEX ( OBJCT2, ' ' )
              OBJCT2(LT:) = '.VOB'
          ELSE
              IF ( OBJCT2(LT+1:LT+3) .NE. 'VOB' ) THEN
                  WRITE( *, * )
                  WRITE( *, * ) 'INVALID FILE NAME'
                  WRITE( *, * )
                  WRITE( *, * )
                  CALL BEEP
                  GO TO 6
              END IF
          END IF
C
C       CHECK IF FILE EXIST.
C
          CALL CONVCI ( OBJCT2, STRBF2, 64 )
          CALL AFILE2 ( VOHAN2, 13, 0, 0, STRBF2, ISTBF )
          IF ( ISTBF(1) .GE. 0 ) THEN
              WRITE( *, * )
              WRITE( *, * ) 'INVALID FILE NAME'
              WRITE( *, * )
              WRITE( *, * )
              CALL BEEP
              GO TO 6
          END IF
C
C       GET GEO-REFERECNCING RECORD.
C
          TEXT = 'LAT/LON GRID'
          CALL CONVCI ( TEXT, GEO(10), 36 )
          DO 7 ITIME = 19, 57
              GEO(ITIME) = 0
        7 CONTINUE
          TEXT = '2.9 '
          CALL CONVCI ( TEXT, GEO(20), 4 )
          GEO(21) = 1
C
C       CREATE NEW VECTOR FILES.
```

```
C
      XMIN   =  99999999.0D0
      XMAX   = -99999999.0D0
      YMIN   =  99999999.0D0
      YMAX   = -99999999.0D0
      STRNAM(1) = OBJCT2
      ISTAT = VEC_FILE ( STRNAM, GEO, XMIN, XMAX,
     *                   YMIN, YMAX )
      IF ( ISTAT .LT. 0 ) THEN
          CALL BEEP
          WRITE( *, * ) 'UNABLE TO CREATE VECTOR FILE'
          GO TO 9999
      END IF
      STATUS = VEC_SET( STRNAM, OBJCT2 )
      VRTEX2 = STRNAM(2)
      LABL2  = STRNAM(3)
C
C     OPEN VECTOR OBJECT FILES NAMES.
C
      CALL CONVCI ( OBJCT1, STRBUF, 64 )
      CALL CONVCI ( OBJCT2, STRBF2, 64 )
C
C     GET LENGTH OF FIRST FILE.
C
      CALL AFILE2 ( VOHAN1, 13, 0, 0, STRBUF, ISTBF )
C
C     OPEN VECTOR OBJECT FILES.
C
      CALL AFILE2 ( VOHAN1, 5, 0, 0, STRBUF, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 5, 0, 0, STRBF2, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     GET FIRST VECTOR VERTEX FILE AND LABEL NAMES.
C
   15 CALL AFILE2 ( VOHAN1, 3, 128, 64, VRTEX1, ISTAT )
      IF ( ISTAT .NE. 64 ) GO TO 995
      CALL AFILE2 ( VOHAN1, 3, 192, 64, LABL1, ISTAT )
      IF ( ISTAT .NE. 64 ) GO TO 995
C
C     TRANSFER SOME HEADER INFORMATION.
C
      CALL AFILE2 ( VOHAN1, 3, 0, 92, TRANBF, ISTAT )
      IF ( ISTAT .NE. 92 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 4, 0, 92, TRANBF, ISTAT )
      IF ( ISTAT .NE. 92 ) GO TO 995
C
C     MOVE REST OF OBJECT FILE.
C
      NBYTL  = ISTBF(4) - 512
      NBYTM  = 8192
      NTIMES = NBYTL / NBYTM
```

```
      LEFT   = AMOD ( NBYTL, NBYTM )
      IF ( LEFT .NE. 0 ) THEN
          NTIMES = NTIMES + 1
      ELSE
          LEFT = NBYTM
      END IF
      ISTRT = 512
      DO 25 IMV2 = 1, NTIMES
          IF ( IMV2 .EQ. NTIMES ) NBYTM = LEFT
          CALL AFILE2 ( VOHAN1, 3, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          CALL AFILE2 ( VOHAN2, 4, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          ISTRT = ISTRT + NBYTM
   25 CONTINUE
C
C     CLOSE VECTOR OBJECT FILE.
C
      CALL AFILE2 ( VOHAN1, 6, 0, 0, STRBUF, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 6, 0, 0, STRBF2, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     OPEN VECTOR VERTEX FILES NAMES.
C
      CALL CONVCI ( VRTEX1, STRBUF, 64 )
      CALL CONVCI ( VRTEX2, STRBF2, 64 )
C
C     GET LENGTH OF FIRST FILE.
C
      CALL AFILE2 ( VOHAN1, 13, 0, 0, STRBUF, ISTBF )
C
C     OPEN VECTOR VERTEX FILES.
C
      CALL AFILE2 ( VOHAN1, 5, 0, 0, STRBUF, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 5, 0, 0, STRBF2, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     TRANSFER SOME HEADER INFORMATION.
C
      CALL AFILE2 ( VOHAN1, 3, 0, 128, TRANBF, ISTAT )
      IF ( ISTAT .NE. 128 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 4, 0, 128, TRANBF, ISTAT )
      IF ( ISTAT .NE. 128 ) GO TO 995
C
C     MOVE REST OF VERTEX FILE.
C
      NBYTL  = ISTBF(4) - 256
      NBYTM  = 8192
      NTIMES = NBYTL / NBYTM
      LEFT   = AMOD ( NBYTL, NBYTM )
```

```
      IF ( LEFT .NE. 0 ) THEN
          NTIMES = NTIMES + 1
      ELSE
          LEFT = NBYTM
      END IF
      ISTRT = 256
      DO 26 IMV2 = 1, NTIMES
          IF ( IMV2 .EQ. NTIMES ) NBYTM = LEFT
          CALL AFILE2 ( VOHAN1, 3, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          CALL AFILE2 ( VOHAN2, 4, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          ISTRT = ISTRT + NBYTM
   26 CONTINUE
C
C     CLOSE VECTOR VERTEX FILES.
C
      CALL AFILE2 ( VOHAN1, 6, 0, 0, STRBUF, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 6, 0, 0, STRBF2, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     OPEN VECTOR LABEL FILES NAMES.
C
      CALL CONVCI ( LABL1, STRBUF, 64 )
      CALL CONVCI ( LABL2, STRBF2, 64 )
C
C     GET LENGTH OF FIRST FILE.
C
      CALL AFILE2 ( VOHAN1, 13, 0, 0, STRBUF, ISTBF)
C
C     OPEN VECTOR LABEL FILES.
C
      CALL AFILE2 ( VOHAN1, 5, 0, 0, STRBUF, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 5, 0, 0, STRBF2, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     TRANSFER SOME HEADER INFORMATION.
C
      CALL AFILE2 ( VOHAN1, 3, 0, 128, TRANBF, ISTAT )
      IF ( ISTAT .NE. 128 ) GO TO 995
      CALL AFILE2 ( VOHAN2, 4, 0, 128, TRANBF, ISTAT )
      IF ( ISTAT .NE. 128 ) GO TO 995
C
C     MOVE REST OF VERTEX FILE.
C
      NBYTL  = ISTBF(4) - 256
      NBYTM  = 8192
      NTIMES = NBYTL / NBYTM
      LEFT   = AMOD ( NBYTL, NBYTM )
      IF ( LEFT .NE. 0 ) THEN
```

```
          NTIMES = NTIMES + 1
        ELSE
          LEFT = NBYTM
        END IF
        ISTRT = 256
        DO 27 IMV2 = 1, NTIMES
          IF ( IMV2 .EQ. NTIMES ) NBYTM = LEFT
          CALL AFILE2 ( VOHAN1, 3, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          CALL AFILE2 ( VOHAN2, 4, ISTRT, NBYTM, TRANBF, ISTAT )
          IF ( ISTAT .NE. NBYTM ) GO TO 995
          ISTRT = ISTRT + NBYTM
   27   CONTINUE
C
C       CLOSE VECTOR LABEL FILES.
C
        CALL AFILE2 ( VOHAN1, 6, 0, 0, STRBUF, ISTAT )
        IF ( ISTAT .NE. 0 ) GO TO 995
        CALL AFILE2 ( VOHAN2, 6, 0, 0, STRBF2, ISTAT )
        IF ( ISTAT .NE. 0 ) GO TO 995
C
C       TRANSFORM THE DATA.
C
C       OPEN NEW VECTOR OBJECT FILE AND START TRANSFORMATION.
C
        CALL CONVCI ( OBJCT2, STRBUF(9), 64 )
        CALL AFILE2 ( HANDL1, 5, 0, 0, STRBUF(9), ISTAT )
        IF ( ISTAT .NE. 0 ) GO TO 995
C
C       FILE HAS NOT BEEN TRANSFORMED - SO LET'S DO IT.
C
   28   CALL AFILE2 ( HANDL1, 3, 128, 64, VECTNM, ISTAT )
        IF ( ISTAT .NE. 64 ) GO TO 995
        CALL AFILE2 ( HANDL1, 3, 192, 64, LABLNM, ISTAT )
        IF ( ISTAT .NE. 64 ) GO TO 995
C
C       SET UP STRUCTURE FILE NAMES.
C
        CALL CONVIC ( STRBUF(9), STRNAM(1), 64 )
        CALL CONVIC ( VECTNM,    STRNAM(2), 64 )
        CALL CONVIC ( LABLNM,    STRNAM(3), 64 )
C
C       CLOSE VECTOR OBJECT FILE.
C
        CALL AFILE2 ( HANDL1, 6, 0, 0, STRBUF(9), ISTAT )
        IF ( ISTAT .NE. 0 ) GO TO 995
C
C       OPEN VERTEX FILE.
C
        CALL AFILE2 ( HANDL2, 5, 0, 0, VECTNM, ISTAT )
        IF ( ISTAT .NE. 0 ) GO TO 995
C
```

```
C       READ IN NUMBER OF VERTICIE STRINGS.
C
        CALL AFILE2 ( HANDL2, 3, 64, 12, STRBF3(1), ISTAT )
        IF ( ISTAT .NE. 12 ) GO TO 995
        ISTRNG = STRBF3(1)
        IFVRT  = STRBF3(2)
        LAVRT  = STRBF3(3)
C
C       WRITE COORDINATE FLAG TO HEADER.
C
        ICT = 0
        CALL AFILE2 ( HANDL2, 4, 80, 4, ICT, ISTAT )
        IF ( ISTAT .NE. 4 ) GO TO 995
C
C       SET MIN AND MAX VARIABLES.
C
        XMIN =  99999999.0D0
        XMAX = -99999999.0D0
        YMIN =  99999999.0D0
        YMAX = -99999999.0D0
C
C       INITIALIZE USER PACIFYING MESSAGE.
C
        WRITE( *, * ) ' TRANSFORMING VERTICIES. PLEASE WAIT ....'
C
C       READ AND CONVERT VERTICIES.
C
        DO 29 ITIME = 1, ISTRNG
C
C           GET FIRST STRING LENGTH.
C
            CALL AFILE2 ( HANDL2, 3, IFVRT, 4, STRBF3(1), ISTAT )
            IF ( ISTAT .NE. 4 ) GO TO 995
            NFVRT = IFVRT + STRBF3(1)
            INVRT = ( STRBF3(1) - 40 ) / 16.0
C
C           GET NUMBER OF TIMES TO LOOP.
C
            NTIMES = INVRT / 512
            LCOMT  = NTIMES
            LOC    = IFVRT + 40
            LEFT   = AMOD ( INVRT, 512 )
            IF ( LEFT .NE. 0 ) THEN
                NTIMES = NTIMES + 1
            ELSE
                LEFT = 512
            END IF
C
C           NUMBER OF BYTES TO READ IN FROM STRING.
C
            NBYTE1 = 8192
            NPARS  = 512
```

```
            NBYTE2 = LEFT * 16
            NPARS2 = LEFT
C
C           TRANSFORM POINTS.
C
            DO 30 IMV1 = 1, NTIMES
                IF ( IMV1 .EQ. NTIMES ) THEN
                    NBYTE1 = NBYTE2
                    NPARS  = NPARS2
                END IF
C
C               READ FROM VERTEX FILE.
C
                CALL AFILE2 ( HANDL2, 3, LOC, NBYTE1, TRANBF, ISTAT )
                IF ( ISTAT .NE. NBYTE1 ) GO TO 995
C
C               PERFORM TRANSFORMATION  27 - 83.
C
                          ISTAT = DSHIFT( NPARS, JDIREC, JMODEL, JUNIT1, JUNIT2,
            *                   DATM1, DATM2, DX, DY, DZ, DS, RX, RY,RZ,
            *                  ORGX, ORGY, ORGZ, ELIPHT, SECFLG,TRANBF)
                LOCX1 = 1
                LOCY1 = 2
                DO 31 J = 1, NPARS
C
C                   FIND MIN AND MAX VALUES.
C
                                  IF (XMIN .GT. TRANBF(LOCX1) ) XMIN = TRANBF(LOCX1)
                                  IF (XMAX .LT. TRANBF(LOCX1) ) XMAX = TRANBF(LOCX1)
                                  IF (YMIN .GT. TRANBF(LOCY1) ) YMIN = TRANBF(LOCY1)
                                  IF (YMAX .LT. TRANBF(LOCY1) ) YMAX = TRANBF(LOCY1)
                    LOCX1 = LOCX1 + 2
                    LOCY1 = LOCY1 + 2
   31           CONTINUE
C
C               WRITE TO VERTEX FILE.
C
                CALL AFILE2 ( HANDL2, 4, LOC, NBYTE1, TRANBF, ISTAT )
                IF ( ISTAT .NE. NBYTE1 ) GO TO 995
C
C               UPDATE LOCATION.
C
                LOC = LOC + NBYTE1
   30        CONTINUE
             IFVRT = NFVRT
   29 CONTINUE
C
C      WRITE NEW MIN AND MAX VERTICIE LIMITS TO FILE.
C
       CALL AFILE2 ( HANDL2, 4,  96, 8, XMIN, ISTAT )
       IF ( ISTAT .NE. 8 ) GO TO 995
       CALL AFILE2 ( HANDL2, 4, 104, 8, XMAX, ISTAT )
```

```
      IF ( ISTAT .NE. 8 ) GO TO 995
      CALL AFILE2 ( HANDL2, 4, 112, 8, YMIN, ISTAT )
      IF ( ISTAT .NE. 8 ) GO TO 995
      CALL AFILE2 ( HANDL2, 4, 120, 8, YMAX, ISTAT )
      IF ( ISTAT .NE. 8 ) GO TO 995
C
C     CLOSE VERTEX FILE FOR NOW.
C
      CALL AFILE2 ( HANDL2, 6, 0, 0, VECTNM, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     OPEN LABEL FILE.
C
      CALL AFILE2 ( HANDL3, 5, 0, 0, LABLNM, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     READ IN NUMBER OF LABELS.
C
      CALL AFILE2 ( HANDL3, 3, 64, 12, STRBF3, ISTAT )
      IF ( ISTAT .NE. 12 ) GO TO 995
      ISTRNG = STRBF3(1)
      IFLBB  = STRBF3(2)
      ILLBB  = STRBF3(3)
C
C     WRITE NEW COORDINATE TYPE FLAG TO LABEL FIELD.
C
      ICT = 0
      CALL AFILE2 ( HANDL3, 4, 80, 4, ICT, ISTAT )
      IF ( ISTAT .NE. 4 ) GO TO 995
C
C     LOOP THROUGH LABELS.
C
      WRITE( *, * ) ' TRANSFORMING LABELS. PLEASE WAIT ....'
      DO 34 IMV1 = 1, ISTRNG
          CALL AFILE2 ( HANDL3, 3, IFLBB, 8, STRBF3, ISTAT )
          IF ( ISTAT .NE. 8 ) GO TO 995
C
C         GET LABEL START X AND Y LOCATIONS.
C
          ILLOC = IFLBB + 24
          CALL AFILE2 ( HANDL3, 3, ILLOC, 16, TRANBF, ISTAT )
          IF ( ISTAT .NE. 16 ) GO TO 995
          NPARS  =  1
          ISTAT  =  DSHIFT( NPARS, JDIREC, JMODEL, JUNIT1, JUNIT2,
          *                 DATM1, DATM2, DX, DY, DZ, DS, RX, RY, RZ,
          *                ORGX, ORGY, ORGZ, ELIPHT, SECFLG, TRANBF)
C
C         WRITE TEXT CORNERS BACK OUT TO FILE.
C
          CALL AFILE2 ( HANDL3, 4, ILLOC, 16, TRANBF, ISTAT )
          IF ( ISTAT .NE. 16 ) GO TO 995
          IFLBB = IFLBB + STRBF3(2)
```

```
   34 CONTINUE
C
C     ANOTHER USER PACIFYING MESSAGE.
C
            TEXT = 'UPDATING DRAWING HEADER INFORMATION. PLEASE WAIT ....'
      WRITE( *, * ) TEXT
C
C     CLOSE LABEL FILE.
C
      CALL AFILE2 ( HANDL3, 6, 0, 0, LABLNM, ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     LET'S REOPEN THE VECTOR OBJECT FILE.
C
   35 CALL AFILE2 ( HANDL1, 5, 0, 0, STRBUF(9), ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     READ IN NUMBER OF OBJECTS.
C
      CALL AFILE2 ( HANDL1, 3, 64, 8, STRBF2, ISTAT )
      IF ( ISTAT .NE. 8 ) GO TO 995
C
C     WRITE NEW COORDINATE TYPE FLAG TO LABEL FIELD.
C
      ICT = 0
      CALL AFILE2 ( HANDL1, 4, 80, 4, ICT, ISTAT )
      IF ( ISTAT .NE. 4 ) GO TO 995
C
C     CHANGE ISLAND PROCESSING FLAG TO REQUIRED.
C
      ICT2 = 1
      CALL AFILE2 ( HANDL1, 4, 92, 4, ICT2, ISTAT )
      IF ( ISTAT .NE. 4 ) GO TO 995
C
C     LOOP THROUGH OBJECTS AND CHANGE AREA FLAGS.
C
      IFBYT = STRBF2(2)
      DO 36 IMV2 = 1, STRBF2(1)
C
C         READ IN LENGTH OF THE OBJECT 1 DESCRIPTOR.
C
          ILOC1 = IFBYT + 4
          CALL AFILE2 ( HANDL1, 3, ILOC1, 4, STRBF3(1), ISTAT )
          IF ( ISTAT .NE. 4 ) GO TO 995
C
C         CHANGE AREA FLAGS.
C
          ILOC1 = IFBYT + 56
          TRANBF(1) = -1.0D0
          CALL AFILE2 ( HANDL1, 4, ILOC1, 8, TRANBF(1), ISTAT )
          IF ( ISTAT .NE. 8 ) GO TO 995
          IFBYT = IFBYT + STRBF3(1)
```

```
   36 CONTINUE
C
C     CLOSE VECTOR OBJECT FILE.
C
      CALL AFILE2 ( HANDL1, 6, 0, 0, STRBUF(9), ISTAT )
      IF ( ISTAT .NE. 0 ) GO TO 995
C
C     FIND MIN AND MAX VALUES AND AREA FOR OBJECT.
C
      IBYT  = 16400
      ISTAT = VO_CALC ( STRNAM, WORK, IBYT )
      IF ( ISTAT .NE. 0 ) THEN
          TEXT = 'UNABLE TO COMPUTE MIN AND MAX VERTICIES'
          WRITE( *, * ) TEXT
          CALL BEEP
          GO TO 9999
      END IF
C
C     SKIP ERROR HANDLING.
C
      GO TO 9999
  995 WRITE( *, * ) 'DISK I/O ERROR'
      CALL BEEP
 9999 CONTINUE
      END
```

B.3.4 Listing of Program MERGVEC

```
      program mergvec
C
C     This program is for merging several mapix related files
C     the first file is the base file,  all subsequent  files
C     are then added to the first one.
C
C     SUBROUTINES USED: UPRCSA.
C                       BEEP.
C                       CONVCI.
C                       AFILE2.
C                       VEC_SET.
C                       VO_CALC.
C ******
C     LANGUAGE:    Microsoft Fortran version 5.1.
C     DATE:        08 November 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      REAL*8     DHEDER8(1), D2EDER8(1)
      REAL*8     R8GEO(32),  XMIN, XMAX, YMIN, YMAX,
     *           R8BUF(2056)
      REAL*8     ISTX, ISTY
      REAL*4     DHEDER4(1), D2EDER4(1)
```

```
      INTEGER*4  VEC_FILE,   VEC_SET,  I4GEO(64), VO_ADD, VV_ADD
      INTEGER*4  DHEDER(1350), PATH(20), PATH2(20), VO_CALC,
     *           D2EDER(1350)
      INTEGER*4  PATH3(20), ISTBF(4)
      INTEGER*2  DHEDER2(1), HANDL3, DATE(3), HANDLV, HANDL4,
     *           D2EDER2(1)
      INTEGER*2  HANDVO, HANDVT, IHREC, IHREC2, IRCLN, IRCLN2
      INTEGER*1  DHEDER1(1), D2EDER1(1)
      CHARACTER*32000 DHEDERC,D2EDERC
      CHARACTER*32000 RECORD
      CHARACTER*64   VPATH,   FILIST(3), FILIST2(3), VPATH2
      CHARACTER*64   FILS(1024)
      CHARACTER*80   TEXT, TEXT2, TEXT3, TEXT4
      CHARACTER*6    ATR
      EQUIVALENCE  ( R8GEO,   I4GEO )
      EQUIVALENCE  ( DHEDER,    DHEDER1 )
      EQUIVALENCE  ( DHEDER,    DHEDER2 )
      EQUIVALENCE  ( DHEDER,    DHEDER4 )
      EQUIVALENCE  ( DHEDER,    DHEDER8 )
      EQUIVALENCE  ( DHEDER,    DHEDERC )
      EQUIVALENCE  ( D2EDER,    D2EDER1 )
      EQUIVALENCE  ( D2EDER,    D2EDER2 )
      EQUIVALENCE  ( D2EDER,    D2EDER4 )
      EQUIVALENCE  ( D2EDER,    D2EDER8 )
      EQUIVALENCE  ( D2EDER,    D2EDERC )
C
C     MIN AND MAXES FOR VECTOR OBJECT FILES.
C
      XMIN =  99999999.0D0
      YMIN =  99999999.0D0
      XMAX = -99999999.0D0
      YMAX = -99999999.0D0
      WRITE (*, 12)
12    FORMAT (////////33X, 'MERGVEC'//27X, 'A PC Utility',
     * /23X,  'for the MERGE of Mapix2 related files. '
     * //28X, 'Proyecto ESPOL - UNO'////////)
      PAUSE
C
C     OUTPUT FILE INFORMATION.
C
    5 WRITE( *, * ) 'ENTER THE BASE DRAWING NAME OR "!" TO EXIT'
      READ( *, '(A80)' ) TEXT
      CALL UPRCSA ( TEXT, 80 )
      IF ( TEXT(1:1) .EQ. '!' ) GO TO 9999
      LT    = INDEX ( TEXT, '.' )
      IF ( LT .EQ. 0 ) THEN
          LT = INDEX ( TEXT, ' ' )
          TEXT(LT:) = '.VOB'
      ELSE
          IF ( TEXT(LT+1:LT+3) .NE. 'VOB' ) THEN
              WRITE( *, * )
              WRITE( *, * ) 'INVALID FILE NAME'
```

```
                WRITE( *, * )
                WRITE( *, * )
                CALL BEEP
                GO TO 5
            END IF
        END IF
C
C       CHECK IF FILE EXIST.
C
        CALL CONVCI ( TEXT, PATH, 80 )
        CALL AFILE2 ( HANDVT, 13, 0, 0, PATH, ISTBF )
        IF ( ISTBF(1) .LT. 0 ) THEN
            WRITE( *, * )
            WRITE( *, * ) 'INVALID FILE NAME'
            WRITE( *, * )
            WRITE( *, * )
            CALL BEEP
            GO TO 5
        END IF
C
C       OPEN BASE FILE TO GET FIRST AVAILABLE OBJECT NAME.
C
        CALL AFILE2 ( HANDVT, 5, 0, 0, PATH, IST )
        CALL AFILE2 ( HANDVT, 3, 88, 4, INOB, IST )
        CALL AFILE2 ( HANDVT, 6, 0, 0, PATH, IST )
C
C       OUTPUT FILE INFORMATION.
C
        INUM = 0
    6   WRITE( *, * )
        WRITE( *, * )
        WRITE( *, * ) 'ENTER THE INPUT FILE DRAWING : '
        WRITE( *, * ) 'ENTER "END" FOR NO MORE DRAWINGS : '
        WRITE( *, * ) 'ENTER "!" TO EXIT : '
        READ( *, '(A80)' ) TEXT2
        CALL UPRCSA ( TEXT2, 80 )
        IF ( TEXT2(1:1) .EQ. '!' ) GO TO 9999
        IF ( TEXT2(1:3) .EQ. 'END' ) GO TO 1000
        LT    = INDEX ( TEXT2, '.' )
        IF ( LT .EQ. 0 ) THEN
            LT = INDEX ( TEXT2, ' ' )
            TEXT2(LT:) = '.VOB'
        ELSE
            IF ( TEXT2(LT+1:LT+3) .NE. 'VOB' ) THEN
                WRITE( *, * )
                WRITE( *, * ) 'INVALID FILE NAME'
                WRITE( *, * )
                WRITE( *, * )
                CALL BEEP
                GO TO 6
            END IF
        END IF
```

```
C
C     CHECK IF FILE EXIST.
C
      CALL CONVCI ( TEXT2, PATH, 80 )
      CALL AFILE2 ( HANDVT, 13, 0, 0, PATH, ISTBF )
      IF ( ISTBF(1) .LT. 0 ) THEN
          WRITE( *, * )
          WRITE( *, * ) 'INVALID FILE NAME'
          WRITE( *, * )
          WRITE( *, * )
          CALL BEEP
          GO TO 6
      END IF
C
C     UPDATE FILES LIST.
C
      INUM = INUM + 1
      FILS(INUM) = TEXT2(1:64)
      IF ( INUM .EQ. 1024 ) THEN
          WRITE( *, * ) 'NO MORE FILE CAN BE ADD CONTINUE'
          CALL BEEP
          READ( *, * )
          GO TO 1000
      END IF
      GO TO 6
C
C     START PROCESSING.
C
 1000 LT    = INDEX ( TEXT, '.' ) - 1
      VPATH  = TEXT(1:64)
      TEXT3  = TEXT(1:LT)
C
C     CHECK IF FILE EXIST AND OPEN.
C
      CALL CONVCI ( TEXT, PATH2, 80 )
      CALL AFILE2 ( HANDLV, 5, 0, 0, PATH2, ISTAT )
      CALL AFILE2 ( HANDLV, 3, 96, 32, R8GEO, ISTAT )
      XMIN =  R8GEO(1)
      YMIN =  R8GEO(3)
      XMAX =  R8GEO(2)
      YMAX =  R8GEO(4)
C
C     WRITE A TWO TO CALCULATE LIMITS.
C
      ICHG = 2
      CALL AFILE2 ( HANDLV, 4, 92, 4, ICHG, ISTAT )
      CALL AFILE2 ( HANDLV, 3, 64, 4, IONB, ISTAT )
      CALL AFILE2 ( HANDLV, 6, 0, 0, PATH2, ISTAT )
      ISTAT = VEC_SET ( FILIST, VPATH )
C
C     OUTPUT DATA BASE FILE.
C
```

```
      TEXT3(LT+1:) = '.DBF'
      CALL CONVCI ( TEXT3, PATH, 80 )
C
C     IF FILE EXIST, OPEN AND APPEND IT.
C
      CALL AFILE2 ( HANDL3, 5, 0, 0, PATH, ISTAT )
C
C     GET HEADER BYTES.
C
      CALL AFILE2 ( HANDL3, 3, 8, 2, IHREC, ISTAT )
      IF ( IHREC .GT. 32000 ) THEN
          WRITE( *, * )
          WRITE( *, * ) 'DATABASE HEADER TOO LARGE'
          WRITE( *, * )
          CALL BEEP
          GO TO 9999
      END IF
      CALL AFILE2 ( HANDL3, 3, 10, 2, IRCLN, ISTAT )
      IF ( IRCLN .GT. 32000 ) THEN
          WRITE( *, * )
          WRITE( *, * ) 'RECORD LENGTH TOO LARGE'
          WRITE( *, * )
          CALL BEEP
          GO TO 9999
      END IF
C
C     READ IN HEADER.
C
      IRDB = IHREC
      CALL AFILE2 ( HANDL3, 3, 0, IRDB, DHEDER, ISTAT )
C
C     LOOP THROUGH AND GET FIELD POSITION TO WRITE TOO.
C
      ISTOP = IHREC - 1
      IVLOC  = 2
      IPLAC = 49
      DO 9551 ITIME = 33, ISTOP, 32
          IF ( DHEDERC(ITIME:ITIME+8) .EQ. '_AGIS_00_' ) GO TO 9552
          IVALB = DHEDER1(IPLAC)
          IVLOC = IVLOC + IVALB
          IPLAC = IPLAC + 32
 9551 CONTINUE
 9552 CONTINUE
C
C     GET FIRST BYTE TO WRITE TOO.
C
      LOC1 = 0
      LOC2 = DHEDER2(5) + DHEDER(2) * IRCLN
C
C     LOOP THROUGH NUMBER OF INPUT FILES.
C
      DO 9899 IBLOOP = 1, INUM
```

```
C
C           CURRENT TO PROCESS.
C
            TEXT2 = FILS(IBLOOP)
            LT2   = INDEX ( TEXT2, '.' ) - 1
            VPATH2 = TEXT2(1:64)
            TEXT4  = TEXT2(1:LT2)
            ISTAT = VEC_SET ( FILIST2, VPATH2 )
            TEXT4(LT2+1:) = '.DBF'
C
C           OPEN OUTPUT DATA BASE FILE.
C
            CALL CONVCI ( TEXT4, PATH3, 80 )
            CALL AFILE2 ( HANDL4, 5, 0, 0, PATH3, ISTAT )
            IF ( ISTAT .LT. 0 ) GO TO 9899
C
C           GET HEADER BYTES.
C
            CALL AFILE2 ( HANDL4, 3, 8, 2, IHREC2, ISTAT )
            IF ( IHREC2 .GT. 32000 ) THEN
                WRITE( *, * )
                WRITE( *, * ) 'DATABASE HEADER TOO LARGE'
                WRITE( *, * )
                CALL BEEP
                GO TO 9999
            END IF
            IF ( IHREC2 .NE. IHREC ) THEN
                WRITE( *, * )
                WRITE( *, * ) 'DATABASE FILES ARE NOT COMPATIBLE'
                WRITE( *, * )
                CALL BEEP
                GO TO 9999
            END IF
            CALL AFILE2 ( HANDL4, 3, 10, 2, IRCLN2, ISTAT )
            IF ( IRCLN .NE. IRCLN2 ) THEN
                WRITE( *, * )
                WRITE( *, * ) 'DATABASE FILES ARE NOT COMPATIBLE'
                WRITE( *, * )
                CALL BEEP
                GO TO 9999
            END IF
C
C           READ IN HEADER.
C
            IRDB = IHREC2
            CALL AFILE2 ( HANDL4, 3, 0, IRDB, D2EDER, ISTAT )
C
C           LOOP THROUGH AND CHECK DATABASE COMPATIBILITY.
C
            ILOP = IHREC2 + 1
            DO 9555 ITIME = 33, ILOP
                IF ( DHEDER1(ITIME) .NE. D2EDER1(ITIME) ) GO TO 9556
```

```
 9555     CONTINUE
          GO TO 9557
 9556     WRITE( *, * ) 'DATABASE FILES ARE NOT COMPATIBLE'
          CALL BEEP
          GO TO 9999
 9557     CONTINUE
          IBYIN = IRCLN2
          IBYOT = IRCLN
          LOC4 = IHREC2
C
C         APPEND AGIS VERTEX FILES.
C
          CALL CONVCI ( FILIST(2), PATH2, 64 )
          CALL AFILE2 ( HANDVO, 5, 0, 0, PATH2, ISTAT  )
          CALL CONVCI ( FILIST2(2), PATH2, 64 )
          CALL AFILE2 ( HANDVT, 5, 0, 0, PATH2, ISTAT  )
          IF ( ISTAT .LT. 0 ) GO TO 9899
C
C         GET INFORMATION ON FILE TO COPY TO.
C
          CALL AFILE2 ( HANDVO, 3, 64, 4, ISTRG, ISTAT )
          CALL AFILE2 ( HANDVO, 3, 72, 4,  ILOC, ISTAT )
          CALL AFILE2 ( HANDVO, 3, ILOC, 4, ISTL, ISTAT )
          IWRT = ILOC + ISTL
          CALL AFILE2 ( HANDVO, 3, 96, 32, R8GEO(9), ISTAT )
C
C         GET INFORMATION ON FILE TO COPY FROM.
C
          CALL AFILE2 ( HANDVT, 3, 64, 4, IOSTR, ISTAT )
          CALL AFILE2 ( HANDVT, 3, 68, 4, LOCAT, ISTAT )
          IBODF = IBODF - 256
          CALL AFILE2 ( HANDVT, 3, 96, 32, R8GEO(13), ISTAT )
          IOFSET = IWRT - LOCAT
          ISTRG  = ISTRG + IOSTR
C
C         CHECK ON LIMITS.
C
          IF ( R8GEO( 9) .GT. R8GEO(13) ) R8GEO( 9) = R8GEO(13)
          IF ( R8GEO(10) .LT. R8GEO(14) ) R8GEO(10) = R8GEO(14)
          IF ( R8GEO(11) .GT. R8GEO(15) ) R8GEO(11) = R8GEO(15)
          IF ( R8GEO(12) .LT. R8GEO(16) ) R8GEO(12) = R8GEO(16)
          DO 1900 LOOP1 = 1, IOSTR
C
C             READ FROM OLD FILE.
C
              CALL AFILE2( HANDVT, 3, LOCAT, 4, ILEN, ISTAT )
              CALL AFILE2( HANDVT, 3, LOCAT, ILEN, R8BUF, ISTAT )
C
C             WRITE TO NEW FILE.
C
              CALL AFILE2( HANDVO, 4, IWRT, ILEN, R8BUF, ISTAT )
              LOCAT = LOCAT + ILEN
```

```
            IWRT  = IWRT  + ILEN
 1900     CONTINUE
C
C        WRITE OUT NEW INFORMATION TO COPY TO.
C
         CALL AFILE2 ( HANDVO, 4, 64, 4, ISTRG, ISTAT )
         ILOC = IWRT - ILEN
         IBDIF = IWRT
         CALL AFILE2 ( HANDVO, 4, 72, 4,  ILOC, ISTAT )
         CALL AFILE2 ( HANDVO, 4, 76, 4, IBDIF, ISTAT )
         CALL AFILE2 ( HANDVO, 4, 96, 32, R8GEO(9), ISTAT )
C
C        CLOSE BOTH VECTOR FILES.
C
         CALL AFILE2 ( HANDVO, 6, 0, 0, PATH2, ISTAT )
         CALL AFILE2 ( HANDVT, 6, 0, 0, PATH2, ISTAT )
C
C        OPEN AGIS VECTOR FILE.
C
         CALL CONVCI ( FILIST2(1), PATH2, 64 )
         CALL AFILE2 ( HANDVO, 5, 0, 0, PATH2, ISTAT )
         CALL CONVCI ( FILIST2(2), PATH2, 64 )
         CALL AFILE2 ( HANDVT, 5, 0, 0, PATH2, ISTAT )
         CALL AFILE2 ( HANDVO, 3, 64, 4, NUMOB, ISTAT )
         CALL AFILE2 ( HANDVO, 3, 68, 4, IFRBY, ISTAT )
         CALL AFILE2 ( HANDVO, 3, 96, 32, R8GEO(5), ISTAT )
C
C        LOOP THROUGH NUMBER OF OBJECTS.
C
         DO 1901 LOOP1 = 1, NUMOB
C
C            GET BYTES TO INCREMENT.
C
             ISTLOC = IFRBY + 4
             CALL AFILE2 ( HANDVO, 3, ISTLOC, 4, INCRE, ISTAT )
C
C            GET OBJECT TYPE FLAG.
C
             ISTLOC = IFRBY + 12
             CALL AFILE2 ( HANDVO, 3, ISTLOC, 4, IOBTY, ISTAT )
             IF ( IOBTY .LT. 0 ) GO TO 1901
C
C            GET NUMBER OF STRINGS.
C
             ISTLOC = IFRBY + 72
             CALL AFILE2 ( HANDVO, 3, ISTLOC, 4, INMSTR, ISTAT )
C
C            LOOP FOR NUMBER OF STRINGS.
C
             LOCSTP = ISTLOC + 4
             DO 1902 LOOP2 = 1, INMSTR
C
```

```
C               GET STRING POINTER AND LENGTH.
C
                    CALL AFILE2 ( HANDVO, 3, LOCSTP, 4, IPTR, ISTAT )
                    LOCSTP = LOCSTP + 4
                    CALL AFILE2 ( HANDVO, 3, LOCSTP, 4, ILEN, ISTAT )
                    LOCSTP = LOCSTP + 4
                    IPTR2 = ABS(IPTR)
C
C                   CHECK ON VALIDITY OF FLAG.
C
                    JPTR = IPTR2 +  4
                    CALL AFILE2 ( HANDVT, 3, JPTR, 4, IGOOD, ISTAT )
                    IF ( IGOOD .EQ. 0 ) GO TO 1902
C
C                   READ STRING FROM OLD VERTEX FILE.
C
                    IPTR2 = IPTR2 + 40
                                CALL AFILE2 ( HANDVT, 3, IPTR2, ILEN, R8BUF, ISTAT)
                    NUMV = ( ILEN / 8 ) / 2
                    ISTAT = IPTR2 + IOFSET - 40
                    IF ( IPTR .LT. 0 ) ISTAT = -ISTAT
                                ISTAT2 = VO_ADD( 1,FILIST(1),0,IOBTY,1,ISTAT, NUMV)
 1902           CONTINUE
                            ISTAT3 = VO_ADD( 4, FILIST(1), 0, IOBTY, 1, ISTAT,NUMV)
C
C               COLLECT DATABASE INFORMATION.
C
 9333           CALL AFILE2 ( HANDL4, 3, LOC4, IBYIN, RECORD, ISTAT )
                LOC4 = LOC4 + IBYIN
                IF ( RECORD(1:1) .EQ. '*' ) GO TO 9333
C
C               FILL OUT DATA BASE RECORD.
C
                WRITE( *, * ) 'PROCESSED OBJECT : ', LOOP1
                WRITE ( RECORD( IVLOC:IVLOC+15 ), '(I16)' ) INOB
                INOB = INOB + 1
                CALL AFILE2( HANDL3, 4, LOC2, IBYOT, RECORD, ISTAT )
                DHEDER(2) = DHEDER(2) + 1
                LOC2 = LOC2 + IBYOT
                IFRBY = IFRBY + INCRE
 1901       CONTINUE
C
C           CLOSE OLD FILE.
C
           CALL AFILE2 ( HANDVO, 6, 0, 0, PATH2, ISTAT )
           CALL AFILE2 ( HANDVT, 6, 0, 0, PATH2, ISTAT )
C
C          CLOSE DATA BASE FILE.
C
           CALL AFILE2 ( HANDL4, 6, 0, 0, PATH3, ISTAT )
 9899 CONTINUE
C
```

```
C     UPDATE DATA BASE INFORMATION.
C
      IRDB = IHREC
      CALL AFILE2 ( HANDL3, 4, 0, IRDB, DHEDER, ISTAT )
      DHEDER1(1) = 26
      LOC1 = IHREC + DHEDER(2) * IBYOT
      CALL AFILE2 ( HANDL3, 4, LOC1, 1, DHEDER1, ISTAT )
      ISTAT = VO_CALC ( FILIST, R8BUF, 16400 )
 9999 CONTINUE
      END
```

B.3.5 Listing of Program FIXDBF

```
      program FIXDBF
C
C     This program fixes the MAPIX Dbase files so they can be merged
C     with the files created from DSCMAPIX.
C
C     SUBROUTINES USED: UPRCSA.
C                       BEEP.
C                       CONVIC.
C                       AFILE2.
C                       DB_FIELD.
C                       DB_SET.
C                       DB_GET.
C                       DB_ADD.
C ******
C     LANGUAGE:    Microsoft Fortran Version 5.1.
C     DATE:        29 November 1994.
C     PROJECT:     ESPOL - UNO.
C ******
      REAL*8     DHEDER8(1)
      REAL*8     R8GEO(32),  XMIN, XMAX, YMIN, YMAX,
     *           R8BUF(2050)
      REAL*8     ISTX, ISTY, STRTX, STRTY
      REAL*4     DHEDER4(1)
      INTEGER*4  VEC_FILE,   VEC_SET,  I4GEO(64), DB_ADD, DB_FIELD
      INTEGER*4  DB_SET,     DB_GET
      INTEGER*4  DHEDER(1284), PATH(20), VO_CALC, PATH2(20)
      INTEGER*4  CODIGO,   TRACT,    TIPOLN,    LNORTE,
     *           TIPOLE,    LESTE,   TIPOLS,    LSUR,
     *           TIPOLO,    LOESTE,  GRADDEF,   USO,
     *           SECTOR,    FOTO,   SISTEMA,   PARROQUIA,
     *           AGIS_00_,   AGIS_00_X,   AGIS_00_A,
     *           AGIS_00_O,   AGIS_00_I
      INTEGER*4  BODIGO,   BRACT,    BIPOLN,    BNORTE,
     *           BIPOLE,    BESTE,   BIPOLS,    BSUR,
     *           BIPOLO,    BOESTE,  BRADDEF,   BSO,
     *           BECTOR,    BOTO,   BISTEMA,   BARROQUIA,
     *           BGIS_00_,   BGIS_00_X,   BGIS_00_A,
```

```
*           BGIS_00_O,   BGIS_00_I
      INTEGER*2  DHEDER2(1), HANDL3, DATE(3), HANDL1
      INTEGER*1  DHEDER1(1)
      CHARACTER*704 DHEDERC
      CHARACTER*321 RECORD
      CHARACTER*64  VPATH,   FILIST(3)
      CHARACTER*80  TEXT, TEXT2, TEXT3
      CHARACTER*30  NAME
      EQUIVALENCE  ( R8GEO,   I4GEO )
      EQUIVALENCE  ( DHEDER,    DHEDER1 )
      EQUIVALENCE  ( DHEDER,    DHEDER2 )
      EQUIVALENCE  ( DHEDER,    DHEDER4 )
      EQUIVALENCE  ( DHEDER,    DHEDER8 )
      EQUIVALENCE  ( DHEDER,    DHEDERC )
      WRITE (*, 12)
 12   FORMAT (////////33X, 'FIXDBF'//27X, 'A PC  Utility',
     * /21X,  'for fixing AGIS database files to be'
     * /21x,  'merged with files created by DSCMAPI1'
     * //28X, 'Proyecto ESPOL - UNO'////////)
      PAUSE
C
C     RETRIEVE INPUT FILE NAME.
C
    1 WRITE( *, * ) 'ENTER THE DATABASE FILE TO FIX : '
      READ( *, '(A80)' ) TEXT2
      CALL UPRCSA ( TEXT2, 80 )
      LT    = INDEX ( TEXT2, '.' )
      IF ( LT .EQ. 0 ) THEN
          LT = INDEX ( TEXT2, ' ' )
          TEXT2(LT:) = '.DBF'
      ELSE
          IF ( TEXT2(LT+1:LT+3) .NE. 'DBF' ) THEN
              WRITE( *, * )
              WRITE( *, * ) 'INVALID FILE NAME'
              WRITE( *, * )
              WRITE( *, * )
              CALL BEEP
              GO TO 1
          END IF
      END IF
C
C     RETRIEVE OUTPUT FILE NAME.
C
    2 WRITE( *, * ) 'ENTER THE NAME OF THE FIXED FILE TO CREATE : '
      READ( *, '(A80)' ) TEXT
      CALL UPRCSA ( TEXT, 80 )
      LT    = INDEX ( TEXT, '.' )
      IF ( LT .EQ. 0 ) THEN
          LT = INDEX ( TEXT, ' ' )
          TEXT(LT:) = '.DBF'
      ELSE
          IF ( TEXT(LT+1:LT+3) .NE. 'DBF' ) THEN
```

```
                WRITE( *, * )
                WRITE( *, * ) 'INVALID FILE NAME'
                WRITE( *, * )
                WRITE( *, * )
                CALL BEEP
                GO TO 2
            END IF
        END IF
C
C       OPEN EXISTING DATABASE.
C
        CALL CONVCI ( TEXT2, PATH2, 80 )
        CALL AFILE2 ( HANDL1, 5, 0, 0, PATH2, ISTAT )
C
C       GET NUMBER OF RECORDS.
C
        CALL AFILE2 ( HANDL1, 3, 4, 4, NUMREC, ISTAT )
C
C       CREATE DATABASE FILE.
C
        CALL CONVCI ( TEXT, PATH, 80 )
        CALL AFILE2 ( HANDL3, 11, 0, 0, PATH, ISTAT )
        CALL GETDAT ( DATE(1), DATE(2), DATE(3) )
        DO 102 LOOP1 = 1, 1284
            DHEDER(LOOP1) = 0
  102   CONTINUE
        DHEDER1(1) = 3
        DHEDER1(2) = MOD ( DATE(1), 100 )
        DHEDER1(3) = DATE(2)
        DHEDER1(4) = DATE(3)
        DHEDER (2) = 0
        DHEDER2(5) = 705
        DHEDER2(6) = 321
C
C       START OF THE USER DEFINED FIELDS.
C       NOTE THAT THE INDEX WIDTH IS THE NUMBER OF CHARACTERS IN THE
C       FIELD NAME,  AND THAT THE FIRST FIELD START IN THE POSITION 33.
C       FOR EVERY FIELD ADDED,  YOU HAVE TO ADD 32 TO THE START OF THE
C       INDEX.
C       DO NOT FORGET TO PUT THE AGIS FIELDS [5] AT THE END.
C
        DHEDERC(  33:38  ) = 'CODIGO'
        DHEDERC(  65:69  ) = 'TRACT'
        DHEDERC(  97:103 ) = 'TIPO_LN'
        DHEDERC( 129:135 ) = 'L_NORTE'
        DHEDERC( 161:167 ) = 'TIPO_LE'
        DHEDERC( 193:198 ) = 'L_ESTE'
        DHEDERC( 225:231 ) = 'TIPO_LS'
        DHEDERC( 257:261 ) = 'L_SUR'
        DHEDERC( 289:295 ) = 'TIPO_LO'
        DHEDERC( 321:327 ) = 'L_OESTE'
        DHEDERC( 353:360 ) = 'GRAD_DEF'
```

```
      DHEDERC( 385:387 ) = 'USO'
      DHEDERC( 417:422 ) = 'SECTOR'
      DHEDERC( 449:452 ) = 'FOTO'
      DHEDERC( 481:487 ) = 'SISTEMA'
      DHEDERC( 513:521 ) = 'PARROQUIA'
C
C     THIS FIELD IS GOING TO BE FILLED UP WITH THE OBJECT NUMBER
C     ( THE LINE BELOW ).
C
      DHEDERC( 545:553 ) = '_AGIS_00_'
      DHEDERC( 577:586 ) = '_AGIS_00_X'
      DHEDERC( 609:617 ) = 'AGIS_00_A'
      DHEDERC( 641:649 ) = 'AGIS_00_O'
      DHEDERC( 673:681 ) = 'AGIS_00_I'
C
C     ADDITIONAL FIELDS CAN BE ADDED HERE.
C
      DHEDERC(  44:44  ) = 'C'
      DHEDERC(  76:76  ) = 'C'
      DHEDERC( 108:108 ) = 'C'
      DHEDERC( 140:140 ) = 'C'
      DHEDERC( 172:172 ) = 'C'
      DHEDERC( 204:204 ) = 'C'
      DHEDERC( 236:236 ) = 'C'
      DHEDERC( 268:268 ) = 'C'
      DHEDERC( 300:300 ) = 'C'
      DHEDERC( 332:332 ) = 'C'
      DHEDERC( 364:364 ) = 'C'
      DHEDERC( 396:396 ) = 'C'
      DHEDERC( 428:428 ) = 'C'
      DHEDERC( 460:460 ) = 'N'
      DHEDERC( 492:492 ) = 'C'
      DHEDERC( 524:524 ) = 'C'
      DHEDERC( 556:556 ) = 'C'
      DHEDERC( 588:588 ) = 'C'
      DHEDERC( 620:620 ) = 'N'
      DHEDERC( 652:652 ) = 'N'
      DHEDERC( 684:684 ) = 'N'
C
      DHEDER1(  49 ) =  8
      DHEDER1(  81 ) =  3
      DHEDER1( 113 ) =  3
      DHEDER1( 145 ) = 35
      DHEDER1( 177 ) =  3
      DHEDER1( 209 ) = 35
      DHEDER1( 241 ) =  3
      DHEDER1( 273 ) = 35
      DHEDER1( 305 ) =  3
      DHEDER1( 337 ) = 35
      DHEDER1( 369 ) =  1
      DHEDER1( 401 ) =  5
      DHEDER1( 433 ) = 40
```

```
      DHEDER1( 465 ) =  4
      DHEDER1( 497 ) =  3
      DHEDER1( 529 ) = 30
      DHEDER1( 561 ) = 16
      DHEDER1( 593 ) =  4
      DHEDER1( 625 ) = 18
      DHEDER1( 657 ) = 18
      DHEDER1( 689 ) = 18
C
      DHEDER1( 50) = 0
      DHEDER1( 82) = 0
      DHEDER1(114) = 0
      DHEDER1(146) = 0
      DHEDER1(178) = 0
      DHEDER1(210) = 0
      DHEDER1(242) = 0
      DHEDER1(274) = 0
      DHEDER1(306) = 0
      DHEDER1(338) = 0
      DHEDER1(370) = 0
      DHEDER1(402) = 0
      DHEDER1(434) = 0
      DHEDER1(466) = 0
      DHEDER1(498) = 0
      DHEDER1(530) = 0
      DHEDER1(562) = 0
      DHEDER1(594) = 0
      DHEDER1(626) = 2
      DHEDER1(658) = 2
      DHEDER1(690) = 2
C
      DHEDER1(705) = 13
C
C     WRITE OUT BACK HEADER RECORD.
C
      CALL AFILE2 ( HANDL3, 4, 0, 705, DHEDER, ISTAT )
      LOC9 = 0
      LOC8 = 705
C
C     INPUT FILE FIELD LOCATIONS.
C
      NAME = 'CODIGO'
      CODIGO    = DB_FIELD( HANDL1, NAME )
      NAME = 'TRACT'
      TRACT     = DB_FIELD( HANDL1, NAME )
      NAME = 'TIPO_LN'
      TIPOLN    = DB_FIELD( HANDL1, NAME )
      NAME = 'L_NORTE'
      LNORTE    = DB_FIELD( HANDL1, NAME )
      NAME = 'TIPO_LE'
      TIPOLE    = DB_FIELD( HANDL1, NAME )
      NAME = 'L_ESTE'
```

```
      LESTE     = DB_FIELD( HANDL1, NAME )
      NAME = 'TIPO_LS'
      TIPOLS    = DB_FIELD( HANDL1, NAME )
      NAME = 'L_SUR'
      LSUR      = DB_FIELD( HANDL1, NAME )
      NAME = 'TIPO_LO'
      TIPOLO    = DB_FIELD( HANDL1, NAME )
      NAME = 'L_OESTE'
      LOESTE    = DB_FIELD( HANDL1, NAME )
      NAME = 'GRAD_DEF'
      GRADDEF   = DB_FIELD( HANDL1, NAME )
      IF ( GRADDEF .LT. 0 ) THEN
         NAME = 'GRAD_DEFLD'
         GRADDEF = DB_FIELD ( HANDL1, NAME )
      END IF
      NAME = 'USO'
      USO       = DB_FIELD( HANDL1, NAME )
      NAME = 'SECTOR'
      SECTOR    = DB_FIELD( HANDL1, NAME )
      NAME = 'FOTO'
      FOTO      = DB_FIELD( HANDL1, NAME )
      NAME = 'SISTEMA'
      SISTEMA   = DB_FIELD( HANDL1, NAME )
      NAME = 'PARROQUIA'
      PARROQUIA = DB_FIELD( HANDL1, NAME )
      NAME = '_AGIS_00_'
      AGIS_00_  = DB_FIELD( HANDL1, NAME )
      NAME = '_AGIS_00_X'
      AGIS_00_X = DB_FIELD( HANDL1, NAME )
      NAME = 'AGIS_00_A'
      AGIS_00_A = DB_FIELD( HANDL1, NAME )
      NAME = 'AGIS_00_O'
      AGIS_00_O = DB_FIELD( HANDL1, NAME )
      NAME = 'AGIS_00_I'
      AGIS_00_I = DB_FIELD( HANDL1, NAME )
C
C     OUTPUT FILE FIELD LOCATIONS.
C
      NAME = 'CODIGO'
      BODIGO    = DB_FIELD( HANDL3, NAME )
      NAME = 'TRACT'
      BRACT     = DB_FIELD( HANDL3, NAME )
      NAME = 'TIPO_LN'
      BIPOLN    = DB_FIELD( HANDL3, NAME )
      NAME = 'L_NORTE'
      BNORTE    = DB_FIELD( HANDL3, NAME )
      NAME = 'TIPO_LE'
      BIPOLE    = DB_FIELD( HANDL3, NAME )
      NAME = 'L_ESTE'
      BESTE     = DB_FIELD( HANDL3, NAME )
      NAME = 'TIPO_LS'
      BIPOLS    = DB_FIELD( HANDL3, NAME )
```

```
      NAME = 'L_SUR'
      BSUR       = DB_FIELD( HANDL3, NAME )
      NAME = 'TIPO_LO'
      BIPOLO     = DB_FIELD( HANDL3, NAME )
      NAME = 'L_OESTE'
      BOESTE     = DB_FIELD( HANDL3, NAME )
      NAME = 'GRAD_DEF'
      BRADDEF    = DB_FIELD( HANDL3, NAME )
      IF ( GRADDEF .LT. 0 ) THEN
         NAME = 'GRAD_DEFLD'
         BRADDEF = DB_FIELD ( HANDL3, NAME )
      END IF
      NAME = 'USO'
      BSO        = DB_FIELD( HANDL3, NAME )
      NAME = 'SECTOR'
      BECTOR     = DB_FIELD( HANDL3, NAME )
      NAME = 'FOTO'
      BOTO       = DB_FIELD( HANDL3, NAME )
      NAME = 'SISTEMA'
      BISTEMA    = DB_FIELD( HANDL3, NAME )
      NAME = 'PARROQUIA'
      BARROQUIA = DB_FIELD( HANDL3, NAME )
      NAME = '_AGIS_00_'
      BGIS_00_   = DB_FIELD( HANDL3, NAME )
      NAME = '_AGIS_00_X'
      BGIS_00_X = DB_FIELD( HANDL3, NAME )
      NAME = 'AGIS_00_A'
      BGIS_00_A = DB_FIELD( HANDL3, NAME )
      NAME = 'AGIS_00_O'
      BGIS_00_O = DB_FIELD( HANDL3, NAME )
      NAME = 'AGIS_00_I'
      BGIS_00_I = DB_FIELD( HANDL3, NAME )
C
C     LOOP FOR NUMBER OF RECORDS IN DATA BASE.
C
      DO 10 I = 1, NUMREC
         RECORD = ' '
         WRITE( *, * ) 'FIXING RECORD : ', I
         IREC   = DB_ADD ( HANDL3, RECORD )
         RECORD = ' '
         IST    = DB_GET ( RECORD, HANDL1, IREC, CODIGO )
         IST    = DB_SET ( HANDL3, IREC, BODIGO, RECORD )
         RECORD = ' '
         IST    = DB_GET ( RECORD, HANDL1, IREC, TRACT )
         IST    = DB_SET ( HANDL3, IREC, BRACT, RECORD )
         RECORD = ' '
         IST    = DB_GET ( RECORD, HANDL1, IREC, TIPOLN )
         IST    = DB_SET ( HANDL3, IREC, BIPOLN, RECORD )
         RECORD = ' '
         IST    = DB_GET ( RECORD, HANDL1, IREC, LNORTE )
         IST    = DB_SET ( HANDL3, IREC, BNORTE, RECORD )
         RECORD = ' '
```

```
            IST     = DB_GET ( RECORD, HANDL1, IREC, TIPOLE )
            IST     = DB_SET ( HANDL3, IREC, BIPOLE, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, LESTE )
            IST     = DB_SET ( HANDL3, IREC, BESTE, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, TIPOLS )
            IST     = DB_SET ( HANDL3, IREC, BIPOLS, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, LSUR )
            IST     = DB_SET ( HANDL3, IREC, BSUR, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, TIPOLO )
            IST     = DB_SET ( HANDL3, IREC, BIPOLO, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, LOESTE )
            IST     = DB_SET ( HANDL3, IREC, BOESTE, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, GRADDEF )
            IST     = DB_SET ( HANDL3, IREC, BRADDEF, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, USO )
            IST     = DB_SET ( HANDL3, IREC, BSO, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, SECTOR )
            IST     = DB_SET ( HANDL3, IREC, BECTOR, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, FOTO )
            IST     = DB_SET ( HANDL3, IREC, BOTO, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, SISTEMA )
            IST     = DB_SET ( HANDL3, IREC, BISTEMA, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, PARROQUIA )
            IST     = DB_SET ( HANDL3, IREC, BARROQUIA, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, AGIS_00_ )
            IST     = DB_SET ( HANDL3, IREC, BGIS_00_, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, AGIS_00_X )
            IST     = DB_SET ( HANDL3, IREC, BGIS_00_X, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, AGIS_00_A )
            IST     = DB_SET ( HANDL3, IREC, BGIS_00_A, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, AGIS_00_O )
            IST     = DB_SET ( HANDL3, IREC, BGIS_00_O, RECORD )
            RECORD = ' '
            IST     = DB_GET ( RECORD, HANDL1, IREC, AGIS_00_I )
            IST     = DB_SET ( HANDL3, IREC, BGIS_00_I, RECORD )
     10 CONTINUE
C
```

```
C       CLOSE DATABASE FILES.
C
        CALL AFILE2 ( HANDL3, 6, 0, 0, PATH, ISTAT )
        CALL AFILE2 ( HANDL1, 6, 0, 0, PATH, ISTAT )
        END
```