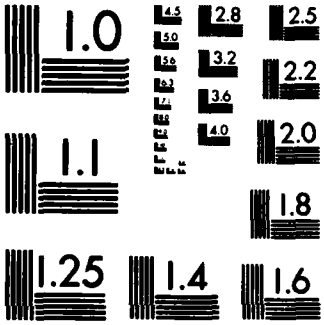
AD-A132 940 CORDIVEM EUROPEAN TERRAIN DATA(U) COMBINED ARMS OPERATIONS RESEARCH ACTIVITY FORT LEAVENWORTH KS W J THOMSON 1983 CAORA/TP-2-83 1/3

UNCLASSIFIED F/G 15/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A132940

ACN 23440

CORDIVEM EUROPEAN TERRAIN DATA

Technical Paper 2-83

UNITED STATES ARMY COMBINED ARMS CENTER

Copy available to DTIC does not permit fully legible reproduction

COMBINED ARMS OPERATIONS RESEARCH ACTIVITY

This document has been approved for public release and sale; its distribution is unlimited.

DTIC ELECTE SEP 27 1983 S A

83-5379

83 09 26 135

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

UNCLASSIFIED

Technical Paper 2-83

Combined Arms Operations Research Activity
Fort Leavenworth, Kansas 66027

CORDIVEM EUROPEAN TERRAIN DATA

by

WALTER J. THOMSON

ACN 23440

Approved by:

DAVID J. FARMER
Director, TSSD

JOHN L. BALLANTYNE
Brigadier General, USA
Commanding

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A132940	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CORDIVEM European Terrain Data		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Walter J. Thomson, Jr.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Combined Arms Operations Research Activity Fort Leavenworth, Kansas 66027		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQ, US Army Training and Doctrine Command ATTN: ATCD-AN Fort Monroe, Virginia 23651		12. REPORT DATE
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Terrain, CORDIVEM, Europe, Corps, Division, Wargame, Simulation, Modelling, Elevation, Data, Hydrography, LOC, Surface Features

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document describes the data sources and the methods used to develop a corps-sized European terrain data base for the Corps/Division Evaluation Model.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CORDIVEM European Terrain Data

DTIC
COPY
INSPECTED

Accession For

A-23

Acknowledgements

I wish to thank all of the personnel in the System Support Branch of CAORA for their technical assistance, my supervisor, Howard Haeker for his sound advice, Barbara Mabry for typing and editorial help, and Roger Lindley for sharing his knowledge of graphics programming.

ABSTRACT

The CORDIVEM combat simulation model requires large amounts of terrain data for successful modelling of corps and division level processes. As of March 1982 the terrain data was limited to the usual Fulda Gap area of the Federal Republic of Germany (FRG). This document describes the data sources and the methods used to develop a corps-sized European terrain data base for the Corps - Division Evaluation Model.

TABLE OF CONTENTS

CHAPTER	Page
Acknowledgements	i
Abstract	ii
Table of Contents	iii
Main Report	1
1. Background	1
a. CORDIVEM requirements	1
(1) Operational data	1
(2) Display data	1
b. LOC/Terrain data base	1
(1) Background	
(a) Data sources	1
(b) Bundespost data	2
(2) Areal	
(3) Linear	2
(4) Final product	2
2. Method	2
a. Preprocessing	2
(1) Inspection	2
(2) Editing	2
(3) Digitization	2
(4) Additions	2
b. Areal data aggregation	3
c. Linear data aggregation	3
(1) LOC - hydrography differences	3
(2) LOC	3
(3) Hydrography	3
(4) Postprocessing	3
3. Results	3
a. Extent	3
b. Format	3
(1) Hex	3
(2) Display	3

Appendices

A. Data Structures and Access Methods	A-1
B. Software	B-1
C. Hex data in ICOR	C-1
D. Distribution List	D-1

MAIN REPORT

1. Background.

a. CORDIVEM requirements.

(1) Operational data. When purchased from BDM, the model ran - and still runs - on hexagonal cells (hexes) of terrain data. The format of this hex data has not been changed from that of ICOR. Each hex has a 3.57 km inner diameter and contains stylized information on terrain roughness, cover and lines-of-communication (LOC, roads) and hydrography (rivers). (See appendix A.) This original hex data covers a roughly oval-shaped area centered near Fulda with a radius of 50 to 120 km. BDM Services Corporation (the developer of ICOR) produced this hex data by a visual analysis of 1:50,000 scale maps.

(2) Display data. Part of the evolution of CORDIVEM from ICOR has been the grafting of display data and software from the Corps Battle Game (CGB) effort onto the model. The display data currently being used for demonstration is the result of patching and reformatting of some special high-resolution digital products prepared by the Defense Mapping Agency (DMA). The file represents an area extending 200 km in the east-west axis and 150 km in the north-south axis centered at 32UPB00. In the display file the data is at 100m resolution with each point having a surface feature code (open, forest, urban) and an elevation value. In addition to this areal data there is a small network of roads and rivers which was digitized by CASAA personnel (now part of CAORA) using software delivered to BDM for use in their LOC/terrain analysis effort.

b. LOC/Terrain database.

(1) Background:

(a) Data sources. In December of 1979 CASAA (now part of CAORA) issued a contract to BDM Services Corporation to produce a terrain data base for the Corps Battle Game, a predecessor of CORDIVEM. Under the terms of this contract BDM was to produce digitized terrain data, loc and hydrography nets for virtually all of Germany. The form of this data was to match the previously-mentioned display data. It was to be developed using DMA elevation data, using CASAA-supplied maps, software and hardware, and the "Bundespost" data.

(b) Bundespost data. The Bundespost data was produced by the post office (Bundespost) of the Federal Republic of Germany. It was provided to Electromagnetic Compatibility Analysis Center (ECAC) in 1976. ECAC reformatted this data and forwarded a copy to TRASANA in 1979. This data contained 100m surface feature codes and elevation data for all of FRG and was the only large-scale terrain database available for Germany. Unfortunately, the data contained many inconsistencies.

(2) Areal. As part of BDM's LOC/Terrain contract the Bundespost data was obtained from TRASANA. This data was used to represent FRG while DMA elevation data was used for German Democratic Republic (East Germany) with no surface feature data.

(3) Linear. The principal effort was an analysis and digitization of the loc and hydrography network for virtually all of Germany - an area represented by about 600 1:50,000 scale maps.

(4) Final product. The final result of this effort was a database extending roughly 400 km east-to-west and 600 km north-to-south to be used as the terrain database for the CBG. The shortcomings of the areal data were recognized and it was originally intended to serve as an interim resource. Nevertheless, this is the source used in our production of operational and display data for CORDIVEM.

2. Method.

a. Preprocessing.

(1) Inspection. After reinspecting the areal data it was determined, since CORDIVEM operates on the 3.57 km hex cell, that the elevation data was accurate enough for CORDIVEM hex data use and that the surface data could be graphically emended.

(2) Editing. The objective of editing the feature data was to ensure that the areas of urban and forest code were in approximately the right location and size for incorporation into the hexes; no other factors were considered. The editing was accomplished by displaying a 20- or 40-km square (operator's choice) on a Tektronix 4027 terminal, comparing the display to a 1:50,000 scale map and making changes to the data with a "rectangular cookie-cutter" routine.

(3) Digitization. To fill part of the GDR void in the data base, the urban and forested areas from 74 1:50,000 scale maps were digitized on a Tektronix 4083 system. This data was then transferred to the VAX, reformatted and packed into the terrain data files. This data was then edited in the same fashion as the Bundespost data.

(4) Additions. The only change to the linear data was to insert end-nodes into the link records. (A link is a section of road or river and a node is an end point of this section; the original links only contained a start node with no stop.)

b. Areal data aggregation. Each hex within the area was approximated by a circle with a radius of 2,000m (the 3.57 km hex has an inner radius of 1785m on an outer radius of 2061m). Tallies were then made of the surface feature codes of the data points within this circle, and average elevation and average absolute slope were then calculated. The hex address, UTM coordinates, average elevation, percent slope and percent open, urban and forest cover were written to a file for further processing.

c. Linear data aggregation.

(1) LOC-hydrography differences. While the loc and hydrography nets are similar, they differ in function. The same is true of their more abstract representation in the hex data: the loc provides, and the hydrography impedes access from one area to another. Thus, slightly different algorithms were used for aggregating each type of data into the hex format.

(2) LOC. The road net was processed by link records. If the terminal nodes of a link were in adjacent hexes, then this connectivity was recorded in the common side of both hexes and the next link was accessed. If not, then the first subnode (essentially a subnode is a curve in the link) was accessed. If the initial node and this subnode were in adjacent hexes, then this was recorded as a hex connectivity; the subnode was then regarded as the initial node whether in the same or adjacent hex and the process iterated to completion. (The case where the subnode was in a nonadjacent hex indicated an error in the source data.)

(3) Hydrography. In converting the hydrography to hex data, each link was inspected subnode-by-subnode so that it could be approximated by hex sides.

(4) Postprocessing. In both cases the source network and the resulting hex data were graphically displayed on a Tektronix 4027 terminal for interactive correction. A subjective judgment is that less than 5 percent of the hex data was changed, so that even without editing the data would probably have been acceptable.

3. Results.

a. Extent. The area represented by the resulting data base is approximately square, it is centered at 32UNC00 and extends 400 km along each axis.

b. Format.

(1) Hex. The hex data is in a file indexed by hex address and UTM coordinates of the hex center; a user can extract a desired rectangular portion in the CORDIVEM input format by running the provided routine 'HEXOUT.'

(2) Display. Each 10 km square of areal data is stored in a file indexed by the UTM coordinate of its southwest corner (i.e., 32UNC45). The linear data is as provided by BDM except that the end node coordinates have been substituted for the seldom-used route numbers (the original data is still available).

APPENDIX A

DATA STRUCTURES AND ACCESS METHODS

APPENDIX A. Data Structure and Access Methods

1. Hex Data.

a. Location. The hex data is stored in the file HEXTERR.DAT in the save-set HEX.BCK on tape #817 in the CGF tape library.

b. File structure. HEXTERR.DAT is an indexed sequential file. The primary index being the hex address with two secondary indices - the UTM coordinates of the hex center.

c. Record structure. Each record contains 7 I*4 words as follows: hex address, easting and northing of the hexcenter (UTM), average elevation, connectivity codes, river codes and surface feature codes. The hex address is the external form; the easting, the northing and the average elevation are in meters; the connectivity and river codes are packed in descending order of sides (i.e., from side 6 to side 1); and the surface feature codes are packed in the order: urbanization, forestation, roughness.

d. Access methods.

(1) A CORDIVEM input file for a selected rectangular area can be obtained by running the routine HEXOUT, which is also in HEX.BCK.

(2) To access the file HEXTERR.DAT from a user-written routine, the file should be opened with:

```
STATUS = 'OLD'
ORG    = 'INDEXED'
ACCESS = 'KEYED'
RECL   = 7
RECORDTYPE = 'FIXED'
FORM = 'UNFORMATTED'
KEY = (1:4:INTEGER, 5:8:INTEGER,9:12:INTEGER)
SHARED, READONLY
```

Alternately, the programer can include 'HEXDAT.TYP' and 'HEXTERR.OPN' from the terrain text library on DBO: of the CGF Vax.

2. Areal data.

a. Location. Each 10 km square of surface/elevation data occupies one file in sav-set 32UDAT.BCK on tape 818 in the CGF library.

b. File structure. Each file contains one 20,000-byte record.

c. Record structure. Each record contains 10,000 I*2 words representing the 100m grid for the given 10 X 10 km area, written columnwise. The data for each point is packed as elevation* 8 + feature code.

d. Access methods. The required open statement is "OPEN (UNIT=LU,FILE=fname)" where "LU" and "fname" are the logical unit and name of the file. The required read statement is just "READ(LU) A" where A is a 100 X 100 I*2 array. If the data is placed into a 400 X 400 array (to represent a 40 km square) then the functions ICODE and IELV from the TERRAIN library may be used.

3. Linear data.

a. Location. Copies of the linear databases produced by BDM are on tape #820 in save-set VECTOR. BCK in the CGF tape library.

b. File and record structure. The following description was obtained from BDM as a clarification of their documentation. The only change here from the original documentation is that word 4 of the link record contains the y, x, coordinates of the terminus of that link.

Supplementary Notes on the Loc Data Base

These notes are intended to clarify and expand upon the technical description of the data base which was previously delivered. Questions concerning physical and logical record sizes and record structures have been specifically addressed along with a few other items which were thought to be of interest.

DATA BASE COORDINATES

Throughout the Loc-Terrain data base the rectangular coordinates of points are specified in 20 meter units offset from an easting of 9° (500,000) (the central meridian of UTM zone 32) and a northing of 5,600,000 meters (relative to the equator). This implies that MGR coordinates are easily converted to data base coordinates by a simple offset and division by 20, provided that the point in question is within UTM zone 32. For points outside of this zone, the procedure used in generating the data was to translate the given UTM designation to GEOREF (Lat/long) and then convert back to UTM coordinates relative to zone 32. The routines necessary to accomplish this are included with the software delivered with the data base.

GRID INDICES

GRID (65,65) is the 10-km grid whose southwest corner lies at the origin of the data base (32UNB00). Thus if (x,y) is the data base representation of a point, then the formula I = (X+32500)/500 and J=(Y+325000)/500 (using integer division) provide the appropriate index GRID (I,J) with which to reference the data.

TYPE CODES

Data for type of node and type of link were encoded as follows:

NODES
1 - intersection of autobahns
2 - built-up area
3 - airfield
4 - open area

LINKS
1 - autobahn
2 - main road
3 - secondary road
4 - fair weather road
5 - rail line
6 - ferry
7 - ford
8 - heavy bridge
9 - dam
10 - road tunnel
11 - rail tunnel
12 - major river
13 - minor river
14 - stream

LOC DATA FILE SPECIFICATIONS

The eight LOC files have the same basic organization. They are each composed of logical records which are 500 I*4 words in length. Furthermore, each is structured as a two-dimensional array.

GRID FILES: Contain pointers to the node files

LENGTH - NEWG.DAT: 33 logical records (16500 words)
NEWHG.DAT: 33 logical records (16500 words)
ARRAY STRUCTURE - 128 X 128
POINTERS - Each entry is either 0 (indicating no data) or else it contains a pointer to the first node in the specified grid.

NODE FILES: Contain linked lists of nodes belonging to a given grid.

LENGTH - NEWN.DAT: 1845 logical records (922500 words)
NEWHN.DAT: 35 logical records (17500 words)
ARRAY STRUCTURE - 5 X N (N=100=(number of logical records))
Entry (I,J) in this array contains information about node J.
(I,J) = pointer to the next node in this linked list
(2,J) = code for this type of node
(3,J) = pointed to first link terminating at this node
(4,J) = pointer to first link originating at this node.
(5,J) = X, Y coordinates of node location (X is in the lower half word)

POINTERS - A zero entry indicates the end of a linked list

LINK FILES: Contain linked lists of inlinks and outlinks for a given node

LENGTH - NEWL.DAT:2698 logical records (1349000 words)
NEWHL.DAT:35 logical records (17500 words)
ARRAY STRUCTURE - 5 X N (N = 100*(number of logical records))
Entry (I,J) in this array contains information about link J.
(1,J) = pointer to next inlink in this linked list
(2,J) = pointer to next outlink in this linked list
(3,J) = length of this link (in 20 meter units)
(4,J) = coordinates of end node
(5,J) = pointer to list of subnodes which describe this link
POINTER - A zero entry indicates the end of a linked list except for (5,J) in which case it would imply that no subnode list is associated with this link.

SUBNODE FILES: Contain a concatenation of the subnode lists associated with links in the link files.

LENGTH - SUB.DAT:3977 logical records (1988500 words)
HSUB.DAT:299 logical records (149500 words)

ARRAY STRUCTURE 500 X N (N = number of logical records)

The array structure imposed upon this file has no direct relevance to the data stored within it. The primary function is to allow the data to be accessed by the same mechanism used by the other files. The pointers to this file from the link file are separated into a record pointer (lower half word) and a word-within-the-record pointer (upper half-word).

SUBNODE LIST STRUCTURE - The first word of a subnode list contains twice the number of subnodes in this list. If this number is N, the next N2 words contain X,Y coordinates which describe the associated link. Note that X is contained in the lower half-word.

c. Access methods.

(1) The user may use routines supplied by BDM with the LOC/terrain documentation or he may use the routines GRIDS, IGRID, GETNDS, and GETREC from the Terrain library.

(2) For accessing this data from a lower level the user is referred to BDM LOC/Terrain documentation.

APPENDIX B

SOFTWARE

APPENDIX B

TABLE OF CONTENTS

	Page
BDMEXT Calling Sequence (Fig. B-1)	B-1
Program: BDMEXT (Fig. B-2)	B-2
DISPLAY Calling Sequence (Fig B-3)	B-11
Program: DISPLAY (Fig B-4)	B-12
DRWHEX Calling Sequence (Fig. B-5)	B-18
Program: DRWHEX (Fig. B-6)	B-19
HEXER Calling Sequence (Fig. B-7)	B-43
Program: HEXER (Fig. B-8)	B-44
HYDROHEXER Calling Sequence (Fig. B-9)	B-55
Program: HYDROHEXER (Fig. B-10)	B-56
PDBPACK Calling Sequence (Fig. B-11)	B-65
Program: PDBPACK (Fig. B-12)	B-66
PDBREAD Calling Sequence (Fig. B-13)	B-71
Program: PDBREAD (Fig. B-14)	B-72
ROADHEXER Calling Sequence (Fig. B-15)	B-86
Program: ROADHEXER	B-87
Terrain Utilities	B-96

BDMEXT CALLING SEQUENCE

```
BDMEXT
 |
 |---INFO
 |
 |---GRIDR
 |
 |---MOVEREC
     |
     |---IGRID
     |
     |---FIX
         |
         |---RECODE
         |
         |---TRANSPOSE
```

Figure B-1

PROGRAM: BDMEXT

ROUTINE \ COMMON	GRAPH	GRID																
BDMEXT		X																
GRIDR		X																
INFO	X																	
MOVEREC	X	X																

Figure B-2

```
      PROGRAM BDMEXT
*************************************************************
*     THIS PROGRAM WAS ADAPTED FROM A ROUTINE WHICH         *
*     ORIGINALLY DISPLAYED THE BDM LOC DATA.                *
*     THIS VERSION TRANSFERS TERRAIN DATA FROM              *
*     THE FILE TERUTM TO 10 KM FILES.                       *
*************************************************************
      INTEGER*4 I,J,X,Y
      INCLUDE 'GRID.CMN'
1 ***********************************************
1 *     GRID CONTAINS THE POINTERS TO THE 10-KM *
1 *     TERRAIN RECORDS IN THE FILE TERUTM.     *
1     INTEGER*2 GRID(128,128)
1     COMMON/GRID/GRID
1 ***********************************************
      CALL INFO(LUIN,LUOUT)
*     READ IN GRID FILE
      CALL GRIDR(LUIN)
*     TRANSFER THE INDICATED RECORDS FROM TERUTM TO
*     THE SMALLER FILES.
      CALL MOVEREC(LUIN,LUOUT)
      END
```

```
      SUBROUTINE FIX(A)
*****************************************************************
*     THIS SUBROUTINE 'FIXES' THE ARRAY A IN THAT THE FEATURE   *
*     AND ELEVATION CODES ARE REPACKED AS FEA+8*ELE AND         *
*     THE ARRAY IS TRANSPOSED.                                  *
*****************************************************************
      IMPLICIT INTEGER*2 (A-Z)
      DIMENSION A(100,100)
      DO J=1,100
         DO I=1,100
            ICODE=ISHFT(A(I,J),-12)
            IELV=A(I,J)-ISHFT(ICODE,12)
            ICODE=RECODE(ICODE)
            IF(IELV.LE.0.OR.IELV.EQ.4095)THEN
              ICNT=ICNT+1
              IELV=0
            ENDIF
            A(I,J)=ICODE+8*IELV
         ENDDO
      ENDDO
      CALL TRANSPOSE(A)
      RETURN
      END
```

```
      SUBROUTINE GRIDR(LUIN)
*********************************************
*       ROUTINE OPENS THE FILE 'TERUTM' AND   *
*       READS THE GRID RECORDS.               *
*********************************************
      INCLUDE 'GRID.CMN'
1 ***********************************************
1 *       GRID CONTAINS THE POINTERS TO THE 10-KM *
1 *       TERRAIN RECORDS IN THE FILE TERUTM.     *
1         INTEGER*2 GRID(128,128)
1         COMMON/GRID/GRID
1 ***********************************************
*****************************************************
      OPEN(ACCESS='DIRECT',ASSOCIATEVARIABLE=IAV,
     +     BLOCKSIZE=20008,FORM='UNFORMATTED',
     +     MAXREC=90000,NAME='DB1:[WALTER]TERUTM.DAT',
     +     RECORDSIZE=5002,RECORDTYPE='FIXED',
     +     TYPE='OLD',UNIT=LUT,EXTENDSIZE=1,SHARED)
*****************************************************
      READ(LUIN'1)((GRID(I,J),I=1,128),J=1,64)
      READ(LUIN'2)((GRID(I,J),I=1,128),J=65,128)
      OPEN(NAME='STATS',TYPE='NEW',UNIT=3)
      RETURN
      END
```

```
      SUBROUTINE INFO(LUIN,LUOUT)
*********************************************
*     PROMPTS THE USER FOR NECESSARY INFO   *
*********************************************
      INCLUDE 'GRAPH.CMN'
1 ************************************************
1 *     THE MIN AND MAX COORDINATE VALUES        *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT
1 ************************************************
      PRINT*,'ENTER THE EASTING AND NORTHING OF THE SW CORNER'
      READ*,XMIN,YMIN
      PRINT*,'NOW ENTER THE EXTENTS, AGAIN IN METERS.'
      PRINT*,'EASTING:'
      READ*,XINT
      PRINT*,'NORTHING:'
      READ*,YINT
      XMAX=XMIN+XINT
      PRINT*,'ENTER THE LOGICAL UNIT NUMBER OF THE INPUT FILE'
      READ*,LUIN
      PRINT*,'ENTER THE LOGICAL UNIT NUMBER OF THE OUTPUT FILE'
      READ*,LUOUT
      YMAX=YMIN+YINT
      RETURN
      END
```

```
      SUBROUTINE MOVEREC(LUIN,LUOUT)
*****************************************************
*     THIS SUBROUTINE IS DESIGNED TO MOVE DESIGNATED  *
*     RECORDS FROM THE FILE 'TERUTM.DAT' TO 10KM      *
*     FILES.                                          *
*****************************************************
*     INPUTS:  LUIN,LUOUT--LOGICAL UNIT NUMBERS       *
*****************************************************
      INTEGER*2 BUFR(10004),SQUARE(100,100)
      CHARACTER*7 MGR
      LOGICAL*1 ERR
      INTEGER*4 X,Y,I,J
      INCLUDE 'GRAPH.CMN'
1 ***********************************************
1 *     THE MIN AND MAX COORDINATE VALUES       *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT
1 ***********************************************
      INCLUDE 'GRID.CMN'
1 ***********************************************
1 *     GRID CONTAINS THE POINTERS TO THE 10-KM *
1 *     TERRAIN RECORDS IN THE FILE TERUTM.     *
1     INTEGER*2 GRID(128,128)
1     COMMON/GRID/GRID
1 ***********************************************
      EQUIVALENCE (BUFR(5),SQUARE(1,1))
C...  FOR EACH GRID
C...  10KM ARE SUBTRACTED TO INDICATE THE SW CORNER OF
C...  THE LAST BLOCK.
      DO X=XMIN,XMAX-10000,10000
C...     DISTANCES ARE MEASURED IN UNITS OF 20 METERS FROM
C...     AN ORIGIN OF 500000M N, 5600000M E.  THE ORIGIN
C...     CORRESPONDS TO  GRID INDICES OF (65,65) IN TERUTM.
C
         DO Y=YMIN,YMAX-10000,10000
            CALL IGRID(X,Y,I,J)
C  D        PRINT*,GRID(I,J),I,J
            IF(GRID(I,J).NE.0)THEN
              READ(LUIN'GRID(I,J))(BUFR(K),K=1,10004)
              CALL FIX(SQUARE)
              CALL UTM2MGR(X,Y,MGR,ERR)
              OPEN(UNIT=LUOUT,NAME=MGR,STATUS='NEW',FORM='UNFORM
              WRITE(LUOUT)SQUARE
              CLOSE(UNIT=LUOUT)
              PRINT*,MGR,X,Y
            ENDIF
            WRITE(3,*)MGR
C  D        WRITE(3,*)(BUFR(K),K=1,4)
C  D        WRITE(3,*)'GRID(I,J)',GRID(I,J)
C  D        WRITE(3,*) ' '
C  D        PRINT*,X,Y
         ENDDO
      ENDDO
      RETURN
      END
```

```
0001            FUNCTION RECODE(I)
0002    ************************************************************
0003    *       THIS FUNCTION TRANSLATES THE 13                    *
0004    *       CODES USED IN THE BDM TERUTM FILE INTO THE 3       *
0005    *       CODES USED IN THE CORDIVEM TERRAIN DISPLAY FILE.   *
0006    ************************************************************
0007    *       INPUTS: I-- THE OLD CODE                           *
0008    *       OUTPUTS: RECODE-- THE NEW ONE                      *
0009    ************************************************************
0010            IMPLICIT INTEGER*2(A-Z)
0011            IF(I.EQ.0)THEN
0012             I=4
0013    *        NO DATA                                  *
0014            ELSE
0015             IF(I.EQ.1.OR.I.EQ.2)THEN
0016              I=2
0017    *         URBAN                                   *
0018             ELSE
0019              IF(I.GE.3.AND.I.LE.5)THEN
0020               I=1
0021    *          FOREST                                 *
0022              ELSE
0023    *           OPEN=7,HEATH/BRUSH=6 ARE UNCHANGED    *
0024               IF(I.EQ.8)THEN
0025                I=3
0026    *           MARSH                                  *
0027               ELSE
0028                IF(I.EQ.9)THEN
0029                 I=5
0030    *            WATER                                 *
0031                ELSE
0032                 IF(I.GT.9)THEN
0033                  I=4
0034    *             BAD DATA IS NO DATA                  *
0035                ENDIF
0036               ENDIF
0037              ENDIF
0038             ENDIF
0039            ENDIF
0040            ENDIF
0041            RECODE=I
0042            RETURN
0043            END
```

```
      SUBROUTINE TRANSPOSE(A)
*****************************************************
*     TRANSPOSES A 100*100 WORD I*2 MATRIX          *
C*****************************************************
      INTEGER*2 A(100,100),B(100,100)
      DO J=1,100
          DO I=1,100
              B(I,J)=A(J,I)
          ENDDO
      ENDDO
      DO J=1,100
         DO I=1,100
            A(I,J)=B(I,J)
         ENDDO
      ENDDO
      END
```

DISPLAY CALLING SEQUENCE

```
DISPLAY
  ——SETCOLOR
  ——SETGEO*
  ——INFO
  ——UNGEN*
  ——MAPPER*
  ——PATCHIT
        |——POLYDEF
        |——FILLUP*
        |——FEATURES*
```

*TERRAIN SYSTEM UTILITIES

Figure B-3.

PROGRAM: DISPLAY

ROUTINES \ COMMON	ANSWER	CMERID	CORNER	MAP	WINDOW
FEATURES			X		X
FILLUP			X	X	X
INFO	X		X		X
MAPPER	X			X	X
PATCHIT			X		
SETGEO		X			
UNGEN			X		

Figure B-4.

```
0001            PROGRAM DISPLAY
0002    ***************************************************
0003    *       THIS ROUTINE DISPLAYS THE CURDIVEM DISPLAY *
0004    *       DATA IN 40KM SQUARES.                      *
0005    ***************************************************
0006
0007            CHARACTER*1 ANS
0008            CALL SETCOLOR
0009            CALL SETGEO
0010
0011    11      CONTINUE
0012            CALL INFO
0013            CALL UNGEN
0014            CALL MAPPER
0015            CALL CMCLOS
0016            PRINT*,'CORRECTIONS TO FEATURES?'
0017            READ(5,10) ANS
0018    10      FORMAT(1A1)
0019            CALL CMOPEN
0020            IF (ANS.EQ.'Y')THEN
0021               CALL PATCHIT(ANS)
0022            ENDIF
0023            GOTO11
0024            END
```

```
      SUBROUTINE INFO
*************************************************************
*     THIS ROUTINE JUST QUERIES THE OPERATOR AS TO WHICH    *
*     DISPLAYS HE WANTS.                                    *
*************************************************************
      IMPLICIT INTEGER*2 (I-N)
      INCLUDE 'WINDO.CMN'
1 ***************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND     *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER      *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS    *
1 *     THE CONTOUR INTERVAL.                                   *
1 ***************************************************************
1     DIMENSION FWINXY(4)
1     COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ***************************************************************
      INCLUDE 'ANSWER.CMN'
1 ***********************************************
1     CHARACTER*1 FEA,CON
1     COMMON/ANSWER/FEA,CON
1 ***********************************************
      INCLUDE 'CORNER.CMN'
1 *********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE  *
1 *     AREA IN THE ARRAY IBUF.                           *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 *********************************************************
      PRINT*,'ENTER THE COORDINATES OF THE SW CORNER '
      READ*,SWX,SWY
      FWINXY(1)=SWX
      FWINXY(2)=FWINXY(1)+40000
      FWINXY(3)=SWY
      FWINXY(4)=FWINXY(3)+40000
      PRINT*,'FEATURES?'
10    FORMAT(A1)
      READ(5,10) FEA
      PRINT*,'CONTOURS'
      READ(5,10)CON
      IF(CON.EQ.'Y')THEN
        PRINT*,'CONTOUR INTERVAL?'
        READ*,ZDELT
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE PATCHIT(ANSWER)
*****************************************************
*     A KLUDGE TO PASS DATA FROM POLYDEF TO FILLUP *
*****************************************************
      INCLUDE 'CORNER.CMN'
1 ************************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE     *
1 *     AREA IN THE ARRAY IBUF.                              *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ************************************************************
      INTEGER*2 ICLR,INCR
      DIMENSION POLY(500,2)
      CHARACTER*1 ANSWER
      DO WHILE(ANSWER.EQ.'Y')
          PRINT*,'COLOR?'
          READ*,ICLR
          CALL CMOPEN
          CALL POLYDEF(N,POLY,ICLR)
          DO I=1,N
             POLY(I,1)=POLY(I,1)-SWX
             POLY(I,2)=POLY(I,2)-SWY
          ENDDO
          CALL FILLUP(N,POLY,ICLR)
          INCR=1
          CALL FEATURES(INCR)
          CALL CMCLOS
          PRINT*,'ANOTHER PATCH?'
10        FORMAT(A1)
          READ(5,10) ANSWER
      ENDDO
      CALL GEN
      RETURN
      END
```

```
      SUBROUTINE POLYDEF(I,VERTEX,ICLR)
*************************************************************
*     THIS SUBROUTINE DISPLAYS A USER-DEFINED POLYGON,      *
*     AND RECORDS THE COORDINATES OF THE VERTICES IN        *
*     THE ARRAY VERTEX. NO MORE THAN 500 EDGES ARE ALLOWED. *
*************************************************************
*     INPUTS: ICLR-- THE LINE COLOR TO BE USED; 0-7         *
*     OUTPUTS: VERTEX-- THE ARRAY OF VERTICES               *
*              I-- THE NUMBER OF VERTICES                   *
*************************************************************
      INTEGER*2 ICLR
      INTEGER*4 IMAXPT,IPT,IRET
      DIMENSION VERTEX(500,2)
      DATA IMAXPT/1/
      CALL CMCLOS
      PRINT*,'TERMINATE POLYGON DEFINITION BY ENTERING 0 AT
     + THE LAST VERTEX.'
      CALL CMOPEN
      CALL LINCLR(ICLR)
C
      I=1
      DO WHILE (I.LT.500)
         CALL LOCATE(IMAXPT,PX,PY,IRET,IPT)
         IF(I.GT.1)THEN
           CALL MOVE(VERTEX(I-1,1),VERTEX(I-1,2))
           CALL DRAW(PX,PY)
         ENDIF
         VERTEX(I,1)=PX
         VERTEX(I,2)=PY
         IF(IRET.EQ.48)THEN !ASCII 0
           CALL DRAW(VERTEX(1,1),VERTEX(1,2))
           I=I+1
           VERTEX(I,1)=VERTEX(1,1)
           VERTEX(I,2)=VERTEX(1,2)
           CALL CMCLOS
           RETURN
         ENDIF
         I=I+1
      ENDDO
      END
```

```
      SUBROUTINE SETCOLOR
*********************************************
*       SETS COLORS AND SPECIFIES THE       *
*       TEKTRONIX 4027 AS THE GRAPHICS DEVICE *
*********************************************
      DIMENSION BLUE(3)
      REAL LTBLUE(3),LTGREEN(3)
      DIMENSION GREEN(3),BKGRND(3),RED(3),BLACK(3)
      DIMENSION YELLOW(3)
      DATA YELLOW/100.,100.,0./
      DATA GREEN/20.,60.,5./,BKGRND/30.,30.,30./
      DATA RED/30.,10.,10./,BLACK/0.,0.,0./
      DATA BLUE/0.,0.,100./,LTGREEN/30.,80.,30./,
     +  LTBLUE/0.,70.,100./
C
      IDEVICE=4027
      IOPT=5
      CALL GRSTRT(IDEVICE, IOPT)
      CALL CLRMAP(0,1,YELLOW)
      CALL CLRMAP(1,1,GREEN)
      CALL CLRMAP(2,1,RED)
      CALL CLRMAP(3,1,LTBLUE)
      CALL CLRMAP(4,1,BLACK)
      CALL CLRMAP(5,1,BLUE)
      CALL CLRMAP(6,1,LTGREEN)
      CALL CLRMAP(7,1,BKGRND)
      CALL BKGCLR(7)
C
      RETURN
      END
```

DRWHEX CALLING SEQUENCE

```
DRWHEX
  |
  |---HXGRDINIT
  |        |
  |        |--- HXINIT*
  |
  |--- GRIDINIT
  |        |
  |        |--- DRWGRD
  |        |        |
  |        |        |--- MARKER
  |        |
  |        |--- SETDEV
  |
  |--- HEXGEN
  |
  |--- SETHXGRD
  |        |
  |        |--- HEXREAD*
  |
  |--- DRWHXGRD
  |        |
  |        |--- DRW*
  |        |
  |        |--- UNPACKER*
  |
  |--- DRWROADS
  |        |
  |        |--- UNPACKER*
  |
  |--- DRAWROADS
  |        |
  |        |--- GRIDR*
  |        |
  |        |--- IGRID*
  |        |
  |        |--- DRWREC
  |        |        |
  |        |        |--- GETNDX*
  |        |        |
  |        |        |--- GETREC*
  |        |        |
  |        |        |--- DRWLNK2
  |        |        |        |
  |        |        |        |--- GETNDX*
  |        |        |        |
  |        |        |        |--- GETREC*
  |        |        |        |
  |        |        |        |--- GETSUB*
  |        |        |        |
  |        |        |        |--- TRANX*
  |        |        |        |
  |        |        |        |--- TRANY*
  |        |        |        |
  |        |        |        |--- DRWNOD
  |        |        |        |        |
  |        |        |        |        |--- TRANX*
  |        |        |        |        |
  |        |        |        |        |--- TRANY*
  |
  |--- FIXER
  |        |
  |        |--- REPACK*
```

* TERRAIN SYSTEM UTILITIES

Figure B-5.

PROGRAM: DRWHEX

ROUTINE \ COMMON	DISPLAY	GRAPH	HEX	HXGRD	HXSTOR	LNKNOD	PACK	SUB	TYPE	UNPACK
DRAWROADS						X				
DRW				X						
DRWGRD		X								
DRWHEX		X	X		X				X	
DRWHXGRD				X	X				X	X
DRWLNK2						X	X		X	
DRWREC						X				
DRWROADS				X	X	X				X
FIXER		X					X		X	
GETSUB						X		X		
GRIDINIT	X	X								
GRIDR						X				
HXGRDINIT	X		X	X						
MARKER		X								
REPACK										X
SETDEV		X								
SETHXGRD	X			X						
UNPACKER										X

Figure B-6.

```
0001            SUBROUTINE DRWNOD(NOD)
0002      ***********************************************
0003      *     THIS ROUTINE EXTRACTS THE XY COORDINATES  *
0004      *     OF THE TERMINUS AND DRAWS FROM THE LAST   *
0005      *     SUBNODE TO THE TERMINUS.                  *
0006      ***********************************************
0007      *     INPUTS: NOD--THE PACKED YX COORDINATES    *
0008      ***********************************************
0009            INTEGER*4 NOD,TMPNOD
0010            INTEGER*2 X,Y
0011            CALL CMOPEN
0012      *     SET X COORDINATE              *
0013            TMPNOD=LIBSEXTZV(0,16,NOD)
0014            CALL LIBSINSV(TMPNOD,0,16,X)
0015      *     SET Y COORDINATE              *
0016            TMPNOD=LIBSEXTZV(16,16,NOD)
0017            CALL LIBSINSV(TMPNOD,0,16,Y)
0018      *     DRAW TO THE END NODE          *
0019            CALL DRAW(TRANX(X),TRANY(Y))
0020            CALL CMCLOS
0021            RETURN
0022            END
```

```
      SUBROUTINE DRWREC(HDNODE)
*****************************************************
*     THIS ROUTINE DRAWS THE ROADNET FOR ONE 10-KM  *
*     SQUARE.                                       *
*****************************************************
*     INPUTS: HDNODE-- THE FIRST NODE POINTER FOR   *
*                      THIS 10-KM SQUARE            *
*****************************************************
      INTEGER*4 HDNODE,TMPNOD,NXTNOD
      INTEGER*2 X,Y,WRDPOS
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ***************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
1 ***************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1   + ,SUBREC(500)
1 ****************************************************************
      NXTNOD=HDNODE
      DO WHILE (NXTNOD.NE.0)
*        GET NODE RECORD      *
         CALL GETNDX(NXTNOD,NRECNUM,WRDPOS)
         LU=2
         CALL GETREC(LU,NRECNUM,NODREC)
         INCLUDE 'SETXY.FOR'
1 C
1 C     SET X COORDINATE
1          TMPNOD=LIBSEXTZV(0,16,NODREC(5,WRDPOS))
1          CALL LIBSINSV(TMPNOD,0,16,X)
1 C     SET Y COORDINATE
1          TMPNOD=LIBSEXTZV(16,16,NODREC(5,WRDPOS))
1          CALL LIBSINSV(TMPNOD,0,16,Y)
1 C
         NXTLNK=NODREC(4,WRDPOS)
         DO WHILE (NXTLNK.NE.0)
*          DRWLNK2 DRAWS ONE 'OUTLINK' AND *
*          GETS THE NEXT LINK POINTER      *
           CALL DRWLNK2(X,Y,NXTLNK)
         ENDDO
         NXTNOD=NODREC(1,WRDPOS)
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE DRWROADS
*****************************************************
*     THIS ROUTINE DRAWS THE HEX 'ROADS' FOR        *
*     EACH OF A SPECIFIED SET OF HEXES.  IT         *
*     DRAWS FROM THE CENTER TO THE SIDES OF         *
*     EACH HEX.  IT IS LIMITED TO A LEVEL 8         *
*     HEX BY THE SIZES OF THE ARRAYS IN HXSTOR.CMN  *
*****************************************************
      IMPLICIT INTEGER*4(H,P)
      INCLUDE 'UTIL:UNPACK.CMN'
1 ****************************************************
1     INTEGER*4 HSIDE(6) ! CONNECTIVITY CODES IN    *
1                        ! NUMERICAL ORDER BY SIDE *
1     COMMON/UNPACK/HSIDE
1 ****************************************************
      INCLUDE 'HXSTOR.CMN'
1 ************************************************************
1 *   THERE ARE 2401 LEVEL 4 HEXES IN A LEVEL 8 HEX          *
1     INTEGER*4 HXNO(2401) ! THE INTERNAL HEX NUMBERS        *
1     INTEGER*4 S(6)       ! S INDEXES THE HEX SIDES         *
1                          ! IN COUNTERCLOCKWISE ORDER       *
1     INTEGER*4 HXSIDES(2401) !PACKED HEX CONNECTIVITIES     *
1     DIMENSION XY(2401,2) ! THE XY COORDINATES OF THE       *
1                          ! HEX CENTERS                     *
1     COMMON /HXSTOR/HXN,HXNO,S,HXSIDES,XY
1 ************************************************************
      INCLUDE 'HXGRD.CMN'
1 ***************************************************
1 *   PARAMETERS USED IN DRAWING THE HEXES AND      *
1 *   THE LOC DATA                                  *
1 ***************************************************
1 *   PI 3.14159                                    *
1 *   X,Y THE RELATIVE POSITIONS OF THE HEX         *
1 *       VERTICES                                  *
1 *   RAD  RADIUS IN METERS OF A HEX                *
1 *   PHI  RELATIVE ROTATION TO THE START OF        *
1 *        THE HEX DRAW;ie,THE 'BOTTOM'             *
1 *   XPHI, THE RELATIVE DISTANCES TO THE START     *
1 *   YPHI   OF THE HEX DRAW                        *
1 ***************************************************
1     REAL PI,PSI,PHI
1     COMMON/HXGRD/X(6),Y(6),RAD,PHI,PI,
1    + XPHI,YPHI
1 ***************************************************
      DATA S/3,1,5,4,6,2/
*     CONVERT BACK TO THE RADIUS OF THE INSCRIBED CIRCLE
      RRAD=RAD*SQRT(3.0)/2.0
              TH=PHI-PI/6.0
*     FOR EACH HEX                    *
      CALL CMOPEN
      DO I=1,HXN
        CALL UNPACKER(HXSIDES(I))
        CALL VECABS
*       MOVE TO HEX CENTER            *
        CALL MOVE(XY(I,1),XY(I,2))
*       FOR EACH SIDE                 *
        DO J=1,6
```

```
0058                  CALL VECREL
0059                  TH=TH+PI/3.0
0060                  IF(HSIDE(S(J)).NE.0)THEN
0061                    LTYP=HSIDE(S(J))
0062                    CALL LINCLR(LTYP)
0063                    CALL DRAW(RRAD*COS(TH),RRAD*SIN(TH))
0064                    CALL VECABS
0065     *              MOVE TO HEX CENTER       *
0066                    CALL MOVE(XY(I,1),XY(I,2))
0067                  ENDIF
0068     *            NEXT SIDE
0069                ENDDO
0070     *          NEXT HEX
0071              ENDDO
0072              CALL VECABS
0073              CALL CMCLOS
0074              RETURN
0075              END
```

```
      SUBROUTINE FIXER
*****************************************************
*     THIS ROUTINE WILL ADD OR DELETE CONNECTIVITIES *
*     FROM THE 'HEXISED' LOC OR HYDRO DATA.          *
*****************************************************
      IMPLICIT INTEGER H
      INCLUDE'UTIL:PACKER.CMN'
1 ***********************************************************
1     INTEGER*4 SIDES ! PACKED CONNECTIVITIES              *
1     INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE      *
1     COMMON/PACK/SIDES,LTYPE
1 ***********************************************************
      INCLUDE 'UTIL:UNPACK.CMN'
1 ****************************************************
1     INTEGER*4 HSIDE(6) ! CONNECTIVITY CODES IN     *
1                        ! NUMERICAL ORDER BY SIDE   *
1     COMMON/UNPACK/HSIDE
1 ****************************************************
      INCLUDE 'TYPE.CMN'
1 ************************************************
1     LOGICAL LOC   ! TYPE=ROADS                 *
1     LOGICAL HYDRO ! TYPE=RIVERS                *
1     COMMON/TYPE/LOC,HYDRO
1 ************************************************
      DIMENSION HSTOR(2)
      CHARACTER ANS
      INTEGER*2 ICHAR
      DIMENSION XCENTR(2),YCENTR(2)
      DIMENSION ICHAR(2),PX(2),PY(2)
      DATA IMAXPT/2/
      INCLUDE 'GRAPH.CMN'
1 ****************************************************
1     REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
1                              ! IN METERS         *
1     INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
1                     ! DATA AND LEVEL OF HEX      *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
1 ****************************************************
      LEV=4
      ANS='Y'
      LTYPE=0
      LU=7
      CALL CMCLOS
      PRINT*,'ENTER 9 TO STOP.'
      DO WHILE(ANS.EQ.'Y')
          DO WHILE(LTYPE.NE.9)
            CALL CMOPEN
            CALL LOCATE(IMAXPT,PX,PY,ICHAR,IGOT)
            DO I=1,2
               X=PX(I)-500000.
               Y=PY(I)-5700000.
               CALL XYL2HA(X,Y,LEV,HSTOR(I))
                CALL HA2XYL(HSTOR(I),XCENTR(I),YCENTR(I),LEV)
                XCENTR(I)=XCENTR(I)+500000
                YCENTR(I)=YCENTR(I)+5700000
            ENDDO
            LTYPE=ICHAR(2)-48
```

```
                IF(LTYPE.GE.0.AND.LTYPE.LE.3)THEN
                  CALL REPACK(HSTOR,LTYPE)
                  IF(LOC)THEN
                    CALL MOVE(XCENTR(1),YCENTR(1))
                    IF(LTYPE.EQ.0)LTYPE=7
                    CALL LINCLR(LTYPE)
                    CALL DRAW(XCENTR(2),YCENTR(2))
                  ENDIF
                  IF(HYDRO)CALL DRW(XCENTR(2),YCENTR(2),HSIDE)
                ENDIF
              ENDDO
            CALL CMCLOS
            PRINT*,'MORE CHANGES?'
            READ(5,10)ANS
            IF(ANS.EQ.'Y')LTYPE=0
          ENDDO
10        FORMAT(1A1)
          RETURN
          END
```

```
      SUBROUTINE GRIDINIT
*****************************************
*     THIS SETS THE X AND Y COORDINATE *
*     LIMITS AND THE GRID INTERVAL.    *
*****************************************
      INCLUDE 'DISPLAY.CMN'
1 ************************************************************
1 *     XO AND YO ARE THE CENTER COORDINATES OF THE DISPLAY *
1 ************************************************************
1     COMMON/DISPLAY/ XO,YO
1 ************************************************************
      INCLUDE 'GRAPH.CMN'
1 ****************************************************
1     REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
1                              ! IN METERS         *
1     INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
1                     ! DATA AND LEVEL OF HEX      *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
1 ****************************************************
      DIMENSION LEVL(4)
      DATA LEVL/1,1,1,4/
      L=LEV-3
*     SET THE X AND Y LIMITS.    *
      XMIN=XO-LEVL(L)*10000
      YMIN=YO-LEVL(L)*10000
      XMAX=XO+LEVL(L)*10000+1
      YMAX=YO+LEVL(L)*10000+1
      INT=10000
      CALL SETDEV
      CALL DRWGRD
      RETURN
      END
```

```
      SUBROUTINE HEXGEN(HXN,HXNO,LEVTOP,LEVBOT)
*****************************************************
*     THIS ROUTINE GENERATES HEX NUMBERS FROM       *
*     'LEVTOP' TO 'LEVBOT',NESTED IN HEX FASHION.   *
*****************************************************
*     INPUTS: LEVTOP-- THE HIGHEST LEVEL OF HEX     *
*                      TO BE GENERATED              *
*             LEVBOT-- THE LEVEL TO BE USED IN      *
*                      FILLING THE HEX TREE         *
*     OUTPUTS: HXN -- THE NUMBER OF HEXES GENERATED *
*              HXNO -- THE ARRAY OF HEX NUMBERS     *
*****************************************************
      IMPLICIT INTEGER (H,P)
      DIMENSION LEVSTOP(4:10)
      DIMENSION HXNO(2401)
      INTEGER*4 ZERO
C...  THE VARIABLE 'LEVMIN' SHOULD BE (AND PROBABLY IS) SET ELSEWH
      LEVMIN=4
      ZERO=77777777
      L=LEVTOP-LEVBOT
      IF(L.EQ.0)THEN
        PRINT*,'ERROR IN HEXGEN'
        RETURN
      ENDIF
      HXN=7**(L-1)  !YOU MUST ASK FOR AT LEAST ONE LEVEL
      DO I=4,10
         LEVSTOP(I)=0
      ENDDO
      DO I=LEVBOT,LEVTOP-1
         LEVSTOP(I)=6
      ENDDO
      HXN=0
C...  IF LEVSTOP(LEVEL)=0 THE LOOP IS ONLY EXECUTED ONCE, AND
C...  THE DIGIT SUBTRACTED IS 0,..NO CHANGE
      DO LEV10=0,LEVSTOP(10)
           L10=LEV10*1000000
           DO LEV9=0,LEVSTOP(9)
              L9=LEV9*100000
              DO LEV8=0,LEVSTOP(8)
                 L8=LEV8*10000
                 DO LEV7=0,LEVSTOP(7)
                   L7=LEV7*1000
                   DO LEV6=0,LEVSTOP(6)
                      L6=LEV6*100
                      DO LEV5=0,LEVSTOP(5)
                         L5=10*LEV5
                         DO LEV4=0,LEVSTOP(4)
                            L4=LEV4
                            HXN=HXN+1
                            HXNO(HXN)=ZERO-L10-L9-L8-L7-L6-L5-L4
C  D                        PRINT*,HXN,HXNO(HXN)
                         ENDDO
                      ENDDO
                   ENDDO
                 ENDDO
              ENDDO
           ENDDO
```

```
0058                 ENDDO
0059 C   D        PRINT*,HXN,(HXNO(I),I=1,HXN)
0060              DO N=1,HXN
0061                 CALL HEXIN(HXNO(N),1,LEVMIN,HSTOR)
0062                 HXNO(N)=HSTOR
0063              ENDDO
0064              RETURN
0065              END
```

```
      SUBROUTINE HXGRDINIT(LEV)
*****************************************************
*     THIS INITIALIZES THE PARAMETERS FOR           *
*     ACCESSING AND DRAWING THE HEX GRID            *
*****************************************************
*     OUTPUTS: LEV-- THE LEVEL OF THE HEX TO BE DRAWN *
*****************************************************
      INCLUDE 'UTIL:HEX.CMN'
1 ***************************************************************
1 *     FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR
1 ***************************************************************
1       IMPLICIT INTEGER(H,P)
1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR,
1     +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
1     +                ICON(70),JCON(70),IMAX(7),JMAX(7)
1 ***************************************************************
      INCLUDE 'DISPLAY.CMN'
1 ***********************************************************
1 *     XO AND YO ARE THE CENTER COORDINATES OF THE DISPLAY *
1 ***********************************************************
1     COMMON/DISPLAY/ XO,YO
1 ***********************************************************
      INCLUDE 'HXGRD.CMN'
1 ***************************************************
1 *     PARAMETERS USED IN DRAWING THE HEXES AND    *
1 *     THE LOC DATA                                *
1 ***************************************************
1 *     PI 3.14159                                  *
1 *     X,Y THE RELATIVE POSITIONS OF THE HEX       *
1 *         VERTICES                                *
1 *     RAD  RADIUS IN METERS OF A HEX              *
1 *     PHI  RELATIVE ROTATION TO THE START OF      *
1 *          THE HEX DRAW;ie,THE 'BOTTOM'           *
1 *     XPHI, THE RELATIVE DISTANCES TO THE START   *
1 *     YPHI   OF THE HEX DRAW                      *
1 ***************************************************
1     REAL PI,PSI,PHI
1     COMMON/HXGRD/X(6),Y(6),RAD,PHI,PI,
1   + XPHI,YPHI
1 ***************************************************
      DATA PI/3.14159/
      PRINT*,'WHAT LEVEL OF HEX DO YOU WANT?'
      READ*,LEV
      PRINT*,'ENTER THE X,Y COORDINATES OF THE CENTER'
      PRINT*,'AS AN OFFSET IN METERS FROM NC00:'
      READ*,XO,YO
*     IT SEEMS EASIER TO HAVE THE OPERATOR PUT THE
*     COORDINATES RELATIVE TO NC00, BUT ALL THE HEX
*     UTILITIES ARE OFFSETS FROM THE CENTER HEX.
      XO=500000+XO
      YO=5700000+YO
      INCLUDE 'UTIL:HEXINIT.PRM'
1 ***************************************************************
1 *     IWRITE:  OUTPUT DEVICE FOR ERROR MESSAGES                *
1 *     LEVMAX:  MAXIMUM LEVEL OF HEX AGGREGATION                *
1 *     LEVMIN:  MINIMUM   "        "   "                       *
1 *     DLT:  LATITUDE OF THE ORIGIN HEX IN FLOATING-            *
```

```
0058  1 *        POINT DEGREES                                          *
0059  1 *        DLN:  LONGITUDE OF ORIGIN HEX                          *
0060  1 *        LEVS Z:  HEX LEVEL AT WHICH THE SCALE OF THE           *
0061  1 *        HEX COORDINATE SYSTEM IS GIVEN                         *
0062  1 *        SIZHEX:  DIAMETER OF HEXES AT SIZE 'LEVSIZ' IN         *
0063  1 *        FLOATING-POINT METERS                                  *
0064  1          IWRITE=6
0065  1          LEVMAX=9
0066  1          LEVMIN=4
0067  1          DLT=51.45
0068  1          DLN=9.00
0069  1          LEVSIZ=6
0070  1          SIZHEX=25000.
0071  1          CALL HXINIT(IWRITE,LEVMAX,LEVMIN,DLT,DLN,LEVSIZ,SIZHEX)
0072  1 ***********************************************************
0073             RAD=DIAMTR/2.0
0074     *       'DIAMTR' IS THE DIAMETER OF THE INSCRIBED CIRCLE
0075             RAD=2*RAD/SQRT(3.0)
0076     *       PHI IS THE ROTATION OF THE HEX GRID FROM NORTH
0077             PHI=(LEVMIN*19.1)*PI/180
0078             XPHI=RAD*COS(PHI)
0079             YPHI=RAD*SIN(PHI)
0080             PSI=PHI+PI/3
0081     *       SET THE RELATIVE POSITIONS OF THE VERTICES
0082             DO I=1,6
0083                PSI=PSI+PI/3
0084                X(I)=COS(PSI)*RAD
0085                Y(I)=SIN(PSI)*RAD
0086             ENDDO
0087             RETURN
0088             END
```

```
      SUBROUTINE MARKER
*************************************************
*     THIS ROUTINE JUST WRITES THE RELATIVE     *
*     COORDINATES OF THE LOWER LEFT CORNER      *
*************************************************
      INCLUDE 'GRAPH.CMN'
1 ******************************************************
1     REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
1                              ! IN METERS         *
1     INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
1                     ! DATA AND LEVEL OF HEX      *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
1 ******************************************************
      CALL CMOPEN
      CALL TXTCLR(0)
      ISIZE=0
      PX=2
      PY=3
      CALL TXAM
      CALL TXSIZE(ISIZE,PX,PY)
*     IMAXX AND IMAXY ARE THE NUMBER OF DIGITS IN  *
*     THE X AND Y COORDINATES, RESPECTIVELY.       *
*     SET X AND Y TO THE UTM COORDINATES OF THE    *
*     SW CORNER
      XX=ALOG10(XMIN)
      IMAXX=NINT(XX)+1
      YY=ALOG10(YMIN)
      IMAXY=NINT(YY)+1
*     PROPORTION THE MOVE RELATIVE TO THE WINDOW SIZE *
*     THE 400 WAS ARRIVED AT HEURISTICALLY            *
      N=(YMAX-YMIN)/10000+1
      CALL MOVE(XMIN,YMIN+400*N)
      CALL RNUMBR(XMIN,-1,IMAXX)
      CALL MOVE(XMIN,YMIN)
      CALL RNUMBR(YMIN,-1,IMAXY)
      CALL CMCLOS
      RETURN
      END
```

```
      SUBROUTINE SETDEV
*****************************************
*       SETS GRAPHICS ENVIRONMENT *
*****************************************
      INCLUDE 'GRAPH.CMN'
1 ******************************************************
1     REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
1                              ! IN METERS         *
1     INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
1                     ! DATA AND LEVEL OF HEX      *
1     COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
1 ******************************************************
*****************************************************
*       INITIALIZING AND DEFINING THE COLORS USED *
*       IN DRAWING THE HEX CONNECTIVITIES AND LOC *
*****************************************************
      DIMENSION RED1(3),RED2(3),BLUE1(3),BLUE2(3)
      DIMENSION GREEN1(3),GREEN2(3),BKGRND(3),BLACK(3)
      DATA RED1/30.,10.,10./,RED2/30.,10.,10./
      DATA BLUE1/10.,20.,100./,BLUE2/10.,20.,100./
      DATA GREEN1/20.,60.,5./,GREEN2/20.,60.,5./
      DATA BKGRND/30.,30.,30./,BLACK/0.,0.,0./
*************************************************************
      IDEVICE=4027
      IOPT=5
      CALL GRSTRT(IDEVICE, IOPT)
      CALL CMOPEN
      XMN=XMIN-5000
      XMX=XMAX+5000
      YMN=YMIN-5000
      YMX=YMAX+5000
      CALL WINDOW(XMN,XMX,YMN,YMX)
*********************************************************
*       THE '1' COLORS ARE USED IN DRAWING THE HEXLOC     *
*       DATA AND THE '2' COLORS FOR THE ORIGINAL LOC DATA *
*********************************************************
      CALL CLRMAP(1,1,GREEN1)
      CALL CLRMAP(2,1,BLUE1)
      CALL CLRMAP(3,1,RED1)
      CALL CLRMAP(4,1,RED2)
      CALL CLRMAP(5,1,BLUE2)
      CALL CLRMAP(6,1,GREEN2)
      CALL CLRMAP(7,1,BKGRND)
      CALL CLRMAP(0,1,BLACK)
      CALL BKGCLR(7)
*
      CALL VWPORT(30.,129.3,0.,99.3)
      CALL CMCLOS
    RETURN
    END
```

```
      SUBROUTINE SETHXGRD(MINLEV)
*********************************************
*     SETS THE XY ADDRESSES OF THE HEXES    *
*     AND READS THE CONNECTIVITIES IN.      *
*********************************************
*     INPUTS: MINLEV-- THE MINIMUM HEX LEVEL *
*********************************************
      IMPLICIT INTEGER(H,P)
      INCLUDE 'HXSTOR.CMN'
1 ************************************************************
1 *     THERE ARE 2401 LEVEL 4 HEXES IN A LEVEL 8 HEX       *
1     INTEGER*4 HXNO(2401) ! THE INTERNAL HEX NUMBERS      *
1     INTEGER*4 S(6)       ! S INDEXES THE HEX SIDES       *
1                          ! IN COUNTERCLOCKWISE ORDER     *
1     INTEGER*4 HXSIDES(2401) !PACKED HEX CONNECTIVITIES *
1     DIMENSION XY(2401,2) ! THE XY COORDINATES OF THE     *
1                          ! HEX CENTERS                   *
1     COMMON /HXSTOR/HXN,HXNO,S,HXSIDES,XY
1 ************************************************************
      INCLUDE 'DISPLAY.CMN'
1 ********************************* *************************
1 *     XO AND YO ARE THE CENTER COORDINATES OF THE DISPLAY *
1 ************************************************************
1     COMMON/DISPLAY/ XO,YO
1 ************************************************************
      INCLUDE 'UTIL:HEXROAD.OPN'
1 *     THIS OPENS THE ISAM FILE USED TO PROCESS AND        *
1 *     DISPLAY THE LOC AND HYDRO DATA .                    *
1     OPEN(UNIT=7,NAME='HEXROAD',STATUS='UNKNOWN',
1    +ORGANIZATION='INDEXED',ACCESS='KEYED',RECL=2,
1    +RECORDTYPE='FIXED',FORM='UNFORMATTED',
1    +KEY=(1:4:INTEGER),SHARED)
1 *
      LU=7
      XO=XO-500000
      YO=YO-5700000
      CALL XYL2HA(XO,YO,MINLEV,HEXO) ! SET CENTER HEX
      DO I=1,HXN ! FOR EACH HEX
         HXNO(I)=HXADD(HXNO(I),HEXO) ! TRANSLATE
         CALL HEXREAD(HXNO(I),HXSIDES(I),LU) ! GET SIDES
         CALL HA2XYL(HXNO(I),X,Y,MINLEV) ! GET XY COORDINATES
         XY(I,1)=X+500000    ! BOTH COORDINATE SYSTEMS ARE
         XY(I,2)=Y+5700000   ! CENTERED AT 32UNC00
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE DRAWROADS(XMIN,XMAX,YMIN,YMAX)
*****************************************************
*     THIS SET OF ROUTINES SUPERIMPOSES THE BDM-    *
*     DERIVED ROADNET ON THE HEX CONNECTIVITIES FOR *
*     COMPARISON.                                   *
*****************************************************
*     INPUTS:XMIN,ETC--BOUNDING COORDINATES IN METERS *
*****************************************************
      INTEGER*4 I,J,X,Y
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ***************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
1 ***************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1   + ,SUBREC(500)
1 ****************************************************************
      INCLUDE 'UTIL:LNKNOD.OPN'
1 *    OPENING THE GRID,NODE,LINK,AND SUBNODE FILES
1       OPEN(UNIT=1,NAME='GRID',TYPE='OLD',READONLY,
1    *ACCESS='DIRECT',BLOCKSIZE=2000,SHARED)
1       OPEN(UNIT=2,NAME='NODE',TYPE='OLD',READONLY,
1    *ACCESS='DIRECT',BLOCKSIZE=2000,SHARED)
1       OPEN(UNIT=3,NAME='ROAD',TYPE='OLD',READONLY,
1    *ACCESS='DIRECT',BLOCKSIZE=2000,SHARED)
1       OPEN(UNIT=4,NAME='SUBN',TYPE='OLD',READONLY,
1    *ACCESS='DIRECT',BLOCKSIZE=2000,SHARED)
1 *
*
*     READ IN GRID FILE
      CALL GRIDR
*     FOR EACH GRID
*
      DO X=XMIN-10000,XMAX,10000
*        DISTANCES ARE MEASURED IN UNITS OF 20 METERS FROM
*        AN ORIGIN OF 500000M N, 5600000M E.  THE ORIGIN
*        CORRESPONDS TO  GRID INDICES OF (65,65).
         DO Y=YMIN-10000,YMAX,10000
              CALL IGRID(X,Y,I,J)
              CALL DRWREC(GRID(I,J))
         ENDDO
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE DRW(XX,YY,SIDES)
*************************************************************
*          INPUTS:                                          *
*                 XX,YY--CENTER OF THE HEX TO BE DRAWN      *
*                 SIDES-- THE COLOR CODES FOR THE HEX       *
*************************************************************
      IMPLICIT INTEGER (H,P)
      INTEGER SIDES(6)
      INCLUDE 'HXGRD.CMN'
1 ***************************************************
1 *     PARAMETERS USED IN DRAWING THE HEXES AND    *
1 *     THE LOC DATA                                *
1 ***************************************************
1 *     PI 3.14159                                  *
1 *     X,Y THE RELATIVE POSITIONS OF THE HEX       *
1 *         VERTICES                                *
1 *     RAD  RADIUS IN METERS OF A HEX              *
1 *     PHI  RELATIVE ROTATION TO THE START OF      *
1 *          THE HEX DRAW;ie,THE 'BOTTOM'           *
1 *     XPHI, THE RELATIVE DISTANCES TO THE START   *
1 *     YPHI   OF THE HEX DRAW                      *
1 ***************************************************
1     REAL PI,PSI,PHI
1     COMMON/HXGRD/X(6),Y(6),RAD,PHI,PI,
1    + XPHI,YPHI
1 ***************************************************
*     THIS IS THE ORDER IN WHICH THE SIDES ARE DRAWN
      INTEGER S(6)
      DATA S/3,1,5,4,6,2/
      CALL VECABS
      CALL MOVE(XX,YY)
      CALL VECREL
      CALL MOVE(XPHI,YPHI)
      DO J=1,6
         LTYP=SIDES(S(J))
         CALL LINCLR(LTYP)
         CALL DRAW(X(J),Y(J))
      ENDDO
      RETURN
      END
```

```
0001            SUBROUTINE DRWGRD
0002    ***********************************************************
0003    *       THIS ROUTINE DRAWS THE GRID LINES *
0004    ***********************************************************
0005            INCLUDE 'GRAPH.CMN'
0006  1 ***************************************************
0007  1         REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
0008  1                                  ! IN METERS        *
0009  1         INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
0010  1                         ! DATA AND LEVEL OF HEX      *
0011  1         COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
0012  1 ***************************************************
0013            CALL CMOPEN
0014            CALL BKGCLR(7)
0015            CALL NEWPAG
0016            CALL LINCLR(0)
0017    *       DRAW THE HORIZONTAL GRID LINES
0018            YINT=INT
0019            DO Y=YMIN,YMAX,YINT
0020                    CALL MOVE(XMIN,Y)
0021                    CALL DRAW(XMAX,Y)
0022            ENDDO
0023    *       DRAW THE VERTICAL GRID LINES
0024            XINT=INT
0025            DO X=XMIN,XMAX,XINT
0026                    CALL MOVE(X,YMIN)
0027                    CALL DRAW(X,YMAX)
0028            ENDDO
0029    *       PRINT THE LOWER LEFT COORDINATES
0030            CALL CMCLOS
0031            CALL MARKER
0032            RETURN
0033            END
```

```
0001            PROGRAM DRWHEX
0002      ****************************************
0003      *       A SET O  ROUTINES TO DISPLAY THE *
0004      *       LOC AND 'HEXISED LOC DATA AND    *
0005      *       THE ASSOCIATED GRIDS.            *
0006      ****************************************
0007            INCLUDE 'UTIL:HEX.CMN'
0008    1 ***************************************************************
0009    1 *       FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR
0010    1 ***************************************************************
0011    1       IMPLICIT INTEGER(H,P)
0012    1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR
0013    1      +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
0014    1      +                ICON(70),JCON(70),IMAX(7),JMAX(7)
0015    1 ***************************************************************
0016            INCLUDE 'HXSTOR.CMN'
0017    1 ************************************************************
0018    1 *       THERE ARE 2401 LEVEL 4 HEXES IN A LEVEL 8 HEX      *
0019    1         INTEGER*4 HXNO(2401) ! THE INTERNAL HEX NUMBERS    *
0020    1         INTEGER*4 S(6)       ! S INDEXES THE HEX SIDES     *
0021    1                              ! IN COUNTERCLOCKWISE ORDER   *
0022    1         INTEGER*4 HXSIDES(2401) !PACKED HEX CONNECTIVITIES *
0023    1         DIMENSION XY(2401,2) ! THE XY COORDINATES OF THE   *
0024    1                              ! HEX CENTERS                 *
0025    1         COMMON /HXSTOR/HXN,HXNO,S,HXSIDES,XY
0026    1 ************************************************************
0027            INCLUDE 'GRAPH.CMN'
0028    1 ***************************************************
0029    1         REAL XMIN,XMAX,YMIN,YMAX ! WINDOW BOUNDARIES *
0030    1                                  ! IN METERS         *
0031    1         INTEGER INT,LEV ! INTERVAL FOR ACCESSING THE *
0032    1                         ! DATA AND LEVEL OF HEX      *
0033    1         COMMON /GRAPH/XMIN,XMAX,YMIN,YMAX,INT,LEV
0034    1 ***************************************************
0035            INCLUDE 'TYPE.CMN'
0036    1 ************************************************
0037    1         LOGICAL LOC   ! TYPE=ROADS              *
0038    1         LOGICAL HYDRO ! TYPE=RIVERS             *
0039    1         COMMON/TYPE/LOC,HYDRO
0040    1 ************************************************
0041            CHARACTER*1 IANS
0042            PRINT*,'LOC OR HYDRO?'
0043            READ(5,10)IANS
0044            IF(IANS.EQ.'L')LOC=.TRUE.
0045            IF(IANS.EQ.'H')HYDRO=.TRUE.
0046            IF(.NOT.LOC.AND..NOT.HYDRO)THEN
0047              PRINT*,'INVALID TYPE; TRY AGAIN'
0048              STOP
0049            ENDIF
0050            IANS='Y'
0051            DO WHILE(IANS.EQ.'Y')
0052              CALL HXGRDINIT(LEV)
0053              CALL GRIDINIT
0054              CALL HEXGEN(HXN,HXNO,LEV,MINLEV)
0055              CALL SETHXGRD(MINLEV)
0056              CALL DRWHXGRD
0057              IF(LOC)CALL DRWROADS
```

```
0058                PRINT*,'ROADS?'
0059                READ(5,10)IANS
0060     10         FORMAT(1A1)
0061                IF(IANS.EQ.'Y')THEN
0062                   CALL DRAWROADS(XMIN,XMAX,YMIN,YMAX)
0063                ENDIF
0064                CALL FIXER
0065                PRINT*,'ANOTHER RUN?'
0066                READ(5,10)IANS
0067             ENDDO
0068             CALL GRSTOP
0069             END
```

```
0001            SUBROUTINE DRWHXGRD
0002      ********************************************
0003      *     CALLS 'DRW' TO DRAW A HEX EITHER IN    *
0004      *     BLACK AND WHITE OR HAVING COLOR, AS    *
0005      *     NEEDS BE.                              *
0006      ********************************************
0007            IMPLICIT INTEGER(H,P)
0008            INCLUDE 'HXGRD.CMN'
0009    1 ****************************************************
0010    1 *     PARAMETERS USED IN DRAWING THE HEXES AND     *
0011    1 *     THE LOC DATA                                 *
0012    1 ****************************************************
0013    1 *     PI 3.14159                                   *
0014    1 *     X,Y THE RELATIVE POSITIONS OF THE HEX        *
0015    1 *         VERTICES                                 *
0016    1 *     RAD  RADIUS IN METERS OF A HEX               *
0017    1 *     PHI  RELATIVE ROTATION TO THE START OF       *
0018    1 *          THE HEX DRAW;ie,THE 'BOTTOM'            *
0019    1 *     XPHI, THE RELATIVE DISTANCES TO THE START    *
0020    1 *     YPHI   OF THE HEX DRAW                       *
0021    1 ****************************************************
0022    1       REAL PI,PSI,PHI
0023    1       COMMON/HXGRD/X(6),Y(6),RAD,PHI,PI,
0024    1     + XPHI,YPHI
0025    1 ****************************************************
0026            INCLUDE 'UTIL:UNPACK.CMN'
0027    1 ******************************************************
0028    1       INTEGER*4 HSIDE(6) ! CONNECTIVITY CODES IN    *
0029    1                          ! NUMERICAL ORDER BY SIDE *
0030    1       COMMON/UNPACK/HSIDE
0031    1 ******************************************************
0032            INCLUDE 'HXSTOR.CMN'
0033    1 ************************************************************
0034    1 *     THERE ARE 2401 LEVEL 4 HEXES IN A LEVEL 8 HEX        *
0035    1       INTEGER*4 HXNO(2401) ! THE INTERNAL HEX NUMBERS      *
0036    1       INTEGER*4 S(6)       ! S INDEXES THE HEX SIDES       *
0037    1                            ! IN COUNTERCLOCKWISE ORDER     *
0038    1       INTEGER*4 HXSIDES(2401) !PACKED HEX CONNECTIVITIES *
0039    1       DIMENSION XY(2401,2) ! THE XY COORDINATES OF THE     *
0040    1                            ! HEX CENTERS                   *
0041    1       COMMON /HXSTOR/HXN,HXNO,S,HXSIDES,XY
0042    1 ************************************************************
0043            INCLUDE 'TYPE.CMN'
0044    1 ***************************************************
0045    1       LOGICAL LOC   ! TYPE=ROADS                  *
0046    1       LOGICAL HYDRO ! TYPE=RIVERS                 *
0047    1       COMMON/TYPE/LOC,HYDRO
0048    1 ***************************************************
0049            CALL CMOPEN
0050            IF (LOC)THEN
0051              DO J=1,6
0052                HSIDE(J)=0
0053              ENDDO
0054            ENDIF
0055            DO I=1,HXN
0056              IF(HYDRO) CALL UNPACKER(HXSIDES(I))
0057              CALL DRW(XY(I,1),XY(I,2),HSIDE)
```

```
0058          ENDDO
0059          CALL VECABS
0060          CALL CMCLOS
0061          RETURN
0062          END
```

```
0001            SUBROUTINE DRWLNK2(X,Y,NXTLNK)
0002      ***************************************************
0003      *     THIS ROUTINE DRAWS THE 'OUTLINK'              *
0004      *     ORIGINATING AT NODE X,Y AND GETS THE POINTER  *
0005      *     TO THE NEXT LINK AT THIS NODE                 *
0006      ***************************************************
0007      *     INPUTS:                                       *
0008      *            X,Y-- COORDINATES OF THE START NODE    *
0009      *            NXTLNK-- THE FIRST OUTLINK AT THE NODE *
0010      *     OUTPUTS: NXTLNK                               *
0011      ***************************************************
0012            INTEGER*4 TMPLINK,SRECNUM,WRDPOS,NOD
0013            INTEGER*2 ICLR,X,Y
0014            INTEGER*2 NUM
0015            INCLUDE 'UTIL:SUB.CMN'
0016    1 ***********************************************************
0017    1 *****   SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES
0018    1 *****             IN ONE LINK (SEE BDM DOCUMENTATION)
0019    1 ***********************************************************
0020    1       INTEGER*2 SUBX(100),SUBY(100)
0021    1       COMMON/SUB/ SUBX,SUBY
0022    1 ***********************************************************
0023            INCLUDE 'UTIL:LNKNOD.CMN'
0024    1 ***************************************************************
0025    1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
0026    1 ***************************************************************
0027    1       INTEGER*4 GRID,NODREC,LNKREC,SUBREC
0028    1       COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
0029    1     + ,SUBREC(500)
0030    1 ****************************************************************
0031            INCLUDE 'TYPE.CMN'
0032    1 ***********************************************
0033    1       LOGICAL LOC   ! TYPE=ROADS                *
0034    1       LOGICAL HYDRO ! TYPE=RIVERS               *
0035    1       COMMON/TYPE/LOC,HYDRO
0036    1 ***********************************************
0037            CALL CMOPEN
0038            CALL GETNDX(NXTLNK,LRECNUM,WRDPOS)
0039            N=WRDPOS
0040            LU=3
0041            CALL GETREC(LU,LRECNUM,LNKREC)
0042      *     SET LINK TYPE                        *
0043            TMPLINK=LIB$EXTZV(0,16,LNKREC(3,N))
0044            CALL LIB$INSV(TMPLINK,0,16,LTYPE)
0045      *
0046            IF(HYDRO)LTYPE=LTYPE-11
0047      *     NOT ALL TYPES OF LINKS ARE DRAWN    *
0048            IF(LTYPE.LE.3)THEN
0049               IF (LTYPE.LT.0)LTYPE=-3
0050               ICLR=LTYPE+3
0051               CALL LINCLR(ICLR)
0052      *
0053      *     SET SUBNODE PHYSICAL RECORD NUMBER  *
0054            TMPLINK=LIB$EXTZV(0,16,LNKREC(5,N))
0055            CALL LIB$INSV(TMPLINK,0,16,SRECNUM)
0056      *     SET SUBNODE WORD NUMBER             *
0057            TMPLINK=LIB$EXTZV(16,16,LNKREC(5,N))
```

```
0058              CALL LIBSINSV(TMPLINK,0,16,WRDPOS)
0059     *
0060     *
0061     *        IF THERE ARE NO SUBNODES SET NXTLNK AND RETURN *
0062                 IF(SRECNUM.GT.0.AND.WRDPOS.GT.0)THEN
0063                    CALL GETSUB(SRECNUM,WRDPOS,NUM)
0064                    CALL MOVE(TRANX(X),TRANY(Y))
0065     *
0066                    DO I=1,NUM
0067     *              DRAW TO EACH SUBNODE                          *
0068                       CALL DRAW(TRANX(SUBX(I)),TRANY(SUBY(I)))
0069                    ENDDO
0070                    NOD=LNKREC(4,N)
0071                    CALL DRWNOD(NOD)
0072     10             CONTINUE
0073                 ENDIF
0074              ENDIF
0075              NXTLNK=LNKREC(2,N)
0076              CALL CMCLOS
0077              RETURN
0078              END
```

HEXER CALLING SEQUENCE

```
HEXER
 |
 |---HEXINIT*
 |
 |---CORNERS
 |
 |---HEXGEN
 |
 |---HEX2XY*
 |
 |---UNGEN*
 |
 |---GATOR
     |
     |---SUMMER
     |     |
     |     |---ICODE*
     |     |
     |     |---IELV
     |
     |---AVG
```

*TERRAIN SYSTEM UTILITIES

Figure B-7.

PROGRAM: HEXER

ROUTINE \ COMMON	CENTER	CODE	CORNER	HEX	HEXRAD	MAXMIN	MAP
AVG		X					
CORNERS						X	
GATOR		X	X		X		
HEX2XY	X						
HEXER			X			X	
HEXINIT				X	X		
ICODE							X
IELV							X
MAPIN							X
SUMMER		X			X		

Figure B-8.

```
0001              SUBROUTINE AVG(HXNO,X,Y)
0002      ***********************************************************
0003      *       THIS ROUTINE AVERAGES THE ELEVATIONS,SLOPES,AND CODES *
0004      *       FOR ONE HEX AND WRITES THE RESULTS.                   *
0005      ***********************************************************
0006      *       INPUTS: HXNO-- EXTERNAL FORM OF THE HEX NUMBER        *
0007      *               X,Y--  UTM COORDINATES OF THE HEX CENTER      *
0008      ***********************************************************
0009              IMPLICIT INTEGER*2 (H-P)
0010              INTEGER*4 HXNO
0011              INCLUDE 'CODE.CMN'
0012    1 ***********************************************************
0013    1 *       ICOD CONTAINS THE COUNTS OF THE RESPECTIVE SURFACE CODES *
0014    1 *       COD CONTAINS THE RESPECTIVE PERCENTAGES                  *
0015    1 *       SSUM IS THE SUM OF THE ABSOLUTE SLOPES                   *
0016    1 *       ZSUM IS THE SUM OF THE ELEVATIONS                        *
0017    1 *       NSLOPES IS THE NUMBER OF SLOPES COUNTED                  *
0018    1 *       AND NPOINTS IS THE NUMBER OF POINTS COUNTED              *
0019    1 ***********************************************************
0020    1         INTEGER*2 ICOD(0:2)
0021    1         DIMENSION COD(0:2)
0022    1         COMMON/CODE/ICOD,COD,SSUM,ZSUM,NSLOPES,NPOINTS
0023    1 ***********************************************************
0024                S=NSLOPES
0025                S=S*100.  ! DATA BASE RESOLUTION OF 100M
0026                SLOPE=SSUM/S
0027                ELEV=ZSUM/NPOINTS
0028      *
0029      *       FOR EACH FEATURE CODE
0030                DO I=0,2
0031      *             COMPUTE % FEATURE TYPE
0032                    COD(I)=ICOD(I)
0033                    COD(I)=COD(I)/NPOINTS
0034      *             NEXT TYPE
0035                ENDDO
0036              LU=3
0037              WRITE(LU,11)HXNO,X,Y,SLOPE,ELEV,(COD(I),I=0,2),NPOINTS
0038          +   ,NSLOPES
0039      11      FORMAT(1X,I8,2F10.0,5F8.2,2I5)
0040              RETURN
0041              END
```

```
      FUNCTION BETWEEN(X,X1,X2)
*************************************************************
*     A ROUTINE WHICH JUST DETERMINES                       *
*     WHETHER OR NOT X IS BETWEEN X1 AND X2.                *
*************************************************************
*     INPUTS: X,X1,X2--UTM COORDINATE VALUES                *
*     OUTPUTS: BETWEEN-- A LOGICAL TRUE OR FALSE            *
*     N.B.  ***SINCE X IS REAL AND X1 AND X2 ARE            *
*           ***INTEGER, CLOSE CALLS MAY BE UNRELIABLE       *
*************************************************************
      INTEGER X1,X2
      LOGICAL BETWEEN
      IF((X1-X).GT.0.OR.(X-X2).GT.0)THEN
        BETWEEN=.FALSE.
      ELSE
        BETWEEN=.TRUE.
      ENDIF
      RETURN
      END
```

```
      FUNCTION BETWEEN(X,X1,X2)
*************************************************************
*     A ROUTINE WHICH JUST DETERMINES                       *
*     WHETHER OR NOT X IS BETWEEN X1 AND X2.                *
*************************************************************
*     INPUTS: X,X1,X2--UTM COORDINATE VALUES                *
*     OUTPUTS: BETWEEN-- A LOGICAL TRUE OR FALSE            *
*     N.B.  ***SINCE X IS REAL AND X1 AND X2 ARE            *
*           ***INTEGER, CLOSE CALLS MAY BE UNRELIABLE       *
*************************************************************
      INTEGER X1,X2
      LOGICAL BETWEEN
      IF((X1-X).GT.0.OR.(X-X2).GT.0)THEN
        BETWEEN=.FALSE.
      ELSE
        BETWEEN=.TRUE.
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE CORNERS
*****************************************************************
*     THIS ROUTINE SETS THE COORDINATES OF THE CORNERS OF       *
*     THE TERRAIN BOX BEING PROCESSED.                          *
*****************************************************************
      INCLUDE 'MAXMIN.CMN'
1 ***************************************************************
1 *     THE UTM LIMITS OF THE RECTANGLE OF INTEREST             *
1 ***************************************************************
1     INTEGER XMIN,XMAX,YMIN,YMAX
1     COMMON /MAXMIN/XMIN,XMAX,YMIN,YMAX
1 ***************************************************************
      PRINT*,'ENTER THE SOUTHWEST COORDINATES IN METERS:'
      PRINT*,'EASTING:'
      READ(5,*)XMIN
      PRINT*,'NORTHING:'
      READ(5,*)YMIN
      PRINT*,'NOW ENTER THE NORTHWEST COORDINATES:'
      PRINT*,'EASTING:'
      READ(5,*)XMAX
      PRINT*,'NORTHING:'
      READ(5,*)YMAX
C...  NOW EXPAND THE BOUNDARY SO THAT HEXES WHICH INTERSECT,
C...  BUT ARE NOT CENTERED IN THE AREA OF INTEREST WILL BE
C...  PROCESSED.
      XMIN=XMIN-12500  !ROUGHLY THE RADIUS OF A LEVEL 6 HEX
      XMAX=XMAX+12500
      YMIN=YMIN-12500
      YMAX=YMAX+12500
      RETURN
      END
```

```
0001            SUBROUTINE GATOR(X,Y,HXNO)
0002      ************************************************************
0003      *     (THIS IS THE AGGRE'GATOR')                           *
0004      *     THIS ROUTINE IS DESIGNED TO AGGREGATE                *
0005      *     HEX DATA FROM A VERSION OF THE                       *
0006      *     TERRAIN DISPLAY FILE.                                *
0007      ************************************************************
0008      *     INPUTS: X,Y-- COORDINATES OF THE CENTER OF THE       *
0009      *                   HEX ( IN METERS FROM HEX ORIGIN)       *
0010      *             HXNO-- EXTERNAL HEX NUMBER                   *
0011      ************************************************************
0012            IMPLICIT INTEGER*2 (H-P)
0013            INTEGER*4 HXNO,INTX,INTY,IX,IY
0014            INCLUDE 'CORNER.CMN'
0015    1 ************************************************************
0016    1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE *
0017    1 *     AREA REPRESENTED BY THE DATA IN IBUF.            *
0018    1 ************************************************************
0019    1       INTEGER*4 SWX,SWY
0020    1       COMMON/CORNER/SWX,SWY
0021    1 ************************************************************
0022            INCLUDE 'HEXRAD.CMN'
0023    1       INTEGER*2 DBRES,HEXR
0024    1       COMMON/HEXRAD/DBRES,RAD2,HEXR
0025            INCLUDE 'CODE.CMN'
0026    1 ************************************************************
0027    1 *     ICOD CONTAINS THE COUNTS OF THE RESPECTIVE SURFACE CODES  *
0028    1 *     COD CONTAINS THE RESPECTIVE PERCENTAGES                   *
0029    1 *     SSUM IS THE SUM OF THE ABSOLUTE SLOPES                    *
0030    1 *     ZSUM IS THE SUM OF THE ELEVATIONS                         *
0031    1 *     NSLOPES IS THE NUMBER OF SLOPES COUNTED                   *
0032    1 *     AND NPOINTS IS THE NUMBER OF POINTS COUNTED               *
0033    1 ************************************************************
0034    1       INTEGER*2 ICOD(0:2)
0035    1       DIMENSION COD(0:2)
0036    1       COMMON/CODE/ICOD,COD,SSUM,ZSUM,NSLOPES,NPOINTS
0037    1 ************************************************************
0038      *
0039            NSLOPES=0
0040            NPOINTS=0
0041            ZSUM=0
0042            SSUM=0
0043            DO I=0,2
0044               ICOD(I)=0
0045            ENDDO
0046      *
0047      *     FOR EACH N-S SCAN LINE INTERSECTING THE CIRCLE
0048      *
0049            INTX=NINT(X/DBRES)*DBRES
0050            INTY=NINT(Y/DBRES)*DBRES
0051                DO IX=INTX-HEXR,INTX+HEXR,DBRES
0052                  YDELTA=NINT(SQRT(RAD2-(IX-INTX)**2)/DBRES)*DBRES
0053      *           YDELTA IS THE DISTANCE (ROUNDED TO DBRES)TO THE
0054      *           CIRCUMFERENCE
0055      *           FOR EACH POINT ON THE SCAN LINE,FROM S TO N
0056                  DO IY=INTY-YDELTA,INTY+YDELTA,DBRES
0057                    J=(IX-SWX)/DBRES+1
```

```
                    I=(IY-SWY)/DBRES+1
                    CALL SUMMER(I,J)
                  ENDDO
*                 NEXT POINT
                ENDDO
*               NEXT SCAN
*           NEXT HEX#
            IF(NPOINTS.NE.0)CALL AVG(HXNO,X,Y)
            RETURN
            END
```

```
      PROGRAM HEXER
*****************************************************
*     'HEXISES' THE SURFACE AND ELEVATION DATA      *
*****************************************************
      IMPLICIT INTEGER(H,P)
      LOGICAL*1 BETWEEN
      INCLUDE 'MAXMIN.CMN'
1 ************************************************************
1 *     THE UTM LIMITS OF THE RECTANGLE OF INTEREST          *
1 ************************************************************
1     INTEGER XMIN,XMAX,YMIN,YMAX
1     COMMON /MAXMIN/XMIN,XMAX,YMIN,YMAX
1 ************************************************************
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE *
1 *     AREA REPRESENTED BY THE DATA IN IBUF.            *
1 ***********************************************************
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      DIMENSION HEX6(2401),HEX4(49)
      CALL HEXINIT
      CALL CORNERS
      CALL HEXGEN(HX6,HEX6,10,6)    !GET ALL LEVEL 6 CENTERS
      CALL HEXGEN(HX4,HEX4,6,4)     !GET CENTER LEVEL 6
      DO I=1,HX6
         CALL HEX2XY(HEX6(I),X,Y)
         IF(BETWEEN(X,XMIN,XMAX).AND.BETWEEN(Y,YMIN,YMAX))THEN
           SWX=NINT(X/10000.)*10000-20000
           SWY=NINT(Y/10000.)*10000-20000
           CALL UNGEN
           DO J=1,HX4
              HEX=HXADD(HEX6(I),HEX4(J))
              CALL HEX2XY(HEX,X,Y)
              CALL HEXOUT(HEX,1,H)
              CALL GATOR(X,Y,H)
           ENDDO
         ENDIF
      ENDDO
      PRINT*,'TH-IH-THATS ALL F-FOLKS.'
      END
```

```
0001            SUBROUTINE HEXGEN(HXN,HXNO,LEVTOP,LEVBOT)
0002    ************************************************************
0003    *       THIS ROUTINE GENERATES HEX NUMBERS                 *
0004    *       FROM LEVEL 'LEVTOP' TO 'LEVBOT',                   *
0005    *       NESTED IN HEX FASHION.                             *
0006    ************************************************************
0007    *       INPUTS: LEVTOP,LEVBOT-- THE TOP AND BOTTOM LEVELS  *
0008    *               OF THE HEX TREE TO BE GENERATED            *
0009    *       OUTPUTS:  HXN-- THE NUMBER OF HEXES GENERATED      *
0010    *                 HXNO-- THE ARRAY CONTAINING THE HEX NUMBERS *
0011    ************************************************************
0012            IMPLICIT INTEGER (H,P)
0013            DIMENSION LEVSTOP(4:10)
0014            DIMENSION HXNO(2401)
0015            INTEGER*4 ZERO
0016    C...    THE VARIABLE 'LEVMIN' SHOULD BE (AND PROBABLY IS) SET ELSEWH
0017            LEVMIN=4
0018            ZERO=77777777
0019            L=LEVTOP-LEVBOT
0020            IF(L.EQ.0)THEN
0021              PRINT*,'ERROR IN HEXGEN'
0022              RETURN
0023            ENDIF
0024            HXN=7**(L-1)  !YOU MUST ASK FOR AT LEAST ONE LEVEL
0025 C  D       PRINT*,'HXN',HXN
0026            DO I=4,10
0027               LEVSTOP(I)=0
0028            ENDDO
0029            DO I=LEVBOT,LEVTOP-1
0030               LEVSTOP(I)=6
0031            ENDDO
0032            HXN=0
0033    C...    IF LEVSTOP(LEVEL)=0 THE LOOP IS ONLY EXECUTED ONCE, AND
0034    C...    THE DIGIT SUBTRACTED IS 0,..NO CHANGE
0035            DO LEV10=0,LEVSTOP(10)
0036                 L10=LEV10*1000000
0037                 DO LEV9=0,LEVSTOP(9)
0038                    L9=LEV9*100000
0039                    DO LEV8=0,LEVSTOP(8)
0040                       L8=LEV8*10000
0041                       DO LEV7=0,LEVSTOP(7)
0042                         L7=LEV7*1000
0043                         DO LEV6=0,LEVSTOP(6)
0044                            L6=LEV6*100
0045                            DO LEV5=0,LEVSTOP(5)
0046                               L5=10*LEV5
0047                               DO LEV4=0,LEVSTOP(4)
0048                                  L4=LEV4
0049                                  HXN=HXN+1
0050                                  HXNO(HXN)=ZERO-L10-L9-L8-L7-L6-L5-L4
0051 C  D                             PRINT*,HXN,HXNO(HXN)
0052                               ENDDO
0053                            ENDDO
0054                         ENDDO
0055                       ENDDO
0056                    ENDDO
0057                 ENDDO
```

```
                    ENDDO
C   D         PRINT*,HXN,(HXNO(I),I=1,HXN)
              DO N=1,HXN
                 CALL HEXIN(HXNO(N),1,LEVMIN,HSTOP)
                 HXNO(N)=HSTOR
              ENDDO
              RETURN
              END
```

```
0001            SUBROUTINE SUMMER(I,J)
0002      ***********************************************************
0003      *     SUMS THE ELEVATION,SLOPE,AND FEATURE DATA AT ONE POINT  *
0004      ***********************************************************
0005      *     INPUTS: I,J-- THE ROW AND COLUMN IN THE ARRAY IBUF      *
0006      ***********************************************************
0007            IMPLICIT INTEGER*2 (H-P)
0008            INCLUDE 'HEXRAD.CMN'
0009    1       INTEGER*2 DBRES,HEXR
0010    1       COMMON/HEXRAD/DBRES,RAD2,HEXR
0011            INCLUDE 'CODE.CMN'
0012    1 ***********************************************************
0013    1 *     ICOD CONTAINS THE COUNTS OF THE RESPECTIVE SURFACE CODES  *
0014    1 *     COD CONTAINS THE RESPECTIVE PERCENTAGES                   *
0015    1 *     SSUM IS THE SUM OF THE ABSOLUTE SLOPES                    *
0016    1 *     ZSUM IS THE SUM OF THE ELEVATIONS                         *
0017    1 *     NSLOPES IS THE NUMBER OF SLOPES COUNTED                   *
0018    1 *     AND NPOINTS IS THE NUMBER OF POINTS COUNTED               *
0019    1 ***********************************************************
0020    1       INTEGER*2 ICOD(0:2)
0021    1       DIMENSION COD(0:2)
0022    1       COMMON/CODE/ICOD,COD,SSUM,ZSUM,NSLOPES,NPOINTS
0023    1 ***********************************************************
0024      C...  COMPUTE THE ABSOLUTE DIFFERENCE IN
0025      C...  ELEVATIONS BETWEEN THIS POINT AND THE
0026      C...  POINTS ADJACENT TO THE N AND E
0027      C...  ADD TO THE CUMULATIVE SUMS OF DIFFERENCES
0028      C...  AND ELEVATIONS,RESPECTIVELY
0029            Z1=IELV(I,J)
0030            IF(Z1.LE.0.OR.Z1.GE.4000)RETURN
0031      C...  INCREASE THE POINT COUNT
0032            NPOINTS=NPOINTS+1
0033      C
0034      C...  INCREASE THE FEATURE CODE TALLIES
0035            IC=ICODE(I,J)
0036            IF(IC.NE.1.AND.IC.NE.2) IC=0
0037            ICOD(IC)=ICOD(IC)+1
0038      C
0039            ZSUM=ZSUM+Z1
0040            IF(J+1.GT.400)GOTO30    !400 COLUMNS IN THE ARRAY
0041            Z2=IELV(I,J+1)
0042            IF(Z2.LE.0.OR.Z2.GE.4000)GOTO30
0043            NSLOPES=NSLOPES+1
0044            SSUM=SSUM+ABS(Z1-Z2)
0045      30    CONTINUE
0046            IF(I+1.GT.400)RETURN    !400 ROWS IN THE ARRAY
0047            Z3=IELV(I+1,J)
0048            IF(Z3.LE.0.OR.Z3.GE.4000)RETURN
0049            NSLOPES=NSLOPES+1
0050            SSUM=SSUM+ABS(Z1-Z3)
0051            RETURN
0052            END
```

HYDROHEXER CALLING SEQUENCE

```
HYDROHEXER
  ├──HEXINIT*
  ├──OPENERS
  ├──GRIDR*
  └──NODEHEX
        ├──GETNDX*
        ├──GETREC
        └──LINKHEX
              ├──GETNDX*
              └──GETREC*
              ──GETSUB*
              ──TRANX*
              ├──TRANY*
              ──HEXSIDE
              └──HXRECORD
                    ├──HEXREAD*
                    └──PACKER*
```

*TERRAIN SYSTEM UTILITIES

Figure B-9.

PROGRAM: HYDROHEXER

ROUTINE \ COMMON	CENTER	HEX	HEXRAD	LNKNOD	PACK	SUB	TRAN
GETSUB				X		X	
GRIDR				X			
HEXINIT		X	X				
HEXSIDE							X
HXRECORD					X		
HYDROHEXR	X			X			X
LINKHEX	X			X	X	X	
NODEHEX				X		X	
PACKER					X		

Figure B-10

```
      SUBROUTINE HEXSIDE(X,Y,HS,HN,W,Z)
*************************************************************
*     THIS SUBROUTINE DETERMINES THE HEX NUMBER             *
*     THAT ANY X,Y COORDINATES LIES WITHIN. THEN            *
*     IT DETERMINES THE SIDE OF THE HEX THAT IS             *
*     NEAREST THE COORDINATES.                              *
*************************************************************
*     INPUTS:                                               *
*            X,Y-- EASTING,NORTHING IN METERS               *
*     OUTPUTS:                                              *
*            HS-- HEX SIDE                                  *
*            HN-- HEX NUMBER                                *
*            W,Z-- COORDINATES OF CENTER OF HEX IN          *
*                  METERS FROM CENTER OF HEX ORIGIN         *
*************************************************************
*     AT=HORIZONTAL DISTANCE OF THE POINT FROM THE          *
*        HEX CENTER.                                        *
*     BT=VERTICAL DISTANCE OF THE POINT FROM THE            *
*        HEX CENTER.                                        *
*     A AND B ARE THE COORDINATES OF THE POINT RELATIVE     *
*     TO A ROTATION OF THE COORDINATE SYSTEM.               *
*     C=THE COSINE OF THE ANGLE OF ROTATION.                *
*     S=THE SINE OF THE ANGLE OF ROTATION.                  *
*     D=TAN(60)*A=SQRT(3)*A                                 *
*       THE SIDE OF THE HEX THE POINT IS NEAREST CAN        *
*     BE DETERMINED BY ITS ANGLE FROM THE HORIZONTAL        *
*     AXIS.                                                 *
*     SIDE3:0-60  SIDE1:60-120  SIDE5:120-180               *
*     SIDE4:180-240  SIDE6:240-300  SIDE2:300-360           *
*       THE POINT IS WITHIN 60 DEGREES OF THE               *
*     HORIZONTAL AXIS IF ABS(B/A)<TAN(60)=SQRT(3).          *
*     THEREFORE:                                            *
*     IF 0<B<D, THEN 0-60 DEGREES.                          *
*     IF 0<B<-D, THEN 120-180 DEGREES.                      *
*     IF 0<B AND NEITHER OF ABOVE, THEN 60-120 DEGREES.     *
*     IF D<B<0, THEN 180-240 DEGREES.                       *
*     IF -D<B<0, THEN 300-360 DEGREES.                      *
*     IF B<0 AND NEITHER OF ABOVE, THEN 240-300 DEGREES.    *
*************************************************************
      IMPLICIT INTEGER (H,P)
      INCLUDE 'TRAN.CMN'
1     COMMON/TRAN/S,C,L,SQ3
      CALL XYL2HA(X,Y,L,HN)
      CALL HA2XYL(HN,W,Z,L)
      AT=X-W
      BT=Y-Z
      A=C*AT+S*BT
      B=C*BT-S*AT
      D=SQ3*A
      IF(B.GE.0) THEN
        IF(B.LE.D) THEN
          HS=3
          ELSE IF(B.LE.-D) THEN
          HS=5
          ELSE
          HS=1
        END IF
```

```
0058            ELSE IF(B.GE.-D) THEN
0059            HS=2
0060            ELSE IF(B.GE.D) THEN
0061            HS=4
0062            ELSE
0063            HS=6
0064          END IF
0065        RETURN
0066        END
```

```
0001            SUBROUTINE HXRECORD(HA,HSIDE,IERR)
0002      ***********************************************************
0003      *     RECORDS THE HYDRO CODE 'LTYPE' AT SIDE 'HSIDE' OF   *
0004      *     THE HEX 'HA'.  THEN IT FINDS THE ADJACENT HEX AND   *
0005      *     THE CORRESPONDING SIDE AND RE-RECORDS THE INFO.     *
0006      ***********************************************************
0007      *     INPUT:  HA--THE HEX ADDRESS OF THE FIRST SUBNODE    *
0008      *             HSIDE-- THE SIDE OF HEX 'HA' AT WHICH THE   *
0009      *                     HYDRO CODE IS BEING RECORDED        *
0010      *     OUTPUT:  IERR-- AN ERROR FLAG                       *
0011      ***********************************************************
0012            IMPLICIT INTEGER(H,P)
0013            INCLUDE 'UTIL:PACK.CMN'
0014    1 ***********************************************************
0015    1       INTEGER*4 SIDES ! PACKED CONNECTIVITIES             *
0016    1       INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE     *
0017    1       COMMON/PACK/SIDES,LTYPE
0018    1 ***********************************************************
0019            LU=7  ! LOGICAL UNIT FOR HEX FILE
0020      *     PUT THE SIDE INDICATOR INTO INTERNAL HEX FORMAT
0021            CALL HEXIN(HSIDE,1,4,HSTORA)
0022      *     FIND THE ADJACENT HEX BY ADDITION
0023            HB=HXADD(HA,HSTORA)
0024      *     FIND THE INVERSE OF THE SIDE
0025            HSTORB=HXINV(HSTORA)
0026      *
0027      *     GET THE RECORD FOR HEX 'HA' OR CAUSE IT TO
0028      *     BE INITIALIZED IF NECESSARY
0029            CALL HEXREAD(HA,SIDES,LU)
0030      *
0031            CALL PACKER(HA,HSTORA,LU,IERR)
0032            IF (IERR.EQ.0)RETURN
0033      *
0034      *     NOW FOR THE SECOND HEX
0035            CALL HEXREAD(HB,SIDES,LU)
0036            CALL PACKER(HB,HSTORB,LU,IERR)
0037            RETURN
0038            END
```

```
      PROGRAM HYDROHEXR
*****************************************************
*     THE SET OF ROUTINES USED TO 'HEXISE' THE      *
*     BDM-PRODUCED GERMAN HYDROGRAPHY DATA.         *
*****************************************************
      IMPLICIT INTEGER(H,P)
      INTEGER*4 XO,YO
      INCLUDE 'TRAN.CMN'
      COMMON/TRAN/S,C,L,SQ3
      INCLUDE 'UTIL:LNKNOD.CMN'
*****************************************************************
*     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES           *
*****************************************************************
      INTEGER*4 GRID,NODREC,LNKREC,SUBREC
      COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
     +,SUBREC(500)
*****************************************************************
      INCLUDE 'UTIL:CENTER.CMN'
*******************************************************
*     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN *
*     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE *
*     TO A GIVEN GRID ZONE.                            *
*******************************************************
      INTEGER*4 XORIGIN,YORIGIN
      COMMON/CENTER/XORIGIN,YORIGIN
      DATA XORIGIN/500000/,YORIGIN/5700000/
*******************************************************
*     INITIALIZE THE HEX PARAMETERS
      CALL HEXINIT
*     OPEN THE LOC AND HEX FILES                        *
*     INITIALIZING VARIABLES FOR USE IN  HEXSIDE        *
      SQ3=SQRT(3.)
      REALEV=LEVMIN
      ANGRAD=REALEV*19.11*3.14159/180.
      S=SIN(ANGRAD)
      C=COS(ANGRAD)
      L=4
*     READ IN THE GRID POINTERS
      CALL GRIDR
      DO J=1,128
         DO I=1,128
            NODE=GRID(I,J)
            IF(NODE.NE.0)CALL NODEHEX(NODE)
         ENDDO
      ENDDO
*
      END
```

```
      SUBROUTINE LINKHEX(NODX,NODY,LNKNUM)
*****************************************************
*     THIS ROUTINE PROCESSES ONE 'OUTLINK',         *
*     CAUSING EACH SUBNODE TO BE RECORDED AS A      *
*     HYDRO CODE AT THE CLOSEST SIDE OF THE HEX     *
*     WHICH CONTAINS THAT SUBNODE.                  *
*****************************************************
*     INPUT:                                        *
*             LNKNUM--THE LINK NUMBER OF THE FIRST  *
*                     LINK INCIDENT TO THIS NODE    *
*             NODX,NODY-- EASTING,NORTHING IN METERS *
*                         OF THE START NODE         *
*****************************************************
      IMPLICIT INTEGER (H,P)
      INTEGER*2 TMPLINK,TERMX,TERMY,NODX,NODY
      INTEGER*2 X(0:100),Y(0:100)
      INTEGER*4 WRDPOS,SRECNUM,SUBWRD,TERMXY
      INCLUDE 'UTIL:SUB.CMN'
1 *********************************************************
1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
1 *********************************************************
1     INTEGER*2 SUBX(100),SUBY(100)
1     COMMON/SUB/ SUBX,SUBY
1 ******************************************************
      INCLUDE 'UTIL:PACK.CMN'
1 ******************************************************
1     INTEGER*4 SIDES ! PACKED CONNECTIVITIES          *
1     INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE  *
1     COMMON/PACK/SIDES,LTYPE
1 ******************************************************
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES      *
1 ************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1   + ,SUBREC(500)
1 ***************************************************************
      INCLUDE 'UTIL:CENTER.CMN'
1 ******************************************************
1 *     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN *
1 *     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE *
1 *     TO A GIVEN GRID ZONE.                            *
1 ******************************************************
1     INTEGER*4 XORIGIN,YORIGIN
1     COMMON/CENTER/XORIGIN,YORIGIN
1     DATA XORIGIN/500000/,YORIGIN/5700000/
1 ******************************************************
*
      X(0)=NODX
      Y(0)=NODY
      NXTLNK=LNKNUM
      LU=3
      DO WHILE(NXTLNK.NE.0)
         CALL GETNDX(NXTLNK,LRECNUM,WRDPOS)
         CALL GETREC(LU,LRECNUM,LNKREC)
```

```
0058    *           SET THE TYPE
0059                N=WRDPOS
0060    *
0061    *            SET LINK TYPE                          *
0062                 TMPLINK=LIBSEXTZV(0,16,LNKREC(3,N))
0063                 CALL LIBSINSV(TMPLINK,0,16,LTYPE)
0064    *
0065                  LTYPE=15-LTYPE
0066                  IF(LTYPE.LT.0.OR.LTYPE.GT.3)LTYPE=0
0067    *             EXTRACT THE TERMINAL COORDINATES    *
0068                  TERMXY=LNKREC(4,WRDPOS)
0069    *             THE X .......
0070                  TMP=LIBSEXTZV(0,16,TERMXY)
0071                  CALL LIBSINSV(TMP,0,16,TERMX)
0072    *             ..... AND THEN THE Y                *
0073                  TMP=LIBSEXTZV(16,16,TERMXY)
0074                  CALL LIBSINSV(TMP,0,16,TERMY)
0075    *
0076    *            SET SUBNODE RECORD POINTER           *
0077                 TMPLINK=LIBSEXTZV(0,16,LNKREC(5,N))
0078                 CALL LIBSINSV(TMPLINK,0,16,SRECNUM)
0079    *            SET SUBNODE WORD POINTER             *
0080                 TMPLINK=LIBSEXTZV(16,16,LNKREC(5,N))
0081                 CALL LIBSINSV(TMPLINK,0,16,SUBWRD)
0082    *
0083    *                IF THERE ARE SUBNODES            *
0084                     IF(SRECNUM.NE.0)THEN
0085    *                   GET THE SUBNODE LIST
0086                        CALL GETSUB(SRECNUM,SUBWRD,NUM)
0087    *                   PROCESS THE SUBNODE LIST     *
0088                        DO I=1,NUM
0089                           X(I)=SUBX(I)
0090                           Y(I)=SUBY(I)
0091                        ENDDO
0092                        X(NUM+1)=TERMX
0093                        Y(NUM+1)=TERMY
0094                        DO I=0,NUM+1
0095                           X1=TRANX(X(I))-XORIGIN
0096                           Y1=TRANY(Y(I))-YORIGIN
0097                           CALL HEXSIDE(X1,Y1,HS,HN,W,Z)
0098                           CALL HXRECORD(HN,HS,IADJ)
0099                        ENDDO
0100                     ENDIF
0101    *           GET THE NEXT LINK RECORD
0102                NXTLNK=LNKREC(2,WRDPOS)
0103             ENDDO
0104             RETURN
0105             END
```

```
      SUBROUTINE NODEHEX(NODE)
*****************************************
*     THIS ROUTINE EXTRACTS THE I*4 WORD  *
*     WHICH CONTAINS THE NODE CORDINATES  *
*     AND THE RECORD # OF THE FIRST LINK  *
*     WHICH IS INCIDENT TO THE NODE       *
*****************************************
*     INPUT:  NODE-- THE HEAD NODE FOR A  *
*                    GRID RECORD FROM THE *
*                    LOC DATA BASE        *
*****************************************
      IMPLICIT INTEGER (H,P)
      INTEGER*4 WRDPOS,TMP
      INTEGER*2 NODX,NODY
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ***************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
1 ***************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1   + ,SUBREC(500)
1 ****************************************************************
      INCLUDE 'UTIL:SUB.CMN'
1 ***********************************************************
1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
1 ***********************************************************
1     INTEGER*2 SUBX(100),SUBY(100)
1     COMMON/SUB/ SUBX,SUBY
1 ***********************************************************
      DO WHILE (NODE.NE.0)
         CALL GETNDX(NODE,NRECNUM,WRDPOS)
         LU=2
         CALL GETREC(LU,NRECNUM,NODREC)
*
         NODXY=NODREC(5,WRDPOS)
*        SET THE X COORDINATE        *
         TMP=LIBSEXTZV(0,16,NODXY)
         CALL LIBSINSV(TMP,0,16,NODX)
*        SET THE Y COORDINATE        *
         TMP=LIBSEXTZV(16,16,NODXY)
         CALL LIBSINSV(TMP,0,16,NODY)
*
         LNK=NODREC(4,WRDPOS)
         CALL LINKHEX(NODX,NODY,LNK)
*        GET NEXT NODE
         NODE=NODREC(1,WRDPOS)
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE OPENERS
*****************************************************
*     THIS ROUTINE SIMPLY OPENS THE GRID,NODE,LINK,AND *
*     SUBNODE FILES, AND THE ISAM FILE WHICH CONTAINS *
*     THE 'HEXISED' LOC OR HYDRO DATA                 *
*****************************************************
      OPEN(UNIT=1,NAME='GRID',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=2,NAME='NODE',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=3,NAME='ROAD',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=4,NAME='SUBN',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
*
*     NOW FOR THE ISAM FILE
      OPEN(UNIT=7,NAME='HEXROAD',STATUS='UNKNOWN',
     + ORGANIZATION='INDEXED',ACCESS='KEYED',RECL=2,
     + RECORDTYPE='FIXED',FORM='UNFORMATTED',
     + KEY=(1:4:INTEGER))
      RETURN
      END
```

PDBPACK CALLING SEQUENCE

```
PDBPACK
 ├──SETCOLOR**
 ├──SETGEO*
 ├──BLOCKIN
 │    ├──MAPZUTM*
 │    └──UNGEN*
 ├──MAPDRAW
 │    ├──FILLUP*
 │    ├──FEATURES*
 │    └──GRIDS*
 ├──LABEL*
 ├──PATCHIT**
      ├──POLYDEF**
      ├──FILLUP*
      ├──FEATURES*
      └──GEN*
```

*TERRAIN SYSTEMS UTILITIES

**"SURFACE" ROUTINES

Figure B-11

PROGRAM: PDBPACK

ROUTINE \ COMMON	CMERID	CORNER	MAP	WINDE	ZDBPRO
BLOCKIN	X	X		X	
FILLUP		X	X	X	
GEN		X			
GRIDS				X	
LABEL				X	
MAPDRAW		X		X	
PATCHIT		X			
SETGEO	X				X
UNGEN		X			

Figure B-12

```
0001              SUBROUTINE BLOCKIN(MAP)
0002      ***********************************************************
0003      *       THIS ROUTINE DETERMINES WHICH 10KM BLOCKS ARE     *
0004      *       NEEDED TO COVER A MAP AND CAUSES THESE TO BE      *
0005      *       READ IN.                                          *
0006      ***********************************************************
0007      *       INPUTS: MAP-- THE NAME OF THE MAP TO BE READ IN   *
0008      *       OUTPUTS: NONE                                     *
0009      ***********************************************************
0010              INCLUDE 'CORNER.CMN'
0011    1 ************************************************************
0012    1 *       SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE  *
0013    1 *       AREA IN THE ARRAY IBUF.                           *
0014    1         INTEGER*4 SWX,SWY
0015    1         COMMON/CORNER/SWX,SWY
0016    1 ************************************************************
0017              INCLUDE 'CMERID.CMN'
0018    1 ***************************************************
0019    1         REAL*8 CMERID
0020    1         REAL P_RAD
0021    1         COMMON/CMERID/CMERID,P_RAD
0022    1 ***************************************************
0023              INCLUDE '[TERRAIN.SURFACE]WINDO.CMN'
0024    1 ***************************************************************
0025    1 *       FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND   *
0026    1 *       MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER    *
0027    1 *       TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS  *
0028    1 *       THE CONTOUR INTERVAL.                                 *
0029    1 ***************************************************************
0030    1         DIMENSION FWINXY(4)
0031    1         COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
0032    1 ***************************************************************
0033              CHARACTER*5 MAP
0034              CHARACTER*7 UTMSW,UTMNE
0035
0036              CALL MAP2UTM(MAP,FEAST,FNORTH,CMERID)
0037              IEAST=NINT(FEAST/10000.)*10000
0038              NORTH=NINT(FNORTH/10000.)*10000
0039 C   D        PRINT*,IEAST,NORTH
0040              SWX=IEAST-20000 ! MAP2UTM RETURNS THE CENTER
0041              SWY=NORTH-20000  !AND A 40KM SQUARE IS NEEDED
0042              CALL UNGEN       !UNGEN CAUSES THE DATA TO BE READ IN
0043              FWINXY(1)=SWX
0044              FWINXY(2)=FWINXY(1)+40000
0045              FWINXY(3)=SWY
0046              FWINXY(4)=FWINXY(3)+40000
0047 C   D        PRINT*,FWINXY
0048 C   D        READ*,JUNK
0049              RETURN
0050              END
```

```
      SUBROUTINE MAPDRAW(MAP,ERR)
*****************************************************
*     THIS ROUTINE DISPLAYS AND FILLS THE DATA FROM  *
*     ONE MAPSHEET.                                  *
*****************************************************
*     INPUTS: MAP-- THE NAME OF THE MAP IN THE M745  *
*                   SERIES                           *
*     OUTPUTS: ERR-- AN ERROR FLAG                   *
*****************************************************
      LOGICAL*1 ERR
      DIMENSION PX(500),PY(500)
      DIMENSION POLY(500,2)
      CHARACTER*5 FNAME(2)
      CHARACTER*5 MAP
      CHARACTER*1 PREFIX(2)
      INTEGER*2 ICLR
      INCLUDE '[TERRAIN.SURFACE]WINDO.CMN'
1 ************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND  *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER   *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS *
1 *     THE CONTOUR INTERVAL.                                *
1 ************************************************************
1     DIMENSION FWINXY(4)
1     COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ************************************************************
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE *
1 *     AREA IN THE ARRAY IBUF.                          *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      DATA PREFIX/'F','U'/
      EQUIVALENCE (POLY(1,1),PX(1))
      EQUIVALENCE (POLY(1,2),PY(1))
      LU=2
      DO ICLR=1,2
        MAP(1:1)=PREFIX(ICLR)
        OPEN(NAME=MAP,UNIT=LU,STATUS='OLD',FORM='UNFORMATTED',ERR=
        X=1     !JUST TO CAUSE A READ TO EOF
        DO WHILE (X.EQ.1)
           READ(LU,END=100)N,(PX(K),PY(K),K=1,N)
           DO J=1,N
              PX(J)=PX(J)-SWX
              PY(J)=PY(J)-SWY
           ENDDO
           CALL FILLUP(N,POLY,ICLR)

        ENDDO
100     CLOSE(UNIT=LU)
      ENDDO
C...  FILLUP CHANGES THE WINDOW SETTING,SO...
      FWINXY(1)=SWX
      FWINXY(2)=SWX+40000
      FWINXY(3)=SWY
      FWINXY(4)=SWY+40000
```

MAPDRAW

```
0058              INCR=2
0059              CALL FEATURES(INCR)
0060              CALL LINCLR(4)
0061              CALL GRIDS
0062              RETURN
0063      200     CALL CMCLOS
0064              ERR=.TRUE.
0065              CLOSE(UNIT=LU)
0066              PRINT*,'ERROR IN MAP: ',MAP
0067              RETURN
0068              END
```

```
      PROGRAM PDBPACK
*************************************************************
*     THIS ROUTINE PACKS THE POLYGONAL AREAL DATA FROM *
*     PDBREAD INTO THE 10KM TERRAIN FILES              *
*************************************************************
      CHARACTER*5 NAM
      LOGICAL*1 ERR
      DATA ERR/.FALSE./
      CALL SETCOLOR
      CALL SETGEO
      DO WHILE (.NOT.ERR)
        PRINT*,'MAP?'
10      FORMAT(A5)
        READ(5,10)NAM
        CALL BLOCKIN(NAM)
        CALL MAPDRAW(NAM,ERR)
        CALL CMOPEN
        CALL NEWPAG
        CALL LABEL
        CALL PATCHIT
      ENDDO
      END
```

PDBREAD CALLING SEQUENCE

```
PDBREAD
 |
 |---SETGEO*
 |
 |---NAMEREAD
 |
 |---FILEREAD
 |
 \---PDB2POLY
      |
      |---CALIBR8
      |    |
      |    |---MOVR
      |    |    |
      |    |    \---PARSER
      |    |
      |    |---DRAWR
      |    |    |
      |    |    |---PARSER
      |    |    |
      |    |    \---POLYWRITE
      |    |                 |
      |    \---MAP2LL        \---DELTA
      |
      |---MOVR
      |    |
      |    \---PARSER
      |
      \---DRAWR
           |
           |---PARSER
           |
           \---POLYWRITE
                     |
                     \---DELTA
```

*Terrain System Utility

Figure B-13

PROGRAM: PDBREAD

ROUTINE \ COMMON	CMERID	MAPDAT	WORDS	ZDBPRO														
CALIBR8		X	X															
DRAWR		X	X															
FILEREAD			X															
MOVR		X	X															
PARSER			X															
PDB2POLY		X	X															
PDBREAD			X															
POLYWRITE	X	X																
SETGEO	X			X														

Figure B-14

```
0001            SUBROUTINE CALIBR8(FNAME,FDM,ERR)
0002      ***********************************************************
0003      *     THIS SUBROUTINE CHECKS THE HEADER FDM, THE SKEW-    *
0004      *     CORRECTION FDM, AND SETS THE X- AND Y-CALIBRATION   *
0005      ***********************************************************
0006      *     INPUTS: FNAME-- THE NAME OF AN M745-SERIES MAP,     *
0007      *             WITH A 'U' OR 'F' (URBAN OR FOREST) AS THE *
0008      *             FIRST CHARACTER.                            *
0009      *     OUTPUTS: FDM-- FUNCTION DEFINITION MODULE TYPE      *
0010      ***********************************************************
0011            LOGICAL ERR
0012            CHARACTER*5 FNAME
0013           -DIMENSION XX(4),YY(4)
0014            INCLUDE 'MAPDAT.CMN'
0015    1 ***********************************************************
0016    1       REAL X,Y      !CURSOR POSITION
0017    1       REAL XCENTR,YCENTR !UTM CENTER OF MAP
0018    1       REAL XSCALE,YSCALE !METERS/TABLET UNIT
0019    1 ***********************************************************
0020    1       COMMON/MAPDAT/X,Y,XCNTR,YCNTR,XSCALE,YSCALE
0021    1 ***********************************************************
0022    1
0023    1
0024            INCLUDE 'WORDS.CMN'
0025    1 ***********************************************************
0026    1       INTEGER*2 PTR          !POINTS TO CURRENT BYTE
0027    1       INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
0028    1       BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
0029    1       INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
0030    1       EQUIVALENCE (WORDS(1),BYTES(1))
0031    1       COMMON/WORDS/BYTES,N_BYTES,PTR
0032    1 ***********************************************************
0033    1
0034            INCLUDE 'FDM.PAR'
0035    1 ***********************************************************
0036    1       INTEGER*2 FDM,L_R_M,L_R_D,S_R_M,S_R_D
0037    1       PARAMETER(
0038    1     + HEADER=16,          !CODE FOR THE HEADER MODULE
0039    1     + L_R_M= 28,          !LONG RELATIVE MOVE
0040    1     + L_R_D= 29,          !LONG RELATIVE DRAW
0041    1     + S_R_M= 32,          !SHORT RELATIVE MOVE
0042    1     + S_R_D= 33)          !SHORT RELATIVE DRAW
0043    1 ***********************************************************
0044      C...  CHECK HEADER FDM
0045            IF(BYTES(2).NE.HEADER.OR.BYTES(1).NE.14)THEN
0046              WRITE(3,1) FNAME,BYTES(2),BYTES(1)
0047      1       FORMAT(1X,'BAD HEADER FDM IN : ',A5,2I2)
0048              ERR=.TRUE.
0049              RETURN
0050            ENDIF
0051      C...  RESETTING THE CURSOR POSITIONS.
0052            X=0
0053            Y=0
0054
0055      C...  CHECK FOR SKEW CORRECTION MOVES
0056            PTR=16
0057            FDM=BYTES(PTR)
```

```
0058              IF(FDM.EQ.L_R_M)THEN
0059                PTR=17
0060                CALL MOVR(FDM,ERR)
0061                IF(ERR)RETURN
0062              ELSE
0063                ERR=.TRUE.
0064                RETURN
0065              ENDIF
0066
0067      C...    CHECK CALIBRATION FDM
0068      C...    SETTING THE SCALE TO 1 STOPS THE INFO FROM
0069      C...    BEING WRITTEN BY POLYWRITE
0070              XSCALE=1
0071              YSCALE=1
0072              IF(FDM.EQ.L_R_D)THEN
0073                CALL DRAWR(FDM,ERR)
0074              ELSE
0075                WRITE(3,*)'BAD CALIBRATION'
0076                ERR=.TRUE.
0077                RETURN
0078              ENDIF
0079
0080      C...    ORIGINALLY THE CALIBRATIONS WERE SET FROM THE
0081      C...    DIGITIZED 10KM LINES
0082      C...    RESET THE X- AND Y-SCALES
0083      C...    THE MAPSHEETS ARE 20 X 12 MINUTES, AND THE
0084      C...    DIGITIZER TABLET WINDOW IS 10000 X 7500, SO...
0085              XSCALE=.000033333
0086              YSCALE=.000026667
0087              ERR=.FALSE.
0088      C...    SET UTM CENTER OF MAP
0089              CALL MAP2LL(FNAME,XCNTR,YCNTR)
0090 C    D       PRINT*,'CNTR IN CALIBR8: ',XCNTR,YCNTR
0091              RETURN
0092              END
```

```
      FUNCTION DELTA(X1,Y1,X2,Y2)
*****************************************************************
*     THIS FUNCTION COMPUTES THE DISTANCE BETWEEN THE POINTS    *
*     (X1,Y2) AND (X2,Y2).                                      *
*****************************************************************
      D=(X1-X2)**2+(Y1-Y2)**2
      DELTA=SQRT(D)
      RETURN
      END
```

```
      SUBROUTINE DRAWR(FD ,ERR)
*************************************************************
*     PUTS THE PARSED MOVES INTO POLY AND CALLS POLYWRITE *
*     IF THE NEXT FDM INDICATES A MOVE.                   *
*************************************************************
      LOGICAL ERR
      REAL POLY(1000,2)
      INTEGER*2 P,ARRAY(1200),W_LEN
      INCLUDE 'MAPDAT.CMN'
1 *************************************************************
1     REAL X,Y      !CURSOR POSITION
1     REAL XCENTR,YCENTR !UTM CENTER OF MAP
1     REAL XSCALE,YSCALE !METERS/TABLET UNIT
1 *************************************************************
1     COMMON/MAPDAT/X,Y,XCNTR,YCNTR,XSCALE,YSCALE
1 *************************************************************
1
1
      INCLUDE 'WORDS.CMN'
1 *************************************************************
1     INTEGER*2 PTR          !POINTS TO CURRENT BYTE
1     INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
1     BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
1     INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
1     EQUIVALENCE (WORDS(1),BYTES(1))
1     COMMON/WORDS/BYTES,N_BYTES,PTR
1 *************************************************************
1
      INCLUDE 'FDM.PAR'
1 *************************************************************
1     INTEGER*2 FDM,L_R_M,L_R_D,S_R_M,S_R_D
1     PARAMETER(
1   +  HEADER=16,          !CODE FOR THE HEADER MODULE
1   +  L_R_M= 28,          !LONG RELATIVE MOVE
1   +  L_R_D= 29,          !LONG RELATIVE DRAW
1   +  S_R_M= 32,          !SHORT RELATIVE MOVE
1   +  S_R_D= 33)          !SHORT RELATIVE DRAW
1 *************************************************************
      DATA P/1/
      ERR=.FALSE.
      IP=(PTR-1)/2
      LEN=ISHFT(ISHFT(WORDS(IP),8),-8)-2  !2-BYTE HEADERS
      W_LEN=30/FDM+1          !OR 29 OR 31 OR 32...
      LEN=LEN/W_LEN
      CALL PARSER(ARRAY,W_LEN,LEN,ERR)
      IF(ERR)RETURN
      DO I=1,LEN,2
         POLY(P,1)=X
         POLY(P,2)=Y
         X=X+ARRAY(I)
         Y=Y+ARRAY(I+1)
         P=P+1
      ENDDO
       POLY(P,1)=X   !THE END OF THE LAST DRAW
       POLY(P,2)=Y
      FDM=BYTES(PTR)
      PTR=PTR+1
```

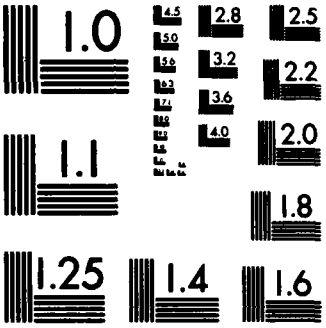
DRAWR

```
C...    IF THE NEXT FDM IS A MOVE OR IF THIS IS THE END OF THE
C...    INPUT FILE THEN WRITE THE CURRENT POLYGON.
        IF(FDM.EQ.L_R_M.OR.FDM.EQ.S_R_M.OR.PTR.GE.N_BYTES)THEN
          CALL POLYWRITE(POLY,P)
          P=1
        ENDIF
        RETURN
        END
```

```
0001
0002            SUBROUTINE FILEREAD(ERR)
0003      ***************************************************************
0004      *     READS A PICTURE DATA BASE (PDB) FILE WHICH HAS BEEN     *
0005      *     TRANSFERRED FROM THE TEK 4081 AND PUTS IT INTO 'BYTES'  *
0006      ***************************************************************
0007            LOGICAL ERR
0008            INCLUDE 'WORDS.CMN'
0009    1 ************************************************************
0010    1       INTEGER*2 PTR          !POINTS TO CURRENT BYTE
0011    1       INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
0012    1       BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
0013    1       INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
0014    1       EQUIVALENCE (WORDS(1),BYTES(1))
0015    1       COMMON/WORDS/BYTES,N_BYTES,PTR
0016    1 ************************************************************
0017    1
0018            ERR=.FALSE.
0019            N_BYTES=0
0020            DO WHILE(.NOT.ERR)
0021               READ(1,10,END=100,ERR=200)N,(BYTES(I),I=N_BYTES+1,N_BYTE
0022               N_BYTES=N_BYTES+N
0023            ENDDO
0024      10    FORMAT(Q,256A1)
0025      100   RETURN
0026      200   ERR=.TRUE.
0027            RETURN
0028            END
```

AD-A132 940 CORDIVEM EUROPEAN TERRAIN DATA(U) COMBINED ARMS OPERATIONS RESEARCH ACTIVITY FORT LEAVENWORTH KS W J THOMSON 1983 CAORA/TP-2-83 2/3

UNCLASSIFIED F/G 15/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      SUBROUTINE MAP2LL(FNAME,FLON,FLAT)
*************************************************
*     THIS ROUTINE COMPUTES THE LAT,LON OF THE *
*     CENTER OF A SHEET FROM THE M745 SERIES.  *
*************************************************
*     INPUTS: FNAME-- THE NAME OF THE MAP      *
*     OUTPUTS: FLON,FLAT-- REAL-VALUED LAT AND *
*              LON OF THE CENTER OF THE MAP    *
*************************************************
      CHARACTER*5 FNAME
      DIMENSION D(2)
      PARAMETER PI=3.141592654
      P_RAD=PI/180.

      DO I=2,4,2
         DECODE(2,10,FNAME(I:I+1)) D(I/2)
      ENDDO
10    FORMAT(F3.0)

      DLAT=50+(59-D(1))/2.*.2+.1
      DLON=9+(D(2)-20)/2./3.+1./6.
      FLAT=DLAT*P_RAD
      FLON=DLON*P_RAD
      RETURN
      END
```

```
0001            SUBROUTINE MOVR(FDM,ERR)
0002      ************************************************************
0003      *     RECOMPUTES THE CURSOR POSITION AND INDEX AFTER A     *
0004      *     RELATIVE MOVE HAS BEEN PARSED.                       *
0005      ************************************************************
0006      *     INPUTS: FDM-- THE CODE FOR THE FDM TYPE              *
0007      *     OUTPUTS: ERR-- ERROR FLAG                            *
0008      ************************************************************
0009            LOGICAL ERR
0010            INTEGER*2 FDM,W_LEN,LEN,ARRAY(20)!MOVES SHOULD BE SHORT
0011            INCLUDE 'MAPDAT.CMN'
0012    1 ***************************************************************
0013    1       REAL X,Y      !CURSOR POSITION
0014    1       REAL XCENTR,YCENTR !UTM CENTER OF MAP
0015    1       REAL XSCALE,YSCALE !METERS/TABLET UNIT
0016    1 ***************************************************************
0017    1       COMMON/MAPDAT/X,Y,XCNTR,YCNTR,XSCALE,YSCALE
0018    1 ***************************************************************
0019    1
0020    1
0021            INCLUDE 'WORDS.CMN'
0022    1 ***************************************************************
0023    1       INTEGER*2 PTR          !POINTS TO CURRENT BYTE
0024    1       INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
0025    1       BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
0026    1       INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
0027    1       EQUIVALENCE (WORDS(1),BYTES(1))
0028    1       COMMON/WORDS/BYTES,N_BYTES,PTR
0029    1 ***************************************************************
0030    1
0031            ERR=.FALSE.
0032            LEN=BYTES(PTR-2)-2  !2-BYTE HEADERS
0033            W_LEN=30/FDM+1     !OR 29,OR 31 OR 32...
0034            LEN=LEN/W_LEN
0035            CALL PARSER(ARRAY,W_LEN,LEN,ERR)
0036            IF(ERR)RETURN
0037            DO I=1,LEN,2
0038               X=X+ARRAY(I)
0039               Y=Y+ARRAY(I+1)
0040            ENDDO
0041            FDM=BYTES(PTR)
0042            PTR=PTR+1
0043            RETURN
0044            END
```

```
      SUBROUTINE NAMEREAD(NAMEIN,EOF)
*****************************************************
*     READS FILE NAMES FOR PROCESSING OF FILES BY   *
*     PDBREAD                                       *
*****************************************************
*     INPUTS: NONE                                  *
*     OUTPUTS: NAMEIN-- NEXT FILE TO BE PROCESSED   *
*              EOF-- FLAG SIGNALLING END OF NAME    *
*                    FILE                           *
*****************************************************
      CHARACTER*9 NAMEIN
      LOGICAL EOF
      EOF=.FALSE.
         READ(7,10,END=100)NAMEIN ! FOR007 CONTAINS
10       FORMAT(A9)               ! THE FILE NAMES
      RETURN
100   EOF=.TRUE.
      RETURN
      END
```

```
0001            SUBROUTINE PARSER(ARRAY,W_LEN,LEN,ERR)
0002      ****************************************************************
0003      *     'PARSES' THE CURRENT FDM (FUNCTION DEFINITION MODULE).   *
0004      ****************************************************************
0005      *     INPUTS:                                                  *
0006      *             W_LEN-- THE WORD LENGTH OF THE DATA(1 OR 2 BYTES) *
0007      *             LEN-- THE LENGTH OF THE ARRAY IN W_LEN UNITS     *
0008      *     OUTPUTS: ARRAY-- THE PARSED DATA                         *
0009      ****************************************************************
0010            INTEGER*2 W_LEN,LEN,ARRAY(0:LEN)
0011            LOGICAL ERR
0012            INCLUDE 'WORDS.CMN'
0013    1 ***********************************************************
0014    1       INTEGER*2 PTR          !POINTS TO CURRENT BYTE
0015    1       INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
0016    1       BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
0017    1       INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
0018    1       EQUIVALENCE (WORDS(1),BYTES(1))
0019    1       COMMON/WORDS/BYTES,N_BYTES,PTR
0020    1 ***********************************************************
0021    1
0022            IF(W_LEN.EQ.1)THEN
0023              DO I=0,LEN-1,2
0024                 ARRAY(I)=BYTES(PTR+I+1)
0025                 ARRAY(I+1)=BYTES(PTR+I)
0026              ENDDO
0027            ELSE IF(W_LEN.EQ.2)THEN
0028              P=(PTR+1)/2
0029              DO I=0,LEN-1
0030                 ARRAY(I)=WORDS(P+I)
0031              ENDDO
0032            ELSE
0033              WRITE(3,*)'INVALID WORD-LENGTH IN PARSER: ',W_LEN
0034              ERR=.TRUE.
0035              RETURN
0036            ENDIF
0037            PTR=PTR+W_LEN*LEN+1 !THE FDM CODE WILL BE ACCESSED NEXT
0038            RETURN
0039            END
```

```
0001            SUBROUTINE PDB2POLY(FNAME)
0002      ************************************************************
0003      *     THIS SUBROUTINE TRANSFORMS THE RELATIVE DATA FROM *
0004      *     THE DIGITIZER-PRODUCED PDB FILES INTO VERTICES OF *
0005      *     POLYGONS IN UTM COORDINATES.                      *
0006      ************************************************************
0007      *     INPUTS: FNAME-- THE NAME OF THE INPUT FILE        *
0008      ************************************************************
0009            CHARACTER*5 FNAME
0010            LOGICAL ERR
0011            INCLUDE 'MAPDAT.CMN'
0012    1 ************************************************************
0013    1       REAL X,Y      !CURSOR POSITION
0014    1       REAL XCENTR,YCENTR !UTM CENTER OF MAP
0015    1       REAL XSCALE,YSCALE !METERS/TABLET UNIT
0016    1 ************************************************************
0017    1       COMMON/MAPDAT/X,Y,XCNTR,YCNTR,XSCALE,YSCALE
0018    1 ************************************************************
0019    1
0020    1
0021            INCLUDE 'WORDS.CMN'
0022    1 ************************************************************
0023    1       INTEGER*2 PTR          !POINTS TO CURRENT BYTE
0024    1       INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
0025    1       BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
0026    1       INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
0027    1       EQUIVALENCE (WORDS(1),BYTES(1))
0028    1       COMMON/WORDS/BYTES,N_BYTES,PTR
0029    1 ************************************************************
0030    1
0031            INCLUDE 'FDM.PAR'
0032    1 ************************************************************
0033    1       INTEGER*2 FDM,L_R_M,L_R_D,S_R_M,S_R_D
0034    1       PARAMETER(
0035    1     + HEADER=16,          !CODE FOR THE HEADER MODULE
0036    1     + L_R_M= 28,          !LONG RELATIVE MOVE
0037    1     + L_R_D= 29,          !LONG RELATIVE DRAW
0038    1     + S_R_M= 32,          !SHORT RELATIVE MOVE
0039    1     + S_R_D= 33)          !SHORT RELATIVE DRAW
0040    1 ************************************************************
0041            ERR=.FALSE.
0042      23    FORMAT(1X,A5)
0043            WRITE(6,23)FNAME
0044            CALL CALIBR8(FNAME,FDM,ERR)
0045             DO WHILE((PTR.LT.N_BYTES).AND.(.NOT.ERR))
0046               IF((FDM.EQ.S_R_M).OR.(FDM.EQ.L_R_M)) THEN
0047                 CALL MOVR(FDM,ERR)
0048               ELSE IF((FDM.EQ.S_R_D).OR.(FDM.EQ.L_R_D)) THEN
0049                 CALL DRAWR(FDM,ERR)
0050               ELSE
0051                 ERR=.TRUE.
0052                 WRITE(3,*)'IN FILE ',FNAME,' BAD FDM: ',FDM
0053                 RETURN
0054               ENDIF
0055             ENDDO
0056            RETURN
0057            END
```

```
      PROGRAM PDBREAD
*****************************************************************
*     THIS PACKAGE PROCESSES THE 4081-PRODUCED PDB FILES INTO *
*     POLYGONS WHICH CAN BE DISPLAYED AND PACKED INTO SURFACE *
*     FEATURE CODES IN THE TERRAIN DATA FILES BY THE ROUTINE  *
*     PDBPACK.                                                *
*****************************************************************
      LOGICAL EOF,ERR
      INCLUDE 'WORDS.CMN'
1 ***********************************************************
1     INTEGER*2 PTR          !POINTS TO CURRENT BYTE
1     INTEGER*2 N_BYTES      !NUMBER OF BYTES IN FILE
1     BYTE BYTES(11264)       !44 BLOCKS ON THE 4081
1     INTEGER*2 WORDS(5632)!THE MAP COORDINATES ARE I*2
1     EQUIVALENCE (WORDS(1),BYTES(1))
1     COMMON/WORDS/BYTES,N_BYTES,PTR
1 ***********************************************************
1
      CHARACTER*9 NAMEIN,NAMEOUT
      DATA EOF/.FALSE./,ERR/.FALSE./
      CHARACTER*5 NAM
      EQUIVALENCE (NAMEIN,NAM)
      OPEN(UNIT=3,NAME='PDBERR.TXT',STATUS='NEW')
      OPEN(UNIT=7,NAME='PDB.TXT',STATUS='OLD')
      CALL SETGEO
      DO WHILE(.NOT.EOF)
         CALL NAMEREAD(NAMEIN,EOF)
         OPEN(UNIT=1,NAME=NAMEIN,TYPE='OLD',FORM='FORMATTED')
         NAMEOUT=NAM//'.DAT'
         OPEN(UNIT=2,NAME=NAMEOUT,TYPE='NEW',FORM='UNFORMATTED')
         CALL FILEREAD(ERR)
         WRITE(3,*)NAMEIN,N_BYTES
         CALL PDB2POLY(NAM,ERR)
         CLOSE(UNIT=1)
         CLOSE(UNIT=2)
      ENDDO
      END
```

```
      SUBROUTINE POLYWRITE(POLY,P)
*****************************************************
*     THIS SCALES,TRANSLATES,AND WRITES THE DIGITIZED *
*     DATA WHICH HAS BEEN TRANSFORMED FROM RELATIVE   *
*     TO ABSOLUTE FORM. AND WRITES THE DATA TO FILE 2 *
*****************************************************

      INTEGER*2 P                 !# OF VERTICES
      DIMENSION POLY(1000,2)    !VERTICES OF A POLYGON
      REAL P1,P2
      INCLUDE 'CMERID.CMN'
1 ***************************************************
1     REAL*8 CMERID
1     REAL P_RAD
1     COMMON/CMERID/CMERID,P_RAD
1 ***************************************************
      INCLUDE 'MAPDAT.CMN'
1 ************************************************************
1     REAL X,Y      !CURSOR POSITION
1     REAL XCENTR,YCENTR !UTM CENTER OF MAP
1     REAL XSCALE,YSCALE !METERS/TABLET UNIT
1 ************************************************************
1     COMMON/MAPDAT/X,Y,XCNTR,YCNTR,XSCALE,YSCALE
1 ************************************************************
C...  THE OLD SCALING INFO IS NOT WRITTEN
      IF(XSCALE.NE.1.)THEN
C...  CONVERT THE ANGULAR MEASUREMENTS TO UTM
          P1=POLY(1,1)*P_RAD*XSCALE+XCNTR
          P2=POLY(1,2)*P_RAD*YSCALE+YCNTR
          CALL ADSMP(P2,P1,CMERID,FEAST,FNORTH)
          POLY(1,1)=FEAST+500000.
          POLY(1,2)=FNORTH
          NPTS=1
          DO I=2,P
            P1=POLY(I,1)*P_RAD*XSCALE+XCNTR
            P2=POLY(I,2)*P_RAD*YSCALE+YCNTR
            CALL ADSMP(P2,P1,CMERID,FEAST,FNORTH)
            P1=FEAST+500000.
            P2=FNORTH
C...      IF THE POINTS ARE REASONABLY FAR APART...
            IF(DELTA(POLY(NPTS,1),POLY(NPTS,2),P1,P2).GT.200.)THEN
              NPTS=NPTS+1
            POLY(NPTS,1)=P1
            POLY(NPTS,2)=P2
            ENDIF
          ENDDO
        IF(DELTA(POLY(NPTS,1),POLY(NPTS,2),POLY(1,1),POLY(1,2)).GT
          NPTS=NPTS+1
        ENDIF
        POLY(NPTS,1)=POLY(1,1)
        POLY(NPTS,2)=POLY(1,2)
        WRITE(2)NPTS,((POLY(I,J),J=1,2),I=1,NPTS)
        NPTS=0
      ENDIF
      RETURN
      END
```

ROADHEXER CALLING SEQUENCE

```
ROADHEXER
 |
 |---HEXINIT*
 |
 |---OPENERS*
 |
 |---GRIDR*
 |
 |---IGRID*
 |
 |---NODEHEX
     |
     |---GETNDX*
     |
     |---GETREC*
     |
     |---NOD2HX
     |     |
     |     |---TRANX*
     |     |
     |     |---TRANY*
     |
     |---LINKHEX
           |
           |---GETNDX*
           |
           |---GETREC*
           |
           |---GETSUB*
           |
           |---HEXSUB
           |     |
           |     |---TRANX*
           |     |
           |     |---TRANY*
           |     |
           |     |---HXRECORD
           |           |
           |           |---HEXREAD*
           |           |
           |           |---PACKER*
           |
           |---NOD2HX
           |     |
           |     |---TRANX*
           |     |
           |     |---TRANY*
           |
           |---HXRECORD
                 |
                 |---HEXREAD*
                 |
                 |---PACKER*
```

*Terrain System Utilities

Figure B-15

PROGRAM: ROADHEXER

ROUTINE \ COMMON	CENTER	HEX	HEXRAD	LUKNOD	PACK	SUB												
GETSUB				X		X												
GRIDR						X												
HEXINIT		X	X															
HEXSUB	X					X												
HXRECORD					X													
LINKHEX				X	X	X												
NOD2HX	X																	
NODEHEX				X		X												
PACKER				X														
ROADHEXER	X			X														

Figure B-16

```
      SUBROUTINE HEXSUB(HEXA,HEXB,N)
*********************************************
*     BEGINNING AT HEXA, WHICH CONTAINS THE *
*     START NODE, A CHAIN OF ADJACENT HEXES *
*     IS GENERATED TO HEXB, WHICH CONTAINS  *
*     THE STOP NODE FOR A LINK.             *
*********************************************
*     INPUTS:                               *
*            HEXA,HEXB-- THE HEXES WHICH    *
*            CONTAIN THE TERMINAL NODES     *
*            N-- THE NUMBER OF SUBNODES,    *
*              NOT COUNTING THE ENDPOINTS.  *
*********************************************
      IMPLICIT INTEGER (H,P)
      INCLUDE'UTIL:SUB.CMN'
1 ************************************************************
1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
1 ************************************************************
1     INTEGER*2 SUBX(100),SUBY(100)
1     COMMON/SUB/ SUBX,SUBY
1 ***********************************************************
      INCLUDE 'UTIL:CENTER.CMN'
1 ***********************************************************
1 *     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN *
1 *     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE *
1 *     TO A GIVEN GRID ZONE.                            *
1 ***********************************************************
1     INTEGER*4 XORIGIN,YORIGIN
1     COMMON/CENTER/XORIGIN,YORIGIN
1     DATA XORIGIN/500000/,YORIGIN/5700000/
1 ***********************************************************
      LEV=4
*
      HEXSTART=HEXA
      HEXSTOP=HEXB
*
      I=0
      HEXC=-1  ! NOT A VALID HEX #
*     PROCESS THE SUBNODE LIST UNTIL THE LINK  *
*     ENTERS THE TERMINAL HEX                  *
      DO WHILE(HEXC.NE.HEXSTOP)
         I=I+1
         X=TRANX(SUBX(I))-XORIGIN
         Y=TRANY(SUBY(I))-YORIGIN
         CALL XYL2HA(X,Y,LEV,HEXC)
*        IF THE SUBNODE IS NOT IN HEXA...         *
         IF(HEXSTART.NE.HEXC)THEN
            CALL HXRECORD(HEXSTART,HEXC,IADJ)
*           ...BUT IS IN AN ADJACENT HEX          *
            IF(IADJ.EQ.1)THEN
*              ...AND IS NOT IN HEXB              *
               IF(HEXC.NE.HEXSTOP)THEN
                  CALL HXRECORD(HEXC,HEXSTOP,IADJ)
*                 ...THEN IF IT IS ADJACENT TO HEXB...
                  IF(IADJ.EQ.1)THEN
*                    DONE                          *
```

```
                                RETURN
                            ENDIF
*                           OTHERWISE START OVER AT THIS SUBNODE
                            HEXSTART=HEXC
                        ENDIF
                    ENDIF
                ENDIF
*               GET THE NEXT SUBNODE
            ENDDO
            RETURN
            END
```

```
      SUBROUTINE HXRECORD(HA,HB,IERR)
*****************************************************************
*     THIS ROUTINE COMPUTES THE SIDE AT WHICH HEXA              *
*     IS ADJACENT TO HEXB. IT PASSES THIS INFORMATION           *
*     TO PACKER.  THEN IT INVERTS THE SIDE AND PASSES           *
*     THE INVERTED SIDE ALONG WITH HEXB AND TYPE TO             *
*     PACKER. THIS IS DONE TO INSURE THAT THE CONNEC-           *
*     TIVITY LOOKS THE SAME FROM BOTH HEXES.  SINCE ONLY        *
*     OUTLINKS ARE CHECKED, SOMETHING OF THIS SORT IS           *
*     NECESSARY.                                                *
*****************************************************************
*       INPUT:  HA--THE HEX ADDRESS OF THE FIRST SUBNODE        *
*               HB--THE HEX ADDRESS OF THE SECOND SUBNODE       *
*       OUTPUT: IERR-- AN ERROR FLAG                            *
*****************************************************************
      IMPLICIT INTEGER(H,P)
      INCLUDE 'UTIL:PACK.CMN'
1 ***********************************************************
1     INTEGER*4 SIDES ! PACKED CONNECTIVITIES                 *
1     INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE         *
1     COMMON/PACK/SIDES,LTYPE
1 ***********************************************************
*
*     SUBTRACT BY ADDING THE INVERSE
*     HSTORB IS THE INVERSE OF THE SIDE AT WHICH
*     HEXB IS ADJACENT TO HEXA
*     HSTORA AND HSTORB ARE IN INTERNAL HEX FORMAT
      HSTORB=HXADD(HA,HXINV(HB))
      HSTORA=HXINV(HSTORB)
*
*     GET THE RECORD FOR HEX 'HA' OR CAUSE IT TO
*     BE INITIALIZED IF NECESSARY
      LU=7
      CALL HEXREAD(HA,SIDES,LU)
*
*     IERR IS A FLAG INDICATING THAT TWO ADJACENT
*     SUBNODES WERE NOT IN ADJACENT HEXES
*
      CALL PACKER(HA,HSTORA,IERR,LU)
      IF (IERR.EQ.0)RETURN
*
*     NOW FOR THE SECOND HEX
*
*     HSTORB IS THE SIDE OF HEXB AT WHICH HEXA
*     IS ADJACENT
*
      CALL HEXREAD(HB,SIDES,LU)
      CALL PACKER(HB,HSTORB,IERR,LU)
*     THERE SHOULD BE NO ERROR IF IT GETS PAST
*     THE FIRST CALL TO PACKER.
      RETURN
      END
```

```
      SUBROUTINE LINKHEX(LNKNUM,HEXA)
*****************************************************
*     THIS ROUTINE CHECKS THE LINK TYPE FOR         *
*     EACH LINK INCIDENT TO A NODE.  IF THE         *
*     TYPE IS BETWEEN 1 AND 3 (ie; IF THE LINK      *
*     REPRESENTS A ROAD) THEN THE TERMINAL HEX      *
*     FOR THIS LINK IS OBTAINED FROM NOD2HX.        *
*     IF THE LINK CONNECTS TWO HEXES THEN THE       *
*     SUBNODES COORDINATES ARE READ IN AND EACH     *
*     PAIR OF ADJACENT SUBNODES(AND THE START       *
*     AND STOP HEXES) ARE PROCESSED BY HXRECORD.    *
*****************************************************
*       INPUT:                                      *
*             LNKNUM-- POINTER TO THE LINK RECORD   *
*             HEXA-- HEX CONTAINING THE NODE        *
*****************************************************
      IMPLICIT INTEGER (H,P)
      INTEGER*2 TMPLINK
      INTEGER*4 WRDPOS,SRECNUM,SUBWRD
*     IF THE LINK CONNECTS TWO DIFFERENT HEXES,THEN
      INCLUDE 'UTIL:SUB.CMN'
1 ***********************************************************
1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
1 ***********************************************************
1     INTEGER*2 SUBX(100),SUBY(100)
1     COMMON/SUB/ SUBX,SUBY
1 ********************************************************
      INCLUDE 'UTIL:PACK.CMN'
1 ********************************************************
1     INTEGER*4 SIDES ! PACKED CONNECTIVITIES          *
1     INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE  *
1     COMMON/PACK/SIDES,LTYPE
1 ********************************************************
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ***************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
1 ***************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1    +  ,SUBREC(500)
1 ****************************************************************
      NXTLNK=LNKNUM
      LU=3
      DO WHILE(NXTLNK.NE.0)
         CALL GETNDX(NXTLNK,LRECNUM,WRDPOS)
         CALL GETREC(LU,LRECNUM,LNKREC)
         N=WRDPOS
*
*        SET LINK TYPE                                  *
         TMPLINK=LIB$EXTZV(0,16,LNKREC(3,N))
         CALL LIB$INSV(TMPLINK,0,16,LTYPE)
*
         IF(LTYPE.LT.4)THEN
           LTYPE=4-LTYPE
           TERMXY=LNKREC(4,WRDPOS)
           CALL NOD2HX(TERMXY,HEXB)
```

```
0058                    IF(HEXA.NE.HEXB)THEN
0059                       CALL HXRECORD(HEXA,HEXB,IADJ)
0060                       IF(IADJ.EQ.0)THEN
0061     *
0062     *                   SET SUBNODE  RECORD POINTER                *
0063                         TMPLINK=LIBSEXTZV(0,16,LNKREC(5,N))
0064                         CALL LIBSINSV(TMPLINK,0,16,SRECNUM)
0065     *                   SET SUBNODE WORD POINTER                   *
0066                         TMPLINK=LIBSEXTZV(16,16,LNKREC(5,N))
0067     *                   CALL LIBSINSV(TMPLINK,0,16,SUBWRD)
0068     *                   GET THE SUBNODE LIST                       *
0069                         IF(SRECNUM.NE.0)THEN
0070                           CALL GETSUB(SRECNUM,SUBWRD,NUM)
0071     *                     PROCESS THE SUBNODE LIST
0072                           CALL HEXSUB(HEXA,HEXB,NUM)
0073                         ENDIF
0074                       ENDIF
0075                    ENDIF
0076                 ENDIF
0077     *           GET THE NEXT LINK RECORD
0078                 NXTLNK=LNKREC(2,WRDPOS)
0079              ENDDO
0080              RETURN
0081              END
```

```
      SUBROUTINE NOD2HX(XY,HSTOR)
*****************************************************
*     THIS ROUTINE UNPACKS THE X AND Y COORD-       *
*     INATES FROM A WORD OF THE LOC DATA BASE,      *
*     (FROM THE NODE OR THE EMMENDED LINK FILE)     *
*     AND TRANSLATES IT TO A HEX ADDRESS.           *
*****************************************************
*     INPUTS: XY-- THE PACKED COORDINATES           *
*     OUTPUTS: HSTOR-- THE INTERNAL HEX NUMBER      *
*****************************************************
      IMPLICIT INTEGER(H,P)
      INTEGER*4 XY,TMPNOD
      INTEGER*2 X,Y
      INCLUDE 'UTIL:CENTER.CMN'
1 ***********************************************************
1 *     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN *
1 *     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE *
1 *     TO A GIVEN GRID ZONE.                            *
1 ***********************************************************
1     INTEGER*4 XORIGIN,YORIGIN
1     COMMON/CENTER/XORIGIN,YORIGIN
1     DATA XORIGIN/500000/,YORIGIN/5700000/
1 ***********************************************************
*
*     SET X COORDINATE                                *
      TMPNOD=LIBSEXTZV(0,16,XY)
      CALL LIBSINSV(TMPNOD,0,16,X)
      XX=TRANX(X)-XORIGIN
*     SET Y COORDINATE
      TMPNOD=LIBSEXTZV(16,16,XY)
      CALL LIBSINSV(TMPNOD,0,16,Y)
      YY=TRANY(Y)-YORIGIN
*
      LEV=4
      CALL XYL2HA(XX,YY,LEV,HSTOR)
      RETURN
      END
```

```
      SUBROUTINE NODEHEX(NODE)
*************************************************
*     THIS ROUTINE ACCESSES THE NODE RECORD     *
*     FINDS THE HEX WHICH CONTAINS THE NODE,    *
*     GETS THE LINK POINTER AND PASSES THIS     *
*     POINTER AND THE HEX NUMBER TO LINKHEX     *
*************************************************
*     INPUT:  NODE-- THE HEAD NODE FOR A        *
*                    GRID RECORD FROM THE       *
*                    LOC DATA BASE              *
*************************************************
      IMPLICIT INTEGER (H,P)
      INTEGER*4 WRDPOS
      INCLUDE 'UTIL:LNKNOD.CMN'
1 ***************************************************************
1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
1 ***************************************************************
1     INTEGER*4 GRID,NODREC,LNKREC,SUBREC
1     COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
1    +,SUBREC(500)
1 ***************************************************************
      INCLUDE 'UTIL:SUB.CMN'
1 ***********************************************************
1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
1 ***********************************************************
1     INTEGER*2 SUBX(100),SUBY(100)
1     COMMON/SUB/ SUBX,SUBY
1 ***********************************************************
C
      DO WHILE (NODE.NE.0)
         CALL GETNDX(NODE,NRECNUM,WRDPOS)
         LU=2
         CALL GETREC(LU,NRECNUM,NODREC)
C
         NODXY=NODREC(5,WRDPOS)
         CALL NOD2HX(NODXY,HEXA)
         LNK=NODREC(4,WRDPOS)
         CALL LINKHEX(LNK,HEXA)
C
*        GET NEXT NODE
         NODE=NODREC(1,WRDPOS)
      ENDDO
      RETURN
      END
```

```
0001            PROGRAM ROADHEXER
0002      ***********************************************
0003      *     THIS SET OF ROUTINES WAS USED TO 'HEXISE'  *
0004      *     THE BDM-PRODUCED LOC DATA FOR GERMANY.     *
0005      ***********************************************
0006            IMPLICIT INTEGER(H,P)
0007            INTEGER*4 XO,YO
0008            INCLUDE 'UTIL:CENTER.CMN'
0009    1 ***********************************************************
0010    1 *     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN *
0011    1 *     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE *
0012    1 *     TO A GIVEN GRID ZONE.                            *
0013    1 ***********************************************************
0014    1       INTEGER*4 XORIGIN,YORIGIN
0015    1       COMMON/CENTER/XORIGIN,YORIGIN
0016    1       DATA XORIGIN/500000/,YORIGIN/5700000/
0017    1 ***********************************************************
0018            INCLUDE 'UTIL:LNKNOD.CMN'
0019    1 ***************************************************************
0020    1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
0021    1 ***************************************************************
0022    1       INTEGER*4 GRID,NODREC,LNKREC,SUBREC
0023    1       COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
0024    1     + ,SUBREC(500)
0025    1 ***************************************************************
0026      *
0027      *     INITIALIZE THE HEX PARAMETERS
0028      *     THE VARIABLES DLT AND DLN IN HEXINIT MUST
0029      *     AGREE WITH THE CENTER COORDINATES XORIGIN AND
0030      *     YORIGIN IN CENTER.CMN
0031            CALL HEXINIT
0032      *     OPEN THE LOC AND HEX FILES
0033            CALL OPENERS
0034      *     READ IN THE GRID POINTERS
0035            CALL GRIDR
0036      *     XO AND YO ARE THE RELATIVE COORDINATES
0037      *     OF THE CENTER OF THE AREA TO BE ACCESSED
0038      *     THAT IS, XO AND YO ARE OFFSETS IN METERS
0039      *     FROM THE CENTER OF THE HEX GRID.
0040            PRINT*,'ENTER THE COORDINATES OF THE CENTER OF THE '
0041            PRINT*,'AREA TO BE PROCESSED AS METERS FROM THE HEX ORIGIN'
0042            READ*,XO,YO
0043            XO=XORIGIN+XO
0044            YO=YORIGIN+YO
0045      *     SET THE GRID POINTERS FOR THE CENTER
0046            CALL IGRID(XO,YO,IO,JO)
0047            WRITE(6,*)'NOW ENTER THE LENGTH OF ONE SIDE OF THE SQUARE A
0048            READ(5,*)IEXT
0049            IEXT=IEXT/20 !DIVIDING BY 2 AND THEN BY 10 (1/2 SIDE,10KM R
0050            DO I=IO-IEXT,IO+IEXT,1
0051               DO J=JO-IEXT,JO+IEXT,1
0052                  NODE=GRID(I,J)
0053                  CALL NODEHEX(NODE)
0054               ENDDO
0055            ENDDO
0056            END
```

TERRAIN UTILITIES

```
      SUBROUTINE ADSCCM(IGRIDN,CMERID)
C
C         ***************************************************************
C         *
C         *  NAME:
C         *      ADSCCM -- DETERMINE CENTRAL MERIDIAN OF A GRIDZONE
C         *
C         *  PURPOSE:
C         *      TO CALCULATE THE CENTRAL MERIDIAN IN RADIANS OF
C         *      A GRIDZONE USING THE INTEGER GRID NUMBER (E.G 32 OF '32
C         *
C         *  DESCRIPTION:
C         *      AUTHOR - P. W. DENNIS
C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
C         *      MOD LEVEL              DATE          DR NUMBERS
C         *         01                 102979          DR 00009
C         *
C         *
C         *  CALLING SEQUENCE:
C         *      CALL ADSCCM (IGRIDN,CMERID)
C         *      WHERE:
C         *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
C         *
C         *      IGRIDN             GRID_NUMBER        INTEGER GRID NUMBER
C         *
C         *      CMERID             CENT_MERID         CENTRAL MERIDIAN
C         *
C         *                                            OF PROJECTION
C         *
C         *
C         *  INPUT/OUTPUT:
C         *
C         *       NONE
C         *
C         *  RESTRICTIONS:
C         *   NONE
C         *
C         ***************************************************************
C
      REAL *8 CMERID
      PARAMETER PI=3.141592654
C
C     COMPUTE CENTRAL MERIDIAN
C
      CMERID = DFLOTJ(IGRIDN*6-183)*PI/180.
C
      RETURN
      END
```

```
0001          SUBROUTINE ADSCDD(FDEG,IDEG,IMIN,SEC)
0002    C
0003    C        ***********************************************************
0004    C        *
0005    C        *  NAME:
0006    C        *      ADSCDD -- CONVERT DECIMAL DEGREES TO DEGREES-MINUTES-SE
0007    C        *
0008    C        *  PURPOSE:
0009    C        *      CONVERTS DEGREES TO INTEGER DEGREES AND MINUTES AND FLO
0010    C        *      POINT S            ECONDS
0011    C        *
0012    C        *  DESCRIPTION:
0013    C        *      AUTHOR - P. W. DENNIS
0014    C        *      LAST MODIFIED BY P. E. KING ON 28 OCT 79
0015    C        *      MOD LEVEL              DATE          DR NUMBERS
0016    C        *         01                 102979          DR 00009
0017    C        *
0018    C        *
0019    C        *  CALLING SEQUENCE:
0020    C        *      CALL ADSCDD (FDEG,IDEG,IMIN,SEC)
0021    C        *      WHERE:
0022    C        *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0023    C        *
0024    C        *      FDEG               INP_DEGREES        F.P. DEGREES
0025    C        *
0026    C        *      IDEG               DEGREES            INTEGER DEGREES
0027    C        *
0028    C        *      IMIN               MINUTES            INTEGER MINUTES
0029    C        *
0030    C        *
0031    C        *      SEC                SECONDS            F.P. SECONDS
0032    C        *
0033    C        *
0034    C        *  INPUT/OUTPUT:
0035    C        *
0036    C        *       NONE
0037    C        *
0038    C        *  RESTRICTIONS:
0039    C        *   NONE
0040    C        *
0041    C        ***********************************************************
0042    C
0043    C         COMPUTE DEGREES BY TAKING INTEGER PART
0044    C         DEGREES = INTEGER PART OF ABSOLUTE VALUE OF [INP_DEGREES]
0045    C
0046          IDEG =  ABS(FDEG)
0047    C
0048    C   GET MINUTES
0049    C   MINUTES = [ABSOLUTE VALUE OF [INP_DEGREES] MINUS DEGREES ] * 60
0050    C
0051          IMIN = (ABS(FDEG) - IDEG) * 60
0052    C
0053    C   GET SECONDS
0054    C   SECONDS = [ABSOLUTE VALUE OF [INP_DEGREES] MINUS DEGREES MINUS MINUT
0055    C              60] * 3600
0056    C
0057          SEC = (ABS(FDEG) - IDEG - FLOAT(IMIN)/60.) * 3600.
```

```
C
C     TRUNCATE TO  .1 SECONDS
C
      SEC = AINT(SEC*10.)/10.
C
C
      RETURN
      END
```

```
0001          SUBROUTINE ADSCFE(IGRIDN,IENUM)
0002    C
0003    C         ***********************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCFE -- DETERMINE THE FIRST 100K SQUARE EAST
0007    C         *                OF THE CENTRAL MERIDIAN
0008    C         *
0009    C         *  PURPOSE:
0010    C         *      TO DETERMINE A NUMBER CORRESPONDING TO THE
0011    C         *      FIRST 100K SQUARE EAST OF THE CENTRAL MERIDIAN
0012    C         *
0013    C         *  DESCRIPTION:
0014    C         *      AUTHOR - P. W. DENNIS
0015    C         *      LAST MODIFIED BY P. W. DENNIS ON 7 JAN 80
0016    C         *      MOD LEVEL              DATE          DR NUMBERS
0017    C         *         01                  102979         DR 00009
0018    C         *
0019    C         *
0020    C         *  CALLING SEQUENCE:
0021    C         *      CALL ADSCFE (IGRIDN,IENUM)
0022    C         *      WHERE:
0023    C         *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0024    C         *
0025    C         *      IGRIDN             INP_GRID_NUM       A GRID NUMBER
0026    C         *
0027    C         *      IENUM              EAST_100K_NUM      THE NUMBER CORRESPO
0028    C         *                                            TO THE FIRST EASTIN
0029    C         *                                            LETTER
0030    C         *
0031    C         *  INPUT/OUTPUT:
0032    C         *
0033    C         *       NONE
0034    C         *
0035    C         *  RESTRICTIONS:
0036    C         *   NONE
0037    C         *
0038    C         ***********************************************************
0039    C
0040    C
0041    C     THE EASTING LETTER ID RANGES FROM 'A-Z' EXCLUDING LETTERS 'I' AND
0042    C     STARTING WITH 'A' AT EVERY THIRD GRIDZONE BEGINNING WITH GRIDZONE
0043    C     I.E. 1,4,7,...,58. SO FOR GRIDZONES 3,6,9,...,60, THE FIRST 100K
0044    C     SQUARE EAST OF THE CENTRAL MERIDIAN CAN BE REPRESENTED BY THE 21ST
0045    C     LETTER, 'W'; FOR GRIDZONES 1,4,7,...,58, THE FIRST 100K SQUARE EAS
0046    C     OF THE CENTRAL MERIDIAN CAN BE REPRESENTED BY THE 5TH LETTER, 'E',
0047    C     AND FOR GRIDZONES 2,5,8,...,59, THE FIRST 100K SQUARE EAST OF THE
0048    C     CENTRAL MERIDIAN CAN BE REPRESENTED BY THE 13TH LETTER, 'N'
0049    C
0050    C          ...FIND THE REMAINDER OF INP_GRID_NUM DIVIDED BY 3
0051    C
0052          II=MOD(IGRIDN,3)
0053    C
0054    C        CHECK FOR GRIDZONE NUMBERS THAT ARE MULTIPLES OF 3, I.E.,GRIDZO
0055    C        3,6,9,... ,60
0056    C
0057    C          IF THE REMAINDER EQUALS 0
```

```
C
        IF (II.EQ.0) THEN
C
C           SET EAST_100K_NUM EQUAL TO 21 (EAST ID LETTER 'W')
C
      IENUM = 21
C
C     CHECK FOR GRIDZONE NUMBERS INCREMENTED BY 3 STARTING WITH 1, I.E.,
C     GRIDZONES 1,4,7,...,58
C
C           ELSEIF REMAINDER EQUALS 1
C
      ELSEIF (II.EQ.1)THEN
C
C           SET EAST_100K_NUM EQUAL TO 5 (EAST ID LETTER 'E')
C
      IENUM = 5
C
C     CHECK FOR GRIDZONE NUMBERS INCREMENTED BY 3 STARTING WITH 2, I.E.,
C     GRIDZONES 2,5,8,...,59
C
      ELSE
C
C           SET EAST_100K_NUM EQUAL TO 13 (EAST ID LETTER 'N')
C
      IENUM = 13
C
      ENDIF
      RETURN
      END
```

```
0001          SUBROUTINE ADSCGL(IGRIDL,IGLN,IERFLG)
0002    C
0003    C         ***************************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCGL -- PERFORM GRIDZONE LETTER TO NUMBER CONVERSION
0007    C         *
0008    C         *  PURPOSE:
0009    C         *      TO CONVERT THE GRIDZONE LETTER TO A  NUMBER FROM
0010    C         *      -10 TO 9 (0 THROUGH 9 NORTH OF EQUATOR)
0011    C         *
0012    C         *  DESCRIPTION:
0013    C         *      AUTHOR - P. W. DENNIS
0014    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015    C         *      MOD LEVEL              DATE          DR NUMBERS
0016    C         *         01                 102979          DR 00009
0017    C         *
0018    C         *
0019    C         *  CALLING SEQUENCE:
0020    C         *      CALL ADSCGL (IGRIDL,IGLN,IERFLG)
0021    C         *      WHERE:
0022    C         *      ARGUMENT NAME     PDL DATA NAME      DESCRIPTION
0023    C         *
0024    C         *      IGRIDL            GRID_LETTER        THE GRIDZONE LETTER
0025    C         *
0026    C         *      IGLN              GRID_LETNUM        A NUMBER CORRESPOND
0027    C         *                                           TO GRID_LETTER
0028    C         *
0029    C         *      IERFLG            ERR_FLAG_UCS       ERROR FLAG (-1 = ER
0030    C         *
0031    C         *
0032    C         *  INPUT/OUTPUT:
0033    C         *
0034    C         *       NONE
0035    C         *
0036    C         *  RESTRICTIONS:
0037    C         *   NONE
0038    C         *
0039    C         ***************************************************************
0040    C
0041          PARAMETER ERROR = -1
0042    C
0043    C
0044          BYTE IGRIDL
0045    C
0046    C
0047    C         THE FIRST 10 LETTERS, 'C-M', REPRESENT GRIDZONES IN THE SOUTH
0048    C         HEMISPHERE. 'C' WILL BE REPRESENTED BY -10, 'D' BY -9,...,'M'
0049    C         THE LAST 10 LETTERS, 'N-X' REPRESENT GRIDZONES IN THE NORTHER
0050    C         HEMISPHERE. 'N' WILL BE REPRESENTED BY 0, 'P' BY 1, 'Q' BY 2,
0051    C         ,'X' BY 9. THESE NUMBERS WILL BE USED LATER FOR CALCULATING T
0052    C         NORTHING
0053    C
0054    C
0055    C
0056    C         GRID_LETTER GREATER THAN OR EQUAL TO 67 (ASCII 'C') ?
0057    C
```

```
0058          IF (IGRIDL.GE.67)  THEN
0059    C
0060    C             GRID_LETTER LESS THAN OR EQUAL TO 88 (ASCII 'X') ?
0061    C
0062          IF (IGRIDL.LE.88) THEN
0063    C
0064    C             GRID_LETTER LESS THAN 73 (ASCII 'I') ?
0065    C
0066          IF (IGRIDL.LT.73) THEN
0067    C
0068    C          SET GRID_LETNUM TO GRID_LETTER MINUS 77
0069    C
0070          IGLN = IGRIDL - 77
0071    C
0072    C          ELSE IS GRID_LETTER GREATER THAN 73 ?
0073    C
0074          ELSEIF (IGRIDL.GT.73) THEN
0075    C
0076    C             GRID_LETTER LESS THAN 79 (ASCII 'O') ?
0077    C
0078          IF (IGRIDL.LT.79) THEN
0079    C
0080    C          SET GRID_LETNUM TO GRID_LETTER MINUS 78
0081    C
0082          IGLN = IGRIDL - 78
0083    C
0084    C          ELSEIF GRID_LETTER IS GREATER THAN 79
0085    C
0086          ELSEIF (IGRIDL.GT.79) THEN
0087    C
0088    C          SET GRID_LETNUM TO GRID_LETTER MINUS 79
0089    C
0090          IGLN = IGRIDL - 79
0091    C
0092          ELSE
0093    C
0094    C          SET ERR_FLAG_UCS
0095    C
0096          IERFLG = ERROR
0097    C
0098          ENDIF
0099          ELSE
0100    C
0101    C          SET ERR_FLAG_UCS
0102    C
0103          IERFLG = ERROR
0104    C
0105          ENDIF
0106          ELSE
0107    C
0108    C          SET ERR_FLAG_UCS
0109    C
0110          IERFLG = ERROR
0111    C
0112          ENDIF
0113          ELSE
0114    C
```

```
C          SET ERR_FLAG_UCS
C
      IERFLG = ERROR
C
      ENDIF
      RETURN
      END
```

```
0001          SUBROUTINE ADSCIE(LEAST,IENUM,IEAST,IERFLG)
0002    C
0003    C         ***************************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCIE -- COMPUTE THE EASTING TO THE NEAREST 100 KILOME
0007    C         *
0008    C         *  PURPOSE:
0009    C         *      TO COMPUTE THE INTEGER EASTING FROM THE CENTRAL MERIDIA
0010    C         *      TO THE NEAREST 100 KILOMETERS
0011    C         *
0012    C         *  DESCRIPTION:
0013    C         *      AUTHOR - P. W. DENNIS
0014    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015    C         *      MOD LEVEL              DATE          DR NUMBERS
0016    C         *         01                 102979          DR 00009
0017    C         *
0018    C         *
0019    C         *  CALLING SEQUENCE:
0020    C         *      CALL ADSCIE (LEAST,IENUM,IEAST,IERFLG)
0021    C         *      WHERE:
0022    C         *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0023    C         *
0024    C         *      LEAST              EAST_100K_LET      THE EASTING LETTER
0025    C         *                                            AN MGR
0026    C         *
0027    C         *      IENUM              EAST_100K_NUM      THE FIRST SQUARE EA
0028    C         *                                            THE CENTRAL MERIDIA
0029    C         *
0030    C         *      IEAST              EAST_INT           INTEGER EASTING IN
0031    C         *                                            100 KM UNITS
0032    C         *
0033    C         *      IERFLG             ERR_FLAG_UCS       ERROR FLAG (-1 = ER
0034    C         *
0035    C         *
0036    C         *  INPUT/OUTPUT:
0037    C         *
0038    C         *       NONE
0039    C         *
0040    C         *  RESTRICTIONS:
0041    C         *   NONE
0042    C         *
0043    C         ***************************************************************
0044    C
0045          PARAMETER ERROR = -1
0046    C
0047          BYTE LEAST
0048    C
0049          INTEGER *4 IEAST
0050    C
0051    C
0052    C       TO GET THE EASTING MEASURED FROM THE CENTRAL MERIDIAN, DETERMINE
0053    C       NUMBER CORRESPONDING TO THE EAST ID LETTER AND SUBTRACT THE
0054    C       NUMBER CORRESPONDING TO THE EAST ID LETTER OF THE FIRST 100K SQU
0055    C       EAST OF THE CENTRAL MERIDIAN. THEN MULTIPLY BY 100,000
0056    C
0057    C
```

```
0058    C           IF EAST_100K_LET IS GREATER THAN OR EQUAL TO 65 (ASCII 'A')
0059    C
0060            IF (LEAST.GE.65)THEN
0061    C
0062    C           IF EAST_100K_LET IS LESS THAN OR EQUAL TO 90 (ASCII 'Z')
0063    C
0064            IF (LEAST.LE.90)THEN
0065    C
0066    C           IF EAST_100K_LET IS LESS THAN 73 (ASCII 'I')
0067    C
0068            IF (LEAST.LE.73) THEN
0069    C
0070    C     SET EAST_INT TO [ EAST_100K_LET MINUS EAST_100K_NUM MINUS 54 ] * 10
0071    C
0072            IEAST = (LEAST - IENUM - 64) *100 000
0073    C
0074    C           ELSEIF EAST_100K_LET IS GREATER THAN 73
0075    C
0076            ELSEIF (LEAST.GT.73) THEN
0077    C
0078    C           IF EAST_100K_LET IS LESS THAN 79 (ASCII 'O')
0079    C
0080            IF (LEAST.LT.79) THEN
0081    C
0082    C     SET EAST_INT TO [ EAST_100K_LET MINUS EAST_100K_NUM MINUS 65 ] * 1
0083    C
0084            IEAST = (LEAST - IENUM - 65) *100 000
0085    C
0086    C           ELSEIF EAST_100K_LET IS GREATER THAN 79
0087    C
0088            ELSEIF (LEAST.GT.79) THEN
0089    C
0090    C     SET EAST_INT TO [ EAST_100K_LET MINUS EAST_100K_NUM MINUS 66 ] * 10
0091    C
0092            IEAST = (LEAST - IENUM - 66) *100 000
0093    C
0094    C           ELSE
0095    C
0096            ELSE
0097    C
0098    C           SET ERR_FLAG_UCS
0099    C
0100            IERFLG = ERROR
0101    C
0102    C           ENDIF
0103    C
0104            ENDIF
0105    C
0106    C           ELSE
0107    C
0108            ELSE
0109    C
0110    C           SET ERR_FLAG_UCS
0111    C
0112            IERFLG = ERROR
0113    C
0114    C           ENDIF
```

```
C
          ENDIF
C
C              ELSE
C
          ELSE
C
C              SET ERR_FLAG_UCS
C
          IERFLG = ERROR
C
C              ENDIF
C
          ENDIF
C
C              ELSE
C
          ELSE
C
C              SET ERR_FLAG_UCS
C
          IERFLG = ERROR
C
C              ENDIF
C
          ENDIF
C
C          CHECK FOR EASTING BLATANTLY OUT OF RANGE
C          (NO EASTING LETTER SHOULD BE MORE THAN 300 KM FROM
C           THE CENTRAL MERIDIAN )
C
          IF (IERFLG.NE.ERROR) THEN
          IF (IEAST.GT.300 000 .OR. IEAST.LT.-300 000) THEN
          IERFLG=ERROR
          ENDIF
          ENDIF
          RETURN
          END
```

```
0001          SUBROUTINE ADSCIN(IGLN,IGRIDN,N100KN,NORTH,ISPHER)
0002    C
0003    C         ***********************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCIN -- COMPUTE THE NORTHING TO THE NEAREST 100
0007    C         *     KILOMETERS
0008    C         *  PURPOSE:
0009    C         *      TO COMPUTE THE INTEGER NORTHING FROM THE EQUATOR
0010    C         *      TO THE NEAREST 100 KILOMETERS
0011    C         *
0012    C         *  DESCRIPTION:
0013    C         *      AUTHOR - P. W. DENNIS
0014    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015    C         *      MOD LEVEL              DATE          DR NUMBERS
0016    C         *         01                 102979          DR 00009
0017    C         *
0018    C         *
0019    C         *  CALLING SEQUENCE:
0020    C         *      CALL ADSCIN (IGLN,IGRIDN,N100KN,NORTH,ISPHER)
0021    C         *      WHERE:
0022    C         *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0023    C         *
0024    C         *      IGLN               GRID_LETNUM        NUMBER CORRESPONDIN
0025    C         *                                            THE GRIDZONE LETTER
0026    C         *
0027    C         *      IGRIDN             INP_GRID_NUM       THE NUMBER OF THE
0028    C         *                                            GRIDZONE
0029    C         *      N100KN             NORTH_100K_NUM     THE NUMBER CORRESPO
0030    C         *                                            100K SQUARE NORTHIN
0031    C         *                                            LETTER
0032    C         *
0033    C         *      NORTH              NORTH_INT          INTEGER NORTHING IN
0034    C         *                                            100 KM UNITS
0035    C         *      ISPHER             INP_SPHEROID       SPHEROID INDEX
0036    C         *
0037    C         *
0038    C         *  INPUT/OUTPUT:
0039    C         *
0040    C         *       NONE
0041    C         *
0042    C         *  RESTRICTIONS:
0043    C         *
0044    C         *      (1) THE REFERENCE GRIDZONE IS IN THE SHARED GLOBAL AREA
0045    C         *      (2) THE MGR INDEXED SPHEROID TABLE IS IN THE
0046    C         *          SHARED GLOBAL AREA.
0047    C         *
0048    C         *
0049    C         ***********************************************************
0050    C
0051    C
0052          INCLUDE 'ZDBPRO.COM'
00100   0053  1 C
00200   0054  1 C      DUMMY COMMON ZDBPRO
00300   0055  1 C
00400   0056  1        INTEGER*4 ZREDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500   0057  1        INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(5
```

```
00600   0058  1        LOGICAL*1 ZISIC(3),ZTDWN(25)
00700   0059  1 C
00800   0060  1        COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZISEC,ZIEX,
00900   0061  1       2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZISIC,
01000   0062  1       3                ZIDWN ,ZRGL  ,ZMGRST,ZLLST
01100   0063  1 C
        0064    C
        0065          PARAMETER PI = 3.141593
        0066    C
        0067    C
        0068    C
        0069    C
        0070          INTEGER *4 NDIST
        0071          INTEGER *4 NORTH
        0072          INCLUDE 'ADSTAB.DAT'
        0073  1 C     ************************************************************
        0074  1 C
        0075  1 C        TABLE OF SPHEROID AXES
        0076  1 C
        0077  1 C     ************************************************************
        0078  1 C
        0079  1       DIMENSION AAXIS(9),BAXIS(9)
        0080  1 C
        0081  1 C                          THE SEMI-MAJOR AXES
        0082  1 C
        0083  1       DATA  AAXIS            /
        0084  1      1                         6378388.,      !    INTERNATIONAL
        0085  1      2                         6378206.,      !    CLARKE 1866
        0086  1      3                         6378249.,      !    CLARKE 1880
        0087  1      4                         6377276.,      !    EVEREST
        0088  1      5                         6377397.,      !    BESSEL
        0089  1      6                         6378160.,      !    AUSTRALIAN NATI
        0090  1      7                         6377397.,      !    AIRY
        0091  1      8                         6378155.,      !    FISCHER
        0092  1      9                         6377304. /     !    MALAYAN
        0093  1 C
        0094  1 C                          THE SEMI-MINOR AXES
        0095  1 C
        0096  1       DATA  BAXIS            /
        0097  1      1                         6356912.,      !    INTERNATIONAL
        0098  1      2                         6356584.,      !    CLARKE 1866
        0099  1      3                         6356515.,      !    CLARKE 1880
        0100  1      4                         6356075.,      !    EVEREST
        0101  1      5                         6356079.,      !    BESSEL
        0102  1      6                         6356775.,      !    AUSTRALIAN NATI
        0103  1      7                         6356257.,      !    AIRY
        0104  1      8                         6356774.,      !    FISCHER
        0105  1      9                         6356102. /     !    MALAYAN
        0106  1 C
        0107  1 C
        0108  1 C                MOD LEVEL          DATE                    DR NUMBE
        0109  1 C                   01            110979                    DR 0000
        0110  1 C
        0111  1 C
        0112  1 C          LAST MODIFIED BY P. E. KING ON 9 NOVEMBER 79
        0113  1 C
        0114  1 C
```

```
0115  1 C     ***********************************************************
0116  1 C       ********************************************************
0117    C
0118    C
0119    C              COMPUTE APPROXIMATE DISTANCE USING GRID_LETNUM
0120    C
0121    C           NORTH_DIST = SEMI_MAJ*(GRID_LETNUM * 8 + 4) * PI_CONS
0122    C
0123            NDIST = AAXIS(ZSPHID)*(FLOAT(IGLN)*8. + 4.)*PI/180.
0124    C
0125    C                CALCULATE TO THE NEAREST 2000 KILOMETERS
0126    C
0127    C      NORTH_INT = 2,000,000 * INTEGER PART OF (NORTH_DIST DIVIDED
0128    C
0129            NORTH = 2 000 000 * (NDIST /2 000 000)
0130    C
0131    C                ADD ON 100 KILOMETER PART
0132    C
0133    C           NORTH_INT = NORTH_INT + 100,000 * NORTH_100K_NUM
0134    C
0135            NORTH = NORTH + 100 000 * N100KN
0136    C
0137    C                CHECK FOR TOO FAR NORTH
0138    C
0139    C           IF NORTH_INT - NORTH_DIST IS GREATER THAN 1,500,000
0140    C
0141            IF (NORTH - NDIST .GT. 1 500 000)  THEN
0142    C
0143    C           SET NORTH_INT TO NORTH_INT MINUS 2,000,000
0144    C
0145            NORTH = NORTH - 2 000 000
0146    C
0147    C                OR TOO FAR SOUTH
0148    C
0149    C           ELSEIF NORTH_DIST - NORTH_INT IS GREATER THAN 1,500,0
0150    C
0151            ELSEIF (NDIST - NORTH .GT. 1 500 000)  THEN
0152    C
0153    C           SET NORTH_INT TO NORTH_INT PLUS 2,000,000
0154    C
0155            NORTH = NORTH + 2 000 000
0156    C
0157    C
0158            ENDIF
0159    C
0160    C             NOW CHECK FOR THE 1000 KILOMETER JUMP ACROSS SPHEROI
0161    C
0162    C           IF ABSOLUTE VALUE OF (NORTH_INT - NORTH_DIST) IS GREA
0163    C
0164    C                          GET INDEX TO MGR INDEXED SPHEROID TABLE
0165    C
0166            IROW = IGRIDN - ZRGN + 2
0167    C
0168    C
0169    C       WATCH OUT FOR  ZONES 1 AND 60
0170    C
0171            IF (IROW.EQ. -57 .OR. IROW .EQ. 61)  THEN
```

```
0172            IROW = 1
0173            ENDIF
0174      C
0175            IF (IABS (NDIST - NORTH).GT. 500 000)  THEN
0176      C
0177      C           IF NORTH_INT IS GREATER THAN NORTH_DIST
0178      C
0179            IF (NORTH .GT. NDIST)  THEN
0180      C
0181      C           SET NORTH_INT TO NORTH_INT MINUS 1,000,000
0182      C
0183            NORTH = NORTH - 1 000 000
0184      C
0185      C
0186            ELSE
0187      C
0188      C           SET NORTH_INT TO NORTH_INT PLUS 1,000,000
0189      C
0190            NORTH = NORTH + 1 000 000
0191      C
0192      C
0193            ENDIF
0194      C
0195      C     SET INP_SPHEROID TO HIGH BYTE OF MGR_SPHEROID_TAB
0196      C
0197            ISPHER = ZMGRST(IROW,IGLN - ZRGL + 2)/256
0198      C
0199      C
0200            ELSE
0201      C
0202      C    SET INP_SPHEROID TO LOW BYTE OF MGR_SPHEROID_TAB
0203      C
0204            ISPHER = MOD (ZMGRST(IROW,IGLN - ZRGL + 2), 256)
0205      C
0206            ENDIF
0207      C
0208            RETURN
0209            END
```

```
0001            SUBROUTINE ADSCLN(LAT,LONG,NORTH,IEAST,IERFLG)
0002     C
0003     C          ***********************************************************
0004     C          *
0005     C          *  NAME:
0006     C          *      ADSCLN  -- LATLONG TO EASTING-NORTHING CONVERSION
0007     C          *
0008     C          *  PURPOSE:
0009     C          *      TO CONVERT ASCII LATITUDE AND LONGITUDE TO
0010     C          *       EASTINGS AND NORTHINGS.
0011     C          *
0012     C          *  DESCRIPTION:
0013     C          *      AUTHOR - P. W. DENNIS
0014     C          *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015     C          *      MOD LEVEL              DATE          DR NUMBERS
0016     C          *         01                 102979          DR 00009
0017     C          *
0018     C          *
0019     C          *  CALLING SEQUENCE:
0020     C          *      CALL ADSCLN (LAT,LONG,NORTH,IEAST,IERFLG)
0021     C          *      WHERE:
0022     C          *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0023     C          *
0024     C          *      IEAST              EAST_LCS           UTM EASTING
0025     C          *
0026     C          *      NORTH             NORTH_LCS          UTM NORTHING
0027     C          *
0028     C          *      IERFLG             ERR_FLAG_LCS       ERROR FLAG (-1 = ER
0029     C          *
0030     C          *
0031     C          *      LAT                LAT_LCS            LATITUDE (ASCII)
0032     C          *
0033     C          *      LONG               LONG_LCS           LONGITUDE (ASCII)
0034     C          *
0035     C          *  INPUT/OUTPUT:
0036     C          *
0037     C          *       NONE
0038     C          *
0039     C          *  RESTRICTIONS:
0040     C          *
0041     C          *   (1) THE REFERENCE GRIDZONE IS IN  THE SGA COMMON
0042     C          *   (2) THE REFERENCE SPHEROID IS IN THE SGA COMMON
0043     C          *
0044     C          ***********************************************************
0045     C
0046            PARAMETER PI = 3.141592654
0047     C
0048            PARAMETER OK=0
0049            PARAMETER ERROR = -1
0050     C
0051     C
0052            INCLUDE 'ZDBPRO.COM'
00100    0053   1 C
00200    0054   1 C       DUMMY COMMON ZDBPRO
00300    0055   1 C
00400    0056   1         INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500    0057   1         INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(5
```

```
00600   0058  1        LOGICAL*1 ZTSIC(3),ZTDWN(25)
00700   0059  1 C
00800   0060  1        COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900   0061  1       2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000   0062  1       3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
01100   0063  1 C
        0064           BYTE LAT(2),LONG(2),IHEMIS
        0065           REAL*8 FLAT,FLONG,CMERID
        0066     C
        0067           INTEGER *4 NORTH,IEAST
        0068     C
        0069           INTEGER *2 IDEG,MIN
        0070     C
        0071     C           INITIALIZE ERR_FLAG_LCS TO 'OK' (=0)
        0072     C
        0073           IERFLG=OK
        0074     C
        0075     C           ...CONVERT ASCII LAT/LONG TO FLOATING POINT RADIANS
        0076     C
        0077     C
        0078     C           ...FIRST THE LATITUDE
        0079     C
        0080     C
        0081     C           ...DECODE LAT_LCS INTO THE VARIABLES DEGREES,MINUTES,
        0082     C           ...AND HEMISPHERE
        0083     C           DECODE CHARACTER DATA ACCORDING TO FORMAT SPECIFICATI
        0084     C
        0085 C   D      WRITE (5,1) LAT,LONG
        0086 C   D1     FORMAT (' LAT= ',9A1,' LONG= ',10A1)
        0087           DECODE (9,1001,LAT,ERR=9999) IDEG,MIN,SEC,IHEMIS
        0088      1001 FORMAT (2I2,F4.1,A1)
        0089     C
        0090     C      CHECK FOR VALID LIMITS OF LATITUDE
        0091     C
        0092           IF(IDEG.LT.90 .AND. MIN.LT.60 .AND. SEC.LT.60.)  THEN
        0093     C
        0094     C      CONVERT TO RADIANS
        0095     C
        0096           FLAT = (DFLOTI(IDEG) + DFLOTI(MIN)/60. + DBLE(SEC)/3600.)
        0097     C
        0098     C           IF HEMISPHERE IS 'S' CHANGE SIGN
        0099     C
        0100                 IF (IHEMIS.EQ.'S')  THEN
        0101     C
        0102     C           SET LAT_COORD TO NEGATIVE LAT_COORD
        0103     C
        0104                 FLAT = -FLAT
        0105     C
        0106     C           ELSEIF HEMISPHERE IS NOT 'N' SET ERROR FLAG
        0107     C
        0108                 ELSEIF (IHEMIS.NE.'N')  THEN
        0109 C   D           WRITE (5,2)
        0110 C   D2          FORMAT (' ERROR--HEMISPHERE IS NOT NORTH')
        0111                 IERFLG = ERROR
        0112                 RETURN
        0113                 ENDIF
        0114           ELSE
```

```
0115 C  D      WRITE (5,3)
0116 C  D3     FORMAT (' DEG IS GT 90 OR MIN IS GT 60 OR SEC IS GT 60.')
0117           IERFLG = ERROR
0118           RETURN
0119           ENDIF
0120     C
0121     C           ...NOW DO LONGITUDE
0122     C
0123     C           ...DECODE LONG_LCS INTO THE VARIABLES DEGREES,MINUTES
0124     C           ...AND HEMISPHERE
0125     C
0126     C           DECODE CHARACTER DATA ACCORDING TO FORMAT SPECIFICATI
0127     C
0128           DECODE(10,1002,LONG,ERR=9999) IDEG,MIN,SEC,IHEMIS
0129      1002 FORMAT (I3,I2,F4.1,A1)
0130     C
0131     C      CHECK FOR VALID LONGITUDE LIMITS
0132     C
0133           IF (IDEG.LE.180 .AND. MIN.LT.60 .AND. SEC.LT.60.)  THEN
0134     C
0135     C      CONVERT TO RADIANS
0136     C
0137           FLONG = (DFLOTI(IDEG)+DFLOTI(MIN)/60.+ DBLE(SEC)/3600.)*PI/
0138     C
0139     C           IF HEMISPHERE IS 'W' CHANGE SIGN OF LONG
0140     C
0141                 IF (IHEMIS.EQ.'W')  THEN
0142     C
0143     C           SET LONG_COORD TO NEGATIVE LONG_COORD
0144     C
0145                 FLONG = -FLONG
0146     C
0147     C           ELSEIF HEMISPHERE IS NOT 'E' SET ERROR FLAG
0148     C
0149                 ELSEIF (IHEMIS.NE.'E')  THEN
0150 C  D            WRITE (5,4)
0151 C  D4           FORMAT (' HEMISPHERE IS NOT WEST')
0152     C
0153                 IERFLG = ERROR
0154                 RETURN
0155                 ENDIF
0156           ELSE
0157 C  D      WRITE (5,5)
0158 C  D5     FORMAT (' DEG IS GT 180 OR MIN IS GT 60 OR SEC IS GT 60.')
0159           IERFLG = ERROR
0160           RETURN
0161           ENDIF
0162     C
0163     C           ...HAVE LAT/LONG IN RADIANS, NOW PROJECT INTO CYLINDER
0164     C
0165     C
0166     C           DETERMINE THE CENTRAL MERIDIAN OF A GRIDZONE(REF_GRID_
0167     C
0168           CALL ADSCCM(ZONE,CMERID)
0169     C
0170     C           SET SPHEROID PARAMETERS(REF_SPHEROID)
0171     C
```

```
0172            CALL ADSSSP(ZSPHID)
0173      C
0174      C           PERFORM UTM PROJECTION
0175      C
0176            CALL ADSMP(FLAT,FLONG,CMERID,FEAST,FNORTH)
0177      C
0178            IEAST = FEAST
0179            NORTH = FNORTH
0180      C
0181            RETURN
0182      C
0183      C     ERROR EXIT FOR DECODE
0184      C
0185       9999 IERFLG = ERROR
0186 C    D     WRITE (5,6)
0187 C    D5    FORMAT (' DECODE ERROR')
0188            RETURN
0189            END
```

```
0001          SUBROUTINE ADSCNI(IGLN,IGRIDN,LNORTH,N100KN,IERFLG)
0002    C
0003    C         ***************************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCNI -- PERFORM 100K NORTHING ID LETTER TO INTEGER
0007    C         *                NORTHING CONVERSION
0008    C         *
0009    C         *  PURPOSE:
0010    C         *      TO CONVERT THE 100K NORTHING ID LETTER TO A
0011    C         *      NUMBER WHICH CORRESPONDS TO THE NUMBER OF 100K SQUARES
0012    C         *      FROM THE EQUATOR (MODULO 20)
0013    C         *
0014    C         *  DESCRIPTION:
0015    C         *      AUTHOR - P. W. DENNIS
0016    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0017    C         *      MOD LEVEL              DATE          DR NUMBERS
0018    C         *         01                 102979          DR 00009
0019    C         *
0020    C         *
0021    C         *  CALLING SEQUENCE:
0022    C         *      CALL ADSCNI (IGLN,IGRIDN,LNORTH,N100KN,IERFLG)
0023    C         *      WHERE:
0024    C         *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0025    C         *
0026    C         *      IGRIDL             GRID_LETTER     THE GRIDZONE LETTER
0027    C         *
0028    C         *      IGLN               GRID_LETNUM        A NUMBER CORRESPOND
0029    C         *                                            TO GRID_LETTER
0030    C         *
0031    C         *      IERFLG             ERR_FLAG_UCS       ERROR FLAG (-1 = ER
0032    C         *
0033    C         *
0034    C         *  INPUT/OUTPUT:
0035    C         *
0036    C         *       NONE
0037    C         *
0038    C         *  RESTRICTIONS:
0039    C         *   NONE
0040    C         *
0041    C         ***************************************************************
0042    C
0043    C
0044          PARAMETER OK =0
0045          PARAMETER ERROR = -1
0046    C
0047          BYTE LNORTH
0048    C
0049    C      NORTHING LETTER IDS GO FROM 'A-V' (NO 'I' OR 'O') AND START OVER
0050    C      EVERY 2,000,000 METERS (20 SQUARES). IN THE NORTHERN HEMISPHERE
0051    C      STARTING AT THE EQUATOR, LETTERS START WITH 'A' FOR THE ODD GRID
0052    C      NUMBERS AND WITH 'F' FOR THE EVEN GRID NUMBERS. SO, FOR ODD GRID
0053    C      NUMBERS, 'A' IS REPRESENTED BY 0, 'B' BY 1, 'C' BY 2,..., 'V' BY
0054    C      AND FOR EVEN GRID NUMBERS, 'F' IS REPRESENTED BY 0, 'G' BY 1, 'H'
0055    C      2,..., 'V' BY 14, 'A' BY 15,..., 'E' BY 19.
0056    C      IN THE SOUTHERN HEMISPHERE STARTING AT THE EQUATOR, LETTER START
0057    C      FROM V AND GO BACKWARDS FOR ODD GRID NUMBERS, AND FROM E AND GO
```

```
0058    C       BACKWARDS FOR EVEN GRID NUMBERS. SO, FOR ODD GRID NUMBERS, 'V' IS
0059    C       REPRESENTED BY -1, 'U' BY -2, 'T' BY -3,..., 'A' BY -19, AND FOR
0060    C       EVEN GRID NUMBERS, 'E' IS REPRESENTED BY -1, 'D' BY -2,..., 'A' B
0061    C       'V' BY -o,..., 'F' BY -19
0062    C
0063    C
0064    C
0065    C
0066    C            IF NORTH_100K_LET IS GREATER OR EQUAL TO 65 (ASCII 'A')
0067    C
0068            IF (LNORTH.GE.65) THEN
0069    C
0070    C            IF NORTH_100K_LET IS LESS THAN OR EQUAL TO 8o (ASCII 'V')
0071    C
0072            IF (LNORTH.LE.8o) THEN
0073    C
0074    C            IF NORTH_100K_LET IS LESS THAN 73 (ASCII 'I')
0075    C
0076            IF (LNORTH.LT.73) THEN
0077    C
0078    C            SET NORTH_100K_NUM TO NORTH_100K_LET MINUS 65
0079    C
0080            N100KN = LNORTH - 65
0081    C
0082    C            ELSE IF NORTH_100K_LET IS GREATER THAN 73
0083    C
0084            ELSE IF (LNORTH.GT.73) THEN
0085    C
0086    C            IF NORTH_100K_LET IS LESS THAN 79 (ASCII 'O')
0087    C
0088            IF (LNORTH.LT.79) THEN
0089    C
0090    C            SET NORTH_100K_NUM TO NORTH_100K_LET MINUS 66
0091    C
0092            N100KN = LNORTH - 66
0093    C
0094    C            ELSE IF NORTH_100K_LET IS GREATER THAN 79
0095    C
0096            ELSE IF (LNORTH.GT.79) THEN
0097    C
0098    C            SET NORTH_100K_NUM TO NORTH_100K_LET MINUS 67
0099    C
0100            N100KN = LNORTH - 67
0101    C
0102            ELSE
0103    C
0104    C            SET ERR_FLAG_UCS TO -1
0105    C
0106            IERFLG = ERROR
0107    C
0108    C            END IF
0109            END IF
0110    C
0111    C
0112            ELSE
0113    C
0114    C            SET ERR_FLAG_UCS TO -1
```

```
C
          IERFLG = ERROR
C
C              END IF
          END IF
C
          ELSE
C
C              SET ERR_FLAG_UCS TO -1
C
          IERFLG = -1
C
          END IF
C
          ELSE
C
C              SET ERR_FLAG_UCS TO -1
C
          IERFLG = ERROR
C
          END IF
C              IF ERR_FLAG_UCS IS 'OK' (=0)
C
          IF (IERFLG.EQ.OK) THEN
C
C              IF INP_GRID_NUM IS EVEN
C
          IF (MOD(IGRIDN,2).EQ.0) THEN
C
C              SET NORTH_100K_NUM TO NORTH_100K_NUM MINUS 5
C
          N100KN = N100KN - 5
C
C              IF NORTH_100K_NUM IS LESS THAN ZERO
C
          IF (N100KN.LT.0) THEN
C
C              SET NORTH_100K_NUM TO NORTH_100K_NUM PLUS 20
C
          N100KN = N100KN + 20
C
C
          END IF
C
          END IF
C
C              IF GRID_LETNUM IS LESS THAN ZERO
C
          IF (IGLN.LT.0) THEN
C
C              SET NORTH_100K_NUM TO NORTH_100K_NUM MINUS 20
C
          N100KN = N100KN - 20
C
          END IF
C
          END IF
```

```
0172          RETURN
0173          END
```

```
0001            SUBROUTINE ADSCNL(NORTH,IEAST,LAT,LONG,IERFLG)
0002     C
0003     C        ***************************************************************
0004     C        *
0005     C        *  NAME:
0006     C        *      ADSCNL -- EASTING-NORTHING TO LATLONG CONVERSION
0007     C        *
0008     C        *  PURPOSE:
0009     C        *      TO CONVERT EASTING NORTHING TO
0010     C        *      ASCII LATITUDE AND LONGITUDE
0011     C        *
0012     C        *  DESCRIPTION:
0013     C        *      AUTHOR - P. W. DENNIS
0014     C        *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015     C        *      MOD LEVEL                DATE          DR NUMBERS
0016     C        *         01                   102979          DR 00009
0017     C        *
0018     C        *
0019     C        *  CALLING SEQUENCE:
0020     C        *      CALL ADSCNL (NORTH,IEAST,LAT,LONG,IERFLG)
0021     C        *      WHERE:
0022     C        *      ARGUMENT NAME      PDL DATA NAME      DESCRIPTION
0023     C        *
0024     C        *      IEAST              EAST_LCS           UTM EASTING
0025     C        *
0026     C        *       NORTH             NORTH_LCS          UTM NORTHING
0027     C        *
0028     C        *      IERFLG             ERR_FLAG_LCS       ERROR FLAG (-1 = ER
0029     C        *
0030     C        *
0031     C        *      LAT                LAT_LCS            LATITUDE (ASCII)
0032     C        *
0033     C        *      LONG               LONG_LCS           LONGITUDE (ASCII)
0034     C        *
0035     C        *  INPUT/OUTPUT:
0036     C        *
0037     C        *      NONE
0038     C        *
0039     C        *  RESTRICTIONS:
0040     C        *
0041     C        *      (1) THE REFERENCE GRIDZONE IS IN THE SGA COMMON
0042     C        *      (2) THE REFERENCE SPHEROID IS IN THE SGA COMMON
0043     C        *
0044     C        *
0045     C        ***************************************************************
0046     C
0047            PARAMETER PI = 3.141592654
0048     C
0049     C
0050     C
0051            INCLUDE 'ZDBPRO.COM'
00100    0052   1 C
00200    0053   1 C      DUMMY COMMON ZDBPRO
00300    0054   1 C
00400    0055   1        INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500    0056   1        INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(5
00600    0057   1        LOGICAL*1 ZTSIC(3),ZTDWN(25)
```

```
00700   0058  1 C
00800   0059  1        COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900   0060  1       2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000   0061  1       3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
01100   0062  1 C
        0063           BYTE LAT(1),LONG(1)
        0064           REAL*8 FLAT,FLONG,CM
        0065     C
        0066           INTEGER *4 NORTH,IEAST
        0067     C
        0068           FEAST = IEAST
        0069           FNORTH = NORTH
        0070     C
        0071     C          ...COMPUTE THE CENTRAL MERIDIAN OF THE REFERENCE GRIDZ
        0072     C          DETERMINE THE CENTRAL MERIDIAN OF A GRIDZONE(REF_GRID_
        0073     C
        0074           CALL ADSCCM(ZRGN,CM)
        0075     C
        0076     C          ...NORMALIZE TO A COMMON PROJECTION BY FINDING THE LAT
        0077     C          ...THROUGH THE INVERSE UTM PROJECTION
        0078     C          SET SPHEROID PARAMETERS(REF_SPHEROID)
        0079     C
        0080           CALL ADSSSP(ZSPHID)
        0081     C
        0082     C          PERFORM INVERSE UTM PROJECTION
        0083     C
        0084           CALL ADSIMP(FEAST,FNORTH,CM,FLAT,FLONG)
        0085     C
        0086     C          ...CONVERT FLOATING POINT RADIANS TO ASCII DEG,MIN AND
        0087     C          SET LAT_COORD EQUAL TO LAT_COORD TIMES 180 DIVIDED BY
        0088     C
        0089           FLAT = FLAT*180./PI
        0090     C
        0091     C          SET LONG_COORD EQUAL TO LONG_COORD TIMES 180 DIVIDED B
        0092     C
        0093           FLONG = FLONG*180./PI
        0094     C
        0095     C          ...CONVERT FROM DECIMAL TO D,M,S
        0096     C          CONVERT DECIMAL DEGREES TO DEGREES-MINUTES-SECONDS
        0097     C
        0098           CALL ADSCDD(FLAT,IDEG,MIN,SEC)
        0099     C
        0100     C          ...PUT VALUES IN LAT_LCS USING ENCODE
        0101     C          ENCODE DATA ACCORDING TO FORMAT SPECIFICATION
        0102     C
        0103           ENCODE (9,1001,LAT,ERR=9999) IDEG,MIN,SEC
        0104      1001 FORMAT (2I2,F4.1)
        0105     C
        0106     C          ...SET HEMISPHERE
        0107     C
        0108     C       IF SOUTHERN HEMISPHERE
        0109     C
        0110           IF (FLAT.LT.0.)  THEN
        0111     C          SET LAT_LCS(9) TO 'S'
        0112     C
        0113           LAT(9)='S'
        0114     C
```

```
C
      ELSE
C
C           ELSE SET LAT_LCS(9) TO 'N'
C
      LAT(9)='N'
      ENDIF
C           ...NOW DO LONGITUDE
C           CONVERT DECIMAL DEGREES TO DEGREES-MINUTES-SECONDS
C
      CALL ADSCDD(FLONG,IDEG,MIN,SEC)
C
C           ...PUT VALUES IN LONG_LCS USING ENCODE FUNCTION
C
C           ENCODE DATA ACCORDING TO FORMAT SPECIFICATION
C
      ENCODE (9,1002,LONG,ERR=9999) IDEG,MIN,SEC
 1002 FORMAT (I3,I2,F4.1)
C
C           ...SET HEMISPHERE
C           IF WESTERN HEMISPHERE
C
      IF (FLONG.LT.0.) THEN
C
C           SET LONG_LCS(10) TO 'W'
C
      LONG(10)='W'
C
C           ELSE SET LONG_LCS(10) TO 'E'
C
      ELSE
C
      LONG(10)='E'
      ENDIF
      DO 1000 IJK=1,9
      IF (LONG(IJK) .EQ. ' ')  THEN
      LONG(IJK)='0'
      ENDIF
      IF (LAT(IJK) .EQ. ' ') THEN
      LAT(IJK)='0'
      ENDIF
 1000 CONTINUE
      RETURN
C
C      ERROR EXIT FOR ENCODE
C
 9999 IERFLG=-1
      RETURN
      END
```

```
0001          SUBROUTINE ADSCNU(NORTH,IEAST,MGR,IERFLG)
0002    C
0003    C         ***********************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSCNU -- PERFORM EASTING/NORTHING TO MGR CONVERSION
0007    C         *
0008    C         *  PURPOSE:
0009    C         *      TO CONVERT AN INTERNAL FLOATING POINT EASTING/NORTHING
0010    C         *      TO AN ASCII MGR
0011    C         *
0012    C         *  DESCRIPTION:
0013    C         *      AUTHOR - P. W. DENNIS
0014    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0015    C         *      MOD LEVEL              DATE          DR NUMBERS
0016    C         *         01                 102979          DR 00009
0017    C         *
0018    C         *
0019    C         *  CALLING SEQUENCE:
0020    C         *      CALL ADSCNU (NORTH,IEAST,MGR,IERFLG)
0021    C         *      WHERE:
0022    C         *
0023    C         *      IEAST              EAST_UCS           UTM EASTING
0024    C         *
0025    C         *       NORTH             NORTH_UCS          UTM NORTHING
0026    C         *
0027    C         *      MGR                UTM_UCS            ASCII MGR STRING
0028    C         *
0029    C         *
0030    C         *
0031    C         *      IERFLG             ERR_FLAG_UCS       ERROR FLAG (-1 = ER
0032    C         *
0033    C         *  INPUT/OUTPUT:
0034    C         *
0035    C         *       NONE
0036    C         *
0037    C         *  RESTRICTIONS:
0038    C         *   (1) REFERENCE GRIDZONE IS IN SHARED GLOBAL AREA
0039    C         *   (2) REFERENCE SPHEROID IS IN SHARED GLOBAL AREA
0040    C         *
0041    C         ***********************************************************
0042    C
0043          PARAMETER PI = 3.141592654
0044    C
0045    C
0046          INCLUDE 'ZDBPRO.COM'
00100   0047  1 C
00200   0048  1 C      DUMMY COMMON ZDBPRO
00300   0049  1 C
00400   0050  1       INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500   0051  1       INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(56
00600   0052  1       LOGICAL*1 ZTSIC(3),ZTDWN(25)
00700   0053  1 C
00800   0054  1       COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900   0055  1      2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000   0056  1      3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
01100   0057  1 C
```

```
0058    C
0059            BYTE MGR(2),ALPHA(24)
0060    C
0061            REAL *8 FLAT,FLONG,CMERID
0062    C
0063            INTEGER *4 IE100,N100,NORTH,IEAST
0064    C
0065    C            SET ARRAY ALPHA_CHAR EQUAL TO LETTERS 'A-Z' EXCLUDING
0066    C
0067            DATA ALPHA              / 'A','B','C','D','E','F','G','H',
0068           2                          'J','K','L','M','N','P','Q','R',
0069           3                          'S','T','U','V','W','X','Y','Z'/
0070    C
0071            FEAST = IEAST
0072            FNORTH = NORTH
0073    C
0074    C            COMPUTE THE CENTRAL MERIDIAN OF THE REFERENCE GRIDZONE
0075    C            DETERMINE THE CENTRAL MERIDIAN OF A GRIDZONE(REF_GRID_
0076    C
0077            CALL ADSCCM (ZRGN,CMERID)
0078    C
0079    C           TO FIND THE MGR COORDINATES, THE POINT MUST BE NORMALIZ
0080    C           PROJECTION BY FINDING THE LATITUDE AND LONGITUDE THEN P
0081    C           A CYLINDER TO GET A NEW EASTING/NORTHING
0082    C
0083    C
0084    C            SET SPHEROID PARAMETERS(REF_SPHEROID)
0085    C
0086            CALL ADSSSP(ZSPHID)
0087    C
0088    C       PERFORM INVERSE UTM PROJECTION (EAST_UCS,NORTH_UCS,CENT_MER
0089    C                                        LONG_COORD)
0090    C
0091            CALL ADSIMP (FEAST,FNORTH,CMERID,FLAT,FLONG)
0092    C
0093    C       GET SPHEROID INDEX FROM LAT-LONG INDEXED SPHEROID TABLE
0094    C
0095            CALL ADSGSI (FLAT,FLONG,ISPHER,NLORIG)
0096    C
0097    C                COMPUTE THE GRID NUMBER OF THE INPUT POINT
0098    C
0099    C        SET INP_GRID_NUM TO INTEGER PART OF [LONG_COORD*180 DIVIDE
0100    C                                            + 186] DIVIDED BY 6
0101    C
0102            IGN =  (FLONG*180./PI + 186.)/6.
0103    C
0104    C            IF INP_GRID_NUM IS GREATER THAN 60
0105    C
0106            IF (IGN .GT. 60) THEN
0107    C
0108    C            SET INP_GRID_NUM TO INP_GRID_NUM - 60
0109    C
0110            IGN = IGN - 60
0111    C
0112    C            ELSEIF INP_GRID_NUM IS LESS THAN 1
0113    C
0114            ELSEIF (IGN .LT. 1)  THEN
```

```
0115    C
0116    C           SET INP_GRID_NUM TO INP_GRID_NUM + 60
0117    C
0118            IGN = IGN + 60
0119    C
0120            ENDIF
0121    C           IF NOT IN REFERENCE GRID ZONE OR SPHEROID TRANSFORM TO
0122    C
0123    C
0124    C       IF INP_SPHEROID IS NOT REF_SPHEROID OR INP_GRID_NUM IS NOT
0125    C
0126            IF (ISPHER.NE.ZSPHID .OR. IGN.NE.ZRGN) THEN
0127    C
0128    C               GET ACTUAL CENTRAL MERIDIAN FOR INPUT POINT
0129    C
0130            CALL ADSCCM(IGN,CMERID)
0131    C
0132    C           SET SPHEROID PARAMETERS(INP_SPHEROID)
0133    C
0134            CALL ADSSSP(ISPHER)
0135    C
0136    C               NOW PROJECT INTO CORRECT SPHEROID
0137    C
0138    C
0139    C               PERFORM UTM PROJECTION
0140    C
0141            CALL ADSMP (FLAT,FLONG,CMERID,CEAST,CNORTH)
0142    C
0143    C           ROUND TO NEAREST TEN METERS
0144    C
0145    C         SET EASTING_NEW_COORD TO 10 TIMES NEAREST INTEGER OF(EAST
0146    C                                                                DIVI
0147    C
0148            CEAST = 10.*ANINT(CEAST/10.)
0149    C
0150    C        SET NORTHING_NEW_COORD TO 10 TIMES NEAREST INTEGER OF(NORT
0151    C                                                                DIVI
0152    C
0153            CNORTH = 10.*ANINT(CNORTH/10.)
0154    C
0155    C
0156            ELSE
0157    C
0158    C           SET EASTING_NEW_COORD TO EAST_UCS
0159    C
0160            CEAST = FEAST
0161    C
0162    C           SET NORTHING_NEW_COORD TO NORTH_UCS
0163    C
0164            CNORTH = FNORTH
0165    C
0166    C
0167            ENDIF
0168    C
0169    C           DETERMINE THE GRIDZONE'S LETTER
0170    C
0171    C       SET GRID_INDEX EQUAL TO INTEGER PART OF(LAT_COORD*180/PI_CO
```

```
0172    C
0173            NDEX = ( FLAT*180./PI + 104. )/8.
0174    C
0175    C            ACCOUNT FOR IRREGULAR GRIDZONE LABELED BY 'X'
0176    C
0177    C            IF GRID_INDEX IS GREATER THAN 22
0178    C
0179            IF (NDEX.GT.22)  THEN
0180    C
0181    C            SET GRID_INDEX EQUAL TO 22
0182    C
0183            NDEX = 22
0184    C
0185    C
0186            ENDIF
0187    C
0188    C            SET GRID_LETTER EQUAL TO ALPHA_CHAR (GRID_INDEX)
0189    C
0190            IGLET = ALPHA(NDEX)
0191    C
0192    C  DETERMINE THE 100K SQUARE WITHIN THE GRIDZONE IN WHICH THE POI
0193    C
0194    C
0195    C      FIND THE FIRST 100K SQUARE EAST OF THE CENTRAL MERIDIAN OF
0196    C
0197    C
0198            CALL ADSCFE(IGN,IENUM)
0199    C
0200    C     FIND THE EASTING ID LETTER BY COUNTING THE NUMBER OF 100K SQ
0201    C     THE EASTING COORDINATE AND ADDING THIS TO THE NUMBER CORRESP
0202    C     1ST SQUARE EAST OF THE CENTRAL MERIDIAN TO GET AN INDEX INTO
0203    C     OF LETTERS REPRESENTING THE EASTING ID LETTER
0204    C
0205    C
0206    C     SET EAST_COUNT EQUAL TO (INTEGER PART OF(EASTING_NEW_COORD +
0207    C
0208            IECNT = (CEAST + 5.) /100 000.
0209    C
0210    C            IF EASTING_NEW_COORD IS NEGATIVE
0211    C
0212            IF (CEAST .LT. 0.) THEN
0213    C
0214    C            DECREMENT EAST_COUNT
0215    C
0216            IECNT = IECNT - 1
0217    C
0218            ENDIF
0219    C
0220    C            SET EAST_100K_LET TO ALPHA_CHAR(EAST_COUNT + EAST_100K
0221    C
0222            IELET = ALPHA(IECNT + IENUM)
0223    C
0224    C        FIND THE NORTHING ID LETTER BY COUNTING THE NUMBER OF 100
0225    C        THE NORTHING COORDINATE
0226    C
0227    C
0228    C  SET NORTH_COUNT TO INTEGER PART OF [(NORTHING_NEW_COORD + 5] /
```

```
0229    C
0230          NCNT = (CNORTH + 5.)/100 000.
0231    C
0232    C           SET NORTH_COUNT2 TO NORTH_COUNT
0233    C
0234          NCNT2 = NCNT
0235    C
0236    C           IF NORTHING_NEW_COORD IS LESS THAN ZERO
0237    C
0238          IF (CNORTH .LT. 0.)  THEN
0239    C
0240    C           SET NORTH_COUNT2 EQUAL TO 99 + NORTH_COUNT2
0241    C
0242          NCNT2 = 99 + NCNT2
0243    C
0244    C
0245          ENDIF
0246    C
0247    C       DETERMINE AN INDEX TO THE TABLE OF LETTERS FOR THE NORTHI
0248    C
0249    C
0250    C           SET NORTH_INDEX EQUAL TO ONE PLUS REMAINDER OF [[NORTH
0251    C
0252    C           NORTH_LETTER_ORIG] DIVIDED BY 20]
0253    C
0254          NDEX = 1 + MOD(NCNT2 + NLORIG,20)
0255    C
0256    C           SET NORTH_100K_LET EQUAL TO ALPHA_CHAR (NORTH_INDEX)
0257    C
0258          INLET = ALPHA(NDEX)
0259    C
0260    C  FIND EASTING COORDINATE WITHIN THE 100K SQUARE BY SUBTRACTING
0261    C  OF 100,000 FROM THE EASTING COORDINATE
0262    C
0263    C
0264    C      SET EAST_100K_COORD EQUAL TO INTEGER PART OF[EASTING_NEW_C
0265    C                                        EAST_COUNT*100,000] DIVIDED
0266    C
0267          IE100 = (CEAST - 100 000.*IECNT)/10.
0268    C
0269    C      FIND MGR NORTHING COORDINATE WITHIN THE 100K SQUARE BY SUB
0270    C      MULTIPLES OF 100,000 FROM THE NORTHING COORDINATE
0271    C
0272    C
0273    C           SET NORTH_100K_COORD EQUAL TO INTEGER PART OF[NORTHING
0274    C           NORTH_COUNT*100,000 ] DIVIDED BY 10
0275    C
0276          N100 = (CNORTH - 100 000.*NCNT) /10.
0277    C
0278    C           IF NORTH_100K_COORD IS LESS THAN 0
0279    C
0280          IF (N100 .LT. 0) THEN
0281    C
0282    C           SET NORTH_100K_COORD EQUAL TO 10,000 + NORTH_100K_COOR
0283    C
0284          N100 = N100 + 10 000
0285    C
```

```
C
      ENDIF
C
C       ENCODE THE COMPUTED VALUES, GRID_NUMBER, GRID_LETTER, EAS
C       NORTH_100K_LET, EAST_100K_COORD, AND NORTH_100K_COORD INT
C       13-CHARACTER ASCII STRING, UTM_UCS
C
C          ENCODE DATA ACCORDING TO FORMAT SPECIFICATION
C
      ENCODE (5,1001,MGR(9)) N100+10 000
      ENCODE (5,1001,MGR(5)) IE100+10 000
 1001 FORMAT (I5)
      ENCODE (5,1002,MGR) IGN,IGLET,IELET,INLET
 1002 FORMAT (I2,A1,A1,A1)
C
      RETURN
      END
```

```
00100   0001          SUBROUTINE ADSCUN(MGR,IN,IE,IERFLG)
00200   0002    C
00300   0003    C         ***********************************************************
00400   0004    C         *
00500   0005    C         *  NAME:
00600   0006    C         *      ADSCUN -- PERFORM MGR TO EASTING/NORTHING CONVER
00700   0007    C         *
00800   0008    C         *  PURPOSE:
00900   0009    C         *      TO CONVERT AN ASCII MGR TO INTERNAL
01000   0010    C         *      EASTING/NORTHING FORMAT
01100   0011    C         *
01200   0012    C         *  DESCRIPTION:
01300   0013    C         *      AUTHOR - P. W. DENNIS
01400   0014    C         *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
01500   0015    C         *      MOD LEVEL              DATE          DR NUMBERS
01600   0016    C         *         01                 102979          DR 00009
01700   0017    C         *
01800   0018    C         *
01900   0019    C         *  CALLING SEQUENCE:
02000   0020    C         *      CALL ADSCUN (MGR,IN,IE,IERFLG)
02100   0021    C         *      WHERE:
02200   0022    C         *      ARGUMENT NAME     PDL DATA NAME     DESCRIPTION
02300   0023    C         *
02400   0024    C         *      IE                EASTING  _COORD     UTM EASTIN
02500   0025    C         *
02600   0026    C         *      IN                NORTHING_COORD    UTM NORTHING
02700   0027    C         *
02800   0028    C         *      MGR               UTM_UCS           ASCII MGR ST
02900   0029    C         *
03000   0030    C         *
03100   0031    C         *      IERFLG            ERR_FLAG_UCS      ERROR FLAG (
03200   0032    C         *
03300   0033    C         *  INPUT/OUTPUT:
03400   0034    C         *
03500   0035    C         *       NONE
03600   0036    C         *
03700   0037    C         *  RESTRICTIONS:
03800   0038    C         *
03900   0039    C         *   (1) REFERENCE GRIDZONE IS IS SHARED GLOBAL AREA
04000   0040    C         *   (2) THE REFERENCE SPHEROID IS IN THE SHARED GLOBAL
04100   0041    C         *
04200   0042    C         ***********************************************************
04300   0043    C
04400   0044    C
04500   0045          INCLUDE 'ZDBPRO.COM'
00100   0046  1 C
00200   0047  1 C      DUMMY COMMON ZDBPRO
00300   0048  1 C
00400   0049  1       INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500   0050  1       INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(56
00600   0051  1       LOGICAL*1 ZTSIC(3),ZTDWN(25)
00700   0052  1 C
00800   0053  1       COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900   0054  1      2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000   0055  1      3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
01100   0056  1 C
04600   0057    C
```

```
04700   0058          PARAMETER OK = 0
04800   0059          PARAMETER ERROR = -1
04900   0060    C
05000   0061          BYTE LNORTH,LEAST,IGRIDL,MGR(2)
05100   0062    C
05200   0063          REAL *8 CMERID,FLAT,FLONG
05300   0064    C
05400   0065          INTEGER *4 NORTH,IEAST,N100,IE100,IN,IE
05500   0066    C
05600   0067    C           INITIALIZE ERR_FLAG_UCS TO 'OK' (=0)
05700   0068    C
05800   0069          IERFLG = OK
05900   0070    C
06000   0071    C      CONVERT THE NUMERIC ASCII OF THE MGR STRING INTO INTEGER F
06100   0072    C      BY DECODING THE FIRST 2 CHARACTERS INTO INP_GRID_NUM, AND
06200   0073    C      CHARACTERS INTO EAST_100K_COORD AND NORTH_100K_COORD RESPE
06300   0074    C
06400   0075          DECODE(13,1001,MGR,ERR=9999)IGRIDN,IGRIDL,LEAST,LNORTH,IE10
06500   0076     1001 FORMAT(I2,3A1,2I4)
06600   0077    C
06700   0078    C    CHECK IF WITHIN ONE GRID NUMBER
06800   0079    C
06900   0080          IDIF = IABS(ZRGN-IGRIDN)
07000   0081    C
07100   0082          IF (IDIF .GT. 1) THEN
07200   0083    C
07300   0084    C    REMEMBER : ZONES 1 AND 60 ARE ADJACENT
07400   0085    C
07500   0086          IF (IDIF .NE. 59)  THEN
07600   0087          IERFLG = ERROR
07700   0088          RETURN
07800   0089          ENDIF
07900   0090          ENDIF
08000   0091    C
08100   0092    C   CONVERT THE 3RD CHARACTER OF THE MGR STRING (GRIDZONE LETTER)
08200   0093    C   GRID_LETNUM, WHERE A POSITIVE NUMBER CORRESPONDS TO THE NORTH
08300   0094    C   PHERE AND A NEGATIVE NUMBER CORRESPONDS TO THE SOUTHERN HEMIS
08400   0095    C
08500   0096    C
08600   0097    C           PERFORM GRIDZONE LETTER TO NUMBER CONVERSION(GRID_LETT
08700   0098    C                                                       ERR_FLAG_
08800   0099    C
08900   0100          CALL ADSCGL(IGRIDL,IGLN,IERFLG)
09000   0101    C
09100   0102    C           IF ERR_FLAG_UCS IS 'OK' (=0)
09200   0103    C
09300   0104                IF (IERFLG.EQ.OK) THEN
09400   0105    C
09500   0106    C
09600   0107    C      CHECK IF WITHIN ONE GRID LETTER
09700   0108    C
09800   0109          IF (IABS(ZRGL-IGLN) .GT. 1)  THEN
09900   0110          IERFLG = ERROR
10000   0111          RETURN
10100   0112          ENDIF
10200   0113    C
10300   0114    C       CONVERT THE 5TH CHARACTER OF THE UTM STRING (100K NORTHIN
```

```
10400   0115   C        AN INTEGER NORTHING, NORTH_100K_NUM
10500   0116   C
10600   0117   C           PERFORM 100K NORTHING ID LETTER TO INTEGER NORTHING CO
10700   0118   C           (GRID_LETNUM,INP_GRID_NUM,NORTH_100K_LET,NORTH_100K_NU
10800   0119   C
10900   0120               CALL ADSCNI(IGLN,IGRIDN,LNORTH,N100KN,IERFLG)
11000   0121   C
11100   0122   C           IF ERR_FLAG_UCS IS 'OK' (=0)
11200   0123   C
11300   0124               IF (IERFLG.EQ.OK) THEN
11400   0125   C
11500   0126   C           ...DETERMINE A NUMBER, EAST_100K_NUM, CORRESPONDING TO
11600   0127   C           ...EAST OF THE CENTRAL MERIDIAN OF THE INPUT GRIDZONE
11700   0128   C           DETERMINE THE FIRST 100K SQUARE EAST OF THE CENTRAL ME
11800   0129   C           (INP_GRID_NUM,EAST_100K_NUM)
11900   0130   C
12000   0131               CALL ADSCFE(IGRIDN,IENUM)
12100   0132   C
12200   0133   C           ...COMPUTE THE EASTING, EAST_INT, MEASURED FROM THE CE
12300   0134   C           ...THE INPUT GRIDZONE, TO THE NEAREST 100K
12400   0135   C           COMPUTE THE EASTING TO THE NEAREST 100 KILOMETERS/
12500   0136   C           (EAST_100K_LET,EAST_100K_NUM,EAST_INT,ERR_FLAG_UCS)
12600   0137   C
12700   0138               CALL ADSCIE(LEAST,IENUM,IEAST,IERFLG)
12800   0139   C
12900   0140   C           IF ERR_FLAG_UCS IS 'OK' (=0)
13000   0141   C
13100   0142               IF (IERFLG.EQ.OK) THEN
13200   0143   C
13300   0144   C           SET EASTING_COORD EQUAL TO EAST_INT PLUS EAST_100K_COO
13400   0145               CEAST = IEAST + 10 * IE100
13500   0146   C
13600   0147   C           ...COMPUTE THE NORTHING, NORTH_INT, MEASURED FROM THE
13700   0148   C           ...NEAREST 100K
13800   0149   C           COMPUTE THE NORTHING TO THE NEAREST 100 KILOMETERS/
13900   0150   C           (GRID_LETNUM,INP_GRID_NUM,NORTH_100K_NUM,NORTH_INT,INP
14000   0151   C
14100   0152               CALL ADSCIN(IGLN,IGRIDN,N100KN,NORTH,ISPHER)
14200   0153   C
14300   0154   C           IF INP_SPHEROID IS NOT ZERO
14400   0155   C
14500   0156               IF (ISPHER.NE.0)  THEN
14600   0157   C
14700   0158   C           SET NORTHING_COORD EQUAL TO NORTH_INT PLUS NORTH_100K_
14800   0159   C
14900   0160               CNORTH = NORTH + 10 * N100
15000   0161   C
15100   0162   C           IF INP_GRID_NUM IS NOT EQUAL TO REF_GRID_NUM OR INP_SP
15200   0163   C
15300   0164   C            NOT EQUAL TO REF_SPHEROID
15400   0165   C
15500   0166               IF (IGRIDN.NE.ZRGN  .OR. ISPHER.NE.ZSPHID)  THEN
15600   0167   C
15700   0168   C           ...COMPUTE THE CENTRAL MERIDIAN OF THE INPUT GRIDZONE
15800   0169   C           DETERMINE THE CENTRAL MERIDIAN OF A GRIDZONE(INP_GRID_
15900   0170   C
16000   0171               CALL ADSCCM(IGRIDN,CMERID)
```

```
16100   0172    C
16200   0173    C           ...NORMALIZE THE EASTING/NORTHING, EASTING_COORD AND N
16300   0174    C           ...COMMON PROJECTION BY CALCULATING THE LATITUDE AND L
16400   0175    C           ...THEN A NEW EASTING/NORTHING.
16500   0176    C
16600   0177    C           SET SPHEROID PARAMETERS (INP_SPHEROID)
16700   0178    C
16800   0179                CALL ADSSSP(ISPHER)
16900   0180    C
17000   0181    C           PERFORM INVERSE UTM PROJECTION(EASTING_COORD,NORTHING_
17100   0182    C           CENT_MERID,LAT_COORD,LONG_COORD)
17200   0183    C
17300   0184                CALL ADSIMP(CEAST,CNORTH,CMERID,FLAT,FLONG)
17400   0185    C
17500   0186    C           ...COMPUTE THE CENTRAL MERIDIAN OF THE REFERENCE GRIDZ
17600   0187    C           DETERMINE THE CENTRAL MERIDIAN OF A GRIDZONE(REF_GRID_
17700   0188    C
17800   0189                CALL ADSCCM(ZRGN,CMERID)
17900   0190    C
18000   0191    C           SET SPHEROID PARAMETERS (REF_SPHEROID)
18100   0192    C
18200   0193                CALL ADSSSP(ZSPHID)
18300   0194    C
18400   0195    C           PERFORM UTM PROJECTION(LAT_COORD,LONG_COORD,CENT_MERID
18500   0196    C           EASTING_COORD,NORTHING_COORD)
18600   0197    C
18700   0198                CALL ADSMP(FLAT,FLONG,CMERID,CEAST,CNORTH)
18800   0199    C
18900   0200                ENDIF
19000   0201    C
19100   0202    C           SET EAST_UCS EQUAL TO EASTING_COORD
19200   0203    C
19300   0204                IE = CEAST
19400   0205    C
19500   0206    C           SET NORTH_UCS EQUAL TO NORTHING_COORD
19600   0207    C
19700   0208                IN = CNORTH
19800   0209    C
19900   0210                ELSE
20000   0211    C
20100   0212    C           SET ERR_FLAG_UCS
20200   0213    C
20300   0214                IERFLG = ERROR
20400   0215    C
20500   0216                ENDIF
20600   0217                ENDIF
20700   0218                ENDIF
20800   0219            ENDIF
20900   0220            RETURN
21000   0221    C
21100   0222    C          ERROR EXIT FOR DECODE
21200   0223    C
21300   0224     9999 IERFLG = ERROR
21400   0225            RETURN
21500   0226            END
```

```
0001            SUBROUTINE ADSGSI(FLAT,FLONG,ISPHER,LNORIG)
0002      C
0003      C          ***********************************************************
0004      C          *
0005      C          *  NAME:
0006      C          *      ADSGSI -- GET SPHEROID INDEX FROM LAT-LONG INDEXED
0007      C          *                SPHEROID TABLE
0008      C          *
0009      C          *  PURPOSE:
0010      C          *      TO DETERMINE IN WHICH SPHEROID A GIVEN LAT-LONG
0011      C          *      IS AND TO RETURN THE NORTHING LETTER ORIGIN
0012      C          *
0013      C          *  DESCRIPTION:
0014      C          *      AUTHOR - P. W. DENNIS
0015      C          *      LAST MODIFIED BY P. W. DENNIS ON 08 JAN 80
0016      C          *      MOD LEVEL              DATE          DR NUMBERS
0017      C          *         01                 102979          DR 00009
0018      C          *
0019      C          *
0020      C          *  CALLING SEQUENCE:
0021      C          *      CALL ADSGSI (FLAT,FLONG,ISPHER,LNORIG)
0022      C          *      WHERE:
0023      C          *      ARGUMENT NAME     PDL DATA NAME     DESCRIPTION
0024      C          *
0025      C          *
0026      C          *      FLAT              LAT_COORD         LATITUDE (RADIANS)
0027      C          *
0028      C          *      FLONG             LONG_COORD        LONGITUDE (RADIANS)
0029      C          *
0030      C          *      ISPHER            SPHEROID_INDEX    INDEX OF SPHEROID
0031      C          *
0032      C          *      NLORIG            NORTH_LETTER_ORIG ORIGIN OF NORTHING
0033      C          *                                          LETTER
0034      C          *
0035      C          *  INPUT/OUTPUT:
0036      C          *
0037      C          *       NONE
0038      C          *
0039      C          *  RESTRICTIONS:
0040      C          *   THE LAT-LONG INDEXED SPHEROID TABLE MUST RESIDE
0041      C          *   IN THE SGA
0042      C          *
0043      C          ***********************************************************
0044      C
0045            PARAMETER PI = 3.141592654
0046      C
0047            INCLUDE 'ZDBPRO.COM'
00100     0048  1 C
00200     0049  1 C       DUMMY COMMON ZDBPRO
00300     0050  1 C
00400     0051  1         INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500     0052  1         INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(56)
00600     0053  1         LOGICAL*1 ZTSIC(3),ZTDWN(25)
00700     0054  1 C
00800     0055  1         COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900     0056  1        2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000     0057  1        3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
```

```
01100   0058  1 C
          0059    C
          0060    C          COVERT RADIANS TO TENTHS OF DEGREES
          0061    C
          0062    C          TENTH_DEGREES = 1800.*LONG_COORD DIVIDED BY PI_CONST
          0063    C
          0064          TEND = 1800./PI * FLONG
          0065    C
          0066    C          SET LONG_INDEX TO  -1
          0067    C
          0068          LONG = -1
          0069    C
          0070    C          DO UNTIL LL_SPHEROID_TAB(LONG_INDEX) IS LESS THAN TENTH_
          0071    C          AND TENTH_DEGREES IS LESS THAN OR EQUAL TO LL_SPHEROID_
          0072    C
          0073    C        INCREMENT LONG_INDEX BY TWO
          0074    C
          0075    C
          0076     100  CONTINUE
          0077          LONG = LONG + 2
          0078          FLOW = ZLLST(LONG)
          0079          FUP  = ZLLST(LONG+2)
          0080          IF (.NOT.(FLOW .LT. TEND .AND. TEND .LE. FUP))
          0081         1      GOTO 100
          0082    C
          0083    C
          0084    C
          0085    C
          0086    C          SET LAT_INDEX TO LL_SPHEROID_TAB(LONG_INDEX PLUS ONE) -
          0087    C
          0088          LAT = ZLLST(LONG + 1) - 3
          0089    C
          0090    C          CONVERT RADIAN LATITUDE TO TENTHS OF DEGREES
          0091    C
          0092    C          TENTH_DEGREES = 1800.*LAT_COORD DIVIDED BY PI_CONST
          0093    C
          0094          TEND = 1800./PI * FLAT
          0095    C
          0096    C          DO UNTIL LL_SPHEROID_TAB(LAT_INDEX) IS LESS THAN TENTH_
          0097    C          TENTH_DEGREES IS LESS THAN OR EQUAL TO LL_SPHEROID_TAB(
          0098    C
          0099    C          INCREMENT LAT_INDEX BY THREE
          0100    C
          0101     200  CONTINUE
          0102          LAT = LAT + 3
          0103          FLOW = ZLLST(LAT)
          0104          FUP  = ZLLST(LAT+3)
          0105          IF(.NOT.(FLOW .LT. TEND .AND. TEND .LE. FUP))
          0106         1         GOTO 200
          0107    C
          0108    C
          0109    C
          0110    C
          0111    C          SET SPHEROID_INDEX TO LL_SPHEROID_TAB(LAT_INDEX+1)
          0112    C
          0113          ISPHER = ZLLST(LAT + 1)
          0114    C
```

```
C           SET NORTH_LETTER_ORIG TO LL_SPHEROID_TAB(LAT_INDEX+2)
C
      LNORIG = ZLLST(LAT + 2)
C
C           IF SPHEROID_INDEX IS NEGATIVE
C
      IF (ISPHER .LT. 0) THEN
C
C           ...SPHEROID JUNCTION NOT ALONG PARALLEL OR MERIDIAN
C           ...SO WE NEED TO INTERPOLATE LINEARLY
C
C           ... THIS CAPABILITY TO BE PROVIDED LATER
C
      ENDIF
      RETURN
      END
```

```
      SUBROUTINE ADSIMP(FEAST,FNORTH,CMERID,FLAT,FLONG)
C
C        ***************************************************************
C        *
C        *  NAME:
C        *      ADSIMP -- PERFORM INVERSE UTM PROJECTION
C        *
C        *  PURPOSE:
C        *      TO PROJECT THE INPUT UTM COORDINATES INTO THE
C        *      EARTH
C        *
C        *  DESCRIPTION:
C        *      AUTHOR - P. W. DENNIS
C        *      LAST MODIFIED BY P. E. KING ON  5 OCT 79
C        *      MOD LEVEL              DATE          DR NUMBERS
C        *         01                 102979          DR 00009
C        *         02                 120579          DR 00089
C        *
C        *
C        *  CALLING SEQUENCE:
C        *      CALL ADSIMP (FEAST,FNORTH,CMERID,FLAT,FLONG)
C        *      WHERE:
C        *      ARGUMENT NAME     PDL DATA NAME     DESCRIPTION
C        *
C        *      FEAST             EASTING_COORD     UTM EASTING
C        *
C        *      FNORTH            NORTHING_COORD    UTM NORTHING
C        *
C        *      CMERID            CENT_MERID        CENTRAL MERIDIAN
C        *
C        *                                          OF PROJECTION
C        *
C        *      FLAT              LAT_COORD         LATITUDE (RADIANS)
C        *
C        *      FLONG             LONG_COORD        LONGITUDE (RADIANS)
C        *
C        *  INPUT/OUTPUT:
C        *
C        *       NONE
C        *
C        *  RESTRICTIONS:
C        *   SPHEROID PARAMETERS MUST BE SET IN COMMON ADSCEAR
C        *
C        ***************************************************************
C
      PARAMETER K0=.9996
      INCLUDE 'ADCEAR.COM'
1 C
1 C
1     COMMON /ADCEAR/LSPHER,AXMAJ,AXMIN,A,B,C,E2
1 C
1 C    PURPOSE:
1 C      CONTAINS SPHEROID PARAMETERS
1 C
1 C
1 C        MOD LEVEL          DATE        DR NUMBERS
1 C           01             110979        DR 00009
```

```
0058  1 C
0059  1 C
0060  1 C          VARIABLE       PDL DATA NAME          DESCRIPTION
0061  1 C
0062  1 C          LSPHER                                LAST SPHEROID USED
0063  1 C          AXMAJ          SEMI_MAJ               SEMI-MAJOR AXIS OF
0064  1 C                                                CURRENT SPHEROID
0065  1 C          AXMIN          SEMI_MIN               SEMI-MINOR AXIS OF
0066  1 C                                                CURRENT SPHEROID
0067  1 C          A              A                      1 ST MERIDIONAL ARC
0068  1 C                                                COEFFICIENT
0069  1 C          B              B                      2 ND MERIDIONAL ARC
0070  1 C                                                COEFFICIENT
0071  1 C          C              C                      3 RD MERIDIONAL ARC
0072  1 C                                                COEFFICIENT
0073  1 C          E2             E2                     SPHEROID ECCENTRICITY
0074  1 C                                                SQUARED
0075  1 C
0076  1 C
0077  1 C                    LAST MODIFIED BY P. E. KING ON 9 NOV 79
0078  1 C  ***************************************************************
0079           REAL*8 PSI,BZ,KZ,KY,AQUAD,BQDB2,CQUAD,X,Y,Z,ROOT,FLAT,FLONG
0080           REAL *8 CMERID
0081 C   D     WRITE (5,*) A,B,C,AXMAJ,AXMIN
0082     C
0083     C...COMPUTE PSI (THE MERIDIONAL ARC PARAMETER)
0084     C
0085           PSI=(FNORTH+B*DSIN(2.*DBLE(FNORTH/A))+
0086          !                C*DSIN(4.*DBLE(FNORTH/A)))/A
0087     C
0088     C      COMPUTE SLOPES AND INTERCEPTS OF PROJECTION RAY
0089     C
0090           BZ = - FEAST / (3.*K0)
0091           KZ = 2.* BZ / ( DBLE(AXMAJ) * DCOS (PSI) )
0092           KY = DTAN ( PSI )
0093     C
0094     C....SOLVE QUADRATIC EQUATION FOR INTERSECTION OF PROJECTION RAY
0095     C....WITH SPHEROID; BUT FIRST CALCULATE COEFFICIENTS.
0096     C
0097           AQUAD = 1 + KY*KY + KZ*KZ
0098           BQDB2 = KZ*BZ
0099           CQUAD = BZ*BZ - DBLE(AXMAJ)*DBLE(AXMAJ)
0100           X = ( -BQDB2 + DSQRT ( BQDB2 * BQDB2 -  AQUAD*CQUAD )) / AQUAD
0101           Y = AXMIN*KY*(X / AXMAJ)
0102           Z = KZ * X + BZ
0103           ROOT = DSQRT (DBLE(AXMIN)*DBLE(AXMIN) - Y*Y)
0104           FLAT = DATAN2 (DBLE(AXMAJ/AXMIN) * Y  ,ROOT )
0105           FLONG = CMERID + DASIN ( -Z/ROOT * DBLE(AXMIN/AXMAJ))
0106           RETURN
0107           END
```

```
0001          SUBROUTINE ADSMP(FLAT,FLONG,CMERID,FEAST,FNORTH)
0002    C
0003    C         ***********************************************************
0004    C         *
0005    C         *  NAME:
0006    C         *      ADSMP -- PERFORM UTM PROJECTION
0007    C         *
0008    C         *  PURPOSE:
0009    C         *      TO PROJECT THE INPUT LAT AND LONG INTO THE
0010    C         *      TRANSVERSE PROJECTION CYLINDER
0011    C         *
0012    C         *  DESCRIPTION:
0013    C         *      AUTHOR - P. W. DENNIS
0014    C         *      LAST MODIFIED BY P. E. KING ON 28 OCT 79
0015    C         *      MOD LEVEL              DATE          DR NUMBERS
0016    C         *         01                 102979          DR 00009
0017    C         *
0018    C         *
0019    C         *  CALLING SEQUENCE:
0020    C         *      CALL ADSMP (FLAT,FLONG,CMERID,FEAST,FNORTH)
0021    C         *      WHERE:
0022    C         *      ARGUMENT NAME     PDL DATA NAME     DESCRIPTION
0023    C         *
0024    C         *      FEAST             EASTING_COORD     UTM EASTING
0025    C         *
0026    C         *      FNORTH            NORTHING_COORD    UTM NORTHING
0027    C         *
0028    C         *      CMERID            CENT_MERID        CENTRAL MERIDIAN
0029    C         *
0030    C         *                                          OF PROJECTION
0031    C         *
0032    C         *      FLAT              LAT_COORD         LATITUDE (RADIANS)
0033    C         *
0034    C         *      FLONG             LONG_COORD        LONGITUDE (RADIANS)
0035    C         *
0036    C         *  INPUT/OUTPUT:
0037    C         *
0038    C         *       NONE
0039    C         *
0040    C         *  RESTRICTIONS:
0041    C         *   SPHEROID PARAMETERS MUST BE SET IN COMMON ADSCEAR
0042    C         *
0043    C         ***********************************************************
0044          PARAMETER K0=.9996
0045          REAL*8 SLAT,SDLONG,CLAT,CDLONG,FLAT,FLONG,CMERID,PSI
0046          INCLUDE 'ADCEAR.COM'
0047    1 C
0048    1 C
0049    1     COMMON /ADCEAR/LSPHER,AXMAJ,AXMIN,A,B,C,E2
0050    1 C
0051    1 C    PURPOSE:
0052    1 C      CONTAINS SPHEROID PARAMETERS
0053    1 C
0054    1 C
0055    1 C        MOD LEVEL          DATE        DR NUMBERS
0056    1 C           01             110979        DR 00009
0057    1 C
```

```
0058  1 C
0059  1 C          VARIABLE       PDL DATA NAME          DESCRIPTION
0060  1 C
0061  1 C          LSPHER                                LAST SPHEROID USED
0062  1 C          AXMAJ          SEMI_MAJ               SEMI-MAJOR AXIS OF
0063  1 C                                                CURRENT SPHEROID
0064  1 C          AXMIN          SEMI_MIN               SEMI-MINOR AXIS OF
0065  1 C                                                CURRENT SPHEROID
0066  1 C          A              A                      1 ST MERIDIONAL ARC
0067  1 C                                                COEFFICIENT
0068  1 C          B              B                      2 ND MERIDIONAL ARC
0069  1 C                                                COEFFICIENT
0070  1 C          C              C                      3 RD MERIDIONAL ARC
0071  1 C                                                COEFFICIENT
0072  1 C          E2             E2                     SPHEROID ECCENTRICITY
0073  1 C                                                SQUARED
0074  1 C
0075  1 C
0076  1 C                 LAST MODIFIED BY P. E. KING ON 9 NOV 79
0077  1 C  ***************************************************************
0078    C
0079    C
0080    C....COMPUTE TRIG FUNCTIONS ONCE AND SAVE!
0081    C
0082          CLAT = DCOS(FLAT)
0083          SLAT = DSIN(FLAT)
0084          CDLONG = DCOS (FLONG - CMERID)
0085          SDLONG = DSIN (FLONG - CMERID)
0086    C
0087    C....COMPUTE PSI [THE MERIDIONAL ARC PARAMETER]
0088    C
0089          PSI=DATAN2(DBLE(AXMIN)*SLAT,DBLE(AXMAJ)*CLAT*CDLONG)
0090    C
0091    C....COMPUTE MERIDIONAL ARC [THE NORTHING]
0092    C....NORTHING_COORD = A_*PSI - B_*SIN [ 2*PSI ] - C_*SIN [ 4*PSI ]
0093    C
0094          FNORTH=DBLE(A)*PSI - DBLE(B)*DSIN(2.*PSI) - C*DSIN(4.*PSI)
0095    C
0096    C....COMPUTE EASTING
0097    C
0098    C....EASTING_COORD = [3*K0*SEMI_MAJ*SDLONG*CLAT] DIVIDED BY
0099    C...[2*SQUARE ROOT OF [CLAT*CLAT*CDLONG*CDLONG+[1-E2]*SLAT*SLAT] + /
0100    C....SQUARE ROOT OF [1-E2*SLAT*SLAT]]
0101    C
0102          FEAST=(3.*K0*DBLE(AXMAJ)*SDLONG*CLAT) /
0103         !        (2.*DSQRT(CLAT*CLAT*CDLONG*CDLONG+(1-E2)*SLAT*SLAT )
0104         !       + DSQRT(1-E2*SLAT*SLAT))
0105          RETURN
0106          END
```

```
0001            SUBROUTINE ADSSSP(ISPHER)
0002    C
0003    C          ***********************************************************
0004    C          *
0005    C          *  NAME:
0006    C          *      ADSSSP -- SETS SPHEROID PARAMETERS IN COMMON ADCEAR
0007    C          *
0008    C          *  PURPOSE:
0009    C          *      TO COMPUTE MERIDIONAL ARC PARAMETERS AND SQUARED
0010    C          *      ECCENTRICITY OF THE SPHEROID SPECIFIED BY ISPHER
0011    C          *
0012    C          *  DESCRIPTION:
0013    C          *      AUTHOR - P. W. DENNIS
0014    C          *      LAST MODIFIED BY P. E. KING ON  4 DEC 79
0015    C          *      MOD LEVEL              DATE          DR NUMBERS
0016    C          *         01                 102979          DR 00009
0017    C          *         02                 120479          DR 00064
0018    C          *
0019    C          *
0020    C          *  CALLING SEQUENCE:
0021    C          *      CALL ADSSSP (ISPHER)
0022    C          *      WHERE:
0023    C          *      ARGUMENT NAME     PDL DATA NAME     DESCRIPTION
0024    C          *
0025    C          *      ISPHER            INP_SPHEROID      SPHEROID INDEX
0026    C          *
0027    C          *
0028    C          *  INPUT/OUTPUT:
0029    C          *
0030    C          *       NONE
0031    C          *
0032    C          *  RESTRICTIONS:
0033    C          *   NONE
0034    C          *
0035    C          ***********************************************************
0036    C
0037            PARAMETER K0=.9996
0038            INCLUDE 'ADCEAR.COM'
0039  1 C
0040  1 C
0041  1         COMMON /ADCEAR/LSPHER,AXMAJ,AXMIN,A,B,C,E2
0042  1 C
0043  1 C       PURPOSE:
0044  1 C         CONTAINS SPHEROID PARAMETERS
0045  1 C
0046  1 C
0047  1 C          MOD LEVEL          DATE        DR NUMBERS
0048  1 C             01             110979        DR 00009
0049  1 C
0050  1 C
0051  1 C          VARIABLE      PDL DATA NAME          DESCRIPTION
0052  1 C
0053  1 C          LSPHER                               LAST SPHEROID USED
0054  1 C          AXMAJ         SEMI_MAJ               SEMI-MAJOR AXIS OF
0055  1 C                                               CURRENT SPHEROID
0056  1 C          AXMIN         SEMI_MIN               SEMI-MINOR AXIS OF
0057  1 C                                               CURRENT SPHEROID
```

```
0058  1 C          A              A                        1 ST MERIDIONAL ARC
0059  1 C                                                  COEFFICIENT
0060  1 C          B              B                        2 ND MERIDIONAL ARC
0061  1 C                                                  COEFFICIENT
0062  1 C          C              C                        3 RD MERIDIONAL ARC
0063  1 C                                                  COEFFICIENT
0064  1 C          E2             E2                       SPHEROID ECCENTRICITY
0065  1 C                                                  SQUARED
0066  1 C
0067  1 C
0068  1 C                  LAST MODIFIED BY P. E. KING ON 9 NOV 79
0069  1 C  ************************************************************
0070          INCLUDE 'ADSTAB.DAT'
0071  1 C     ********************************************************
0072  1 C
0073  1 C        TABLE OF SPHEROID AXES
0074  1 C
0075  1 C     ********************************************************
0076  1 C
0077  1       DIMENSION AAXIS(9),BAXIS(9)
0078  1 C
0079  1 C                          THE SEMI-MAJOR AXES
0080  1 C
0081  1       DATA  AAXIS            /
0082  1      1                         6378388.,      !    INTERNATIONAL
0083  1      2                         6378206.,      !    CLARKE 1866
0084  1      3                         6378249.,      !    CLARKE 1880
0085  1      4                         6377276.,      !    EVEREST
0086  1      5                         6377397.,      !    BESSEL
0087  1      6                         6378160.,      !    AUSTRALIAN NATIONAL
0088  1      7                         6377397.,      !    AIRY
0089  1      8                         6378155.,      !    FISCHER
0090  1      9                         6377304. /     !    MALAYAN
0091  1 C
0092  1 C                          THE SEMI-MINOR AXES
0093  1 C
0094  1       DATA  BAXIS            /
0095  1      1                         6356912.,      !    INTERNATIONAL
0096  1      2                         6356584.,      !    CLARKE 1866
0097  1      3                         6356515.,      !    CLARKE 1880
0098  1      4                         6356075.,      !    EVEREST
0099  1      5                         6356079.,      !    BESSEL
0100  1      6                         6356775.,      !    AUSTRALIAN NATIONAL
0101  1      7                         6356257.,      !    AIRY
0102  1      8                         6356774.,      !    FISCHER
0103  1      9                         6356102. /     !    MALAYAN
0104  1 C
0105  1 C
0106  1 C                MOD LEVEL            DATE                  DR NUMBERS
0107  1 C                   01               110979                 DR 00009
0108  1 C
0109  1 C
0110  1 C          LAST MODIFIED BY P. E. KING ON 9 NOVEMBER 79
0111  1 C
0112  1 C
0113  1 C  ************************************************************
0114  1 C     ********************************************************
```

```
C
C
C        THE FOLLOWING 'IF TEST' SHOULD BE PERFORMED
C        IN FUTURE IMPLEMENTATIONS IN ORDER TO AVOID
C        PERFORMING CALCULATIONS WHICH WILL BE REDUNDANT
C        FOR A MAJORITY OF OCCURENCES
C
C           IF INP_SPHEROID IS NOT EQUAL TO SPHEROID_INDEX
C
C           SET SPHEROID_INDEX EQUAL TO INP_SPHEROID
         LSPHER = ISPHER
C  D     WRITE (5,*) 'SET PARAMETERS FOR SPHEROID ',LSPHER
C           SET SEMI_MAJ BY PERFORMING TABLE LOOK-UP INTO SPHEROID_TAB
         AXMAJ = AAXIS(LSPHER)
C
C           SET SEMI_MIN BY PERFORMING TANLE LOOK-UP INTO SPHEROID_TAB
C
         AXMIN = BAXIS(LSPHER)
C           ...COMPUTE E2 [ECCENTRICITY SQUARED]
C           E2 = [ 1 - [SEMI_MIN DIVIDED BY SEMI_MAJ] ^ 2]
C
         E2 = 1. - (AXMIN/AXMAJ)**2
C
C           ...COMPUTE MERIDIONAL ARC EXPANSION COEFFICIENTS
C
C           A_ = SEMI_MAJ*K0*[ 1 - E2 DIVIDED BY 4 -3*E2*E2 DIVIDED BY 64
C
         A = AXMAJ*K0*(1. - E2/4. - 3.*E2*E2/64.)
C
C           B_ = SEMI_MAJ*K0*E2* [ 1 + E2 DIVIDED BY 4 ] DIVIDED BY 8
C
         B = AXMAJ*K0*E2*(1. + E2/4.) /8.
C
C           C_ = SEMI_MAJ*K0*E2*E2 DIVIDED BY 256
C
         C = AXMAJ*K0*E2*E2 /256.
C
C        ENDIF WILL GO HERE
C
         RETURN
         END
```

```
      SUBROUTINE DRWCON(WINXY,DBRES,CONRES,ZMIN,ZMAX,ZDELT)
*****************************************************************
*       DRAWS ELEVATION CONTOURS                                *
*****************************************************************
*       INPUTS: WINXY--MIN,MAX OF EASTING AND NORTHING,RESPECTIVELY *
*               DBRES-- DATABASE RESOLUTION, USUALLY 100M       *
*               CONRES-- DISTANCE BETWEEN CONTOURS, AT LEAST DBRES *
*               ZMIN,ZMAX-- MINIMUM AND MAXIMUM ELEVATION IN DATA *
*               ZDELT-- CONTOUR RESOLUTION, USER-DEFINED        *
*       OUTPUTS: NONE                                           *
*       N.B. ALL UNITS ARE IN METERS.                           *
*       *** *H. JONES    R. LINDLEY ***                         *
*****************************************************************
      IMPLICIT INTEGER*2 (I-N)
      INCLUDE 'MASK.DIM'
1 ******************************************************
1     BYTE MASK(400,400,3)
1 ******************************************************
      DIMENSION WINXY(4)
C
C           SET LOWER LEFT AND UPPER RIGHT INDICES.
C
      ILL=1
      IUR=(WINXY(2)-WINXY(1))/DBRES
      JLL=1
      JUR=(WINXY(4)-WINXY(3))/DBRES
C
C           SET I,J INCREMENT.
C
      IJDELTA=MAX1(CONRES/DBRES,1.)
C
C           SET X,Y INCREMENT
C
      XYDELTA=DBRES*FLOAT(IJDELTA)
C
C           FORCE MINIMUM TO BE NON INTEGER.
C
      ZMIN=ANINT(ZMIN)+0.5
C
C           FORCE Z INCREMENT TO BE INTEGER.
C
      ZDELT=ANINT(ZDELT)
C
C    *DRAW CONTOURS.
      DO HEIGHT=ZMIN+ZDELT,ZMAX,ZDELT
C
C       *SET VALUE OF MASK FOR THE THREE SIDES THUS:
C       *VALUE OF  0   --   NOT CHECKED
C       *VALUE OF  1   --   INTERCEPT
C       *VALUE OF 10   --   NO INTERCEPT
        DO K=1,3
           DO J=JLL,JUR,IJDELTA
              DO I=ILL,IUR,IJDELTA
                 MASK(I,J,K) = 0
              ENDDO
           ENDDO
        ENDDO
```

```
C        *
C        *      IY+1
C        *      .........
C        *      ..       .
C        *      . .      .
C        *      .  .     .
C        * 2    .   .3   . IX+1
C        *      .    .   .
C        *      .     . .
C        *      .      ..
C        *      .........
C        *     IX,IY    1
C        *
C
C        *SCAN BOTTOM EDGE FOR TRACE STARTING POINTS.
          IY = JLL
          YY=FLOAT(IY/IJDELTA)*XYDELTA
          DO IX=ILL,IUR,IJDELTA
             ICROSS = 0
             IF(MASK(IX,IY,1) .EQ. 0) THEN
                 CALL INTERS(IX,IY,IX+
     +           IJDELTA,IY,HEIGHT,FRAC,ICROSS)
             ENDIF
             IF(ICROSS .EQ. 0) THEN
                MASK(IX,IY,1) = 10
             ELSE
                IENTER = 1
                ITOP = 0
                X=FLOAT(IX/IJDELTA)*XYDELTA
                XX = X + XYDELTA * FRAC
                CALL MOVE (XX,YY)
                IXX=IX
                IYY=IY
                CALL TRACE (IXX,IYY,IENTER,ITOP,HEIGHT,MASK,
     $                      XYDELTA,ILL,IUR,JLL,JUR,IJDELTA)
             ENDIF
          ENDDO
C
C        *SCAN SIDE 2 OF RESOLUTION ELEMENTS FOR TRACE STARTING POINTS.
          DO IY = JLL,JUR,IJDELTA
             Y=FLOAT(IY/IJDELTA)*XYDELTA
             DO IX = ILL,IUR,IJDELTA
C
                ICROSS = 0
                IF(MASK(IX,IY,2) .EQ. 0) THEN
                 CALL INTERS(IX,IY,IX,
     +           IY+IJDELTA,HEIGHT,FRAC,ICROSS)
                ENDIF
C
                IF(ICROSS .EQ. 0) THEN
                   MASK(IX,IY,2) = 10
                ELSE
                   ITOP = 0
                   IENTER = 2
                   XX=FLOAT(IX/IJDELTA)*XYDELTA
                   YY = Y + XYDELTA * FRAC
                   CALL MOVE (XX,YY)
```

```
                    IXX=IX
                    IYY=IY
                    CALL TRACE (IXX,IYY,IENTER,ITOP,HEIGHT,MASK,
                                XYDELTA,ILL,IUR,JLL,JUR,IJDELTA)
     $
                    ITOP = 1
                    IENTER = 2
                    CALL MOVE (XX,YY)
                    IXX=IX-IJDELTA
                    IYY=IY
                    CALL TRACE (IXX,IYY,IENTER,ITOP,HEIGHT,MASK,
                                XYDELTA,ILL,IUR,JLL,JUR,IJDELTA)
     $
                ENDIF
            ENDDO
        ENDDO
C
      ENDDO
      RETURN
      END
```

```
0001            SUBROUTINE FEATURES(INCR)
0002      ************************************************************
0003      *     THE FEATURE CODES ARE DISPLAYED ON A TEK 4027 BY     *
0004      *     DRAWING THE COLUMNS IN APPROPRIATE COLORS            *
0005      ************************************************************
0006      *     INPUTS:  INCR, THE INCRIMENT FOR MOVES AND DRAWS     *
0007      *              IT IS USUALLY SET TO 1 OR 2.                 *
0008      *     OUTPUTS: NONE                                        *
0009      ************************************************************
0010            INTEGER*2 INCR
0011            DIMENSION IWIN(4)
0012            INCLUDE 'WINDO.CMN'
0013    1 ************************************************************
0014    1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND  *
0015    1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER   *
0016    1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS *
0017    1 *     THE CONTOUR INTERVAL.                                *
0018    1 ************************************************************
0019    1       DIMENSION FWINXY(4)
0020    1       COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
0021    1 ************************************************************
0022            INCLUDE 'CORNER.CMN'
0023    1 ***********************************************************
0024    1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE     *
0025    1 *     AREA IN THE ARRAY IBUF.                              *
0026    1       INTEGER*4 SWX,SWY
0027    1       COMMON/CORNER/SWX,SWY
0028    1 ***********************************************************
0029            INTEGER*2 IX,IY,INDX,INDY,ICODE
0030      C
0031      C.....  IWIN INDEXES IBUF,WHILE FWINXY SETS THE WINDOW
0032      C.....  FOR MOVES AND DRAWS
0033      C...    THE FOLLOWING IS A KLUDGE ON A MORE FLEXIBLE VERSION
0034      C...    OF THIS ROUTINE
0035            IWIN(1)=(FWINXY(1)-SWX)/100+1
0036            IWIN(2)=(FWINXY(2)-SWX)/100
0037            IWIN(3)=(FWINXY(3)-SWY)/100+1
0038            IWIN(4)=(FWINXY(4)-SWY)/100
0039 C    D     CALL CMCLOS
0040 C    D     PRINT*,INCR,IWIN
0041 C    D     READ*,JUNK
0042 C    D     CALL CMOPEN
0043      C     FOR EACH X
0044            CALL CMOPEN
0045            DO IX=IWIN(1),IWIN(2),INCR
0046               XI=(IX-1)*100.+SWX
0047      C        SET FEATURE CODE VALUE FOR SCAN
0048                    IC=ICODE(IWIN(3),IX)
0049                    YI=FWINXY(3)
0050      C             MOVE TO BOTTOM OF WINDOW FOR THIS SCAN
0051                    CALL MOVE(XI,YI)
0052      C             FOR EACH Y
0053                    DO IY=IWIN(3),IWIN(4),INCR
0054      C                THE POINTS IN THE DATA BASE ARE 100M APART
0055                       YI=(IY-1)*100.+SWY
0056      C                IF NO CHANGE,CONTINUE READING
0057                       IF(ICODE(IY,IX).EQ.IC)GOTO10
```

```
C.....                ELSE
C                        DRAW A LINE IN THE APPROPRIATE COLOR
C                        RED FOR 'URBAN' AND GREEN FOR 'FOREST'
                         CALL LINCLR(IC)
                         CALL DRAW(XI,YI)
C                     SET THE NEW FEATURE CODE
                      IC=ICODE(IY,IX)
10                    CONTINUE
                   ENDDO
C.....             DRAW TO THE TOP IF NECESSARY
                   CALL LINCLR(ICODE(IY-INCR,IX))
                   CALL DRAW(XI,YI)
            ENDDO
            CALL CMCLOS
            RETURN
            END
```

```
      SUBROUTINE FILLOP(N,VVERTS,ICOD)
*****************************************************************
*   N IS THE INPUT NUMBER OF ORDERED VERTICES WHICH SPECIFY THE
*   BOUNDARY OF THE SIMPLE CLOSED POLYGON. SINCE THE FIRST AND LAST
*   VERTEX ARE UNDERSTOOD TO BE IDENTICAL, THE NUMBER OF PHYSICAL
*   VERTICES IS N-1.
*   VVERTS IS THE ARRAY CONTAINING THE POLYGON VERTICES.
*      X-COORDINATE OF THE K-TH VERTEX = VVERTS(K,1)
*      Y-COORDINATE OF THE K-TH VERTEX =VVERTS(K,2)
*   IVERT(1,*)=IVERT(N,*) IS ASSUMED.
*
*   THE WORLD GRID POINTS LYING WITHIN THE POLYGON ARE FILLED BY THIS
*   SUBROUTINE.  THE METHOD OF OPERATION IS AS FOLLOWS. EACH VERTICAL
*   LINE THROUGH THE POLYGON IS EXAMINED. IN PRACTICE, THE LINE
*   INTERSECTS THE POLYGON BOUNDARY AT AN EVEN NUMBER OF POINTS
*   (IF NOT, THE LINE IS SHIFTED SLIGHTLY). THEN, FROM BOTTOM TO TOP,
*   WE "PAINT" THE LINE SEGMENTS BOUNDED BY THE FIRST AND SECOND
*   INTERSECTIONS, BY THE THIRD AND FOURTH INTERSECTIONS, ETC.
*
*   THE LINE SEGMENTS WHICH FORM THE POLYGON BOUNDARY ARE ORDERED BY
*   INCREASING VALUES OF X. THIS HELPS TO SPEED UP THE IDENTIFICATION
*   OF INTERSECTIONS BETWEEN THE BOUNDARY SEGMENTS AND THE VERTICAL
*   "PAINT LINES".
*****************************************************************
      DIMENSION VVERTS(500,2),VERTS(500,2),LOWLI(500),ICROSS(500)
      DIMENSION YHOLD(20),ILIST(20)
        INTEGER*2 IELV,ICOD
        INCLUDE 'MAP.CMN'
1 ***********************************************************
1 *       IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH    *
1 *       THE FIRST INDEX CORRESPONDS TO NORTHING, AND       *
1 *       THE SECOND TO EASTING.                             *
1 ***********************************************************
1       INTEGER*2 IBUF(400,400)
1     COMMON /MAP/IBUF
1 ***********************************************************
        INCLUDE 'WINDO.CMN'
1 ***********************************************************
1 *       FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND  *
1 *       MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER   *
1 *       TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS *
1 *       THE CONTOUR INTERVAL.                                *
1 ***********************************************************
1       DIMENSION FWINXY(4)
1       COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ***********************************************************
        INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *       SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE     *
1 *       AREA IN THE ARRAY IBUF.                              *
1       INTEGER*4 SWX,SWY
1       COMMON/CORNER/SWX,SWY
1 ***********************************************************
      NS=N-1
C
C   FIND THE MINIMUM & MAXIMUM VALUES OF X WITHIN THE POLYGON,
C   XXMIN AND XXMAX.  ALSO FING THE CENTER-OF-MASS- OF THE POLYGON VERTIC
```

```
C    (XCM,YCM).
C
        XXMIN=1.0E10
        XXMAX=-1.0E10
        YYMIN=1.0E10
        YYMAX=-1.0E10
        XCM=0.0
        YCM=0.0
        DO 10 I=1,NS
        XCM=XCM+VVERTS(I,1)
        YCM=YCM+VVERTS(I,2)
        IF(VVERTS(I,1).GT.XXMIN) XXMIN=VVERTS(I,1)
        IF(VVERTS(I,1).GT.XXMAX) XXMAX=VVERTS(I,1)
        IF(VVERTS(I,2).GT.YYMIN) YYMIN=VVERTS(I,2)
        IF(VVERTS(I,2).GT.YYMAX) YYMAX=VVERTS(I,2)
     10 CONTINUE
        XCM=XCM/NS
        YCM=YCM/NS
C
C    SHIFT EACH VERTEX SLIGHTLY (APPROXIMATELY AWAY FROM THE CM).
C    THIS SHIFT AVOIDS HAVING "WORKING" VERTICIES WHICH LIE DIRECTLY ATOP
C    WORLD GRID POINTS.
C
        DO 20 I=1,N
        VERTS(I,1)=VVERTS(I,1) + .0001*(VVERTS(I,1)-XCM)
        VERTS(I,2)=VVERTS(I,2) + .0001*(VVERTS(I,2)-YCM)
C
C    INITIALIZE AN ARRAY TO AID IN SORTING VECTORS.
C
        ICROSS(I)=0
     20 CONTINUE
C
C    THE POLYGON SIDE VECTOR NUMBERED I IS UNDERSTOOD TO HAVE ENDPOINTS
C    VERTS(I,*) AND VERTS(I+1,*).   THERE ARE N-1 SUCH VECTORS, AND WE NOW
C    SORT THEM ACCORDING TO INCREASING VALUES OF "MINIMUM X".
C    THE ARRAY LOWGI(*) CONTAINS THE RESULTS OF THE SORTING.
C    E.G., LOWGI(*)= 1,50,2,3,49,... WOULD MEAN THAT VECTOR NUMBER 1
C    HAS THE SMALLEST LEFTMOST X-COORDINATE, VECTOR NUMBER 50 IS SECOND
C    SMALLEST IN X, ETC.
C
        DO 100 K=1,NS
C                                FIND THE K-TH SMALLEST (IN X) VECTOR.
        XMIN=1.0E10
        DO 50 I=1,NS
        IF(ICROSS(I).EQ.1) GO TO 50
        X=AMIN1(VERTS(I,1),VERTS(I+1,1))
        IF(X.GT.XMIN) GO TO 50
        KEY=I
        XMIN=X
     50 CONTINUE
        LOWGI(K)=KEY
C                              CROSS OFF THE K-TH SMALLEST VECTOR
C                              FROM FURTHER CONSIDERATION.
        ICROSS(KEY)=1
    100 CONTINUE
C
C    FIND X-COORDINATES WHICH BRACKET THE WORLD REGION TO BE FILLED.
```

```
C
      IXXMIN=IFIX(XXMIN/100)*100 - 100
      IXXMAX=IFIX(XXMAX/100)*100 + 100
C
C   LOOP OVER VERTICAL COLUMNS OF GRID POINTS WITHIN THE POLYGON.
C
      KSTART=1
      DO 300 IC=IXXMIN,IXXMAX,100
      X=IC
C   FIND NUMBER OF INTERSECTIONS (NINTER) OF THE VERTICAL COLUMN
C   WITH THE POLYGON SIDE VECTORS.
  150 NINTER=0
      DO 200 K=KSTART,NS
      KSO=1
      I=LOWLI(K)
C
C            SEE IF VECTOR I INTERSECTS THE COLUMN LINE X=IC.
C
      IF(X.LT.AMIN1(VERTS(I,1),VERTS(I+1,1))) GO TO 210
      IF(X.GT.AMAX1(VERTS(I,1),VERTS(I+1,1))) GO TO 200
C
C   AN INTERSECTION HAS BEEN FOUND. CALCULATE THE Y-COORDINATE (Y)
C   OF THE INTERSECTION POINT.
C
      X1=VERTS(I,1)
      X2=VERTS(I+1,1)
      Y1=VERTS(I,2)
      Y2=VERTS(I+1,2)
      SLOPE=(Y2-Y1)/(X2-X1)
      Y=SLOPE*(X-X1)+Y1
      NINTER=NINTER+1
      YHOLD(NINTER)=Y
C   FOR THE FIRST INTERSECTION, REDEFINE ISTART FOR THE NEXT COLUMN.
      IF(NINTER.EQ.1) KSO=K
  200 CONTINUE
  210 KSTART=KSO
C   MAKE SURE THAT THERE ARE AN EVEN NUMBER OF INTERSECTIONS.
      IF(MOD(NINTER,2).EQ.0) GO TO 220
      X=X+1.0
      GO TO 150
  220 IF(NINTER.EQ.0) GO TO 300
C
C   SORT THE Y-COORDINATES OF THE INTERSECTION POINTS BY INCREASING
C   VALUE. ILIST(*)=3,7,2,... MEANS THAT THE THIRD INTERSECTION FOUND HAS
C   THE SMALLEST Y-COORDINATE, THE 7-TH INTERSECTION HAS THE NEXT
C   SMALLEST Y, ETC.
C   NOTE THAT NOW ICROSS(*)=1
      DO 225 K=1,NINTER
  225 ICROSS(K)=1
C
      DO 250 K=1,NINTER
      YMIN=1.0E10
      DO 230 L=1,NINTER
      IF(ICROSS(L).EQ.0) GO TO 230
      Y=YHOLD(L)
      IF(Y.GT.YMIN) GO TO 230
      KEY=L
```

```
          YMIN=Y
      230 CONTINUE
          ILIST(K)=KEY
          ICROSS(KEY)=0
      250 CONTINUE
C
C   FOR TESTING, DRAW THE COLUMN ON THE TERMINAL SCREEN.
C
          DO 280 K=2,NINTER,2
          Y1=YHOLD(ILIST(K-1))
          Y2=YHOLD(ILIST(K))

            IX=IC/100
            IY1=YHOLD(ILIST(K-1))/100
            IY2=YHOLD(ILIST(K))/100
            DO IY=IY1,IY2
               IBUF(IY,IX)=IBUF(IY,IX)*8+ICOD
            ENDDO
          X=IC
      280 CONTINUE
C
      300 CONTINUE
            FWINXY(1)=SWX+XXMIN
            FWINXY(2)=SWX+XXMAX
            FWINXY(3)=SWY+YYMIN
            FWINXY(4)=SWY+YYMAX
C  D        CALL CMCLOS
C  D        PRINT*,FWINXY
C  D        CALL CMOPEN
          RETURN
          END
```

```
      SUBROUTINE GEN
*************************************************
*     THIS ROUTINE GENERATES THE FILE NAMES AND *
*     RELATIVE INDICES FOR REWRITING THE DATA   *
*     INTO THE TORM FILES; AND REWRITES THE DATA *
*************************************************
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE    *
1 *     AREA IN THE ARRAY IBUF.                             *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      CHARACTER*7 MGR
      LOGICAL*1 ERR
      DO J=0,3
         IEAST=SWX+J*10000
         DO I=0,3
            NORTH=SWY+I*10000
            CALL UTM2MGR(IEAST,NORTH,MGR,ERR)
            CALL MAPOUT(I,J,MGR,ERR)
         ENDDO
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE GETNDX(RECNUM,INDEX,POS)
       IMPLICIT INTEGER*4(A-Z)
C      A ROUTINE TO DETERMINE ON WHICH 500 WORD PHYSICAL RECORD
C      OF UNIT 'LU' THE 5 WORD LOGICAL RECORD 'RECNUM' RESIDES.
C
C      SUBTRACT 1 TO TAKE CARE OF MULTIPLES OF 100
      N=RECNUM-1
      INDEX=N/100+1
C      INDEX IS THE PHYSICAL RECORD INDEX
C      POS IS THE POSITION OF THE LOGICAL RECORD  WITHIN THE
C      PHYSICAL RECORD
      POS=RECNUM-(INDEX-1)*100
      RETURN
      END
```

```
0001            SUBROUTINE GETREC(LU,NUM,PREC)
0002            IMPLICIT INTEGER*4(A-Z)
0003             DIMENSION PRECNUM(3),PREC(500)
0004             DATA PRECNUM/3*0/
0005      C      A ROUTINE TO RETRIEVE PHYSICAL RECORD NUMBER 'NUM'
0006      C      FROM UNIT 'LU'
0007      C
0008      C      FIRST CHECK TO SEE IF THE PHYSICAL RECORD IS
0009      C      ALREADY IN CORE.
0010      C
0011 C    D        PRINT*,NUM,LU
0012            IF(NUM.EQ.PRECNUM(LU-1)) RETURN
0013            READ(LU'NUM) PREC
0014            PRECNUM(LU-1)=NUM
0015            RETURN
0016            END
```

```
0001            SUBROUTINE GETSUB(SRECNUM,WRDPOS,N)
0002      C*************************************************
0003      C*****  REVISED:  6/11/82
0004      C*****  FOR USE WITH ROADHEXER
0005      C*************************************************
0006      C
0007      C       THIS ROUTINE TAKES THE SUBNODE COORDINATES FOR
0008      C       ONE LINK AND PUTS THEM INTO THE ARRAY SUBXY
0009      C
0010            IMPLICIT INTEGER*4 (A-Z)
0011            REAL Y
0012            INTEGER*2 N
0013            INCLUDE 'LNKNOD.CMN'
0014    1 ***************************************************************
0015    1 *     ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES         *
0016    1 ***************************************************************
0017    1       INTEGER*4 GRID,NODREC,LNKREC,SUBREC
0018    1       COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
0019    1     + ,SUBREC(500)
0020    1 ****************************************************************
0021            INCLUDE 'SUB.CMN'
0022    1 ***********************************************************
0023    1 *     SUBX,SUBY THE X AND Y COORDINATES OF THE SUBNODES *
0024    1 *     IN ONE LINK (SEE BDM DOCUMENTATION)               *
0025    1 ***********************************************************
0026    1       INTEGER*2 SUBX(100),SUBY(100)
0027    1       COMMON/SUB/ SUBX,SUBY
0028    1 ***********************************************************
0029  C   D     PRINT*,'INPUTS:',SRECNUM,WRDPOS
0030            LU=4
0031                CALL GETREC(LU,SRECNUM,SUBREC)
0032      C    THE FIRST ENTRY IN THE SUBNODE LOGICAL RECORD IS TWICE
0033      C    THE NUMBER OF POINTS IN THE SUBNODE LIST
0034      C
0035                N=SUBREC(WRDPOS)/2
0036  C   D         PRINT*,'N',N
0037
0038      C    CHECK TO SEE IF ALL OF THE LOGICAL RECORD IS WITHIN THE
0039      C    PRESENT PHYSICAL RECORD.
0040      C
0041                IF(N+WRDPOS.LE.500) THEN
0042                    DO K=1,N
0043                        L=WRDPOS+K
0044
0045                        INCLUDE 'SUBNODE.SET'
0046    1 ***********************************************
0047    1 *****   UNPACKING THE SUBNODE COORDINATES
0048    1 ***********************************************
0049    1 *     SETTING THE X COORDINATE
0050    1       TMP=LIB$EXTZV(0,16,SUBREC(L))
0051    1       CALL LIB$INSV(TMP,0,16,SUBX(K))
0052    1 *     SETTING THE Y COORDINATE
0053    1       TMP=LIB$EXTZV(16,16,SUBREC(L))
0054    1       CALL LIB$INSV(TMP,0,16,SUBY(K))
0055    1 ***********************************************
0056                    ENDDO
0057                ELSE
```

```
C            GET THE BEGINNING OF THE LOGICAL RECORD
             NBEG=500-WRDPOS
                 DO  K=1,NBEG
                   L=WRDPOS+K
                   INCLUDE 'SUBNODE.SET'
1 ***********************************************
1 *****   UNPACKING THE SUBNODE COORDINATES
1 ***********************************************
1 *        SETTING THE X COORDINATE
1          TMP=LIBSEXTZV(0,16,SUBREC(L))
1          CALL LIBSINSV(TMP,0,16,SUBX(K))
1 *        SETTING THE Y COORDINATE
1          TMP=LIBSEXTZV(16,16,SUBREC(L))
1          CALL LIBSINSV(TMP,0,16,SUBY(K))
1 ***********************************************
                 ENDDO
C
C       GET NEXT PHYSICAL SUBNODE RECORD
             SRECNUM=SRECNUM+1
             CALL GETREC(LU,SRECNUM,SUBREC)
C       GET THE REMAINDER OF THE LOGICAL SUBNODE RECORD
             NREM=N-NBEG
C D       PRINT*,'NREM',NREM,'K',K,'L',L
             DO  L=1,NREM
                K=NBEG+L
                   INCLUDE 'SUBNODE.SET'
1 ***********************************************
1 *****   UNPACKING THE SUBNODE COORDINATES
1 ***********************************************
1 *        SETTING THE X COORDINATE
1          TMP=LIBSEXTZV(0,16,SUBREC(L))
1          CALL LIBSINSV(TMP,0,16,SUBX(K))
1 *        SETTING THE Y COORDINATE
1          TMP=LIBSEXTZV(16,16,SUBREC(L))
1          CALL LIBSINSV(TMP,0,16,SUBY(K))
1 ***********************************************
                 ENDDO
          ENDIF
C
          RETURN
          END
```

```
0001          SUBROUTINE GRIDR
0002          INCLUDE 'LNKNOD.CMN'
0003    1 ***************************************************************
0004    1 *        ARRAYS FOR THE GRID,NODE,LINK,AND SUBNODE FILES      *
0005    1 ***************************************************************
0006    1          INTEGER*4 GRID,NODREC,LNKREC,SUBREC
0007    1          COMMON /LNKNOD/GRID(128,128),NODREC(5,100),LNKREC(5,100)
0008    1        + ,SUBREC(500)
0009    1 ****************************************************************
0010      C
0011          INTEGER*4 BUFR(16384)
0012          EQUIVALENCE (GRID(1,1),BUFR(1))
0013          DO 11 IG=1,32
0014          K=(IG-1)*500
0015          READ(1'IG) (BUFR(JG),JG=K+1,K+500)
0016    11      CONTINUE
0017            READ(1'33)(BUFR(J),J=16001,16384)
0018          RETURN
0019          END
```

```
      SUBROUTINE HA2IJM(HADR,I,J,LEV)
C     *ROUTINE(CONVERT HEX ADDRESS TO MIN LEVEL I,J COORDINATES-HA2IJM)
C     ****************************************************************
C     *
C     *   DESIGNER/PROGRAMMER:
C     *      DON KRECKER  19 SEPTEMBER 1980
C     *   PURPOSE:
C     *      HA2IJM CONVERTS A HEX ADDRESS IN OCTAL REPRESENTATION TO
C     *      ITS EQUIVALENT I,J OBLIQUE COORDINATES AT THE MINIMUM LEVEL
C     *      OF HEX AGGREGATION.  THESE I,J COORDINATES ARE EXPRESSED IN
C     *      UNITS OF HEX DIAMETERS OF THE SMALLEST SIZE HEX IN THE
C     *      CURRENT CONFIGURATION AND CORRESPOND TO THE CENTER OF THE
C     *      GIVEN HEX.  THE LEVEL OF AGGREGATION OF THE HEX IS ALSO
C     *      RETURNED.  HA2IJM IS THE INVERSE OF THE FUNCTION IJ2HA.
C     *      THE ALGORITHM PULLS DIGITS OFF THE HEX ADDRESS ONE BY ONE
C     *      FROM RIGHT TO LEFT.  AS EACH DIGIT IS PULLED OFF, IT IS
C     *      CONSIDERED TO BE THE LEFTMOST DIGIT AND THEREFORE REPRE-
C     *      SENT THE HEX AT THE HIGHEST LEVEL OF AGGREGATION CONTAINING
C     *      THE GIVEN HEX.  ACCORDINGLY, THE I,J COORDINATES (AT THE
C     *      MINIMUM LEVEL) CORRESPONDING TO THIS LARGEST SIZE HEX ARE
C     *      ADDED TO RUNNING I AND J TOTALS.  IF ANOTHER HEX DIGIT IS
C     *      FOUND, THEN THE PREVIOUS DIGIT(S) ACTUALLY REPRESENT HEXES
C     *      OF LOWER LEVEL.  A TRANSFORMATION IS APPLIED TO SHRINK THE
C     *      CURRENT I,J VECTOR TO THE NEXT LOWER LEVEL, AND THE ALGO-
C     *      RITHM CONTINUES WITH THE NEW DIGIT.  THE ALGORITHM TER-
C     *      MINATES WHEN NO MORE NONZERO HEX DIGITS ARE FOUND.  THE
C     *      LEVEL OF THE HEX IS DETERMINED AS THE MAXIMUM NUMBER OF
C     *      LEVELS OF HEX AGGREGATION MINUS THE NUMBER OF DIGITS IN
C     *      THE HEX ADDRESS.  HA2IJM CHECKS TO ENSURE THAT THE INPUT
C     *      HEX ADDRESS IS POSITIVE AND HAS A VALID NUMBER OF DIGITS.
C     *   CALLING SEQUENCE:
C     *      CALL HA2IJM(HADR,I,J,LEV)
C     *   INPUT:
C     *      HADR    - HEX ADDRESS FOR WHICH EQUIVALENT I,J COORDINATES
C     *                AT THE MINIMUM HEX LEVEL ARE TO BE COMPUTED
C     *      NHLEV   - MAXIMUM NUMBER OF LEVELS OF HEX AGGREGATION.
C     *                (IN COMMON/HEX/)
C     *      MINLEV  - MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *      IMAX(HDIG)
C     *      JMAX(HDIG)
C     *              - ARRAYS CONTAINING THE I,J COORDINATES (AT THE
C     *                MINIMUM HEX LEVEL) OF THE CENTERS OF EACH OF THE
C     *                7 HEXES OF MAXIMUM LEVEL.  (THE MAXIMUM HEX LEVEL
C     *                IS NHLEV - 1.)
C     *   OUTPUT:
C     *      I,J     - OBLIQUE COORDINATES (AT THE MINIMUM HEX LEVEL) OF
C     *                THE GIVEN HEX ADDRESS
C     *      LEV     - LEVEL OF AGGREGATION OF THE GIVEN HEX ADDRESS
C     *
C     ****************************************************************
      IMPLICIT INTEGER(H,P)
      COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMT
     SR,
     +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
     +                ICON(70),JCON(70),IMAX(7),JMAX(7)
      DIMENSION IVAL(7),JVAL(7)
      EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
```

```
      SUBROUTINE GRIDS
*****************************************************************
*       THIS ROUTINE JUST DRAWS THE 10KM GRID SQUARES.          *
*****************************************************************
        INTEGER*2 IX,IY,IMAX
        INCLUDE 'WINDO.CMN'
1 ***************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND     *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER      *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS    *
1 *     THE CONTOUR INTERVAL.                                   *
1 ***************************************************************
1       DIMENSION FWINXY(4)
1       COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ***************************************************************
        CALL CMOPEN
        GRSIZE=10000.
      YB=FWINXY(3)
      YT=FWINXY(4)
        CALL LINCLR(4)
      DO XL=FWINXY(1),FWINXY(2),GRSIZE
         CALL MOVE(XL,YB)
         CALL DRAW(XL,YT)
           IF(AMOD(XL,10000.).EQ.0.)THEN
C.....       REDRAW THE GRID LINE
             CALL MOVE(XL+50.,YB)
             CALL DRAW(XL+50.,YT)
           ENDIF
      ENDDO
        XL=FWINXY(1)
        XR=FWINXY(2)
      DO YB=FWINXY(3),FWINXY(4),GRSIZE
         CALL MOVE(XL,YB)
         CALL DRAW(XR,YB)
           IF(AMOD(YB,10000.).EQ.0.)THEN
C.....       REDRAW THE GRID LINE
             CALL MOVE(XL,YB+50.)
             CALL DRAW(XR,YB+50.)
           ENDIF
      ENDDO
C       CALL NUMBR
        CALL CMCLOS
      RETURN
      END
```

```
C         *INITIALIZE I,J COORDINATES TO 0
           I = 0
           J = 0
C         *IF(HEX ADDRESS IS POSITIVE)THEN
           IF(HADR.GE.0) GOTO 1300
C            *INITIALIZE LEVEL TO MAXIMUM NUMBER OF HEX LEVELS
              LEV = NHLEV
C            *GET LEAST SIGNIFICANT (RIGHTMOST) HEX DIGIT
              HEX  = HADR
              HDIG = IAND(HEX,7)
C            *LOOP UNTIL(NO MORE HEX DIGITS)
 1100         CONTINUE
C               *DECREMENT LEVEL BY ONE
                 LEV = LEV - 1
C               *SHRINK PREVIOUS I,J VECTOR BY ONE LEVEL
                 INEW = (3*I - J)/7
                 JNEW = (I + J + J)/7
C               *ADD I,J VECTOR CORRESPONDING TO CURRENT HEX DIGIT
                 I = INEW + IMAX(HDIG)
                 J = JNEW + JMAX(HDIG)
C               *GET NEXT HEX DIGIT
                 HEX  = ISHFT(HEX,-3)
                 HDIG = IAND(HEX,7)
C            *ENDLOOP(HEX DIGIT LOOP)
              IF(HDIG.NE.0) GOTO 1100
C            *IF(LEVEL OF HEX ADDRESS IS INVALID)THEN
              IF(LEV.GE.MINLEV) GOTO 1200
C               *INCLUDE(GENERATE HEX ERROR MESSAGE - HXERR)
                 CALL HXERR(6HHA2IJM,2,HADR,LEV,0,0)
C            *ENDIF(LEVEL CHECK)
 1200         CONTINUE
C         *ELSE(HEX ADDRESS IS NOT POSITIVE)
           GOTO 1400
 1300      CONTINUE
C            *SET RETURN LEVEL TO ZERO
              LEV = 0
C            *INCLUDE(GENERATE HEX ERROR MESSAGE - HXERR)
              CALL HXERR(6HHA2IJM,1,HADR,0,0,0)
C         *ENDIF(CHECK FOR POSITIVE HEX ADDRESS)
 1400      CONTINUE
C      *ENDROUTINE(HA2IJM)
        RETURN
        END
```

```
0001          SUBROUTINE HA2XYL(HADR,X,Y,LEV)
0002    C    *ROUTINE(CONVERT HEX ADDRESS TO X,Y COORDINATES AND LEVEL-HA2XYL)
0003    C    ***************************************************************
0004    C    *
0005    C    *   DESIGNER/PROGRAMMER:
0006    C    *      DON KRECKER  21 SEPTEMBER 1980
0007    C    *   PURPOSE:
0008    C    *      HA2XYL CONVERTS A HEX ADDRESS TO THE X,Y CARTESIAN COORDI-
0009    C    *      NATES OF THE CENTER OF THE HEX AND THE LEVEL OF AGGREGATION
0010    C    *      OF THE HEX.  THE X,Y COORDINATES ARE EXPRESSED IN METERS.
0011    C    *      THIS ROUTINE IS THE INVERSE OF THE SUBROUTINE XYL2HA.
0012    C    *      HA2XYL FIRST CALLS THE ROUTINE HA2IJM TO CONVERT THE HEX
0013    C    *      ADDRESS TO EQUIVALENT I,J OBLIQUE COORDINATES AT THE MINI-
0014    C    *      MUM HEX LEVEL AND TO RETURN THE LEVEL OF THE GIVEN HEX.
0015    C    *      ERROR CHECKING IS DONE IN THIS SUBORDINATE ROUTINE.  THEN
0016    C    *      THE I,J COORDINATES ARE CONVERTED TO X,Y COORDINATES IN
0017    C    *      METERS BY CALLING THE ROUTINE IJM2XY.
0018    C    *   CALLING SEQUENCE:
0019    C    *      CALL HA2XYL(HADR,X,Y,LEV)
0020    C    *   INPUT:
0021    C    *      HADR    - HEX ADDRESS FOR WHICH THE EQUIVALENT X,Y COORDI-
0022    C    *                NATES AND LEVEL OF AGGREGATION ARE TO BE COMPUTED
0023    C    *   OUTPUT:
0024    C    *      X,Y     - REAL-VALUED CARTESIAN COORDINATES OF THE CENTER
0025    C    *                OF THE GIVEN HEX EXPRESSED IN METERS
0026    C    *      LEV     - LEVEL OF AGGREGATION OF THE GIVEN HEX ADDRESS
0027    C    *
0028    C    ***************************************************************
0029          IMPLICIT INTEGER(H,P)
0030    C       *INCLUDE(CONVERT HEX ADDRESS TO MIN LEVEL I,J AND LEVEL-HA2IJM)
0031             CALL HA2IJM(HADR,I,J,LEV)
0032    C       *INCLUDE(CONVERT MIN LEVEL I,J TO X,Y COORDINATES - IJM2XY)
0033             CALL IJM2XY(I,J,X,Y)
0034    C    *ENDROUTINE(HA2XYL)
0035          RETURN
0036          END
```

```
      SUBROUTINE HEXIN(HREAD,IBASE,LEVEL,HSTOR)
C    *ROUTINE( CONSTRUCT INTERNAL HEX ADDRESS - HEXIN)
C    *   DESIGNER/PROGRAMMER\
C    *      DON KRECKER  10 SEPTEMBER 1980
C    *   PURPOSE\
C    *      HEXIN TAKES AN INPUT HEX QUANTITY IN EITHER OCTAL OR
C    *      DECIMAL REPRESENTATION AND CONSTRUCTS A HEX ADDRESS AT
C    *      THE REQUESTED LEVEL IN THE REQUIRED INTERNAL FORMAT.
C    *      THE INPUT HEX QUANTITY IS TREATED AS A HEX VECTOR FROM
C    *      THE ORIGIN OF THE HEX COORDINATE SYSTEM, AND THEREFORE
C    *      THE PRESENCE OR ABSENCE OF LEADING 7 HEX DIGITS PLAYS
C    *      NO ROLE.  THE RETURN HEX ADDRESS WILL CONTAIN LEADING 7
C    *      HEX DIGITS AS NEEDED TO INDICATE THE REQUESTED HEX LEVEL.
C    *      THE NUMBER OF LEADING7 HEX DIGITS DEPENDS ON THE NUMBER
C    *      OF LEVELS OF HEX AGGREGATION IN USE IN THE CURRENT CON-
C    *      FIGURATION, AS INITIALIZED IN A DATA STATEMENT.  HEXIN
C    *      ALSO CHECKS FOR INVALID INPUTS.  IN THE CASE OF AN ERROR,
C    *      AN ERROR MESSAGE IS PRINTED, AND THE RETURN HEX ADDRESS
C    *      IS SET TO ZERO.
C    *
C    ***************************************************************
C    *   CALLING SEQUENCE\
C    *      CALL HEXIN(HREAD,IBASE,LEVEL,HSTOR)
C    *   INPUT:
C    *      HREAD  - HEX QUANTITY AS READ IN EITHER OCTAL OR DECIMAL
C    *               REPRESENTATION, WITH OR WITHOUT LEADING 7 DIGITS
C    *      IBASE  - FLAG INDICATING THAT HREAD IS IN OCTAL (0) OR
C    *               DECIMAL (1) REPRESENTATION
C    *      LEVEL  - LEVEL OF HEX ADDRESS TO BE CONSTRUCTED
C    *   OUTPUT:
C    *      HSTOR  - HEX ADDRESS AT REQUESTED LEVEL IN REQUIRED
C    *               INTERNAL FORMAT, OR ZERO IF ANY ERRORS OCCURRED
C    *
C    ***************************************************************
      IMPLICIT INTEGER(H,P)
C    *OUTPUT DEVICE NUMBER CONSTANT
      DATA IRT/6/
C    *NUMBER OF LEVELS OF HEX AGGREGATION CONSTANT
      DATA NHEXLV/10/
C    *MINIMUM HEX LEVEL CONSTANT
      DATA MINHLV/ 2/
C       *INITIALIZE RETURN HEX ADDRESS TO ZERO
        HSTOR = 0
C       *IF(VALID LEVEL AND POSITIVE HEX QUANTITY)THEN
        IF(LEVEL.LT.MINHLV) GOTO 1800
        IF(LEVEL.GE.NHEXLV) GOTO 1800
        IF(HREAD.LE.0)      GOTO 1800
C         *INITIALIZE MULTIPLIER CORRESPONDING TO OCTAL OR DECIMAL
          MULT = 2*IBASE + 8
C         *CALCULATE MAXIMUM SHIFT CONSTANT AS FUNCTION OF LEVEL
          MSHIFT = 3*(NHEXLV-LEVEL)
C         *INITIALIZE PARAMETERS FOR DIGIT LOOP
          HLOC   = HREAD
          LSHIFT = 0
C         *LOOP UNTIL(ALL DIGITS CHECKED OR INVALID DIGIT FOUND)
 1100     CONTINUE
C           *STRIP OFF NEXT DIGIT
```

```
0058                    HTMP = HLOC/MULT
0059                    HDIG = HLOC - HTMP*MULT
0060     C             *IF(DIGIT LIMIT NOT EXCEEDED)THEN
0061                    IF(LSHIFT.GE.MSHIFT) GOTO 1400
0062     C                *IF(DIGIT IS A VALID HEX DIGIT)THEN
0063                      IF(HDIG.EQ.0.OR.HDIG.GT.7) GOTO 1200
0064     C                  *INSERT DIGIT IN OUTPUT HEX ADDRESS
0065                         HSTOR = IOR(HSTOR,ISHFT(HDIG,LSHIFT))
0066     C                *ELSE(DIGIT IS INVALID)
0067                      GOTO 1300
0068      1200            CONTINUE
0069     C                  *SET RETURN HEX ADDRESS TO ZERO
0070                         HSTOR = 0
0071     C                *ENDIF(HEX DIGIT VALIDITY CHECK)
0072      1300            CONTINUE
0073     C             *ELSE(DIGIT LIMIT EXCEEDED)
0074                    GOTO 1600
0075      1400          CONTINUE
0076     C                *IF(HEX DIGIT NOT 7)THEN
0077                      IF(HDIG.EQ.7) GOTO 1500
0078     C                  *SET RETURN HEX ADDRESS TO ZERO
0079                         HSTOR = 0
0080     C                *ENDIF(CHECK FOR 7 HEX DIGIT)
0081      1500            CONTINUE
0082     C             *ENDIF(DIGIT LIMIT CHECK)
0083      1600          CONTINUE
0084     C             *UPDATE LOOP PARAMETERS FOR NEXT DIGIT
0085                    HLOC   = HTMP
0086                    LSHIFT = LSHIFT + 3
0087     C          *ENDLOOP(DIGIT LOOP)
0088                 IF(HLOC.NE.0.AND.HSTOR.NE.0) GOTO 1100
0089     C          *IF(NO ERROR AND LEADING 7 HEX DIGITS NEEDED)THEN
0090                 IF(HSTOR.EQ.0) GOTO 1700
0091                 IF(LSHIFT.GE.MSHIFT) GOTO 1700
0092     C             *SET UP REQUIRED LEADING 7 DIGITS
0093                    HDIG = ISHFT(1,MSHIFT-LSHIFT) - 1
0094     C             *INSERT DIGITS IN OUTPUT HEX ADDRESS
0095                    HSTOR = IOR(HSTOR,ISHFT(HDIG,LSHIFT))
0096     C          *ENDIF(ERROR AND LEADING 7 HEX DIGIT CHECK)
0097      1700        CONTINUE
0098     C        *ENDIF(VALID LEVEL AND POSITIVE HEX QUANTITY CHECK)
0099      1800     CONTINUE
0100     C        *IF(ANY ERRORS)THEN
0101               IF(HSTOR.NE.0) GOTO 1900
0102     C           *WRITE ERROR MESSAGE
0103                 IF(IBASE.NE.0) GOTO 1810
0104                    WRITE(IRT,9001) HREAD,LEVEL
0105                 GOTO 1920
0106      1810       CONTINUE
0107                    WRITE(IRT,9002) HREAD,LEVEL
0108      1820       CONTINUE
0109     C        *ENDIF(ERROR CHECK)
0110      1900     CONTINUE
0111           RETURN
0112      9001 FORMAT(//42H *****  INVALID PARAMETERS IN HEXIN  *****/
0113          +        21H   OCTAL HEX NUMBER =,O10/
0114          +        32H   REQUESTED HEX ADDRESS LEVEL =,I10//)
```

HEXIN

```
0115      9002 FORMAT(//42H *****  INVALID PARAMETERS IN HEXIN  *****/
0116          +          23H   DECIMAL HEX NUMBER =,I10/
0117          +          32H   REQUESTED HEX ADDRESS LEVEL =,I10//)
0118           END
```

```
      SUBROUTINE HEX2XY(HEX,X,Y)
*************************************************************
*     THIS SUBROUTINE IS USED TO TRANSLATE AN INTERNAL HEX *
*     ADDRESS TO STANDARD UTM COORDINATES.                 *
*************************************************************
      IMPLICIT INTEGER(H,P)
      INCLUDE 'CENTER.CMN'
     1 ********************************************************
     1 *     THE CENTER OF THE HEX GRID IS AT XORIGIN,YORIGIN  *
     1 *     WHERE THE COORDINATES ARE IN METERS UTM RELATIVE  *
     1 *     TO A GIVEN GRID ZONE.                             *
     1 ********************************************************
     1      INTEGER*4 XORIGIN,YORIGIN
     1      COMMON/CENTER/XORIGIN,YORIGIN
     1      DATA XORIGIN/500000/,YORIGIN/5700000/
     1 C*******************************************************
      CALL HA2XYL(HEX,X,Y,LEV)
C  D   PRINT*,'LEV IN HEX2XY: ',LEV
      X=X+XORIGIN
      Y=Y+YORIGIN
      RETURN
      END
```

```
0001            SUBROUTINE HEXINIT
0002      ***************************************************************
0003      *     THIS JUST KEEPS ALL OF THE HEXINIT.PRM JUNK OUT OF THE *
0004      *     MAIN ROUTINE. WHILE INITIALIZING THE HEX PARAMETERS.   *
0005      ***************************************************************
0006            INCLUDE 'HEX.CMN'
0007    1 ***********************************************************************
0008    1 *     FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR                     *
0009    1 ***********************************************************************
0010    1       IMPLICIT INTEGER(H,P)
0011    1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR,
0012    1      +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
0013    1      +                ICON(70),JCON(70),IMAX(7),JMAX(7)
0014    1 ***********************************************************************
0015            INCLUDE 'HEXRAD.CMN'
0016    1       INTEGER*2 DBRES,HEXR
0017    1       COMMON/HEXRAD/DBRES,RAD2,HEXR
0018      ***************************************************************
0019      *     IWRITE:  OUTPUT DEVICE FOR ERROR MESSAGES               *
0020      *     LEVMAX:  MAXIMUM LEVEL OF HEX AGGREGATION               *
0021      *     LEVMIN:  MINIMUM   "        "   "                       *
0022      *     DLT:  LATITUDE OF THE ORIGIN HEX IN FLOATING-           *
0023      *     POINT DEGREES                                           *
0024      *     DLN:  LONGITUDE OF ORIGIN HEX                           *
0025      *     LEVSIZ:  HEX LEVEL AT WHICH THE SCALE OF THE            *
0026      *     HEX COORDINATE SYSTEM IS GIVEN                          *
0027      *     SIZHEX:  DIAMETER OF HEXES AT SIZE 'LEVSIZ' IN          *
0028      *     FLOATING-POINT METERS                                   *
0029            IWRITE=6
0030            LEVMAX=9
0031            LEVMIN=4
0032            DLT=51.45  ! LAT AND
0033            DLN=9.00   ! LON OF  32UNC00
0034            LEVSIZ=6
0035            SIZHEX=25000.
0036            CALL HXINIT(IWRITE,LEVMAX,LEVMIN,DLT,DLN,LEVSIZ,SIZHEX)
0037      ***************************************************************
0038            DBRES=100   !DATA BASE RESOLUTION
0039      *     THE TRUE RADIUS OF A 3.57 KM HEX IS 2061,BUT...
0040            HEXR=2000
0041            R=HEXR  !HEXR OVERFLOWS WHEN IT IS SQUARED
0042            RAD2=R**2
0043            RETURN
0044            END
```

```
0001          SUBROUTINE HEXOUT(HSTOR,IBASE,HWRYT)
0002    C    *ROUTINE( FORMAT HEX ADDRESS FOR OUTPUT - HEXOUT)
0003    C    ***************************************************************
0004    C    *
0005    C    *   DESIGNER/PROGRAMMER:
0006    C    *      DON KRECKER  11 SEPTEMBER 1980
0007    C    *   PURPOSE:
0008    C    *      HEXOUT TAKES A HEX ADDRESS IN INTERNAL FORMAT AND REFORMATS
0009    C    *      IT FOR OUTPUT.  IF REQUESTED, THE HEX ADDRESS IS CONVERTED
0010    C    *      FROM OCTAL TO DECIMAL REPRESENTATION.  LEADING 7 HEX DIGITS
0011    C    *      ARE APPENDED TO THE ADDRESS SO THAT IT IS IN STANDARD FORM
0012    C    *      BASED ON 12 LEVELS OF HEX AGGREGATION.  THE NUMBER OF LEAD-
0013    C    *      ING 7 HEX DIGITS TO BE APPENDED DEPENDS ON THE NUMBER OF
0014    C    *      LEVELS OF HEX AGGREGATION IN USE IN THE CURRENT CONFIGURA-
0015    C    *      TION, AS INITIALIZED IN A DATA STATEMENT.
0016    C    *   CALLING SEQUENCE:
0017    C    *      CALL HEXOUT(HSTOR,IBASE,HWRYT)
0018    C    *   INPUT:
0019    C    *      HSTOR  - HEX ADDRESS IN INTERNAL FORMAT
0020    C    *      IBASE  - FLAG INDICATING THAT REQUESTED OUTPUT FORMAT IS
0021    C    *               OCTAL (0) OR DECIMAL (1)
0022    C    *   OUTPUT:
0023    C    *      HWRYT  - HEX ADDRESS IN REQUESTED OUTPUT FORMAT, EITHER
0024    C    *               OCTAL OR DECIMAL, WITH STANDARD LEADING 7 HEX
0025    C    *               DIGITS APPENDED
0026    C    *
0027    C    ***************************************************************
0028            INCLUDE 'HEX.CMN'
0029  1 ***************************************************************
0030  1 *       FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR           *
0031  1 ***************************************************************
0032  1       IMPLICIT INTEGER(H,P)
0033  1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR,
0034  1      +              XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
0035  1      +              ICON(70),JCON(70),IMAX(7),JMAX(7)
0036  1 ***************************************************************
0037    C    *OUTPUT DEVICE NUMBER CONSTANT
0038          DATA IRT/6/
0039    C    *NUMBER OF LEVELS OF HEX AGGREGATION CONSTANT
0040          DATA NHEXLV/10/
0041    C    *MINIMUM HEX LEVEL CONSTANT
0042          DATA MINHLV/ 2/
0043    C       *INITIALIZE RETURN HEX ADDRESS TO ZERO
0044             HWRYT = 0
0045    C       *IF(POSITIVE HEX ADDRESS)THEN
0046             IF(HSTOR.LE.0) GOTO 1400
0047    C          *COMPUTE NUMBER OF DIGITS IN HEX ADDRESS
0048               NDIG = IFIX(0.480898*ALOG(FLOAT(HSTOR))) + 1
0049    C          *IF(VALID NUMBER OF DIGITS)THEN
0050               IF(NDIG.GT.NHEXLV-MINHLV) GOTO 1300
0051    C             *SET UP LEADING 7 DIGITS TO BE APPENDED
0052                  HDIG = ISHFT(1,3*(12-NHEXLV)) - 1
0053    C             *CONSTRUCT OUTPUT HEX ADDRESS IN OCTAL FORMAT
0054                  HWRYT = IOR(HSTOR,ISHFT(HDIG,3*NDIG))
0055    C             *IF(DECIMAL FORMAT REQUESTED)THEN
0056                  IF(IBASE.EQ.0) GOTO 1200
0057    C                *INITIALIZE CONVERSION LOOP
```

```
0058                        HLOC  = HWRYT
0059                        HWRYT = 0
0060                        IPOWR = 1
0061     C                 *LOOP UNTIL(ALL DIGITS CONVERTED)
0062      1100              CONTINUE
0063     C                    *STRIP OFF NEXT DIGIT
0064                           HDIG = IAND(HLOC,7)
0065     C                    *APPEND DIGIT TO OUTPUT HEX ADDRESS
0066                           HWRYT = HWRYT + IPOWR*HDIG
0067     C                    *UPDATE LOOP PARAMETERS FOR NEXT DIGIT
0068                           HLOC  = ISHFT(HLOC,-3)
0069                           IPOWR = IPOWR*10
0070     C                 *ENDLOOP(DIGIT CONVERSION LOOP)
0071                        IF(HLOC.GT.0) GOTO 1100
0072     C              *ENDIF(DECIMAL FORMAT CHECK)
0073      1200           CONTINUE
0074     C           *ENDIF(DIGIT COUNT CHECK)
0075      1300        CONTINUE
0076     C        *ENDIF(POSITIVE HEX ADDRESS CHECK)
0077      1400     CONTINUE
0078      1500     CONTINUE
0079           RETURN
0080      9001 FORMAT(//44H *****  INVALID HEX ADDRESS IN HEXOUT  *****/
0081          +         35H   HEX ADDRESS IN INTERNAL FORMAT =,O12//)
0082           END
```

```
0001            SUBROUTINE HEXREAD(HSTOR,SIDES,LU)
0002    C*********************************************
0003    C*****  THIS ROUTINE IS DESIGNED TO READ THE
0004    C*****  HEX NUMBER AND CONNECTIVITY OF ITS
0005    C*****  SIDES FROM THE FILE 'HEXROAD.DAT'
0006    C*********************************************
0007    C
0008            IMPLICIT INTEGER(H,P)
0009            INTEGER*4 SIDES
0010    C
0011            CALL HEXOUT(HSTOR,1,HEX)
0012            READ(UNIT=LU,KEY=HEX,KEYID=0,IOSTAT=IOS
0013         +  ,ERR=999) HEX,SIDES
0014            CALL HEXIN(HEX,1,4,HSTOR)
0015    C
0016    C       IF ALL GOES WELL...
0017            RETURN
0018    C
0019    C       IF NOT...
0020    999     CONTINUE
0021    C       ERROR CODE 36 IS RETURNED ON AN ATTEMPT TO
0022    C       READ A NONEXISTANT RECORD. (AND MAYBE FOR
0023    C       OTHER REASONS,TOO.)
0024    C
0025            SIDES=0
0026            IF(IOS.EQ.36)THEN
0027              WRITE(UNIT=LU)HEX,SIDES
0028            ELSE
0029              CALL LIB$SIGNAL(%VAL(IOS))
0030            ENDIF
0031            RETURN
0032            END
```

```
      SUBROUTINE HEXWRITE(HSTOR,SIDES,LU)
C*********************************************
C*****  THIS ROUTINE IS DESIGNED TO WRITE THE
C*****  HEX NUMBER AND THE CONNECTIVITY OF ITS
C*****  SIDES TO THE FILE 'HEXROAD.DAT'.
C*********************************************
C
      IMPLICIT INTEGER (H,P)
      INTEGER*4 SIDES
C
      CALL HEXOUT(HSTOR,1,HEX)
      WRITE(UNIT=LU,IOSTAT=IOS,ERR=999)HEX,SIDES
C
C     IF ALL GOES WELL...
      RETURN
C
C     IF NOT...
999   CONTINUE
      IF(IOS.EQ.50)THEN
        CALL HEXREAD(HSTOR,ISIDES,LU)!ISIDES IS A DUMMY
        REWRITE(UNIT=LU)HEX,SIDES
      ELSE
        CALL LIB$SIGNAL(%VAL(IOS))
      ENDIF
      RETURN
      END
```

```
      INTEGER FUNCTION HXADD(HEXA,HEXB)
C    *ROUTINE(ADD TWO OCTAL HEX NUMBERS - HXADD)
C    ******************************************************************
C    *
C    *   DESIGNER/PROGRAMMER:
C    *      DON KRECKER  11 SEPTEMBER 1980
C    *   PURPOSE:
C    *      HXADD ADDS TWO HEX NUMBERS EXPRESSED IN OCTAL REPRESENTA-
C    *      TION.  THE ALGORITHM FIRST ADDS COLUMNS OF HEX DIGITS IN
C    *      PARALLEL TO GET HEX SUM DIGITS.  THE CORRESPONDING HEX
C    *      CARRY DIGITS ARE THEN CALCULATED USING LOGICAL OPERATIONS.
C    *      IF NOT ALL CARRY DIGITS ARE ZERO, THEY ARE SHIFTED ONE
C    *      COLUMN LEFT AND TREATED AS A NEW ADDEND TO BE ADDED TO THE
C    *      SUM DIGITS.  THE PROCEDURE CONTINUES UNTIL NO NEW CARRIES
C    *      ARE GENERATED.  IF N IS THE MAXIMUM NUMBER OF HEX DIGITS
C    *      IN EITHER ADDEND, THE ALGORITHM WILL TERMINATE IN AT MOST
C    *      N+1 STEPS.  THE HEX SUM WILL CONTAIN N OR N+1 HEX DIGITS,
C    *      AND ANY N+1ST HEX DIGIT WILL NOT BE A 7.
C    *      GIVEN A PAIR OF HEX DIGITS, HA AND HB, THEIR HEX SUM DIGIT
C    *      IS CALCULATED BY ADDITION MODULO 7 WITH THE RESULT IN THE
C    *      RANGE 1 THROUGH 7 (RATHER THAN 0 THROUGH 6).
C    *         HSUM(HA,HB)=HA+HB(MOD 7)
C    *      THE CORRESPONDING HEX CARRY DIGIT IS CALCULATED BY A SERIES
C    *      OF LOGICAL OPERATIONS.
C    *         HCAR(HA,HB)=XOR(OR(HA,HB),AND(XOR(HA,HB),HSUM(HA,HB)))
C    *      IN SOME INSTANCES THIS FORMULA WILL GIVE A CARRY DIGIT
C    *      OF 7 WHICH MUST BE RESET TO 0.
C    *   CALLING SEQUENCE:
C    *      HXADD = HXADD(HEXA,HEXB)
C    *   INPUT:
C    *      HEXA   - FIRST HEX NUMBER TO BE ADDED
C    *      HEXB   - SECOND HEX NUMBER TO BE ADDED
C    *   OUTPUT:
C    *      HXADD  - HEX SUM OF HEXA AND HEXB
C    *
C    ******************************************************************
      IMPLICIT INTEGER(H,P)
C    *MASK OF OCTAL 1 DIGITS
      DATA M1/'1111111111'O/
C    *MASK OF OCTAL 3 DIGITS
      DATA M3/'3333333333'O/
C    *MASK OF OCTAL 4 DIGITS
      DATA M4/'4444444444'O/
C       *INITIALIZE ADDENDS AS LOCAL VARIABLES
        HA = HEXA
        HB = HEXB
C       *GENERATE MASK OF SIGNIFICANT HEX DIGIT POSITIONS
        LOR  = IOR(HA,HB)
        LOR3 = IAND(LOR,M3)
        MDIG = IAND(IOR(LOR3+M3,LOR),M4)
C       *SET UP HEX DIGIT 7 IN NEXT MOST SIGNIFICANT POSITION
        HCHK = (MDIG - ISHFT(MDIG,-3)) * 14
C       *LOOP UNTIL(NO HEX CARRIES TO BE ADDED IN)
 1100   CONTINUE
C         *EVALUATE BASIC LOGICAL FUNCTIONS OF ADDENDS
          LOR   = IOR(HA,HB)
          LOR3  = IAND(LOR,M3)
```

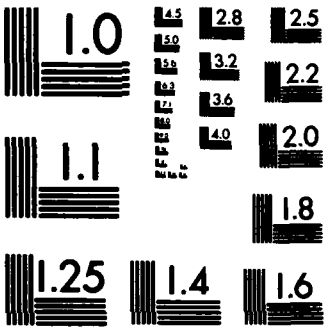
```
0058                 LAND  = IAND(HA,HB)
0059                 LAND4 = IAND(LAND,M4)
0060                 LEOR  = IEOR(HA,HB)
0061                 LEOR4 = IAND(LEOR,M4)
0062     C          **COMPUTE HEX SUM DIGITS
0063     C             *ADD DIGITS IN EACH COLUMN MODULO 8 (NO COLUMN OVERFLOWS)
0064                    HA = IEOR(IAND(HA,M3)+IAND(HB,M3),LEOR4)
0065     C             *GENERATE MASK OF COLUMNS WHICH OVERFLOW WHEN ADDED
0066                    MDIG = IOR(IAND(LEOR4,NOT(HA)),LAND4)
0067     C             *INCREMENT COLUMNS WITH OVERFLOW TO GET SUM MODULO 7
0068                    HA = HA + ISHFT(MDIG,-2)
0069     C          **END BLOCK(COMPUTE HEX SUM DIGITS)
0070     C          **COMPUTE HEX CARRY DIGITS
0071     C             *COMPUTE GENERALIZED CARRY DIGITS (POSSIBLY WITH 7S)
0072                    HB = IEOR(LOR,IAND(LEOR,HA))
0073     C             *GENERATE MASK OF COLUMNS GIVING CARRY DIGIT 7
0074                    MDIG = IAND(IAND(LOR,LOR3+M1),MDIG)
0075     C             *RESET CARRY DIGITS 7 TO 0
0076                    HB = IEOR(HB,ISHFT(MDIG,-2)*7)
0077     C          **END BLOCK(COMPUTE HEX CARRY DIGITS)
0078     C          *SHIFT CARRY DIGITS ONE COLUMN LEFT TO FORM NEW ADDEND
0079                 HB = ISHFT(HB,3)
0080     C       *ENDLOOP(ADDITION UNTIL NO MORE CARRIES LOOP)
0081              IF(HB.NE.0) GOTO 1100
0082     C       *REMOVE EXTRANEOUS LEADING 7 FROM SUM IF PRESENT
0083              IF(HA.GT.HCHK) HA = HA - HCHK
0084     C       *SET RETURN VALUE TO HEX SUM
0085              HXADD = HA
0086     C    *ENDROUTINE(HXADD)
0087           RETURN
0088           END
```

```
0001          SUBROUTINE HXERR(NAMSUB,ICASE,IPAR1,IPAR2,IPAR3,IPAR4)
0002    C    *ROUTINE(GENERATE HEX ERROR MESSAGE - HXERR)
0003    C    ***************************************************************
0004    C    *
0005    C    *   DESIGNER/PROGRAMMER:
0006    C    *      DON KRECKER  24 SEPTEMBER 1980
0007    C    *   PURPOSE:
0008    C    *      HXERR WRITES AN ERROR MESSAGE DESCRIBING AN ERROR WHICH HAS
0009    C    *      BEEN DETECTED IN ONE OF THE HEX LIBRARY ROUTINES.  IT ALSO
0010    C    *      CALLS A USER DEBUG ROUTINE, HXDBUG, FOR ADDITIONAL ERROR
0011    C    *      PROCESSING.  A STUB VERSION OF HXDBUG IS INCLUDED IN THE
0012    C    *      HEX LIBRARY IN CASE THE USER DOES NOT PROVIDE A VERSION TO
0013    C    *      OVERRIDE IT.
0014    C    *   CALLING SEQUENCE:
0015    C    *      CALL HXERR(NAMSUB,ICASE,IPAR1,IPAR2,IPAR3,IPAR4)
0016    C    *   INPUT:
0017    C    *      NAMSUB - NAME OF THE HEX LIBRARY ROUTINE IN WHICH THE ERROR
0018    C    *               WAS DETECTED.  THIS MAY BE A LOWER LEVEL ROUTINE
0019    C    *               INSTEAD OF A HEX ROUTINE CALLED DIRECTLY FROM OUT-
0020    C    *               SIDE THE HEX LIBRARY.  THIS PARAMETER IS PASSED
0021    C    *               USING A 6H HOLLERITH CONSTANT.
0022    C    *      ICASE  - CASE NUMBER IDENTIFYING THE TYPE OF ERROR WHICH
0023    C    *               WAS DETECTED:
0024    C    *               1  ZERO OR NEGATIVE HEX ADDRESS
0025    C    *               2  HEX ADDRESS WITH HEX LEVEL OUT OF RANGE
0026    C    *               3  HEX LEVEL OUT OF RANGE
0027    C    *               4  TWO HEX CENTERS COINCIDE
0028    C    *               5  TWO HEX ADDRESSES AT DIFFERENT HEX LEVELS
0029    C    *               6  TWO HEX ADDRESSES COINCIDE
0030    C    *      IPAR1,IPAR2,IPAR3,IPAR4
0031    C    *             - PARAMETERS WHICH GIVE ADDITIONAL INFORMATION ABOUT
0032    C    *               THE ERROR.  THE MEANING OF THESE PARAMETERS IS
0033    C    *               DEPENDENT ON THE TYPE OF ERROR:
0034    C    *               ICASE=1 (ZERO OR NEGATIVE HEX ADDRESS)
0035    C    *                       IPAR1=HEX ADDRESS
0036    C    *                       IPAR2=IPAR3=IPAR4=0
0037    C    *               ICASE=2 (HEX ADDRESS WITH LEVEL OUT OF RANGE)
0038    C    *                       IPAR1=HEX ADDRESS
0039    C    *                       IPAR2=LEVEL OF HEX ADDRESS
0040    C    *                       IPAR3=IPAR4=0
0041    C    *               ICASE=3 (HEX LEVEL OUT OF RANGE)
0042    C    *                       IPAR1=HEX LEVEL
0043    C    *                       IPAR2=IPAR3=IPAR4=0
0044    C    *               ICASE=4 (HEX CENTERS COINCIDE)
0045    C    *                       IPAR1=ADDRESS OF FIRST HEX
0046    C    *                       IPAR2=ADDRESS OF SECOND HEX
0047    C    *                       IPAR3=IPAR4=0
0048    C    *               ICASE=5 (HEX ADDRESSES AT DIFFERENT LEVELS)
0049    C    *                       IPAR1=FIRST HEX ADDRESS
0050    C    *                       IPAR2=LEVEL OF FIRST HEX ADDRESS
0051    C    *                       IPAR3=SECOND HEX ADDRESS
0052    C    *                       IPAR4=LEVEL OF SECOND HEX ADDRESS
0053    C    *               ICASE=6 (HEX ADDRESSES COINCIDE)
0054    C    *                       IPAR1=COMMON HEX ADDRESS
0055    C    *                       IPAR2=IPAR3=IPAR4=0
0056    C    *      IHXOUT - OUTPUT DEVICE NUMBER TO WHICH HEX ERROR MESSAGES
0057    C    *               ARE TO BE WRITTEN.  (IN COMMON/HEX/)
```

```
C     *
C     ****************************************************************
      IMPLICIT INTEGER(H,P)
      COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMT
     $R,
     +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
     +                ICON(70),JCON(70),IMAX(7),JMAX(7)
      DIMENSION IVAL(7),JVAL(7)
      INTEGER*2 NAMSUB(3)
      EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
C       *WRITE ERROR MESSAGE HEADING
         WRITE(IHXOUT,9000)
C       *CASE(TYPE OF ERROR)
         GOTO (1100,1200,1300,1400,1500,1600) ICASE
C          *TYPE = ZERO OR NEGATIVE HEX ADDRESS
 1100       CONTINUE
C             *WRITE MESSAGE AND INVALID HEX ADDRESS
               WRITE(IHXOUT,9001) NAMSUB,IPAR1
C          *TYPE = HEX ADDRESS WITH LEVEL OUT OF RANGE
            GOTO 1700
 1200       CONTINUE
C             *WRITE MESSAGE AND INVALID HEX AND LEVEL
               WRITE(IHXOUT,9002) NAMSUB,IPAR1,IPAR2
C          *TYPE = HEX LEVEL OUT OF RANGE
            GOTO 1700
 1300       CONTINUE
C             *WRITE MESSAGE AND INVALID HEX LEVEL
               WRITE(IHXOUT,9003) NAMSUB,IPAR1
C          *TYPE = HEX CENTERS COINCIDE
            GOTO 1700
 1400       CONTINUE
C             *WRITE MESSAGE AND INVALID HEX PAIR
               WRITE(IHXOUT,9004) NAMSUB,IPAR1,IPAR2
C          *TYPE = HEX ADDRESSES AT DIFFERENT LEVELS
            GOTO 1700
 1500       CONTINUE
C             *WRITE MESSAGE AND INVALID HEXES AND LEVELS
               WRITE(IHXOUT,9005) NAMSUB,IPAR1,IPAR2,IPAR3,IPAR4
C          *TYPE = HEX ADDRESSES COINCIDE
            GOTO 1700
 1600       CONTINUE
C             *WRITE MESSAGE AND COMMON HEX ADDRESS
               WRITE(IHXOUT,9006) NAMSUB,IPAR1
C       *END CASE(TYPE OF ERROR)
 1700    CONTINUE
C       *INCLUDE(USER HEX DEBUG ROUTINE - HXDBUG)
         CALL HXDBUG(NAMSUB,ICASE,IPAR1,IPAR2,IPAR3,IPAR4)
C     *ENDROUTINE(HXERR)
      RETURN
 9000 FORMAT(54H0++++++++++     HXERR - ERROR DETECTED IN HEX ROUTINE:,
     +        15H     ++++++++++,/)
 9001 FORMAT(5X,3A2,47H HAS BEEN PASSED A ZERO OR NEGATIVE HEX ADDRESS,
     +        /,5X,5HHEX =,O12)
 9002 FORMAT(5X,3A2,47H HAS BEEN PASSED A HEX ADDRESS WITH AN INVALID ,
     +        5HLEVEL,/,5X,5HHEX =,O12,9H  LEVEL =,I3)
 9003 FORMAT(5X,3A2,37H HAS BEEN PASSED AN INVALID HEX LEVEL,/,5X,
     +        7HLEVEL =,I8)
```

AD-A132 940 CORDIVEM EUROPEAN TERRAIN DATA(U) COMBINED ARMS OPERATIONS RESEARCH ACTIVITY FORT LEAVENWORTH KS W J THOMSON 1983 CAORA/TP-2-83 3/3

UNCLASSIFIED F/G 15/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

HXERR

```
 9004 FORMAT(5X,3A2,45H HAS BEEN PASSED HEX ADDRESSES WHOSE CENTERS ,
     +        8HCOINCIDE,/,5X,6HHEXA =,O12,6H  HEXB =,O12)
 9005 FORMAT(5X,3A2,44H HAS BEEN PASSED HEX ADDRESSES AT DIFFERENT ,
     +        6HLEVELS,/,5X,6HHEXA =,O12,10H  LEVELA =,I3,8H  HEXB =,
     +        O12,10H  LEVELB =,I3)
 9006 FORMAT(5X,3A2,45H HAS BEEN PASSED HEX ADDRESSES WHICH COINCIDE,/,
     +        5X,13HHEXA = HEXB =,O12)
      END
```

```
0001          SUBROUTINE HXINIT(IWRITE,LEVMAX,LEVMIN,DLT,DLN,LEVSIZ,SIZHEX)
0002    C    *ROUTINE(INITIALIZE HEX LIBRARY PARAMETERS - HXINIT)
0003    C    ***********************************************************************
0004    C    *
0005    C    *   DESIGNER/PROGRAMMER:
0006    C    *      DON KRECKER  25 SEPTEMBER 1980
0007    C    *   PURPOSE:
0008    C    *      HXINIT INITIALIZES PARAMETERS FOR A GIVEN CONFIGURATION
0009    C    *      OF THE HEX LIBRARY.  A PROGRAM USING THE HEX LIBRARY MUST
0010    C    *      CALL HXINIT BEFORE CALLING ANY OTHER HEX LIBRARY ROUTINES.
0011    C    *      THE INITIALIZATION REQUIRES INPUT PARAMETERS SPECIFYING
0012    C    *      THE DEVICE NUMBER TO WHICH ERROR MESSAGES ARE TO BE WRIT-
0013    C    *      TEN, THE MAXIMUM AND MINIMUM HEX LEVELS, THE LATITUDE AND
0014    C    *      LONGITUDE OF THE ORIGIN OF THE COORDINATE SYSTEM, AND THE
0015    C    *      DIAMETER OF HEXES AT SOME CONVENIENT LEVEL OF AGGREGATION.
0016    C    *      THESE AND/OR FUNCTIONS OF THESE PARAMETERS ARE SAVED IN
0017    C    *      THE COMMON BLOCK /HEX/, WHICH IS RESERVED FOR HEX LI-
0018    C    *      BRARY USE.
0019    C    *   CALLING SEQUENCE:
0020    C    *      CALL HXINIT(IWRITE,LEVMAX,LEVMIN,DLT,DLN,LEVSIZ,SIZHEX)
0021    C    *   INPUT:
0022    C    *      IWRITE - DEVICE NUMBER TO WHICH HEX ERROR MESSAGES ARE TO
0023    C    *               BE WRITTEN
0024    C    *      LEVMAX - MAXIMUM LEVEL OF HEX AGGREGATION.  THE MOST SIG-
0025    C    *               NIFICANT DIGIT IN HEX ADDRESSES WILL REPRESENT
0026    C    *               A HEX AT THIS LEVEL.  NOTE THAT THIS TOGETHER
0027    C    *               WITH THE DIAMETER OF HEXES AS SPECIFIED BY LEVSIZ
0028    C    *               AND SIZHEX WILL BOUND THE AREA COVERED BY THE HEX
0029    C    *               COORDINATE SYSTEM.
0030    C    *      LEVMIN - MINIMUM LEVEL OF HEX AGGREGATION.  THE CONFIGURA-
0031    C    *               TION WILL TREAT HEXES AT THIS LEVEL AS REGULAR
0032    C    *               HEXAGONS.  LARGER HEXES AT HIGHER LEVELS WILL ONLY
0033    C    *               APPROXIMATE REGULAR HEXAGONS IN SHAPE.
0034    C    *      DLT    - LATITUDE OF THE ORIGIN OF THE HEX COORDINATE
0035    C    *               SYSTEM EXPRESSED IN DEGREES AS A FLOATING POINT
0036    C    *               NUMBER
0037    C    *      DLN    - LONGITUDE OF THE ORIGIN OF THE HEX COORDINATE
0038    C    *               SYSTEM EXPRESSED IN DEGREES AS A FLOATING POINT
0039    C    *               NUMBER
0040    C    *      LEVSIZ - HEX LEVEL IN TERMS OF WHICH THE SCALE OF THE HEX
0041    C    *               COORDINATE SYSTEM IS GIVEN
0042    C    *      SIZHEX - DIAMETER OF HEXES AT LEVEL LEVSIZ EXPRESSED IN
0043    C    *               METERS AS A FLOATING POINT NUMBER.  DIAMETERS OF
0044    C    *               HEXES AT OTHER LEVELS ARE COMPLETELY DETERMINED
0045    C    *               GIVEN THIS ONE DIAMETER.
0046    C    *   OUTPUT:
0047    C    *      IHXOUT - OUTPUT DEVICE NUMBER TO WHICH HEX ERROR MESSAGES
0048    C    *               ARE TO BE WRITTEN.  (IN COMMON/HEX/)
0049    C    *      NHLEV  - MAXIMUM NUMBER OF LEVELS OF HEX AGGREGATION.
0050    C    *               THIS INCLUDES LEVMAX THROUGH LEVEL 0 EVEN IF
0051    C    *               LEVMIN IS GREATER THAN 0.  (IN COMMON/HEX/)
0052    C    *      MINLEV - MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
0053    C    *      SLTO   - SINE OF THE LATITUDE OF THE ORIGIN OF THE HEX
0054    C    *               COORDINATE SYSTEM.  (IN COMMON/HEX/)
0055    C    *      CLTO   - COSINE OF THE LATITUDE OF THE ORIGIN OF THE HEX
0056    C    *               COORDINATE SYSTEM.  (IN COMMON/HEX/)
0057    C    *      DLNO   - LONGITUDE OF THE ORIGIN OF THE HEX COORDINATE
```

```
C     *                 SYSTEM EXPRESSED AS A REAL-VALUED QUANTITY IN
C     *                 DEGREES.  (IN COMMON/HEX/)
C     *       DIAM(NDIG)
C     *               - ARRAY CONTAINING THE DIAMETER IN METERS OF A HEX
C     *                 WHOSE ADDRESS CONTAINS NDIG HEX DIGITS.  (IN
C     *                 COMMON/HEX/)
C     *       DIAMTR - DIAMETER IN METERS OF HEXES AT THE MINIMUM HEX
C     *                 LEVEL.  (IN COMMON/HEX/)
C     *       XOFI   - X COORDINATE OF THE VECTOR (I,J) = (1,0) AT THE
C     *                 MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *       YOFI   - Y COORDINATE OF THE VECTOR (I,J) = (1,0) AT THE
C     *                 MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *       XOFJ   - X COORDINATE OF THE VECTOR (I,J) = (0,1) AT THE
C     *                 MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *       YOFJ   - Y COORDINATE OF THE VECTOR (I,J) = (0,1) AT THE
C     *                 MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *       RIOFX  - REAL I COORDINATE AT THE MINIMUM HEX LEVEL OF THE
C     *                 VECTOR (X,Y) = (1,0).  (IN COMMON/HEX/)
C     *       RJOFX  - REAL J COORDINATE AT THE MINIMUM HEX LEVEL OF THE
C     *                 VECTOR (X,Y) = (1,0).  (IN COMMON/HEX/)
C     *       RIOFY  - REAL I COORDINATE AT THE MINIMUM HEX LEVEL OF THE
C     *                 VECTOR (X,Y) = (0,1).  (IN COMMON/HEX/)
C     *       RJOFY  - REAL J COORDINATE AT THE MINIMUM HEX LEVEL OF THE
C     *                 VECTOR (X,Y) = (0,1).  (IN COMMON/HEX/)
C     *       ICON(HDIG + (NDX-1)*7)
C     *       JCON(HDIG + (NDX-1)*7)
C     *               - ARRAYS CONTAINING THE I AND J CONTRIBUTIONS OF
C     *                 EACH POSSIBLE HEX DIGIT (1-7) AT EACH POSSIBLE
C     *                 DIGIT POSITION IN A HEX ADDRESS.  HDIG INDICATES
C     *                 THE HEX DIGIT, AND NDX INDICATES ITS POSITION
C     *                 COUNTING FROM THE RIGHT.  (IN COMMON/HEX/)
C     *       IVAL(HDIG)
C     *       JVAL(HDIG)
C     *               - ARRAYS CONTAINING THE I AND J COORDINATES CORRE-
C     *                 SPONDING TO EACH OF THE 7 SINGLE DIGIT HEX VEC-
C     *                 TORS (1-7).  (IN COMMON/HEX/)
C     *       IMAX(HDIG)
C     *       JMAX(HDIG)
C     *             - ARRAYS CONTAINING THE I AND J COORDINATES (AT THE
C     *                 MINIMUM HEX LEVEL) OF THE CENTERS OF EACH OF THE
C     *                 7 HEXES OF MAXIMUM LEVEL.  (IN COMMON/HEX/)
C     *
C     ***********************************************************************
          INCLUDE 'HEX.CMN'
1 ***********************************************************************
1 *       FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR                     *
1 ***********************************************************************
1       IMPLICIT INTEGER(H,P)
1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR,
1     +                 XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
1     +                 ICON(70),JCON(70),IMAX(7),JMAX(7)
1 ***********************************************************************
        DIMENSION IVAL(7),JVAL(7)
        EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
C     *HEX ROTATIONAL CONSTANT IN DEGREES - ARCTAN(SQRT(3)/5)
        DATA RCON/19.10661/
C     *DEGREES TO RADIANS CONVERSION CONSTANT
```

```
0115          DATA DG2RD/0.01745329/
0116    C       *SET OUTPUT DEVICE NUMBER FOR HEX ERROR MESSAGES
0117             IHXOUT = IWRITE
0118    C       *SET MAXIMUM NUMBER OF LEVELS TO MAXIMUM LEVEL PLUS ONE
0119             NHLEV = LEVMAX + 1
0120    C       *SET MINIMUM LEVEL
0121             MINLEV = LEVMIN
0122    C       *COMPUTE SINE AND COSINE OF LATITUDE OF ORIGIN
0123             RLT  = DLT * DG2RD
0124             SLTO = SIN(RLT)
0125             CLTO = COS(RLT)
0126    C       *SET LONGITUDE OF ORIGIN
0127             DLNO = DLN
0128    C       *DETERMINE NUMBER OF DIGITS IN MINIMUM LEVEL HEX ADDRESS
0129             NDIG = NHLEV - MINLEV
0130    C       *COMPUTE DIAMETER OF HEX AT MINIMUM LEVEL
0131             SQRT7  = SQRT(7.0)
0132             DIAMTR = SIZHEX/(SQRT7**(LEVSIZ-MINLEV))
0133    C       *INITIALIZE DIAMETER COMPUTATION LOOP AT MINIMUM HEX LEVEL
0134             NHD = NDIG
0135             SIZ = DIAMTR
0136    C       *LOOP(FOR ALL HEX LEVELS)
0137     1100    CONTINUE
0138    C          *SET DIAMETER OF HEX AT THIS LEVEL
0139                DIAM(NHD) = SIZ
0140    C          *UPDATE NUMBER OF DIGITS AND DIAMETER FOR NEXT LEVEL
0141                NHD = NHD - 1
0142                SIZ = SIZ * SQRT7
0143    C       *ENDLOOP(HEX LEVEL LOOP)
0144             IF(NHD.GT.0) GOTO 1100
0145    C       *COMPUTE POLAR COORDINATE ANGLES OF I- AND J-AXES AT MIN LEVEL
0146             ROT8   = RCON * FLOAT(MINLEV)
0147             ANGLEI = (+90.0 + ROT8) * DG2RD
0148             ANGLEJ = (-30.0 + ROT8) * DG2RD
0149    C       *COMPUTE X,Y COORDINATES OF UNIT I AND J VECTORS AT MIN LEVEL
0150             XOFI = COS(ANGLEI) * DIAMTR
0151             YOFI = SIN(ANGLEI) * DIAMTR
0152             XOFJ = COS(ANGLEJ) * DIAMTR
0153             YOFJ = SIN(ANGLEJ) * DIAMTR
0154    C       *COMPUTE I,J COORDINATES AT MIN LEVEL OF UNIT X AND Y VECTORS
0155             DTERM = XOFI*YOFJ - XOFJ*YOFI
0156             RIOFX = +YOFJ/DTERM
0157             RJOFX = -YOFI/DTERM
0158             RIOFY = -XOFJ/DTERM
0159             RJOFY = +XOFI/DTERM
0160    C       *LOOP(FOR ALL SINGLE DIGIT HEX VECTORS)
0161             DO 1200 HDIG = 1,7
0162    C          *SET CORRESPONDING I AND J COORDINATES
0163                IPART = IAND(HDIG,1)
0164                JPART = IAND(ISHFT(HDIG,-1),1)
0165                KPART = ISHFT(HDIG,-2)
0166                IVAL(HDIG) = IPART - KPART
0167                JVAL(HDIG) = JPART - KPART
0168    C       *ENDLOOP(SINGLE DIGIT HEX VECTOR LOOP)
0169     1200    CONTINUE
0170    C       *INITIALIZE I AND J CONTRIBUTION COMPUTATION LOOP INDICES
0171             IXOLD = 0
```

HXINIT

```
0172                IXNEW = 7
0173                IXLIM = 7*MDIG
0174      C        *LOOP(FOR ALL POSSIBLE DIGIT POSITIONS PAST FIRST)
0175       1300     CONTINUE
0176      C           *LOOP(FOR ALL HEX DIGITS)
0177                   DO 1400 HDIG = 1,7
0178      C              *COMPUTE I AND J CONTRIBUTIONS FROM LAST POSITION VALUES
0179                      ICON(HDIG+IXNEW) = 2*ICON(HDIG+IXOLD) + JCON(HDIG+IXOLD)
0180                      JCON(HDIG+IXNEW) = 3*JCON(HDIG+IXOLD) - ICON(HDIG+IXOLD)
0181      C           *ENDLOOP(HEX DIGIT LOOP)
0182       1400        CONTINUE
0183      C           *UPDATE LOOP INDICES FOR NEXT DIGIT POSITION
0184                   IXOLD = IXNEW
0185                   IXNEW = IXOLD + 7
0186      C        *ENDLOOP(DIGIT POSITION LOOP)
0187                IF(IXNEW.LT.IXLIM) GOTO 1300
0188      C        *LOOP(FOR ALL HEXES OF MAXIMUM LEVEL)
0189                DO 1500 HDIG = 1,7
0190      C           *SET I AND J COORDINATES AT MINIMUM HEX LEVEL
0191                   IMAX(HDIG) = ICON(HDIG+IXOLD)
0192                   JMAX(HDIG) = JCON(HDIG+IXOLD)
0193      C        *ENDLOOP(HEXES OF MAXIMUM LEVEL LOOP)
0194       1500     CONTINUE
0195      C     *ENDROUTINE(HXINIT)
0196            RETURN
0197            END
```

```
      INTEGER FUNCTION HXINV(HEX)
C    *ROUTINE(FIND HEX INVERSE OF AN OCTAL HEX NUMBER - HXINV)
C    ***************************************************************
C    *
C    *   DESIGNER/PROGRAMMER:
C    *      DON KRECKER  12 SEPTEMBER 1980
C    *   PURPOSE:
C    *      HXINV COMPUTES THE INVERSE OF A HEX NUMBER EXPRESSED IN
C    *      OCTAL REPRESENTATION.  THE ALGORITHM INVERTS ALL HEX DIGITS
C    *      IN PARALLEL.  HEX DIGITS 1 THROUGH 6 ARE COMPLEMENTED MOD
C    *      7, WHILE THE HEX DIGIT 7 REMAINS UNCHANGED.  OCTAL 0 DIGITS
C    *      TO THE LEFT OF THE MOST SIGNIFICANT HEX DIGIT ALSO REMAIN
C    *      UNCHANGED.  THE ALGORITHM USES SHIFT AND LOGICAL OPERATIONS
C    *      TO FLAG THE DIGITS 1 THROUGH 6.  THEN THESE DIGITS ARE IN-
C    *      VERTED WHILE THE 0 AND 7 DIGITS ARE UNTOUCHED.
C    *   CALLING SEQUENCE:
C    *      HXINV = HXINV(HEX)
C    *   INPUT:
C    *      HEX     - HEX NUMBER TO BE INVERTED
C    *   OUTPUT:
C    *      HXINV   - HEX INVERSE OF THE ARGUMENT HEX NUMBER
C    *
C    ***************************************************************
      IMPLICIT INTEGER(H,P)
C    *MASK OF OCTAL 1 DIGITS
      DATA M1/'1111111111'O/
C       *SET LOCAL VARIABLES TO HEX NUMBER SHIFTED 0, 1, AND 2 BITS
        HSHFT0 = HEX
        HSHFT1 = ISHFT(HEX,-1)
        HSHFT2 = ISHFT(HEX,-2)
C       *FLAG DIGITS 1 THROUGH 6 IN LOW ORDER BIT POSITION OF DIGITS
        M1T06 = IAND(IOR(IEOR(HSHFT0,HSHFT1),IEOR(HSHFT1,HSHFT2)),M1)
C       *EXTEND FLAGS TO MASK ALL 3 BITS OF DIGITS 1 THROUGH 6
        M1T06 = M1T06 * 7
C       *INVERT DIGITS 1 THROUGH 6 AND RETURN RESULT AS HEX INVERSE
        HXINV = IEOR(HEX,M1T06)
C    *ENDROUTINE(HXINV)
      RETURN
      END
```

```
      SUBROUTINE IGRID(X,Y,I,J)
*****************************************************************
*       THIS ROUTINE CONVERTS THE UTM COORDINATES X,Y TO THE    *
*       GRID INDICES I,J.                                       *
*       THE ORIGIN IS 500,000 5,600,000 IN GRID ZONE 32U AND    *
*       THE UNITS ARE 20 METERS. THE INDEX FOR THE ORIGIN IS    *
*       IN RECORD (65,65) OUT OF 128*128 RECORDS.               *
*****************************************************************
       IMPLICIT INTEGER*4 (A-Z)
      I=((X-500000)/20+32500)/500
      J=((Y-5600000)/20+32500)/500
      RETURN
      END
```

```
0001            FUNCTION ICODE(I,J)
0002      ***********************************************************
0003      *       EXTRACTS THE FEATURE CODE FROM IBUF(I,J)          *
0004      ***********************************************************
0005      *       INPUTS: I,J; THE Y AND X INDICES,RESPECTIVELY     *
0006      *       OUTPUTS:  ICODE, THE SURFACE FEATURE CODE         *
0007      ***********************************************************
0008            IMPLICIT INTEGER*2 (I-N)
0009            INCLUDE 'MAP.CMN'
0010    1 ************************************************************
0011    1 *       IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH   *
0012    1 *       THE FIRST INDEX CORRESPONDS TO NORTHING, AND      *
0013    1 *       THE SECOND TO EASTING.                            *
0014    1 ************************************************************
0015    1       INTEGER*2 IBUF(400,400)
0016    1     COMMON /MAP/IBUF
0017    1 ************************************************************
0018            ICODE=IBUF(I,J)-IELV(I,J)*8
0019 C  D       CALL CMCLOS
0020 C  D       PRINT*,ICODE,IELV(I,J)
0021 C  D       CALL CMOPEN
0022            RETURN
0023            END
```

```
0001              FUNCTION IELV(I,J)
0002      ************************************************************
0003      *       EXTRACTS THE ELEVATION  FROM IBUF(I,J)             *
0004      ************************************************************
0005      *       INPUTS:  I,J; THE Y AND X COORDINATES,RESPECTIVELY *
0006      *       OUTPUTS: IELV, THE ELEVATION OF THE POINT          *
0007      ************************************************************
0008              IMPLICIT INTEGER*2 (I-N)
0009              INCLUDE 'MAP.CMN'
0010    1 ***********************************************************
0011    1 *       IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH   *
0012    1 *       THE FIRST INDEX CORRESPONDS TO NORTHING, AND      *
0013    1 *       THE SECOND TO EASTING.                            *
0014    1 ***********************************************************
0015    1         INTEGER*2 IBUF(400,400)
0016    1       COMMON /MAP/IBUF
0017    1 ***********************************************************
0018  C   D       PRINT*,'IELV',I,J
0019              IELV=IBUF(I,J)/8
0020              RETURN
0021              END
```

```
      INTEGER FUNCTION IJL2HA(I,J,LEVIJ)
C     *ROUTINE(CONVERT I,J, AND LEVEL TO EQUIVALENT HEX ADDRESS-IJL2HA)
C     ****************************************************************
C     *
C     *   DESIGNER/PROGRAMMER:
C     *      DON KRECKER  18 SEPTEMBER 1980
C     *   PURPOSE:
C     *      IJL2HA CONVERTS A GIVEN I,J, AND LEVEL TRIPLE TO AN EQUI-
C     *      VALENT HEX ADDRESS IN OCTAL REPRESENTATION.  I AND J ARE
C     *      OBLIQUE COORDINATES EXPRESSED IN UNITS CORRESPONDING TO
C     *      HEX DIAMETERS AT THE GIVEN LEVEL OF HEX AGGREGATION, AND
C     *      THE COMPUTED HEX ADDRESS WILL BE AT THIS SAME LEVEL.
C     *      IJL2HA IS THE INVERSE OF THE SUBROUTINE HA2IJL WHICH COM-
C     *      PUTES THE I,J, AND LEVEL TRIPLE CORRESPONDING TO A GIVEN
C     *      HEX ADDRESS.
C     *      AFTER CHECKING THE VALIDITY OF THE REQUESTED HEX LEVEL,
C     *      THE ALGORITHM CONSTRUCTS THE HEX ADDRESS AT THAT LEVEL
C     *      CENTERED AT THE ORIGIN.  THIS IS A STRING OF (NHLEV-LEVEL)
C     *      HEX DIGITS, EACH EQUAL TO 7, WHERE NHLEV IS THE MAXIMUM
C     *      NUMBER OF LEVELS OF HEX AGGREGATION.  THEN, WORKING FROM
C     *      RIGHT TO LEFT, SUCCESSIVE HEX DIGITS ARE EXTRACTED FROM
C     *      THE I,J COORDINATES AND INSERTED IN PLACE OF 7S IN THE
C     *      HEX ADDRESS.  THE HEX DIGITS ARE COMPUTED AS DESCRIBED IN
C     *      THE ROUTINE IJ2HV.
C     *   CALLING SEQUENCE:
C     *      IJL2HA = IJL2HA(I,J,LEVIJ)
C     *   INPUT:
C     *      I,J     - INTEGER-VALUED OBLIQUE COORDINATES WHICH ARE TO
C     *                BE CONVERTED TO AN EQUIVALENT HEX ADDRESS AT THE
C     *                SPECIFIED LEVEL OF HEX AGGREGATION
C     *      LEVIJ   - LEVEL OF HEX AGGREGATION WITH RESPECT TO WHICH
C     *                THE I,J COORDINATES ARE EXPRESSED AND AT WHICH
C     *                THE HEX ADDRESS IS TO BE COMPUTED
C     *      NHLEV   - MAXIMUM NUMBER OF LEVELS OF HEX AGGREGATION.
C     *                (IN COMMON/HEX/)
C     *      MINLEV  - MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *      IVAL(HDIG)
C     *      JVAL(HDIG)
C     *              - ARRAYS CONTAINING THE I,J COORDINATES CORRESPON-
C     *                DING TO EACH OF THE 7 SINGLE DIGIT HEX VECTORS
C     *                (1-7).  FOR EXAMPLE, SINCE THE HEX VECTOR 1 COR-
C     *                RESPONDS TO (I,J) = (1,0), IVAL(1) = 1 AND
C     *                JVAL(1) = 0.  (IN COMMON/HEX/)
C     *   OUTPUT:
C     *      IJL2HA  - HEX ADDRESS CORRESPONDING TO THE GIVEN I,J OBLIQUE
C     *                COORDINATES AT THE SPECIFIED LEVEL OF AGGREGATION
C     *
C     ****************************************************************
      INCLUDE 'HEX.CMN'
1 ***********************************************************************
1 *       FOR DEFINITIONS OF VARIABLES SEE HXINIT.FOR                   *
1 ***********************************************************************
1       IMPLICIT INTEGER(H,P)
1       COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMTR,
1      +           XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
1      +           ICON(70),JCON(70),IMAX(7),JMAX(7)
1 ***********************************************************************
```

```
      DIMENSION IVAL(7),JVAL(7)
      EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
C       *IF(VALID HEX LEVEL)THEN
         IF(LEVIJ.LT.MINLEV) GOTO 1200
         IF(LEVIJ.GE.NHLEV)  GOTO 1200
C          *CONSTRUCT HEX ADDRESS AT GIVEN LEVEL CENTERED AT ORIGIN
            HADR = ISHFT(1,3*(NHLEV-LEVIJ)) - 1
C          *INITIALIZE HEX DIGIT EXTRACTION LOOP
            ILOCAL = I
            JLOCAL = J
            HSHIFT = 0
C          *LOOP UNTIL(NO MORE HEX DIGITS TO EXTRACT FROM (I,J))
 1100       CONTINUE
C             *COMPUTE NEXT HEX DIGIT = (I + 2*J) MOD 7
              HDIG = ILOCAL + JLOCAL + JLOCAL
              HDIG = HDIG - (HDIG/7)*7
              IF(HDIG.LE.0) HDIG = HDIG + 7
C             *INSERT HEX DIGIT AT FRONT OF HEX ADDRESS
              HADR   = HADR - ISHFT(7-HDIG,HSHIFT)
              HSHIFT = HSHIFT + 3
C             *SUBTRACT (I,J) CORRESPONDING TO NEWLY FOUND HEX DIGIT
              INEW = ILOCAL - IVAL(HDIG)
              JNEW = JLOCAL - JVAL(HDIG)
C             *SHRINK NEW I,J VECTOR TO NEXT LOWER HEX LEVEL
              ILOCAL = (3*INEW - JNEW)/7
              JLOCAL = (INEW + JNEW + JNEW)/7
C          *ENDLOOP(HEX DIGIT EXTRACTION LOOP)
            IF(ILOCAL.NE.0) GOTO 1100
            IF(JLOCAL.NE.0) GOTO 1100
C          *RETURN COMPUTED HEX ADDRESS
            IJL2HA = HADR
C       *ELSE(INVALID HEX LEVEL)
         GOTO 1300
 1200    CONTINUE
C          *SET RETURN HEX ADDRESS TO ZERO
            IJL2HA = 0
C          *INCLUDE(GENERATE HEX ERROR MESSAGE - HXERR)
            CALL HXERR(6HIJL2HA,3,LEVIJ,0,0,0)
C       *ENDIF(HEX LEVEL CHECK)
 1300    CONTINUE
C    *ENDROUTINE(IJL2HA)
      RETURN
      END
```

```
0058    C           *SET RETURN HEX ADDRESS TO ZERO
0059                 IJM2HA = 0
0060    C           *INCLUDE(GENERATE HEX ERROR MESSAGE - HXERR)
0061                 CALL HXERR(6HIJM2HA,3,LEV,0,0,0)
0062    C        *ENDIF(HEX LEVEL CHECK)
0063     1200     CONTINUE
0064    C     *ENDROUTINE(IJM2HA)
0065           RETURN
0066           END
```

```
      INTEGER FUNCTION IJM2HA(I,J,LEV)
C     *ROUTINE(CONVERT MIN LEVEL I,J COORDINATES TO HEX ADDRESS-IJ42HA)
C     ****************************************************************
C     *
C     *   DESIGNER/PROGRAMMER:
C     *      DON KRECKER  20 SEPTEMBER 1980
C     *   PURPOSE:
C     *      IJM2HA TAKES A POINT EXPRESSED IN I,J OBLIQUE COORDINATES
C     *      AT THE MINIMUM HEX LEVEL AND COMPUTES THE ADDRESS OF THE
C     *      HEX AT THE SPECIFIED LEVEL WHICH CONTAINS THE POINT.  THE
C     *      HEX ADDRESS IS COMPUTED IN OCTAL REPRESENTATION.  IF THE
C     *      SPECIFIED HEX LEVEL IS GREATER THAN THE MINIMUM HEX LEVEL,
C     *      THE COMPUTED HEX MAY DIFFER FROM THE HEX (AT THIS LEVEL)
C     *      WHOSE CENTER IS CLOSEST TO THE GIVEN POINT.  THE REASON IS
C     *      THAT HEXES AT HIGHER LEVELS OF AGGREGATION ARE NOT TRUE
C     *      REGULAR HEXAGONS BUT ONLY APPROXIMATE REGULAR HEXAGONS IN
C     *      SHAPE.  IJM2HA IS THE INVERSE OF THE SUBROUTINE HA2IJM,
C     *      WHICH CONVERTS A HEX ADDRESS TO I,J COORDINATES AT THE
C     *      MINIMUM HEX LEVEL.
C     *      AFTER CHECKING THE VALIDITY OF THE REQUESTED HEX LEVEL,
C     *      IJM2HA USES IJL2HA TO COMPUTE THE MINIMUM LEVEL HEX ADDRESS
C     *      WITH THE GIVEN I,J COORDINATES.  THEN THE ADDRESS OF THE
C     *      HEX AT THE SPECIFIED LEVEL IS FOUND BY TRUNCATING THE AP-
C     *      PROPRIATE NUMBER OF HEX DIGITS.
C     *   CALLING SEQUENCE:
C     *      IJM2HA = IJM2HA(I,J,LEV)
C     *   INPUT:
C     *      I,J     - INTEGER-VALUED OBLIQUE COORDINATES AT THE MINIMUM
C     *                HEX LEVEL OF A POINT WHOSE CONTAINING HEX AT A
C     *                SPECIFIED LEVEL IS TO BE COMPUTED
C     *      LEV     - LEVEL OF AGGREGATION OF THE HEX ADDRESS TO BE
C     *                COMPUTED
C     *      NHLEV   - MAXIMUM NUMBER OF LEVELS OF HEX AGGREGATION.
C     *                (IN COMMON/HEX/)
C     *      MINLEV  - MINIMUM HEX LEVEL.  (IN COMMON/HEX/)
C     *   OUTPUT:
C     *      IJM2HA  - ADDRESS OF THE HEX AT THE REQUESTED LEVEL WHICH
C     *                CONTAINS THE GIVEN I,J POINT
C     *
C     ****************************************************************
      IMPLICIT INTEGER(H,P)
      COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMT
     $R,
     +           XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
     +           ICON(70),JCON(70),IMAX(7),JMAX(7)
      DIMENSION IVAL(7),JVAL(7)
      EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
C       *IF(VALID HEX LEVEL)THEN
        IF(LEV.LT.MINLEV) GOTO 1100
        IF(LEV.GE.NHLEV)  GOTO 1100
C         *CONVERT I,J COORDINATES TO MINIMUM LEVEL HEX ADDRESS
          HADR = IJL2HA(I,J,MINLEV)
C         *TRUNCATE HEX DIGITS TO GET HEX ADDRESS AT REQUIRED LEVEL
          IJM2HA = ISHFT(HADR,3*(MINLEV-LEV))
C       *ELSE(INVALID HEX LEVEL)
        GOTO 1200
 1100   CONTINUE
```

```
0001          SUBROUTINE IJM2XY(I,J,X,Y)
0002    C    *ROUTINE(CONVERT MIN LEVEL I,J TO X,Y COORDINATES - IJ42XY)
0003    C    ***************************************************************
0004    C    *
0005    C    *   DESIGNER/PROGRAMMER:
0006    C    *      DON KRECKER  20 SEPTEMBER 1980
0007    C    *   PURPOSE:
0008    C    *      IJM2XY CONVERTS A PAIR OF I,J OBLIQUE COORDINATES AT THE
0009    C    *      MINIMUM HEX LEVEL TO THE EQUIVALENT X,Y CARTESIAN COORDI-
0010    C    *      NATES IN METERS.  THIS ROUTINE IS THE INVERSE OF THE SUB-
0011    C    *      ROUTINE XY2IJM.
0012    C    *      THE CONVERSION IS EFFECTED BY APPLYING A LINEAR TRANSFOR-
0013    C    *      MATION (IN THE FORM OF A MATRIX MULTIPLICATION) TO THE
0014    C    *      I,J VECTOR TO OBTAIN AN X,Y VECTOR.
0015    C    *   CALLING SEQUENCE:
0016    C    *      CALL IJM2XY(I,J,X,Y)
0017    C    *   INPUT:
0018    C    *      I,J     - INTEGER-VALUED OBLIQUE COORDINATES EXPRESSED IN
0019    C    *                HEX DIAMETERS AT THE MINIMUM HEX LEVEL WHICH ARE
0020    C    *                TO BE CONVERTED TO CARTESIAN COORDINATES
0021    C    *      XOFI    - X COORDINATE OF THE VECTOR (I,J) = (1,0).
0022    C    *                (IN COMMON/HEX/)
0023    C    *      YOFI    - Y COORDINATE OF THE VECTOR (I,J) = (1,0).
0024    C    *                (IN COMMON/HEX/)
0025    C    *      XOFJ    - X COORDINATE OF THE VECTOR (I,J) = (0,1).
0026    C    *                (IN COMMON/HEX/)
0027    C    *      YOFJ    - Y COORDINATE OF THE VECTOR (I,J) = (0,1).
0028    C    *                (IN COMMON/HEX/)
0029    C    *   OUTPUT:
0030    C    *      X,Y     - REAL-VALUED CARTESIAN COORDINATES EXPRESSED IN
0031    C    *                METERS EQUIVALENT TO THE GIVEN OBLIQUE COORDINATES
0032    C    *
0033    C    ***************************************************************
0034          IMPLICIT INTEGER(H,P)
0035          COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMT
0036         SR,
0037         +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
0038         +                ICON(70),JCON(70),IMAX(7),JMAX(7)
0039          DIMENSION IVAL(7),JVAL(7)
0040          EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
0041    C       *CONVERT INTEGER-VALUED I,J COORDINATES TO REAL
0042             RI = FLOAT(I)
0043             RJ = FLOAT(J)
0044    C       *TRANSFORM TO EQUIVALENT X,Y COORDINATES IN METERS
0045             X = XOFI * RI + XOFJ * RJ
0046             Y = YOFI * RI + YOFJ * RJ
0047    C    *ENDROUTINE(IJM2XY)
0048          RETURN
0049          END
```

```
      SUBROUTINE LABEL
*************************************************************
*     THIS SUBROUTINE LABELS THE COLOR CODES USED IN        *
*     THE MAPPED-SECTION DISPLAY FILE.                      *
*************************************************************
      REAL LEFT
      INTEGER*4 LABELS(7,2)
      DATA LABELS/'FORE','URBA','MARS','NULL','WATE','HEAT',
     + 'OPEN','ST','N','H',' ','R','H',' '/
      INCLUDE 'WINDO.CMN'
1 ************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND   *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER    *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS  *
1 *     THE CONTOUR INTERVAL.                                 *
1 ************************************************************
1     DIMENSION FWINXY(4)
1     COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ************************************************************
      CALL CMOPEN
      CALL VWPORT(0.,40.,0.,90.)
      CALL WINDOW(0.,40.,0.,90.)
      CALL TXTCLR(4)
      CALL TXAM
      CALL TXSIZE(0,2.,2.)
      LEFT=5.
      RIGHT=15.
      DO I=1,7
         BOTTOM=10.*I
         TOP=BOTTOM+8.
         CALL LINCLR(I)
         DO X=LEFT,RIGHT,.5
            CALL MOVE(X,BOTTOM)
            CALL DRAW(X,TOP)
         ENDDO
            CALL MOVE(X,BOTTOM+5.)
            CALL TEXT(4,LABELS(I,1))
            CALL TEXT(2,LABELS(I,2))
            CALL MOVE(X,BOTTOM+2.)
            CALL TEXT(5,'CODE:')
            CALL INUMBR(I,1)
      ENDDO
      FX=FWINXY(1)
      FY=FWINXY(3)
      CALL MOVE(20.,6.)
      CALL RNUMBR(FX,-1,8)
      CALL MOVE(20.,0.)
      CALL RNUMBR(FY,-1,8)
      R=1.
      T=1.
      R=R*98.8+40.5
      T=T*98.8+.5
       CALL VWPORT(40.,R,0.,T)
      CALL WINDOW(FWINXY(1),FWINXY(2),FWINXY(3),FWINXY(4))
      CALL CMCLOS
      RETURN
      END
```

```
      SUBROUTINE INTERS(IX1,IY1,IX2,IY2,HEIGHT,FRAC,ICROSS)
*****************************************************************
*       FINDS INTERSECTION OF SEGMENT X1,Y1 TO X2,Y2 WITH HEIGHT
*****************************************************************
*       INPUTS:
*               IX1,IY1,IX2,IY2-- INDICES TO 2 POINTS IN THE DATA ARRAY
*               HEIGHT__ ELEVATION OF THE CONTOUR BEING TRACED
*       OUTPUTS: FRAC -- FRACTION OF DISTANCE FROM POINT1 TOWARDS POINT2
*       ***  H. JONES ***
*****************************************************************
      IMPLICIT INTEGER*2 (I-N)
C
C    *CHECK FOR NO INTERSECTION
      ICROSS = 0
      Z1=IELV(IY1,IX1)
      Z2=IELV(IY2,IX2)
      IF(Z1 .GE. HEIGHT  .AND. Z2 .GE. HEIGHT) THEN
         GO TO 10
      ENDIF
      IF(Z1 .LE. HEIGHT  .AND. Z2 .LE. HEIGHT) THEN
         GO TO 10
      ENDIF
C
C    *COMPUTE INTERSECTION
      ICROSS = 1
      FRAC = (HEIGHT -Z1) / (Z2-Z1)
C
   10 RETURN
      END
```

```
0001            SUBROUTINE MAP2LL(FNAME,FLON,FLAT)
0002      ************************************************
0003      *     THIS ROUTINE COMPUTES THE LAT,LON OF THE *
0004      *     CENTER OF A SHEET FROM THE M745 SERIES.  *
0005      ************************************************
0006      *     INPUTS: FNAME-- THE NAME OF THE MAP      *
0007      *     OUTPUTS: FLON,FLAT-- REAL-VALUED LAT AND *
0008      *              LON OF THE CENTER OF THE MAP    *
0009      ************************************************
0010            CHARACTER*5 FNAME
0011            DIMENSION D(2)
0012            PARAMETER PI=3.141592654
0013            P_RAD=PI/180.
0014
0015            DO I=2,4,2
0016               DECODE(2,10,FNAME(I:I+1)) D(I/2)
0017            ENDDO
0018      10    FORMAT(F3.0)
0019
0020            DLAT=50+(59-D(1))/2.*.2+.1
0021            DLON=9+(D(2)-20)/2./3.+1./6.
0022            FLAT=DLAT*P_RAD
0023            FLON=DLON*P_RAD
0024            RETURN
0025            END
```

```
0001            SUBROUTINE MAP2UTM(FNAME,FEAST,FNORTH,CMERID)
0002    ************************************************************
0003    *****  THIS ROUTINE COMPUTES THE UTM CENTER OF A SHEET FROM
0004    *****  THE M745 SERIES.
0005    ************************************************************
0006            CHARACTER*5 FNAME
0007            DIMENSION D(2)
0008            PARAMETER PI=3.141592654
0009            P_RAD=PI/180.
0010
0011            DO I=2,4,2
0012               DECODE(2,10,FNAME(I:I+1)) D(I/2)
0013            ENDDO
0014    10      FORMAT(F3.0)
0015
0016            DLAT=50+(59-D(1))/2.*.2+.1
0017            DLON=9+(D(2)-20)/2./3.+1./6.
0018            FLAT=DLAT*P_RAD
0019            FLON=DLON*P_RAD
0020            CALL ADSMP(FLAT,FLON,CMERID,FEAST,FNORTH)
0021            FEAST=FEAST+500000.
0022            RETURN
0023            END
```

```
      SUBROUTINE MAPPER
*****************************************************************
*     CALLS THE ROUTINES TO DISPLAY THE TERRAIN DATA            *
*****************************************************************
*     INPUTS:  NONE                                             *
*     OUTPUTS: NONE                                             *
*****************************************************************
      IMPLICIT INTEGER*2 (I-N)
      INCLUDE 'WINDO.CMN'
1 ***************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND     *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER      *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS    *
1 *     THE CONTOUR INTERVAL.                                   *
1 ***************************************************************
1     DIMENSION FWINXY(4)
1     COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ***************************************************************
      INCLUDE 'MAP.CMN'
1 *************************************************************
1 *     IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH       *
1 *     THE FIRST INDEX CORRESPONDS TO NORTHING, AND          *
1 *     THE SECOND TO EASTING.                                *
1 *************************************************************
1     INTEGER*2 IBUF(400,400)
1    COMMON /MAP/IBUF
1 *************************************************************
      INCLUDE 'ANSWER.CMN'
1 ***********************************************
1     CHARACTER*1 FEA,CON
1     COMMON/ANSWER/FEA,CON
1 ***********************************************
      CALL CMOPEN
      CALL NEWPAG

      IF(FEA.EQ.'Y')THEN
        CALL LABEL
        INCR=2
        CALL FEATURES(INCR)
      ENDIF
      IF(CON.EQ.'Y')THEN
        CALL MAXMIN
        DBRES=100
        ZMIN=MIN
        ZMAX=MAX
        CONRES=400. ! CHECK ELEVATION EVERY 400 M
        IF(MAX.GT.-32767.AND.MIN.LT.32767)THEN
         CALL CMOPEN
         CALL LINCLR(4)
         CALL VWPORT(40.,139.3,0.,99.3)
         CALL WINDOW(0.,40000.,0.,40000.)
         CALL DRWCON(FWINXY,DBRES,CONRES,ZMIN,ZMAX,ZDELT)
        ELSE
          CALL CMCLOS
          PRINT*,'NO DATA IN THIS AREA.'
        ENDIF
      ENDIF
```

```
30      CONTINUE
        CALL WINDOW(FWINXY(1),FWINXY(2),FWINXY(3),FWINXY(4))
        CALL GRIDS
        CALL CMCLOS
        RETURN
        END
```

```
      SUBROUTINE MAPIN(IR,JC,MGR,ERR)
*****************************************************************
*     READS IN A 10KM FILE AND PLACES IT IN THE BUFFER, IBUF    *
*****************************************************************
*     INPUTS:                                                   *
*             IR,JC-- THE 'ROW' AND 'COLUMN' OF THE FILE,       *
*                     EACH RUNS FROM 0 TO 3                     *
*             MGR-- THE MILITARY GRID REFERENCE (UTM) NAME OF   *
*                   THE FILE TO BE OPENED                       *
*     OUTPUTS: ERR-- AN ERROR FLAG                              *
*****************************************************************
      LOGICAL*1 ERR
      CHARACTER*7 MGR
      INTEGER*2 ROW,COL
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE    *
1 *     AREA IN THE ARRAY IBUF.                             *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      INCLUDE 'MAP.CMN'
1 ***********************************************************
1 *     IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH     *
1 *     THE FIRST INDEX CORRESPONDS TO NORTHING, AND        *
1 *     THE SECOND TO EASTING.                              *
1 ***********************************************************
1     INTEGER*2 IBUF(400,400)
1   COMMON /MAP/IBUF
1 ***********************************************************
      INTEGER*2 SQUARE(100,100)
      OPEN(UNIT=9,NAME=MGR,STATUS='OLD',FORM='UNFORMATTED',ERR=10)
      READ(9)SQUARE

20    CONTINUE
      IROW=100*IR
      JCOL=100*JC
        DO J=1,100
        COL=JCOL+J
          DO I=1,100
          ROW=IROW+I
          IBUF(ROW,COL)=SQUARE(I,J)
          ENDDO
        ENDDO
      CLOSE(UNIT=9)
      RETURN
10    ERR=.TRUE.
      PRINT*,'NO FILE: ',MGR
      DO I=1,100
         DO J=1,100
            SQUARE(I,J)=0
         ENDDO
      ENDDO
      GOTO20
      END
```

```
      SUBROUTINE MAPOUT(IR,JC,MGR,ERR)
*****************************************************************
*     REWRITES THE 10KM FILES.                                  *
*****************************************************************
*     INPUTS:                                                   *
*            IR,JC--THE 'ROW' AND 'COLUMN' OF THE 100*100 ARRAY TO *
*                   BE WRITTEN;  THE VALUES RUN FROM 0 TO 3     *
*            MGR--  THE NAME OF THE FILE                        *
*     OUTPUTS:  ERR-- AN ERROR FLAG                             *
*****************************************************************
      LOGICAL*1 ERR
      CHARACTER*7 MGR
      INTEGER*2 ROW,COL
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE  *
1 *     AREA IN THE ARRAY IBUF.                           *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      INCLUDE 'MAP.CMN'
1 ***********************************************************
1 *     IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH   *
1 *     THE FIRST INDEX CORRESPONDS TO NORTHING, AND      *
1 *     THE SECOND TO EASTING.                            *
1 ***********************************************************
1     INTEGER*2 IBUF(400,400)
1    COMMON /MAP/IBUF
1 ***********************************************************
      INTEGER*2 SQUARE(100,100)
      OPEN(UNIT=9,NAME=MGR,STATUS='UNKNOWN',FORM='UNFORMATTED',ERR=10)

      IROW=100*IR
      JCOL=100*JC
        DO J=1,100
        COL=JCOL+J
          DO I=1,100
          ROW=IROW+I
          SQUARE(I,J)=IBUF(ROW,COL)
          ENDDO
        ENDDO
      WRITE(9)SQUARE
      CLOSE(UNIT=9)
      RETURN
10    ERR=.TRUE.
      PRINT*,'ERROR ON OPENING FILE ',MGR
      RETURN
      END
```

```
      SUBROUTINE MAXMIN
*****************************************************
*     FINDS THE MAX AND MIN ELEVATION VALUES IN THE   *
*     DATA ARRAY.                                     *
*****************************************************
    IMPLICIT INTEGER*2 (I-N)
    CHARACTER*14 NAMEF
      INCLUDE 'WINDO.CMN'
1 ************************************************************
1 *     FWINXY CONTAINS THE X MIN AND MAX AND THE Y MIN AND  *
1 *     MAX RESPECTIVELY FOR THE WINDOW. MIN AND MAX REFER   *
1 *     TO THE MIN AND MAX OF ELEVATION VALUES, AND ZDELT IS *
1 *     THE CONTOUR INTERVAL.                                *
1 ************************************************************
1     DIMENSION FWINXY(4)
1     COMMON/WINDO/FWINXY,MIN,MAX,ZDELT
1 ************************************************************
      INCLUDE 'MAP.CMN'
1 *********************************************************
1 *     IBUF HOLDS A 40*40KM ARRAY OF DISPLAY DATA,WITH   *
1 *     THE FIRST INDEX CORRESPONDS TO NORTHING, AND      *
1 *     THE SECOND TO EASTING.                            *
1 *********************************************************
1     INTEGER*2 IBUF(400,400)
1    COMMON /MAP/IBUF
1 *********************************************************
      MIN=32767
      MAX=-32767
      X1=1
      X2=400
      Y3=1
      Y4=400
C     IN THE INTEREST OF TIME,ONLY EVERY FOURTH POINT IS CHECKED
      X=4.
      Y=4.
      DO J=X1,X2,X
        DO I=Y3,Y4,Y
          IMIN=MIN0(IELV(I,J),MIN)
          MAX=MAX0(IELV(I,J),MAX)
          IF(IMIN.GT.0)THEN
           MIN=IMIN
          ENDIF
        ENDDO
      ENDDO
      RETURN
    END
```

```
      SUBROUTINE OPENERS
*****************************************************
*     THIS ROUTINE SIMPLY OPENS THE GRID,NODE,LINK,AND *
*     SUBNODE FILES, AND THE ISAM FILE WHICH CONTAINS *
*     THE 'HEXISED' LOC OR HYDRO DATA                 *
*****************************************************
      OPEN(UNIT=1,NAME='GRID',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=2,NAME='NODE',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=3,NAME='ROAD',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
      OPEN(UNIT=4,NAME='SUBN',TYPE='OLD',READONLY,SHARED,
     *ACCESS='DIRECT',BLOCKSIZE=2000)
*
*     NOW FOR THE ISAM FILE
      OPEN(UNIT=7,NAME='HEXROAD',STATUS='UNKNOWN',
     + ORGANIZATION='INDEXED',ACCESS='KEYED',RECL=2,
     + RECORDTYPE='FIXED',FORM='UNFORMATTED',
     + KEY=(1:4:INTEGER))
      RETURN
      END
```

```
      SUBROUTINE PACKER(HEX,HSIDE,IADJ,LU)
C*********************************************************
C****  THIS PACKS THE MAX VALUE OF CONNECTIVITY 'TYPE'
C****  AT 'SIDE' OF THE HEX INTO THE APPROPRIATE DIGIT
C****  OF 'SIDES'
C*********************************************************
C*****  INPUT:
C*****        HEX--INTERNAL HEX#
C*****        HSIDE-- HEX SIDE CURRENTLY BEING PROCESSED
C*****        IADJ-- ERROR FLAG DENOTING THAT THE DIFFERENCE
C*****               IN THE TWO HEXES WAS TOO LARGE
C*****  OUTPUT:
C*****        HEX
C*****        HSIDE
C*****        SIDE-- DECIMAL NUMBER OF THE SIDE;RANGE:1-6
C*****        SIDES-- THE PACKED CONNECTIVITIES OF THE HEX
C*********************************************************
C
      IMPLICIT INTEGER (H,P)
      INTEGER*4 TOP,TMP,BOTTOM,SID9,SID10,SIDE,HSIDE
      INCLUDE 'PACKER.CMN'
1 *********************************************************
1     INTEGER*4 SIDES ! PACKED CONNECTIVITIES          *
1     INTEGER*4 LTYPE ! CONNECTIVITY FOR CURRENT SIDE  *
1     COMMON/PACK/SIDES,LTYPE
1 *********************************************************
C D   PRINT*,'LU IN PACKER',LU
C
C*****  IADJ=1 IF THE HEXES WERE ADJACENT;IE IF THEIR
C*****  VECTOR DIFFERENCE IS FROM 1 TO 6, AND 0 IF NOT
      IADJ=1
      CALL STRIPPER(HSIDE,SIDE)
      IF(SIDE.LE.0)THEN
         IADJ=0
         RETURN
      ENDIF
      SID9=10**(SIDE-1)
      SID10=10**SIDE
C
C     STRIP OFF THE UPPER DIGITS
      TOP=INT(SIDES/SID10)*SID10
C
C     NOW GET THE LOWER ONES
      TMP=SIDES-TOP
      ISIDE=INT(TMP/SID9)
      BOTTOM=TMP-ISIDE*SID9
C
C     RESET THE VALUE FOR THE CONNECTIVITY
      ITYPE=MAX(ISIDE,LTYPE)
C
C     REPACK
      SIDES=TOP+ITYPE*SID9+BOTTOM
      CALL HEXWRITE(HEX,SIDES,LU)
      RETURN
      END
```

```
      SUBROUTINE REPACK(HSTOR,LTYPE)
*************************************************************
*     REPACKS THE CONNECTIVITY OR HYDRO LEVELS FOR A HEX    *
*************************************************************
*     INPUTS: HSTOR-- INTERNAL HEX NUMBER                   *
*             LTYPE-- LOC OR HYDRO LEVEL AT THE GIVEN SIDE  *
*************************************************************
      IMPLICIT INTEGER(H,P)
      INTEGER*4 SIDES
      DIMENSION HSTOR(2),HXSIDE(2)
      INCLUDE'UNPACK.CMN'
1 ****************************************************
1     INTEGER*4 HSIDE(6) ! CONNECTIVITY CODES IN    *
1                        ! NUMERICAL ORDER BY SIDE *
1     COMMON/UNPACK/HSIDE
1 ****************************************************
      HXSIDE(1)=HXADD(HSTOR(2),HXINV(HSTOR(1)))
      HXSIDE(2)=HXINV(HXSIDE(1))
      LU=7
      DO J=1,2
        CALL HEXREAD(HSTOR(J),SIDES,LU)
        CALL UNPACKER(SIDES)
        CALL STRIPPER(HXSIDE(J),ISIDE)
        HSIDE(ISIDE)=LTYPE
        NEWSIDE=0
        DO I=1,6
           NEWSIDE=NEWSIDE+HSIDE(I)*10**(I-1)
        ENDDO
        CALL HEXWRITE(HSTOR(J),NEWSIDE,LU)
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE RIJ2IJ(RI,RJ,I,J)
C     *ROUTINE(CONVERT REAL TO NEAREST INTEGER I,J COORDINATES - RIJ2IJ)
C     ******************************************************************
C     *
C     *   DESIGNER/PROGRAMMER:
C     *      DON KRECKER  17 SEPTEMBER 1980
C     *   PURPOSE:
C     *      RIJ2IJ TAKES A POINT SPECIFIED BY REAL-VALUED OBLIQUE
C     *      COORDINATES (RI,RJ) AND DETERMINES THE NEAREST POINT WITH
C     *      INTEGER-VALUED OBLIQUE COORDINATES (I,J).  ALL TIES ARE
C     *      RESOLVED IN FAVOR OF THE POINT WITH THE LARGER I AND/OR J
C     *      COORDINATE.
C     *      THE SYSTEM OF INTEGER I,J OBLIQUE COORDINATES CORRESPONDS
C     *      TO THE SET OF HEX CENTERS IN A HEXAGONAL GRID.  WHEN COM-
C     *      PUTATIONS ON SUCH A GRID LEAD TO POINTS OTHER THAN HEX
C     *      CENTERS, IT IS OFTEN NECESSARY TO DETERMINE WHICH HEX CON-
C     *      TAINS THE COMPUTED POINT.  THE HEX CONTAINING THE POINT IS
C     *      THE ONE WHOSE CENTER IS CLOSEST TO THE POINT.  THUS THE
C     *      PROBLEM REDUCES TO FINDING THE POINT WITH INTEGER I,J
C     *      COORDINATES WHICH IS CLOSEST TO A GIVEN POINT WITH REAL
C     *      I,J COORDINATES.  BECAUSE THE AXES IN THE OBLIQUE COORDI-
C     *      NATE SYSTEM ARE NOT ORTHOGONAL, IT IS NOT ALWAYS CORRECT
C     *      TO SIMPLY ROUND THE REAL I AND J COORDINATES TO THE NEAR-
C     *      EST INTEGERS.  RIJ2IJ IMPLEMENTS THE PROPER TRANSFORMATION.
C     *      GIVEN RI AND RJ, THE ALGORITHM FIRST FINDS THE GREATEST
C     *      INTEGERS LESS THAN OR EQUAL TO THEM, NAMELY IO AND JO.
C     *      THIS DETERMINES A RHOMBUS CONTAINING (RI,RJ) WITH VERTICES
C     *      (IO,JO), (IO+1,JO), (IO+1,JO+1), AND (IO,JO+1).  IN ORDER
C     *      TO DECIDE WHICH VERTEX IS CLOSEST TO (RI,RJ), THE RHOMBUS
C     *      IS DIVIDED INTO BANDS OF WIDTH 1/2 PERPENDICULAR TO THE
C     *      I, J, AND K AXES.  THERE ARE THREE I-BANDS NUMBERED 0, 1,
C     *      AND 2 IN THE DIRECTION OF THE POSITIVE I-AXIS.  LIKEWISE,
C     *      THERE ARE THREE J-BANDS.  THERE ARE ONLY 2 K-BANDS NUM-
C     *      BERED 0 AND 1 IN THE DIRECTION OF THE NEGATIVE K-AXIS.
C     *      ONCE THE I, J, AND K BAND NUMBERS FOR (RI,RJ) HAVE BEEN
C     *      COMPUTED, THE COORDINATES (I,J) OF THE PROPER VERTEX ARE
C     *      GIVEN BY:
C     *            I = IO + (IBAND+KBAND)/2
C     *            J = JO + (JBAND+KBAND)/2
C     *   CALLING SEQUENCE:
C     *      CALL RIJ2IJ(RI,RJ,I,J)
C     *   INPUT:
C     *      RI      - REAL-VALUED I COORDINATE
C     *      RJ      - REAL-VALUED J COORDINATE
C     *   OUTPUT:
C     *      I       - INTEGER-VALUED I COORDINATE
C     *      J       - INTEGER-VALUED J COORDINATE
C     *
C     ******************************************************************
      IMPLICIT INTEGER(H,P)
C       *COMPUTE GREATEST INTEGERS LE RI AND RJ
        IO = IFIX(RI)
        JO = IFIX(RJ)
        IF(RI.LT.FLOAT(IO)) IO = IO - 1
        IF(RJ.LT.FLOAT(JO)) JO = JO - 1
C       *FIND NONNEGATIVE FRACTIONAL PARTS OF RI AND RJ
        FI = RI - FLOAT(IO)
```

```
0058              FJ = RJ - FLOAT(J0)
0059      C      *COMPUTE I, J, AND K BAND NUMBERS
0060              IBAND = IFIX(FI + FI - FJ + 1.0)
0061              JBAND = IFIX(FJ + FJ - FI + 1.0)
0062              KBAND = IFIX(FI + FJ)
0063      C      *DETERMINE NEAREST INTEGER OBLIQUE COORDINATES
0064              I = I0 + ISHFT(IBAND+KBAND,-1)
0065              J = J0 + ISHFT(JBAND+KBAND,-1)
0066      C    *ENDROUTINE(RIJ2IJ)
0067           RETURN
0068           END
```

```
0001              SUBROUTINE SETGEO
0002      ***************************************************
0003      *       THIS ROUTINE SETS THE GEOGRAPHIC PARAMETERS   *
0004      *       NEEDED TO RUN THE LAT/LON-->UTM STUFF.        *
0005      ***************************************************
0006              INCLUDE 'CMERID.CMN'
0007    1 ***************************************************
0008    1         REAL*8 CMERID
0009    1         REAL P_RAD
0010    1         COMMON/CMERID/CMERID,P_RAD
0011    1 ***************************************************
0012              INCLUDE 'UTIL:ZDBPRO.CMN'
00100     0013  1 C
00200     0014  1 C       DUMMY COMMON ZDBPRO
00300     0015  1 C
00400     0016  1        INTEGER*4 ZRFDAY,ZIDCNT,ZYDOG,ZTSEC(4),ZTEX(5)
00500     0017  1        INTEGER*2 ZSPHID,ZYGOAT,ZTSN,ZRGN,ZRGL,ZMGRST(3,3),ZLLST(56)
00600     0018  1        LOGICAL*1 ZTSIC(3),ZTDWN(25)
00700     0019  1 C
00800     0020  1        COMMON /ZDBPRO/ ZRFDAY,ZIDCNT,ZYDOG,ZTSEC,ZTEX,
00900     0021  1       2                ZSPHID,ZYGOAT,ZTSN ,ZRGN ,ZTSIC,
01000     0022  1       3                ZTDWN ,ZRGL  ,ZMGRST,ZLLST
01100     0023  1 C
          0024             PARAMETER PI=3.141592654
          0025             P_RAD=PI/180.
          0026             ZSPHID=1 ! INTERNATIONAL SPHEROID (?)
          0027             ZRGN=32  ! UTM ZONE FOR MOST OF EUROPE
          0028             CALL ADSCCM(ZRGN,CMERID) ! SET CENTRAL MERIDIAN
          0029             CALL ADSSSP(ZSPHID)      ! SET SPHEROID IN ADCEAR.CMN
          0030             END
```

```
      SUBROUTINE STRIPPER(HSIDE,ISIDE)
C*****************************************
C*****  THIS ROUTINE ACCEPTS A HEX VECTOR IN
C*****  INTERNAL FORMAT,CONVERTS IT TO EXTERNAL
C*****  FORMAT AND THEN STRIPS OFF THE LEADING
C*****  7'S.
C*****************************************
C*****  INPUT:  HSIDE--THE HEX SIDE IN INTERNAL
C*****                 FORMAT
C*****  OUTPUT: ISIDE--THE HEX SIDE AS A DECIMAL
C*****                 DIGIT
C*****************************************
C
      IMPLICIT INTEGER(H,P)
C
      PARAMETER H7=77777770
C*****  CONVERT THE INTERNAL HEX # TO DECIMAL
      CALL HEXOUT(HSIDE,1,ISIDE)
      ISIDE=ISIDE-H7
      RETURN
      END
```

```
      SUBROUTINE TRACE (IX,IY,IENTER,ITOP,HEIGHT,MASK,
     $   XYDELTA,ILL,IUR,JLL,JUR,IJDELTA)
*****************************************************************
*       TRACES CONTOUR UNTIL WINDOW EDGE OR CLOSURE.            *
*       CONTOUR ENTERED RES. ELEMENT "IX,IY" FROM SIDE "IENTER" *
*****************************************************************
*       INPUTS:                                                 *
*             IX,IY-- INDICES OF CURRENT RESOLUTION ELEMENT     *
*             IENTER-- THE SIDE OF THE ELEMENT ENTERED (1,2,OR 3) *
*             ITOP -- A FLAG (0 UPON ENTRY) WHETHER THE Y INDEX HAS *
*                     REACHED THE "TOP" OF THE ARRAY            *
*            HEIGHT-- THE ALTITUDE OF THE TRACE                 *
*            MASK-- AN ARRAY OF FLAGS INIDCATING THE STATUS OF EACH *
*                   SIDE OF EACH ELEMENT (0-NOT CHECKED,        *
*                   1-INTERCEPT,10-NO INTERCEPT)                *
*            XYDELTA-- HORIZONTAL DISTANCE BETWEEN CONTOUR INTERVAL *
*                      CHECKS                                   *
*            ILL,IUR-- LOWER LEFT AND UPPER RIGHT LIMITS TO THE ROW *
*                      INDEX                                    *
*            JLL,JUR-- SAME FOR THE COLUMN INDEX                *
*            IJDELTA-- RATIO OF THE CONTOUR ELEMENTS TO THE DATA *
*                      ELEMENTS                                 *
*       OUTPUTS: NONE                                           *
*       ***  H.JONES ***                                        *
*****************************************************************
      IMPLICIT INTEGER*2 (I-N)
      INCLUDE 'MASK.DIM'
1 ***********************************************************
1     BYTE MASK(400,400,3)
1 ***********************************************************
C
   10 CONTINUE
      IF(IX .LT. ILL .OR. IX .GT. IUR) THEN
         GO TO 20
      ENDIF
      IF(IY .LT. JLL .OR. IY .GT. JUR) THEN
         GO TO 20
      ENDIF
C
      IF(IENTER .NE. 1) THEN
         IF(ITOP .EQ. 0) THEN
            IYIND = IY
         ELSE
            IYIND = IY+IJDELTA
         ENDIF
         IF(MASK(IX,IYIND,1) .EQ. 0) THEN
               CALL INTERS(IX,IYIND,IX+
     $         IJDELTA,IYIND,HEIGHT,FRAC,ICROSS)
            IF(ICROSS .EQ. 1) THEN
               X=FLOAT(IX/IJDELTA)*XYDELTA
               XX = X + XYDELTA * FRAC
               YY=FLOAT(IYIND/IJDELTA)*XYDELTA
C  D             CALL CMCLOS
C  D             PRINT*,XX,YY,'      TRACE'
C  D             CALL CMOPEN
               CALL DRAW (XX,YY)
               MASK(IX,IYIND,1) = 1
```

```
0058                    IY = IYIND - IJDELTA + ITOP*IJDELTA
0059                    IENTER = 1
0060                    ITOP = 1 - ITOP
0061                    GO TO 10
0062                 ELSE
0063                    MASK(IX,IYIND,1) = 10
0064                 ENDIF
0065              ENDIF
0066           ENDIF
0067     C
0068           IF(IENTER .NE. 2) THEN
0069              IF(ITOP .EQ. 0) THEN
0070                 IXIND = IX
0071              ELSE
0072                 IXIND = IX+IJDELTA
0073              ENDIF
0074              IF(MASK(IXIND,IY,2) .EQ. 0) THEN
0075             CALL INTERS(IXIND,IY,IXIND,IY+
0076          +  IJDELTA,HEIGHT,FRAC,ICROSS)
0077                 IF(ICROSS .EQ. 1) THEN
0078                    XX=FLOAT(IXIND/IJDELTA)*XYDELTA
0079                    Y=FLOAT(IY/IJDELTA)*XYDELTA
0080                    YY = Y + XYDELTA * FRAC
0081                    CALL DRAW (XX,YY)
0082                    MASK(IXIND,IY,2) = 1
0083                    IX = IXIND - IJDELTA + ITOP*IJDELTA
0084                    IENTER = 2
0085                    ITOP = 1 - ITOP
0086                    GO TO 10
0087                 ELSE
0088                    MASK(IXIND,IY,2) = 10
0089                 ENDIF
0090              ENDIF
0091           ENDIF
0092     C
0093           IF(IENTER .NE. 3 .AND. MASK(IX,IY,3) .EQ. 0) THEN
0094             CALL INTERS(IX,IY+IJDELTA,IX+
0095          $  IJDELTA,IY,HEIGHT,FRAC,ICROSS)
0096              IF(ICROSS .EQ. 1) THEN
0097                 X=FLOAT(IX/IJDELTA)*XYDELTA
0098                 Y=FLOAT(IY/IJDELTA+1)*XYDELTA
0099                 XX = X + XYDELTA * FRAC
0100                 YY = Y - XYDELTA * FRAC
0101                 CALL DRAW (XX,YY)
0102                 MASK(IX,IY,3) = 1
0103     C
0104                 IENTER = 3
0105                 ITOP = 1 - ITOP
0106                 GO TO 10
0107              ELSE
0108                 MASK(IX,IY,3) = 10
0109              ENDIF
0110           ENDIF
0111     C
0112        2  RETURN
0113           END
```

```
0001              FUNCTION TRANX(X)
0002      C
0003      C       TRANSLATES THE X VALUES STORED IN THE NODE AND
0004      C       SUBNODE RECORDS OF THE BDM DATA FILES BACK TO
0005      C       STANDARD UTM COORDINATES.THE STORED DATA HAS HAD
0006      C       500,000M SUBTRACTED AND BEEN DIVIDED BY 20,SO...
0007              INTEGER*2 X
0008              X1=X
0009 C    D       CALL CMCLOS
0010      C
0011              TRANX=20*X1+500000
0012 C    D       PRINT*,'X',X,TRANX
0013 C    D       CALL CMOPEN
0014              RETURN
0015              END
```

```
0001              FUNCTION TRANY(Y)
0002        C
0003        C     TRANSLATES THE Y VALUES STORED IN THE NODE AND
0004        C     SUBNODE RECORDS OF THE BDM DATA FILES BACK TO
0005        C     STANDARD UTM COORDINATES.THE STORED DATA HAS HAD
0006        C     5,600,000M SUBTRACTED AND BEEN DIVIDED BY 20,SO...
0007        C
0008              INTEGER*2 Y
0009              Y1=Y
0010              TRANY=20*Y1+5600000
0011 C   D        CALL CMCLOS
0012 C   D        PRINT*,'Y',Y,TRANY
0013 C   D        CALL CMOPEN
0014              RETURN
0015              END
```

```
      SUBROUTINE UNGEN
*************************************************
*     THIS ROUTINE GENERATES THE FILE NAMES AND  *
*     RELATIVE INDICES FOR PLACING THE FILE      *
*     DATA INTO THE ARRAY IBUF.                  *
*************************************************
      INCLUDE 'CORNER.CMN'
1 ***********************************************************
1 *     SWX,SWY ARE THE SOUTHWEST UTM COORDINATES OF THE  *
1 *     AREA IN THE ARRAY IBUF.                           *
1     INTEGER*4 SWX,SWY
1     COMMON/CORNER/SWX,SWY
1 ***********************************************************
      CHARACTER*7 MGR
      LOGICAL*1 ERR
      DO J=0,3
         IEAST=SWX+J*10000
         DO I=0,3
            NORTH=SWY+I*10000
            CALL UTM2MGR(IEAST,NORTH,MGR,ERR)
            CALL MAPIN(I,J,MGR,ERR)
         ENDDO
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE UNPACKER(SIDES)
C*****************************************
C*****  THIS ROUTINE IS DESIGNED TO UNPACK
C*****  THE CONNECTIVITY CODES FROM THE DATA
C*****  IN THE ISAM FILE CREATED BY ROADHEXER.
C*****************************************
C*****  INPUT:  SIDES--THE PACKED CONNECTIVITY
C*****                 CODES
C*****  OUTPUT: HSIDE(6)--THE UNPACKED CONNEC-
C*****                    TIVITY CODES
C*****************************************
C
      IMPLICIT INTEGER (H,P)
      INTEGER*4 SIDES
      INCLUDE 'UNPACK.CMN'
1 ****************************************************
1     INTEGER*4 HSIDE(6) ! CONNECTIVITY CODES IN    *
1                        ! NUMERICAL ORDER BY SIDE *
1     COMMON/UNPACK/HSIDE
1 ****************************************************
C
C     IN ORDER TO AVOID CHANGING 'SIDES'
      ISIDES=SIDES
      DO I=6,1,-1
         IEXP=10**(I-1)
         HSIDE(I)=INT(ISIDES/IEXP)
         ISIDES=ISIDES-HSIDE(I)*IEXP
      ENDDO
      RETURN
      END
```

```
      SUBROUTINE XY2IJM(X,Y,I,J)
C    *ROUTINE(CONVERT X,Y TO MIN LEVEL I,J COORDINATES - XY2IJM)
C    ***************************************************************
C    *
C    *   DESIGNER/PROGRAMMER:
C    *      DON KRECKER  20 SEPTEMBER 1980
C    *   PURPOSE:
C    *      XY2IJM CONVERTS A PAIR OF X,Y CARTESIAN COORDINATES EX-
C    *      PRESSED IN METERS TO I,J OBLIQUE COORDINATES AT THE MINI-
C    *      MUM HEX LEVEL.  THE RESULT IS THE PAIR OF INTEGER-VALUED
C    *      I,J COORDINATES CORRESPONDING TO THE CENTER OF THE MINIMUM
C    *      LEVEL HEX WHICH CONTAINS THE GIVEN POINT.  THIS ROUTINE IS
C    *      THE INVERSE OF THE SUBROUTINE IJM2XY.
C    *      THE CONVERSION IS CARRIED OUT BY FIRST APPLYING A LINEAR
C    *      TRANSFORMATION (IN THE FORM OF A MATRIX MULTIPLICATION) TO
C    *      THE X,Y VECTOR TO OBTAIN A PAIR OF REAL-VALUED I,J COORDI-
C    *      NATES.  THE RESULT IS THEN ROUNDED TO THE NEAREST INTEGER-
C    *      VALUED I,J COORDINATES BY THE ROUTINE RIJ2IJ.
C    *   CALLING SEQUENCE:
C    *      CALL XY2IJM(X,Y,I,J)
C    *   INPUT:
C    *      X,Y     - REAL-VALUED CARTESIAN COORDINATES EXPRESSED IN
C    *                METERS WHICH ARE TO BE CONVERTED TO OBLIQUE COOR-
C    *                DINATES AT THE MINIMUM HEX LEVEL
C    *      RIOFX   - REAL I COORDINATE OF THE VECTOR (X,Y) = (1,0).
C    *                (IN COMMON/HEX/)
C    *      RJOFX   - REAL J COORDINATE OF THE VECTOR (X,Y) = (1,0).
C    *                (IN COMMON/HEX/)
C    *      RIOFY   - REAL I COORDINATE OF THE VECTOR (X,Y) = (0,1).
C    *                (IN COMMON/HEX/)
C    *      RJOFY   - REAL J COORDINATE OF THE VECTOR (X,Y) = (0,1).
C    *                (IN COMMON/HEX/)
C    *   OUTPUT:
C    *      I,J     - INTEGER-VALUED OBLIQUE COORDINATES CORRESPONDING
C    *                TO THE CENTER OF THE MINIMUM LEVEL HEX CONTAINING
C    *                THE GIVEN X,Y POINT
C    *
C    ***************************************************************
      IMPLICIT INTEGER(H,P)
      COMMON/HEX/IHXOUT,NHLEV,MINLEV,SLTO,CLTO,DLNO,DIAM(10),DIAMT
     $R,
     +                XOFI,YOFI,XOFJ,YOFJ,RIOFX,RJOFX,RIOFY,RJOFY,
     +                ICON(70),JCON(70),IMAX(7),JMAX(7)
      DIMENSION IVAL(7),JVAL(7)
      EQUIVALENCE(IVAL(1),ICON(1)),(JVAL(1),JCON(1))
C       *TRANSFORM X,Y COORDINATES TO EQUIVALENT REAL-VALUED I,J PAIR
        RI = RIOFX * X + RIOFY * Y
        RJ = RJOFX * X + RJOFY * Y
C       *INCLUDE(CONVERT REAL I,J COORDINATES TO INTEGER - RIJ2IJ)
        CALL RIJ2IJ(RI,RJ,I,J)
C    *ENDROUTINE(XY2IJM)
      RETURN
      END
```

```
      SUBROUTINE XYL2HA(X,Y,LEV,HADR)
C    *ROUTINE(CONVERT X,Y COORDINATES AND LEVEL TO HEX ADDRESS-XYL2HA)
C    ****************************************************************
C    *
C    *   DESIGNER/PROGRAMMER:
C    *      DON KRECKER  21 SEPTEMBER 1980
C    *   PURPOSE:
C    *      XYL2HA TAKES A POINT EXPRESSED IN X,Y CARTESIAN COORDINATES
C    *      IN METERS AND COMPUTES THE ADDRESS OF THE HEX AT THE SPECI-
C    *      FIED LEVEL WHICH CONTAINS THE POINT.  IF THE SPECIFIED HEX
C    *      LEVEL IS GREATER THAN THE MINIMUM HEX LEVEL, THE COMPUTED
C    *      HEX MAY DIFFER FROM THE HEX (AT THIS LEVEL) WHOSE CENTER IS
C    *      CLOSEST TO THE GIVEN POINT.  THE REASON IS THAT HEXES AT
C    *      HIGHER LEVELS OF AGGREGATION ARE NOT TRUE REGULAR HEXAGONS
C    *      BUT ONLY APPROXIMATE REGULAR HEXAGONS IN SHAPE.  XYL2HA IS
C    *      THE INVERSE OF THE SUBROUTINE HA2XYL, WHICH CONVERTS A HEX
C    *      ADDRESS TO THE X,Y COORDINATES OF THE CENTER OF THE HEX.
C    *      XYL2HA FIRST CALLS THE ROUTINE XY2IJM TO CONVERT THE X,Y
C    *      COORDINATES TO I,J OBLIQUE COORDINATES AT THE MINIMUM HEX
C    *      LEVEL.  THEN THE FUNCTION IJM2HA IS USED TO CONVERT THE I,J
C    *      COORDINATES TO A HEX ADDRESS AT THE REQUESTED LEVEL.  ERROR
C    *      CHECKING IS DONE BY THIS SUBORDINATE FUNCTION.
C    *   CALLING SEQUENCE:
C    *      CALL XYL2HA(X,Y,LEV,HADR)
C    *   INPUT:
C    *      X,Y     - REAL-VALUED CARTESIAN COORDINATES EXPRESSED IN
C    *                METERS OF A POINT WHOSE CONTAINING HEX AT A SPE-
C    *                CIFIED LEVEL IS TO BE COMPUTED
C    *      LEV     - LEVEL OF AGGREGATION OF THE HEX ADDRESS TO BE
C    *                COMPUTED
C    *   OUTPUT:
C    *      HADR    - ADDRESS OF THE HEX AT THE REQUESTED LEVEL WHICH
C    *                CONTAINS THE GIVEN X,Y POINT
C    *
C    ****************************************************************
      IMPLICIT INTEGER(H,P)
C       *INCLUDE(CONVERT X,Y TO MIN LEVEL I,J COORDINATES - XY2IJM)
        CALL XY2IJM(X,Y,I,J)
C       *INCLUDE(CONVERT MIN LEVEL I,J AND LEVEL TO HEX ADDRESS-IJM2HA)
        HADR = IJM2HA(I,J,LEV)
C    *ENDROUTINE(XYL2HA)
      RETURN
      END
```

APPENDIX C

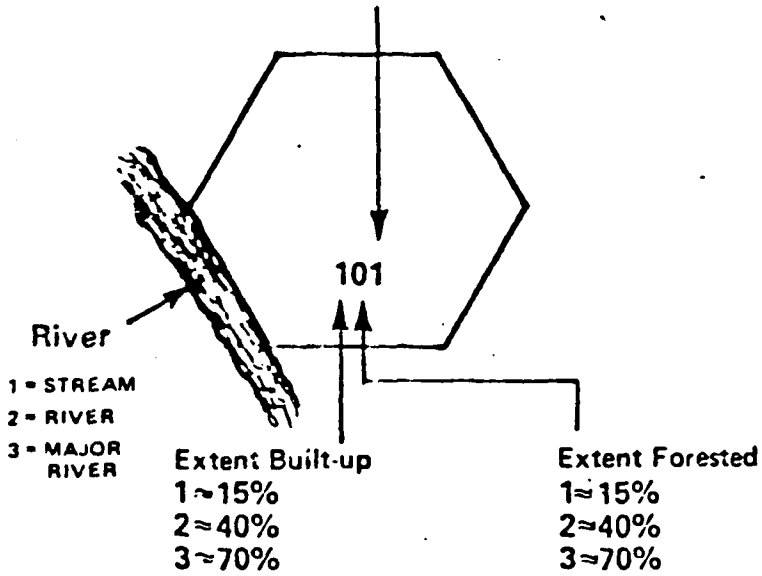
HEX DATA IN ICOR

Appendix C - Hexagonal Terrain in ICOR.

The following information was excerpted from the ICOR User's Manual published by BDM.

HEX Terrain Codes

Terrain Roughness
1 = terrain slope avg>.03 overall or≈15% hills or rugged terrain
2 = terrain slope avg>.06 overall or≈40% hills or very rugged terrain
3 = terrain slope avg>.1 or most of hex impassable to vehicles



Roads:
Roads do not always correspond one to one with actual highways, but rather indicate the extent to which two hexes are connected.

Autobahn: 3
Primary: 2
Secondary: 1

Figure C-1

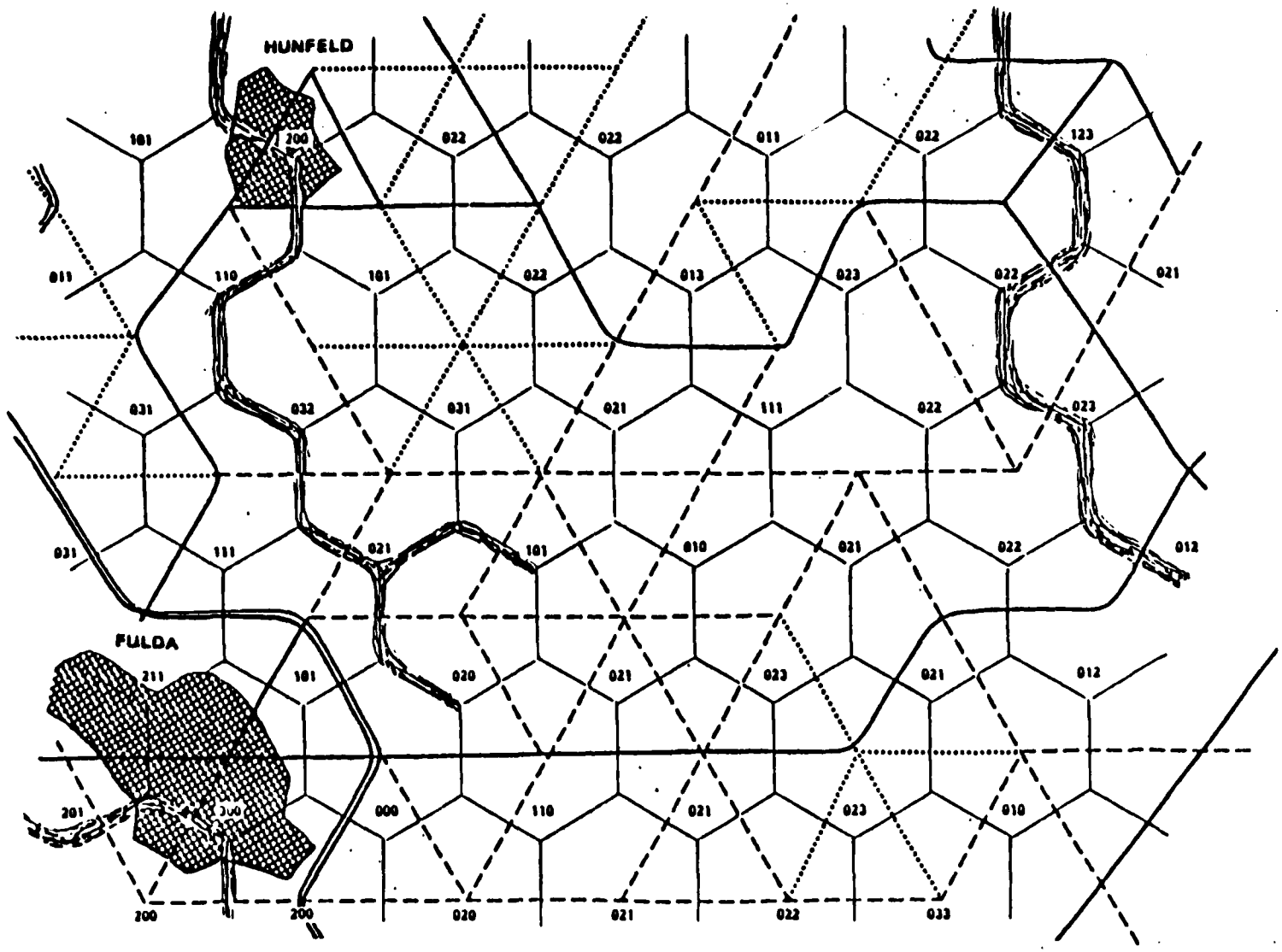


Figure C-2

Sample ICOR Terrain

THE HEXAGONAL COORDINATE SYSTEM

In the ICOR simulation the plane, or surface of the earth on which units move and fight, is broken up into discrete points as a means of organizing the data base and the operation of the model. Thus, it is possible to refer to all units and terrain as being at a particular location, meaning in the neighborhood of a given discrete point. This allows the locations of units and terrain features to be represented in the model much more compactly than would be possible if they were represented, for example, as a floating point coordinate pair on a cartesian plane along with given shapes. More important, the discrete points provide a means of reference as in the expression "all units at location X". Thus, each point can be represented in the computer as a block in a data structure, which contains terrain information about the neighborhood at that point, and a list of units in that neighborhood.

A hexagonal grid has been chosen as a means of defining the points, or neighborhoods, so represented. This has several advantages. The most important is that in a hexagonal grid, any neighborhood, or hexagon, which is adjacent to another also shares a side of finite length. This is not true with a square grid, where neighborhoods can be adjacent at a corner. This eliminates the tactical problem of a unit moving diagonally between two adjacent enemy units, without entering the neighborhood of either. If this problem is eliminated in a square grid by disallowing diagonal movement, then it restricts movement direction choice to only four directions, requiring units to move up to 45° off of their desired direction, with a loss of 29% of their effective speed. In a hexagonal grid the corresponding maximum loss is 13%. A hexagonal grid also eliminates complications from the two types of adjacencies when evaluating a situation, choosing movement direction, and calculating speed and arrival time at the next location. Another benefit is that a locus of points at a given count of hexes away from a center hex more closely approximates a circle than a similar locus on a square grid. This allows distance considerations, such

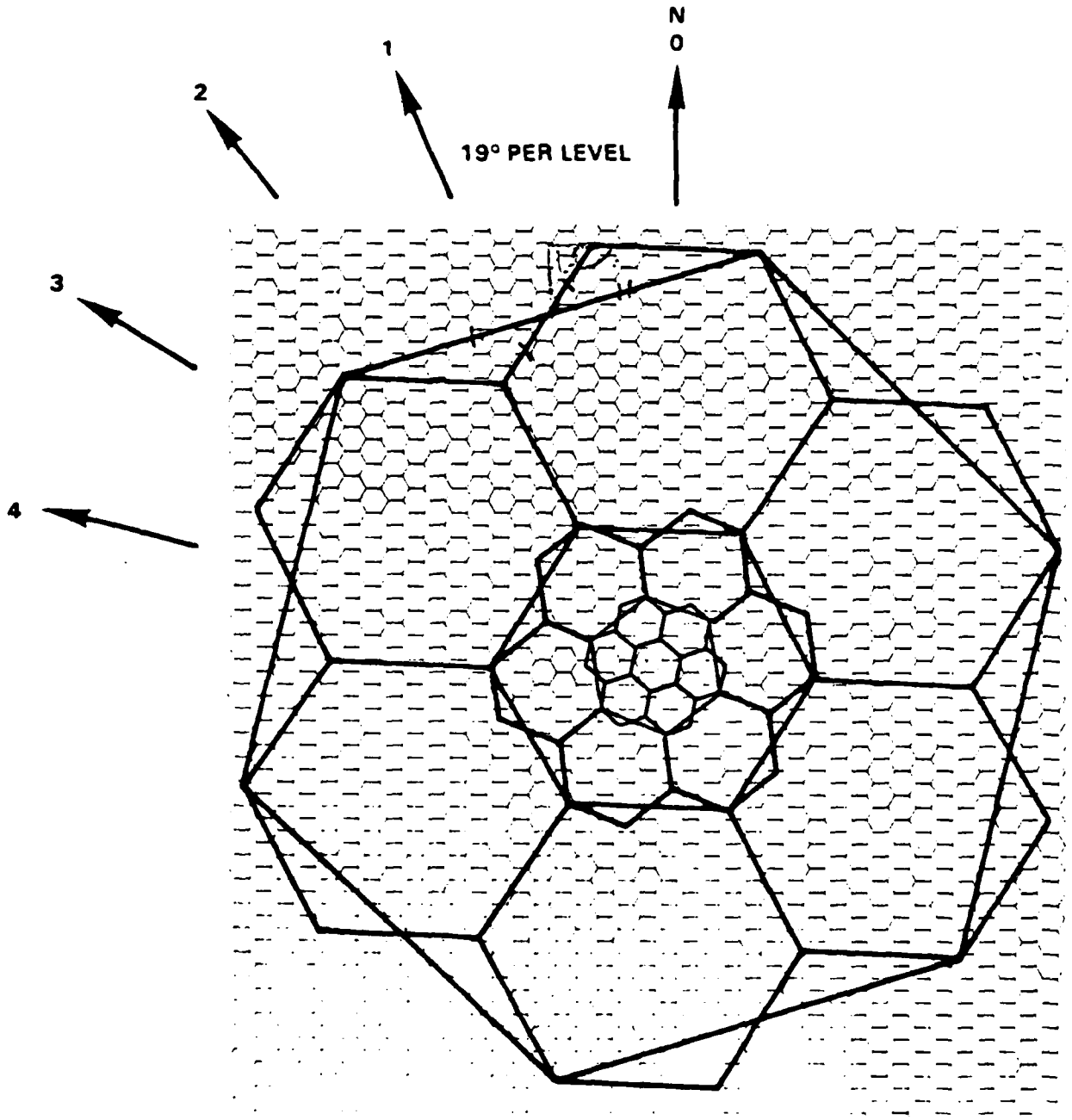
as the range of artillery, to be expressed as a given number of hexes. With a square grid such a procedure would introduce unacceptable errors.

One ultimate goal in a modeling system, of which ICOR is but one member, is the provision for scale change of the system to different levels of detail. It is possible with a hexagonal grid to divide each neighborhood into seven smaller neighborhoods which, when this is done for all hexes, creates a new hexagonal grid of smaller hexes. Conversely, groups of seven hexes can be grouped together to form larger hexes. Figure B-1 illustrates. For each level of aggregation, the size (diameter) of the next larger hex is $\sqrt{7}$ times that of the smaller hexes. The axis of straight rows, or the "grain" of the hex field, rotates approximately 19° counterclockwise. In the BDM hexagonal system, levels of hexes are defined as shown in Figure B-2. The size hex used for the units in ICOR is level 4, or 3.57 Km diameter (and center to center).

A numbering scheme for hexes must define the level of the hex, and its position in the plane of hexes at that level. In the BDM hex numbering system, the level of the hex is equal to 12 minus the number of digits in the hex address. Thus, a single digit hex address is a level 11 hex of 3,241 Km diameter. A two digit hex adds resolution of an additional level, to 1225 km. In ICOR, the 8 digit hex numbers, or addresses, give 8 levels of resolution, which corresponds to level 4, or 3.57 km hexes.

As a hex address is read from left to right one reads from most significant to least significant digit. At each digit, one can consider a selection of a smaller hex within the larger, or higher level, hex given by the preceeding digits. Figure B-3 illustrates this disaggregation.

At each level, a single digit represents the seven possible smaller hexes, and corresponding directions. These directions are shown in Figure B-4. If the digit is considered to be an octal number of three bits, each bit equal to one indicates a one hex diameter vector in the i, j, or k directions. Thus the vectors in each of the three basic directions are 001 or 1 for i, 010 or 2 for j, and 100 or 4 for k. Other directions are represented by combinations. The combination 111 or 7 is used for the null vector rather than 0.



ARROWS SHOW "DIRECTION 1" AT LEVELS 0 TO 4

Figure C-3

Hexes of Levels 0 Through 4 (Detailed and Idealized Boundaries)

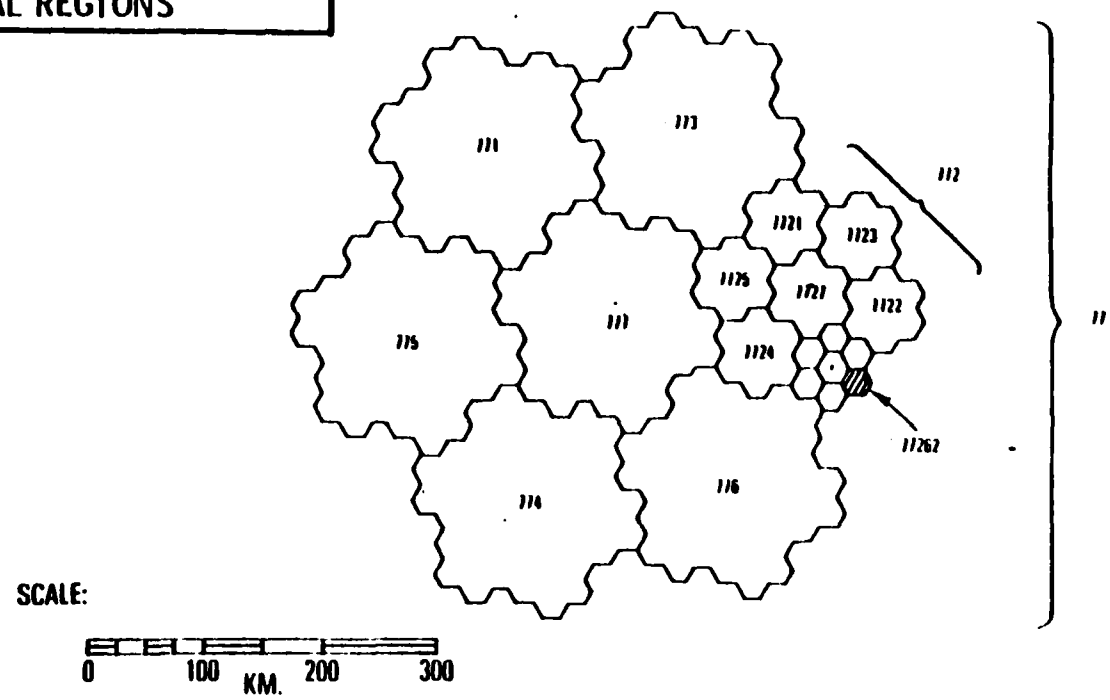
LEVEL	HEX DIAMETER	HEX AREA
0	72.9 M	4601 M^2
1	192.8 M	32205 M^2
2	510.2 M	225434 M^2
3	1.35 KM	1578035 M^2
4	3.57 KM	11.0 KM^2
5	9.45 KM	77.3 KM^2
6	25.00 KM	541.3 KM^2
7	66.14 KM	3788.9 KM^2
8	175.00 KM	26522 KM^2
9	463.01 KM	185654 KM^2
10	1225.00 KM	1299579 KM^2
11	3241.05 KM	9097056 KM^2
12	8575.00 KM	63679389 KM^2

Figure C-4

Hex Dimensions

HEX POSITION LOCATION SYSTEM

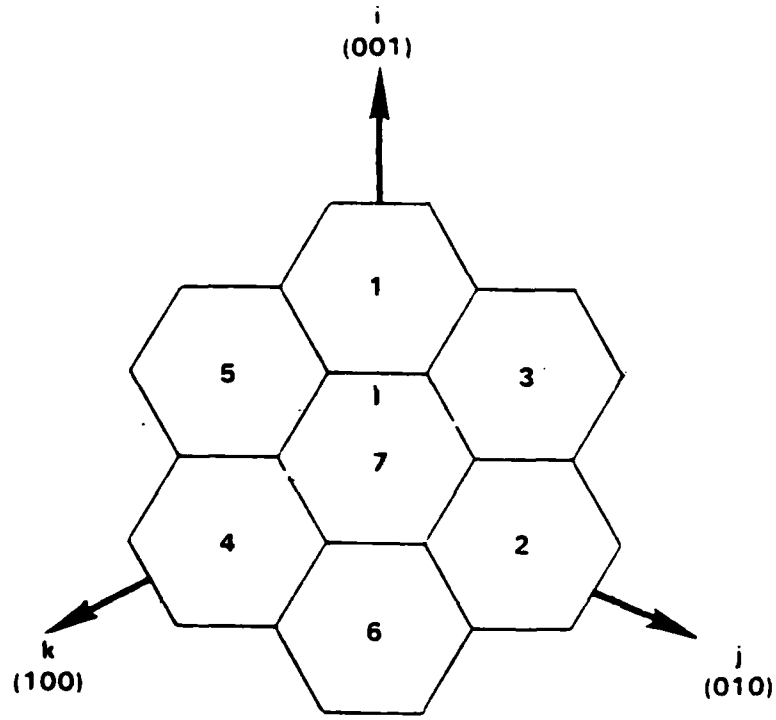
INTERNAL "COORDINATE SYSTEM" BASED ON NESTED HEXAGONAL REGIONS



- REALISTIC REPRESENTATION OF MANEUVER
- WIDER POSSIBILITIES FOR UNIT INTERACTIONS
- SIGNIFICANT COMPUTATIONAL EFFICIENCY

Figure C-5

Hex Position Location System



Hex Numbers and Directions at Level 4

Figure C-6

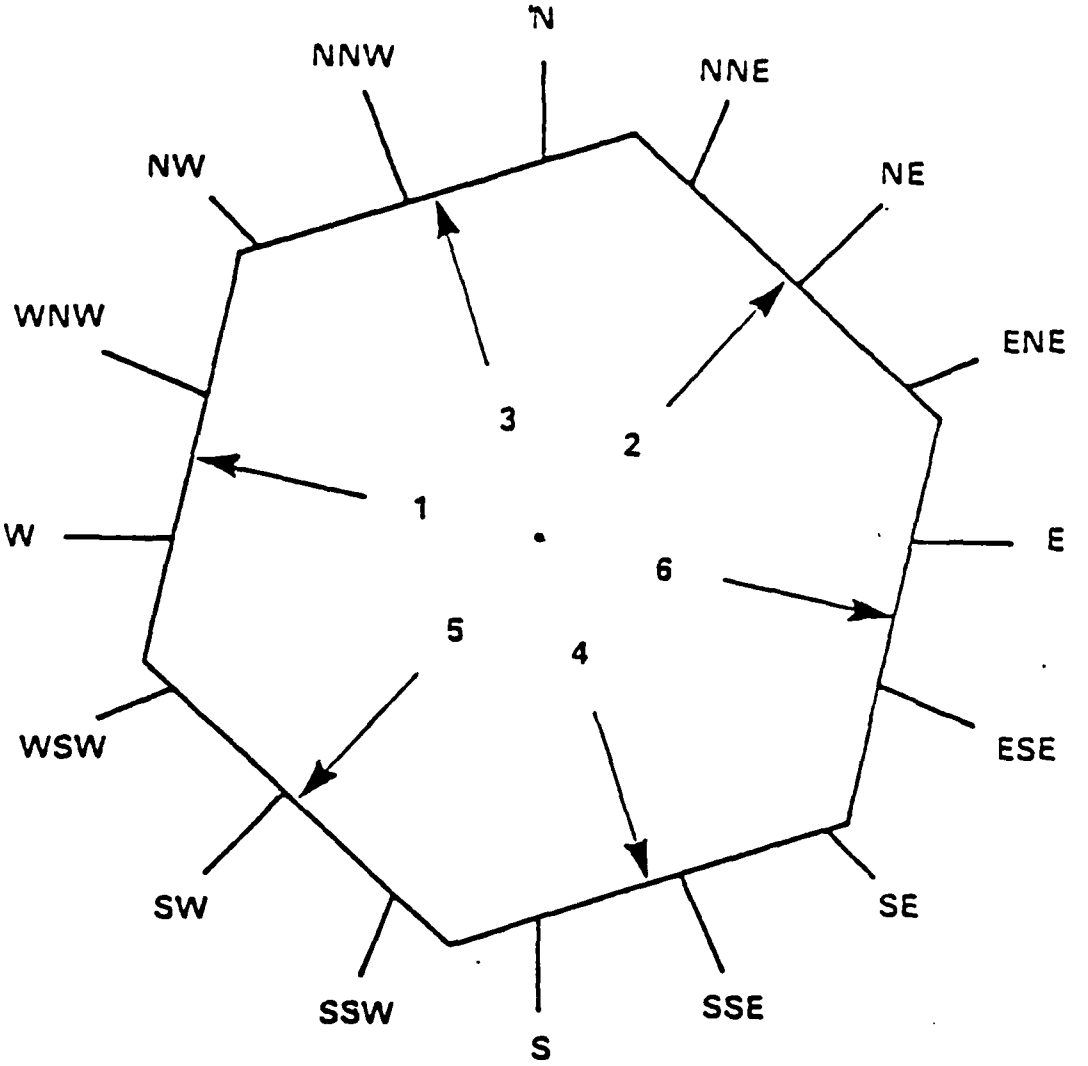
As a practical matter, a hex grid having hex addresses on it for cluster-center hexes but not others is usually used in the play of the ICOR simulation. There are two cases:

(1) In computer generated hex maps, the number given in the center hex is that of the higher level hex at the next level. To get the actual hex number, append 7 to the hex address of the center hex (in which the number is given). For each adjacent hex append the number for the given hex direction of that hex relative to the center hex (as shown in Figure B-5 for level 4).

(2) In some other manually-generated maps, the hex address of the center hex at the level of the hexes shown is given. This results in all hex addresses for center hexes ending in the digit 7. If this is the case, the hex address of each adjacent hex may be found by deleting the last 7 and appending the hex direction from the center hex to that hex.

The hex coordinate system used in the ICOR model is centered near the town of Fulda. The hex 7, and all hexes 7....7, are centered at 50°30'N, 9°30'E or NA3594 in UTM. If one starts with a blank (no addresses) hex sheet, the origin would then be labeled with the number 7....7 with the number of 7's indicating the level of hexes. Figure B-6 illustrates how all other hexes on the sheet can then be numbered. Summarizing, to plot a center hex at the given level, count hexes as follows:

Level	Hexes in Given Hex Direction	and Hexes to Left
Same	1	0
1 higher	2	1
2 higher	3	5
3 higher	1	18



Hex Directions at Level 4

Figure C-7

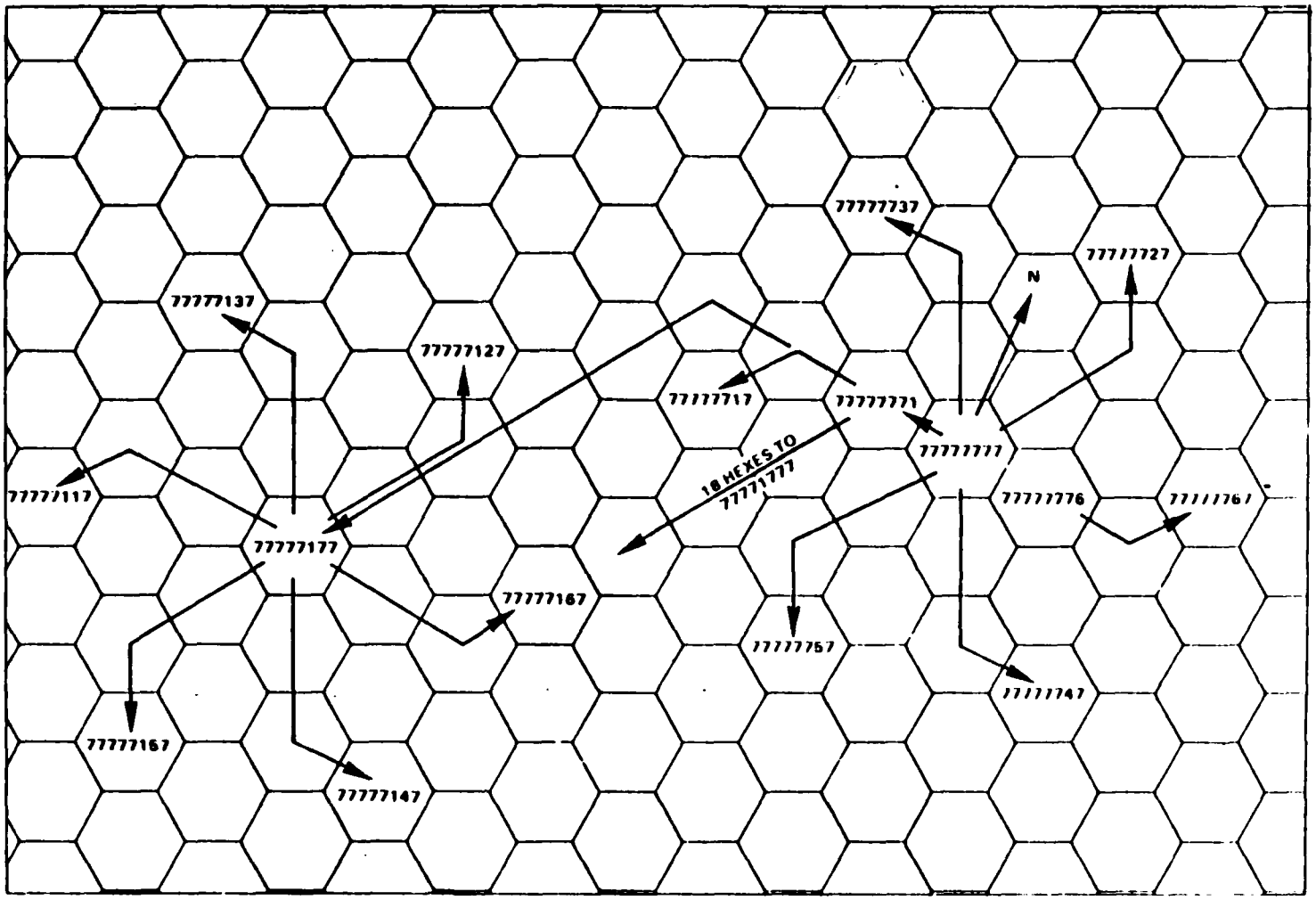


Figure C-8

Plotting Hex Numbers

APPENDIX D

DISTRIBUTION

APPENDIX D

DISTRIBUTION

Organization	Number of Copies
Commander Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
HQDA Army Study & Documentation and Information Retrieval System Washington, DC 20310	1
Commander USATRADOC ATTN: TRADOC Library Fort Monroe, VA 23651	1
Commander USA War College ATTN: Library BB-424 Carlisle Barracks, PA 17013	1
Commandant USACGSC ATTN: ATZLSW-TA CGSC Library Fort Leavenworth, KS 66027	2
Commander USATRASANA ATTN: Technical Library ATAA-SL White Sands Missile Range, NM 88002	1
Commander USATRASANA ATTN: ATOR-TFC (Mr. Hue McCoy) White Sands Missile Range, NM 88002	1
Commander USACACDA Fort Leavenworth, KS 66027	1
Commander CAORA ATOR-CA Fort Leavenworth, KS 66027	5

FILM