File

Institute of
Oceanographic Sciences
Deacon Laboratory

INTERNAL DOCUMENT No. 300

DBDB5 data set of global gridded bathymetry

N P Plummer

1991

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES
DEACON LABORATORY

INTERNAL DOCUMENT No. 300

DBDB5 data set of global gridded bathymetry

N P Plummer

1991

Wormley
Godalming
Surrey GU8 5UB
Tel 0428 684141
Telex 858833 OCEANS G
Telefax 0428 683066

CONTENTS

INTRODUCTION

The DBDB5 data set contains digital bathymetric data for the world oceans interpolated on to a 5min by 5min grid. The data set was compiled in the United States by the Naval Ocean Research and Development Activity (NORDA) and the Naval Oceanographic Office (USNOO) with support from the Ofice of Naval Research.

Each depth value is expressed in uncorrected whole metres and was derived from echo soundings assuming a sound speed of 1500m/s . Grid points on land are indicated by a depth value of -10.

The data set is contained on two magnetic tapes; one for the Southern Hemisphere and one for the Northern Hemisphere respectively. The tapes are registered as F339 (Southern Hemisphere) and F340 (Northern Hemisphere) with each one divided into 16 individual files of 45° by 45°. Each separate file is then broken down into 81 blocks each of 5° by 5°. Each block contains 3721 data points. Figures 1,2 and 3 show a diagramatic representation of this. Each 5° square is preceded by a header which denotes the longitude and latitude and a flag indicates whether the block is entirely land or not. If the flag is denoted by 1 then the 5° square is entirely land ,otherwise it contains one or more sea points.

AIM

The principal aim of the investigation was to obtain median depth values for an area of 1/4° square and thus produce a representation of the ocean topography at this resolution. As a first exercise the data set was broken down to produce median data values for an area of 1° square. This involved taking successive blocks of 169 elements from each block of the 5° squares and deriving the median value of each set of 169 elements. This process was repeated until the entire globe had been covered. In this particular case the true mathematical median could be obtained , i.e. the 85^{th} element of the 169 sorted integers. It was then decided to produce the median data values for a finer resolution of 1/4° square. The programs written to produce the 1° resolution were slightly modified to do this. This involved taking successive blocks of 16 data points from the data set and repeating until a 5° square area had been covered. This process was then repeated until the entire globe had been covered. Refer to fig. 3 for organisation of data points within a 5° square.

The true mathematical median value could not be used during this investigation; i.e. the mean of the 8^{th} and 9^{th} elements of the sorted 16 values. As previously mentioned the depth values are given in uncorrected whole metres. It was therefore considered appropriate to give the

mathematical median as an integer value. In this investigation this was achieved by selecting the 8^{th} element of the 16 sorted integers as an estimate to the true median. This means that the median data values are an 'under estimate' of the true depth but only by a relatively small error margin.

PROBLEMS

The main problem encountered was concerned with accessibility to the data as it is stored on several tapes and also in a character format foreign to the IBM. Another problem was unveiled as a shortage of disk-space meant that only one tape could be stored at any one instant. A program was developed to read in one tape at a time, translate the ASCII character format of the tapes and store the data on to a temporary disk. A series of further Fortran programs was then written to;

(i) Obtain median data depth values and store in new files.

(ii) Convert the new files into ASCII character encoded format so that they may be used in existing graphics programs.

(iii) Combine the data from the two hemispheres and output to a final file again in a form suitable for the graphics programs .

All programs were written on the IBM mainframe computer and the final output transferred to the SUN workstations for viewing .

The file containing the information about the DBDB5 data set explains how the 61 grid points on each of the four boundaries of the 5° square are repeated on the boundaries of adjacent 5° squares. This was thought to be a problem at first as data would be read in twice during each calculation of the median but it was later decided that this would have a negilible effect on the final median depth results.

PROGRAMS

Program 1: TAPFILE FORTRAN

This program transfers the data from the tapes on to a temporary disk on the IBM mainframe. The physical tape characteristics are; 9 track , 1600 bpi , ASCII , unlabelled , record length of 80 bytes and block size of 3120 bytes.

The program is designed to read in all 16 files of each tape and store on a disk. As already mentioned the tape is in ASCII character form. When transfering data to an IBM a change of format is necessary. The character set for the IBM differs from the standard ASCII format and is called

EBCDIC. The data is transferred into a readable form through calling the routine call ASCEBC. (Most other computer systems do not require this conversion process as the tape can be read directly.) The output file to the disk is in character A80 format as this is the format specified in the DBDB5 file.

The variables (st=st+80 and ed=st+80 ; refer to appendix 1 for program listing) ensure that all of the 81 blocks of each file are read into a larger file which contains the information from the 16 files on each tape.

Program 2 : 025DEG FORTRAN

This program was developed primarily to extract the median depth values from the data set. The program accesses the file created by the previous program. The initial idea was to read in a maximum of 4 files at a time and thus increase the speed of processing the data but as previously mentioned the disk space was not readily available.

Once the file required has been read in from the original large file , the information is sent to a new file. This is a check to ensure that the correct file has been obtained. At this point the data is in character A80 format. This part of the program may be ommitted in future use as it was only a check during the development of the program. After this, a new file is opened and the data is rewritten with a record length of 305 (ie 61I5) format. This is so that the data may be represented as shown in fig. 3; ie as a two dimensional array of size (61,61) .This is done so that the required data points can be calculated more easily.

The 16 values required are located within the (61,61) grid and they are sorted by calling the call RSORT command. The eighth element of the set is extracted and passed to a new file for future reference. Refer to appendix 2 for program listing.

Program 3: 025LAND FORTRAN

The output file from the previous program contains 400 median depth values for each 5° square. These values are then stored in an array of size (20,20). 025land Fortran is concerned with putting the files into the correct order for output , (ie starting with file 1 in the bottom left-hand corner of a large array and terminating with file 16 in the top right-hand corner of the array (refer to fig . 1 for representation of how the files are stored on each tape) and converting the median depth values into ASCII encoded character format through several subroutines. The ASCII encoded character format is the same as that required by the graphics programs M-PLOT and G-PLOT used by the FRAM group to display the model data (see Hateley, 1991). Subroutines ASCIIN and ASCOUT are used to read in or write out data in this format. The array of size (1440,360) contains the total median values at a 1/4° resolution for each individual hemisphere.

Sections of the program are involved with defining the depth values of -10 as land and assigning everything else as depths of the ocean topography. Refer to appendix 3 for program listing.

Program 4: 025COLOUR FORTRAN

This program was written so that the information from the two hemispheres could be combined on to a single file so that it could be implemented into the graphics programs. This requirement was necessary as when the two individual files where joined together directly, a narrow line was visible at the equator due to scaling differences between the two hemispheres. This program removes that line to produce an uninterrupted image for viewing.

The array of size (1440,720) contains the information from both the Northern and Southern Hemispheres at a resolution of 1/4° by 1/4° .The ASCII characters which make up the final output file are then passed to the graphics programs G-PLOT and M-PLOT where the median depth values may be viewed graphically. Refer to appendix 4 for a program listing.of 025colour Fortran

RESULTS

Refer to figures 4-11.

CONCLUSION

In conclusion it may be observed that the 1/4° resolution obtained from the DBDB5 data does provide a good representation of the world ocean topography. Although this is a very fine resolution, the programs developed to do this may be further modified to produce an even finer resolution of 5min by 5min; ie the original bathymetric gridded data readings produced by the DBDB5 data set. This is very simple to do in principle although a major problem would be the immense size of the output files and also the time needed to produce them.

The new topography may be employed in future models which simulate ocean circulation. One factor which will be important in future models is the simulation of the Agulhas Current off South Africa. One of the shortcomings of FRAM (Fine Resolution Antarctic Model) is its poor representation of some of the features of this current. Present conclusions suggest that this occurs due to smoothing of the topography which occured when the model was set-up. At present eddy formation and retroflection of the current occur too far up-stream, ie at a longitude east of where it occurs in reality. It would be interesting to see if the new topography is employed in future models and to see whether the Agulhas Current can be simulated more accurately using this topography.

ACKNOWLEDGEMENTS

Valuable assistance in developing the programs was provided by the FRAM Core Team including Andrew Coward ,Tim Hateley and Simon Thompson. Support was also received with much gratitude from Des Bulpett and Gwyneth Jones.

DBDB5 DATA TAPE STRUCTURE

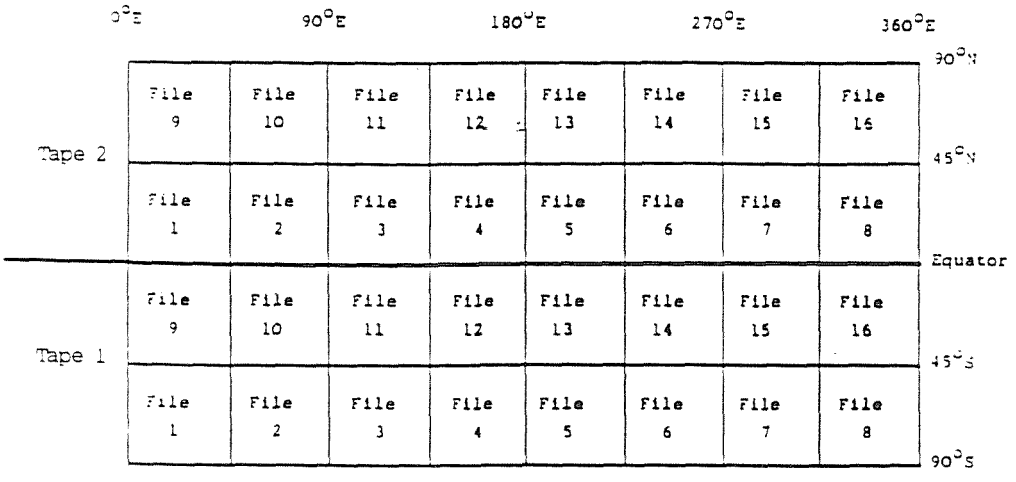

Figure 1. File sequence on tape.

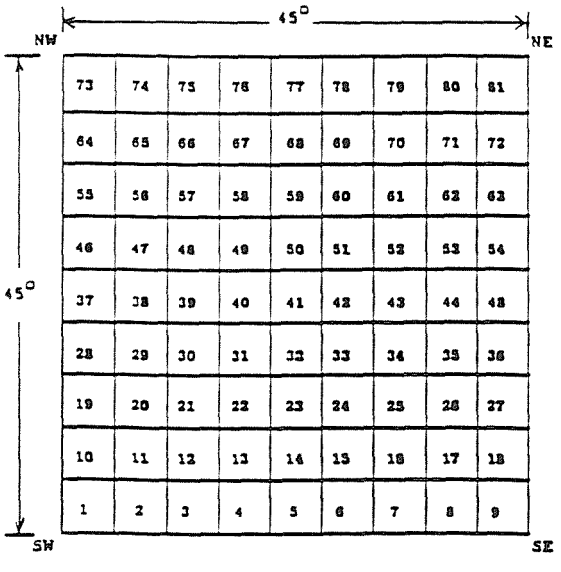

Figure 2. 5° square sequence within file.

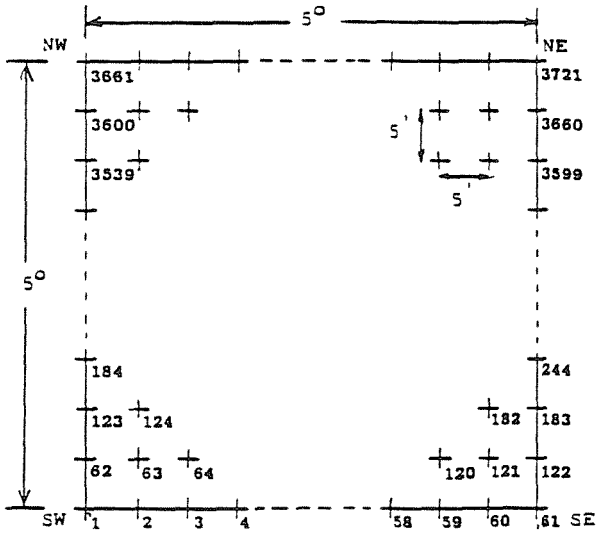

Figure 3. Grid point sequence within 5° square.

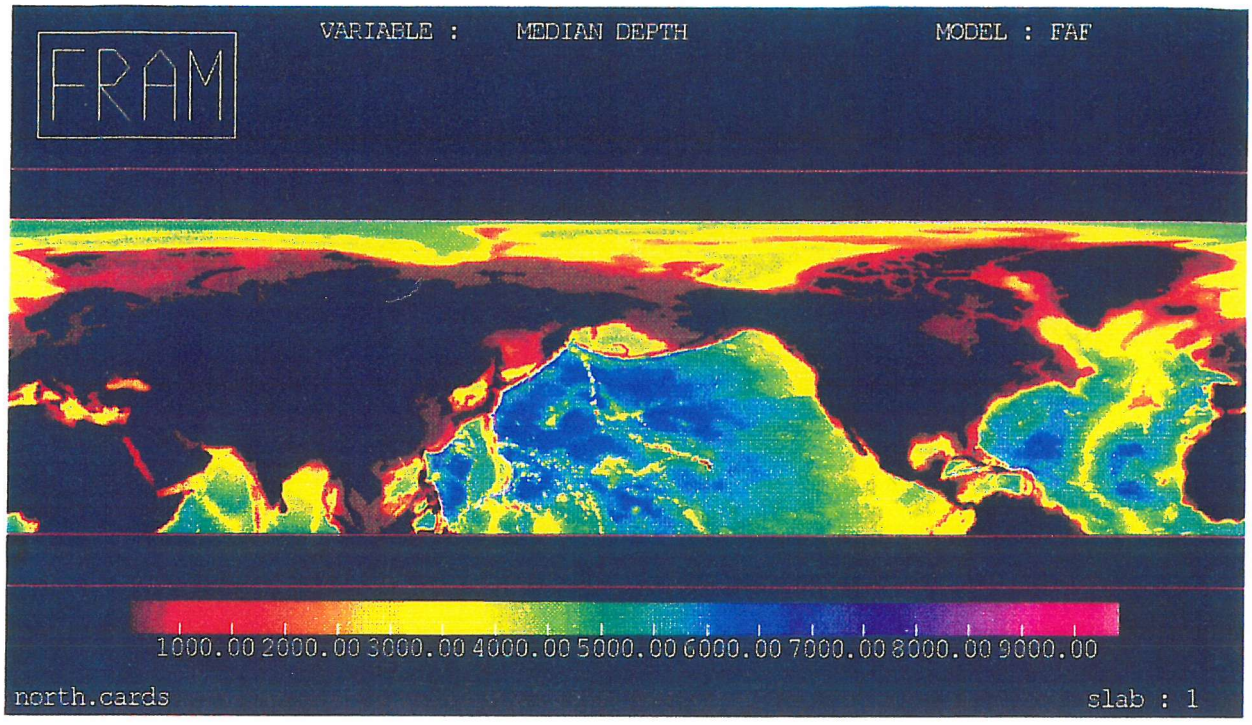

Figure 4. Northern Hemisphere 1/4° resolution.

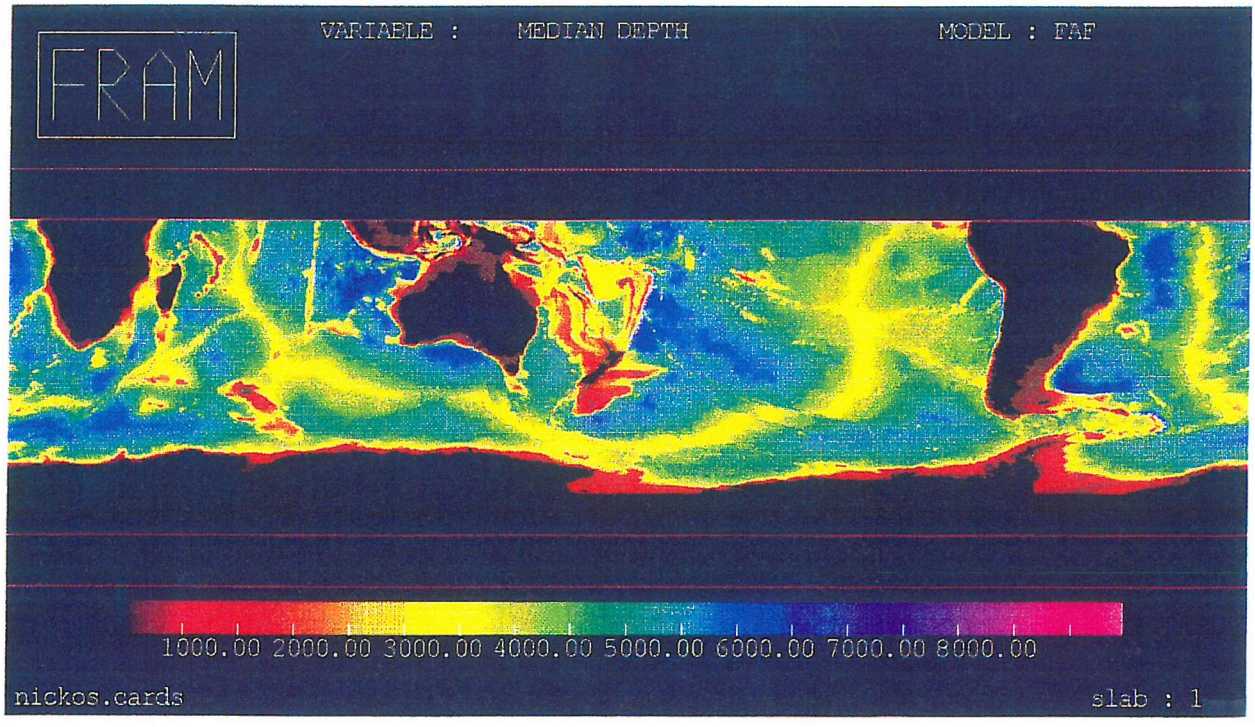

Figure 5. Southern Hemisphere 1/4° resolution

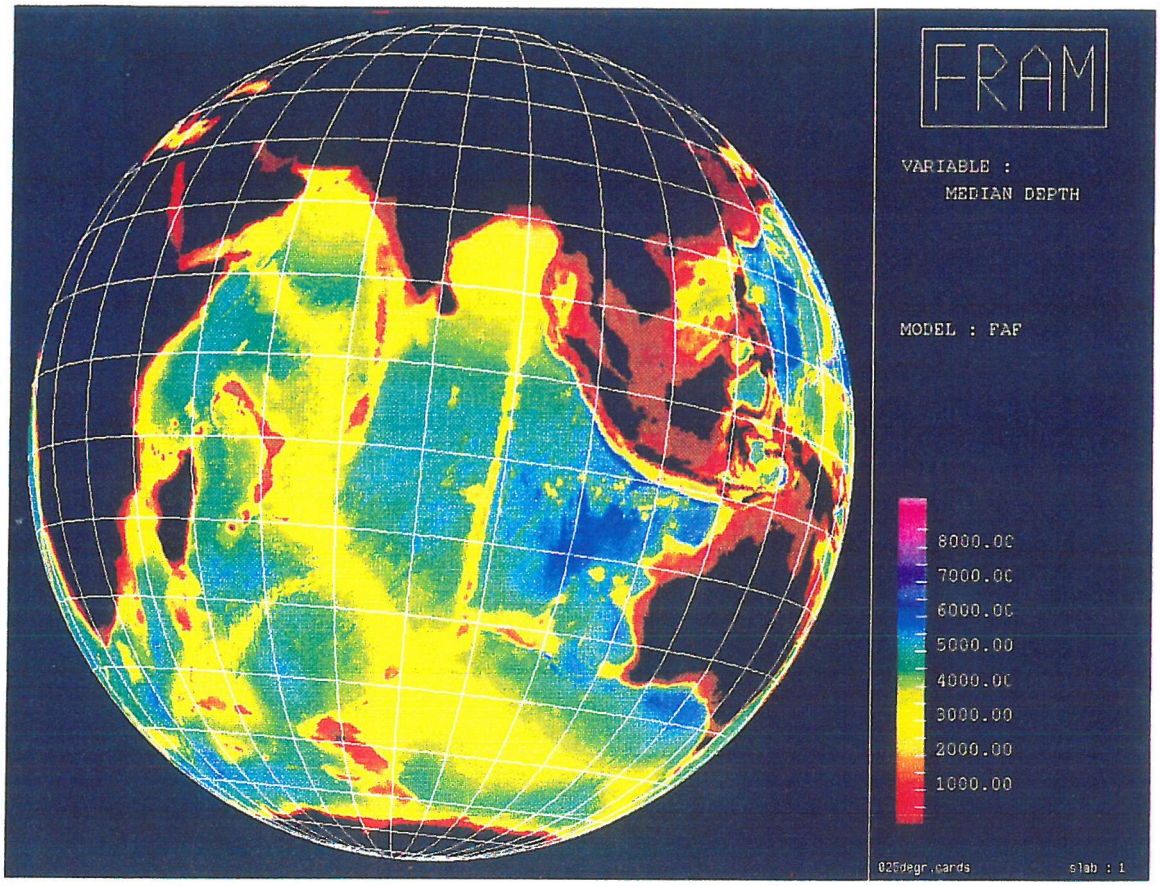

Figure 6. Indian Ocean 1/4° resolution.

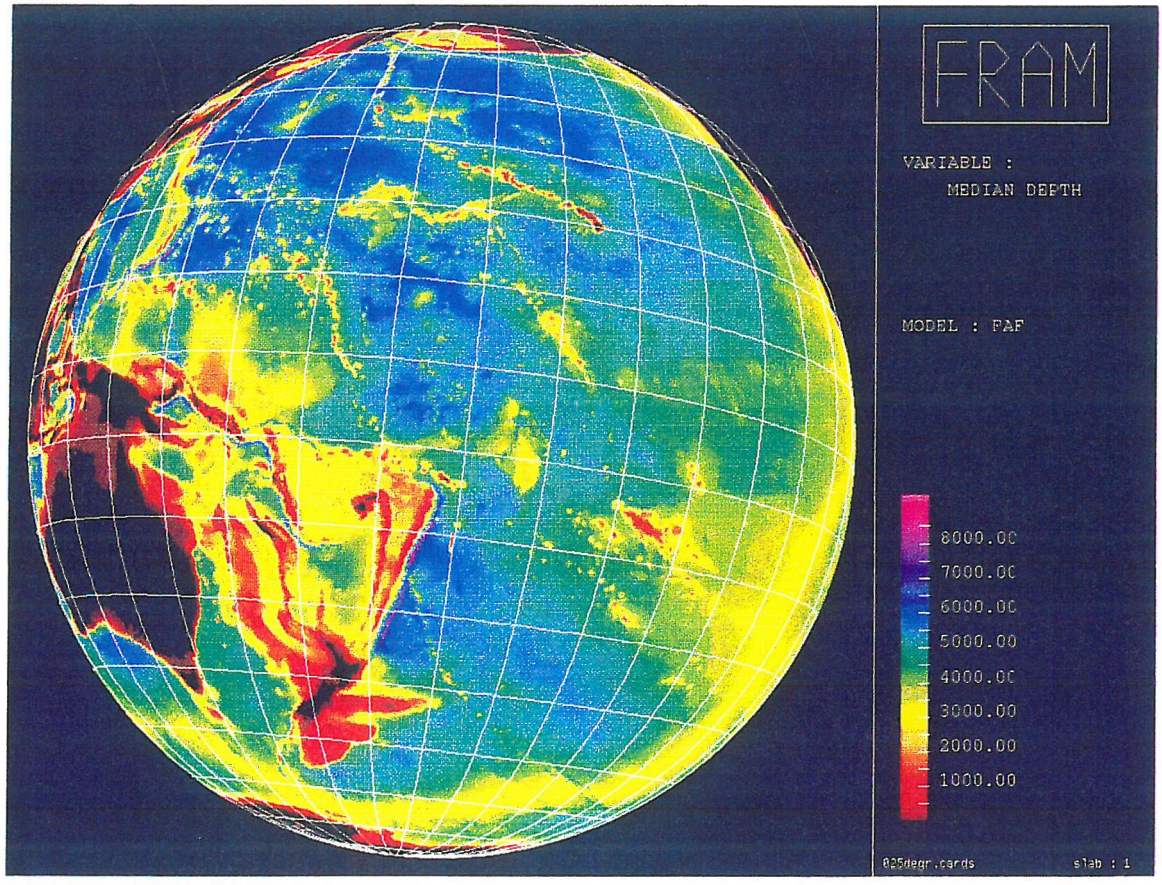

Figure 7. Pacific Ocean 1/4° resolution.

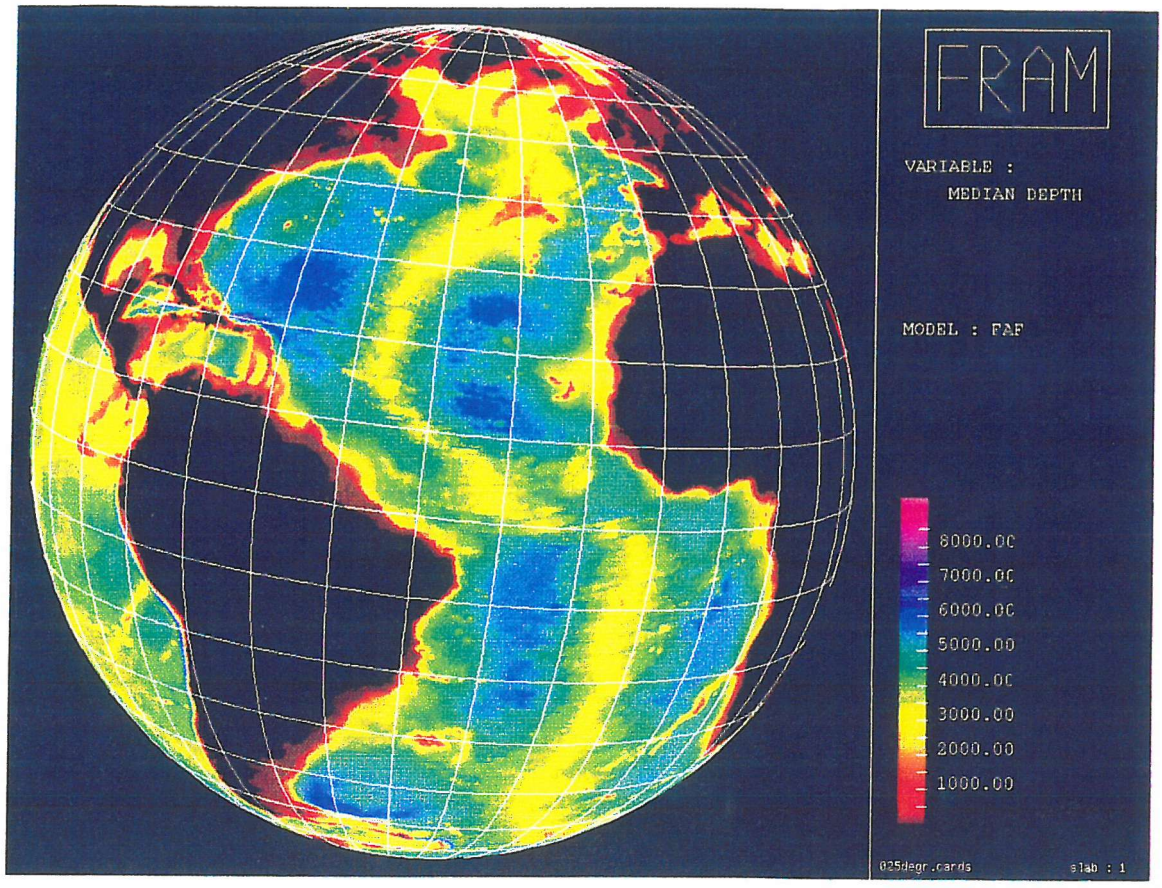

Figure 8. Atlantic Ocean 1/4° resolution.

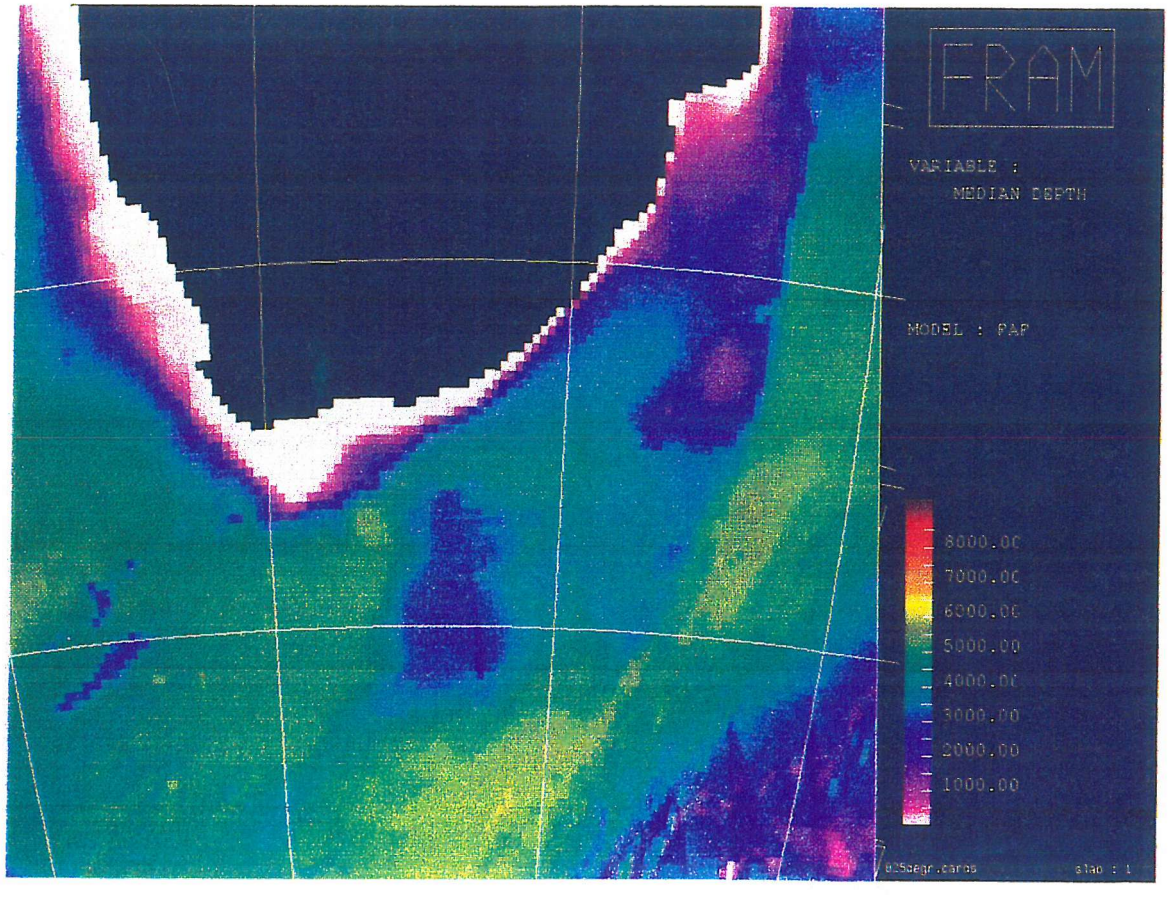

Figure 9. Southern African coastline 1/4° resolution.

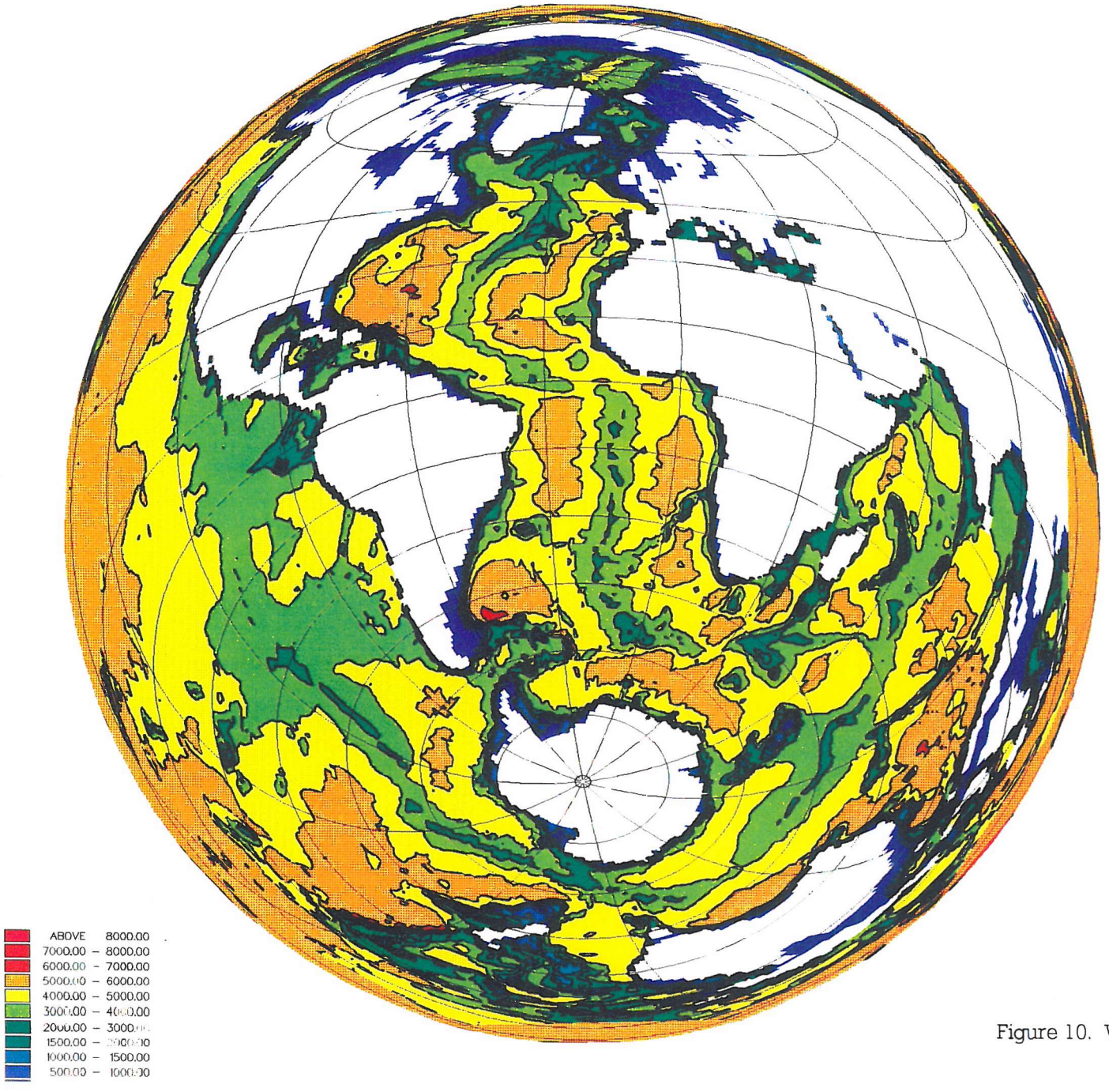

Figure 10. WOCE Projection 1° resolution.

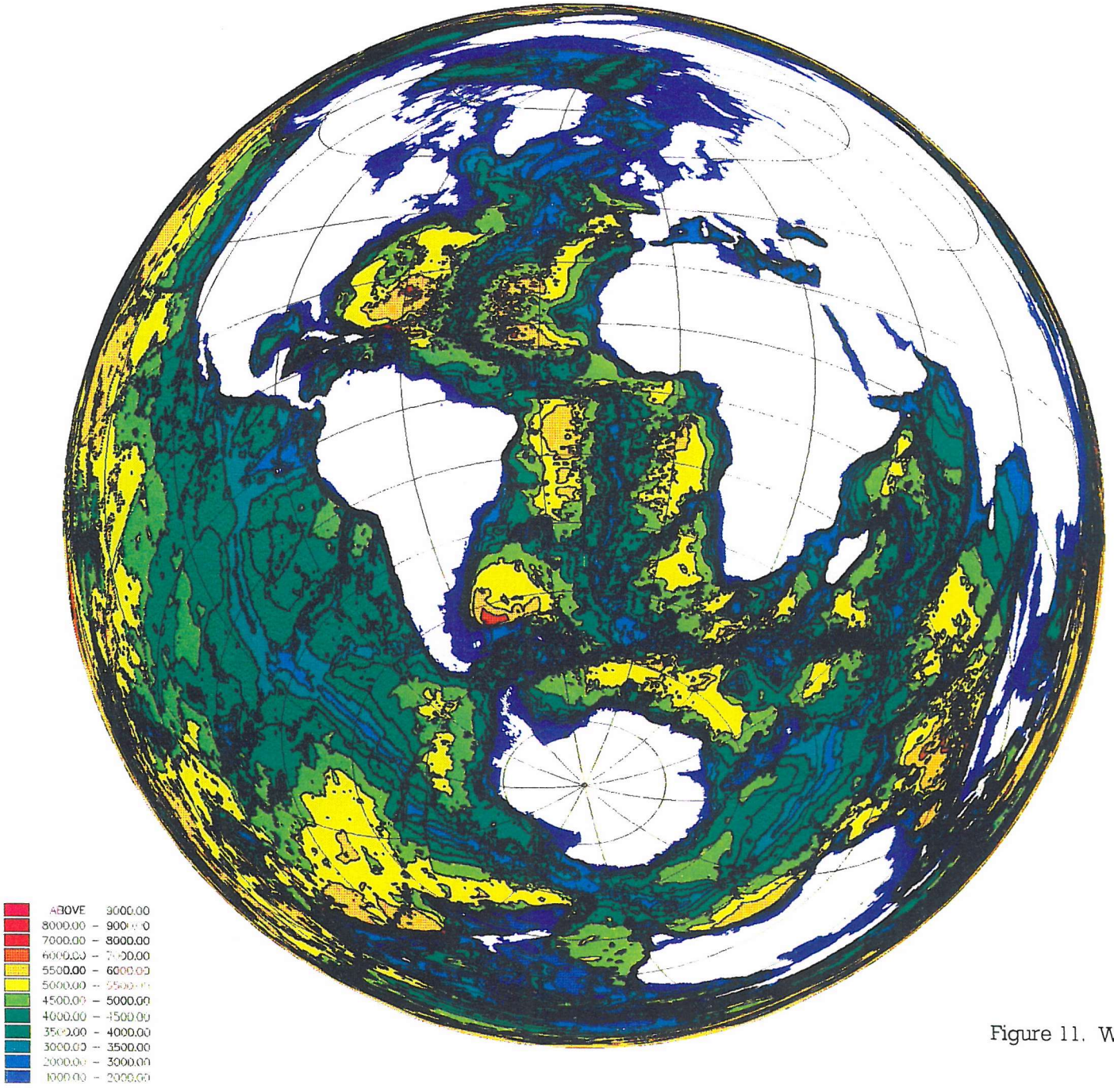

Figure 11. WOCE Projection 1/4° resolution.

APPENDIX 1

```
        PROGRAM TAPFILE

C     *PROGRAM TO READ AN ASCII TAPE WITH MULTIPLE FILES

       CHARACTER *20 FNAME
       CHARACTER * 1 ANS
       INTEGER ENDREC, ST, ED
       PARAMETER (LREC=3120)
       CHARACTER BLOCK * 3120
       FNAME='NICE DATA C'
       CALL NDOPEN (3,FNAME,1,2,'UNKNOWN',80,0,ISTAT)
       IF (ISTAT .NE. 0) THEN
         WRITE (6,*) 'ERROR IN OPENING FILE FILE ONE A'
         WRITE (6,*) 'ISTAT = ', ISTAT
         STOP
       ENDIF
C     SET UP TAPDEF TO ATTACH TAPE TO STREAM 4

       WRITE (6,*) 'DO YOU WISH TO MOVE PAST ANY FILES ? (Y/N)'
       READ(5,110)ANS
  110  FORMAT (A1)
       WRITE (6,'(A1)') ANS
       IF (ANS.EQ.'Y') THEN
         WRITE (6,*)'ENTER NUMBER OF FILES YOU WISH TO MOVE PASS'
         READ (5,*)MOVE
         CALL FILFSP(4,MOVE)
       ENDIF
C     ***************************************************
C     *START READING FROM TAPE A BLOCK AT A TIME *
C     *READ IN ALL 16 FILES                      *
C     ***************************************************
       DO 70,K=1,16
       CALL TAPDEF (4,K,'TAP1','BLP',1600,'FB',3120,3120)
C     *READ IN 81 BLOCKS*
   11    READ (4,200,END=99)BLOCK
C     CONVERT TAPE FROM ASCII TO EBCDIC SO THE IBM UNDERSTANDS IT
       CALL ASCEBC (BLOCK,BLOCK)

C     ********************************************
C     *WRITE OUT BLOCK TO OUTPUT FILE   *
C     *IE THE 81 BLOCKS OF EACH FILE    *
C     ********************************************

       ST = 1
       ED = 80
       DO 30,I=1,39
         WRITE (3,'(A80)')  BLOCK(ST:ED)
         ST = ST + 80
         ED = ST + 80
   30  CONTINUE
       GOTO 11
   99  CONTINUE
   70  CONTINUE
  200  FORMAT(A3120)
       STOP
       END
```

APPENDIX 2

```
      PROGRAM DEGREESQUARES
C     ****************************

C     ***************************************************************************************
C     *FIRST SECTION OF THE PROGRAM IS CONCERNED WITH SELECTING THE *
C     *FILES YOU WISH TO WORK WITH.EACH FILE CONTAINS DEPTH VALUES  *
C     *FROM THE NORTHERN/SOUTHERN HEMISPHERES                       *
C     ***************************************************************************************

      CHARACTER * 80 RECORD,HEADER
      CHARACTER * 20 FNAME, FILENAME
      INTEGER FILE1,FILE2,FILE3,FILE4,NREC,NFILES,POSITION
      INTEGER A(4000),B(61,61),MEDIAN,N,LOOP,F,Z,MED(400),IDEG(16)
      NREC=18954

      FNAME='NICE DATA C'
      CALL NDOPEN(3,FNAME,3,1,'OLD',80,303264,ISTAT)
      IF (ISTAT.NE.0) THEN
        WRITE(6,*)'ERROR IN OPENING FILE ', FNAME
        WRITE(6,*)'ISTAT= ',ISTAT
        STOP
      ENDIF
      FILENAME='STORDAT DATA C'
      CALL NDOPEN(2,FILENAME,1,3,'UNKNOWN',80,18954,ISTAT)
      IF (ISTAT.NE.0) THEN
        WRITE(6,*)'ERROR IN OPENING FILE ', FILENAME
        WRITE(6,*)'ISTAT= ',ISTAT
        STOP
      ENDIF

    1 WRITE (6,*) 'HOW MANY FILES DO YOU REQUIRE?'
      WRITE (6,*) 'MAXIMUM OF 4 FILES'
      READ (5,*) NFILES
      IF (NFILES .GT. 4) THEN
        WRITE (6,*) 'PLEASE TRY AGAIN,MAX NUMBER OF FILES IS 4'
        GOTO 1
      ENDIF
      IF (NFILES .EQ. 1) THEN
        WRITE(6,*)'PLEASE ENTER THE FILE YOU REQUIRE'
        READ(5,*) FILE1
      ELSEIF (NFILES .EQ. 2) THEN
        WRITE(6,*)'PLEASE ENTER WHICH TWO FILES YOU REQUIRE'
        READ(5,*) FILE1, FILE2
      ELSEIF (NFILES .EQ. 3) THEN
        WRITE(6,*)'PLEASE ENTER WHICH THREE FILES YOU REQUIRE'
        READ(5,*) FILE1, FILE2, FILE3
      ELSE
        WRITE(6,*)'PLEASE ENTER WHICH FOUR FILES YOU REQUIRE'
        READ(5,*) FILE1, FILE2, FILE3, FILE4
      ENDIF
      IF (FILE1 .EQ. 1) THEN
        POSITION=1
      ELSE
        POSITION=(18954*(FILE1-1)+1)
      ENDIF
```

```
      DO I=1,NFILES
        DO J=1,NREC
          READ (3,'(A80)',REC=POSITION)RECORD
          WRITE (2,'(A80)')RECORD
          POSITION=POSITION+1
        ENDDO
        IF (I .EQ. 1) THEN
          IF (FILE2 .NE. (FILE1 + 1)) THEN
            POSITION=(18954 * FILE1) + (18954 * ((FILE2 - FILE1) - 1)+1)
          ENDIF
        ELSEIF (I .EQ. 2) THEN
          IF (FILE3 .NE. (FILE2 + 1)) THEN
            POSITION=(18954*FILE2) + (18954 * ((FILE3 - FILE2) - 1)+1)
          ENDIF
        ELSE
          IF (FILE4 .NE. (FILE3 + 1)) THEN
            POSITION=(18954 * FILE3) + (18954 * ((FILE4 - FILE3) - 1)+1)
          ENDIF
        ENDIF
        WRITE (6,*) 'POSITION BEFORE THE END IS ', POSITION
      ENDDO
C
C     OUTPUT 4 FILES IN DIFF FORMAT TO O/P FILE
C
C
      CALL NDOPEN(15,'STORDAT DATA C',1,1,'OLD',80,18954,ISTAT)
      IF (ISTAT.NE.0) THEN
           WRITE(6,*)'ERROR IN OPENING FILE',STORDAT DATA C
           WRITE(6,*)'ISTAT = ',ISTAT
      ENDIF
      CALL NDOPEN(16,'TEST DATA D',1,3,'UNKNOWN',3005,0,ISTAT)
      IF (ISTAT.NE.0) THEN
           WRITE(6,*)'ERROR IN OPENING FILE',TEST DATA A
           WRITE(6,*)'ISTAT = ',ISTAT
      ENDIF

C     ********************************************************************************
C     *LOOP 81 TIMES FOR ALL BLOCKS IN A FILE                                        *
C     *SORT DATA INTO 61I5 FORMAT AND OUTPUT TO 'TEST DATA' FILE*
C     ********************************************************************************

      DO K=1,81
        READ (15,'(A80)') HEADER
        WRITE(16,'(A80)') HEADER
        WRITE(16,*)

        DO I=1,3728,16
          READ (15,'(16I5)') (A(I+J-1),J=1,16)
        ENDDO
        WRITE (6,*) 'BEFORE DO I=1,3668'
        DO I=1,3668,61
          WRITE (16,'(61I5)') (A(I+J-1),J=1,61)
        ENDDO
      ENDDO
C
C     CLOSE 'TEST DATA' FILE
C
      CALL NCLOSE(16,ISTAT)
      WRITE (6,*) 'AFTER DO I=1,3668'
```

```
C     ***************************************************************************
C     *READ INFORMATION FROM A 5 DEGREE SQUARE;BREAKDOWN THAT      *
C     *INFORMATION INTO 1/4 DEGREE SQUARES AND OBTAIN MEDIAN PTS.  *
C     *THEN STORE RESULTS GATHERED INTO VARIOUS FILES.             *
C     ***************************************************************************


      FILENAME='TEST DATA D'
      CALL NDOPEN (16,FILENAME,1,1,'OLD',3005,0,ISTAT)
      IF (ISTAT.NE.0) THEN
         WRITE(6,*)'ERROR IN OPENING FILE',FNAME
         WRITE(6,*)'ISTAT = ',ISTAT
      ENDIF
      FILENAME='MEDIAN DATA D'
      CALL NDOPEN (4,FILENAME,1,3,'UNKNOWN',110,0,ISTAT)
      IF (ISTAT.NE.0) THEN
         WRITE(6,*)'ERROR IN OPENING FILE',FNAME
         WRITE(6,*)'ISTAT = ',ISTAT
      ENDIF

      WRITE(6,*)'GIVE US A CHANCE'
C
C     *READ ALL NUMBERS IN FILE INTO THE B ARRAY*
C
      DO Z=1,81
         READ(16,'(A80)')HEADER
         READ(16,*) B
         ICOUNTER=0
         WRITE(4,*)
         WRITE(4,*)HEADER

C     ***************************************************************************
C     *THE 3721 DATA POINTS ARE ARRANGED IN A (61,61) ARRAY  *
C     *THE CALCULATIONS BELOW LOCATE THE DATA POINTS REQUIRED*
C     *RESULTING IN 400 1/4 DEGREE SQUARES FROM THE ORIGINAL *
C     *5 DEGREE SQUARE.                                      *
C     ***************************************************************************
       LOOP=0
       DO L=1,20
         DO K=1,20
           DO I=((L-1)*3)+1,((L-1)*3)+4
             DO J=((K-1)*3)+1,((K-1)*3)+4
               ICOUNTER=ICOUNTER+1
               IDEG(ICOUNTER)=B(J,I)
             ENDDO
           ENDDO
           N=ICOUNTER
           CALL RSORT(IDEG,16,ISTAT)

C     ***************************************************************************
C     *THE NUMBERS WITHIN THE 16 ARRAY ARE NOW SORTED AND THE *
C     *MEDIAN VALUE IS CALCULATED                             *
C     ***************************************************************************
           LOOP = LOOP + 1
           CALL AMEDIAN (IDEG,N,MEDIAN,LOOP,MED)
           ICOUNTER=0
         ENDDO
       ENDDO
```

```
C
C      *OUTPUT MED ARRAY TO MEDIAN DATA A (STREAM 4)*

       DO I=1,400,20
         WRITE (4,'(20I5)') (MED(I+J-1),J=1,20)
       ENDDO
       ENDDO
C      *CLOSE ALL STREAMS*

       CALL NCLOSE(16,ISTAT)
       IF (ISTAT.NE.0) THEN
          WRITE (6,*)'ERROR IN CLOSING FILE',FILENAME
          WRITE (6,*)'ISTAT =',ISTAT
       ENDIF
       CALL NCLOSE(4,ISTAT)
       IF (ISTAT.NE.0) THEN
WRITE (6,*)'ERROR IN CLOSING FILE',FILENAME
          WRITE (6,*)'ISTAT =',ISTAT
       ENDIF
       CALL NCLOSE(7,ISTAT)
       IF (ISTAT.NE.0) THEN
          WRITE (6,*)'ERROR IN CLOSING FILE',FILENAME
          WRITE (6,*)'ISTAT =',ISTAT
       ENDIF
       STOP
       END

C      ************************************
C      *NOW FIND THOSE MEDIAN VALUES*
C      ************************************

       SUBROUTINE AMEDIAN (IDEG,N,MEDIAN,LOOP,MED)
       REAL IDEG(16), MED(400)
       IF (MOD(N,2).EQ.0) THEN
         MED(LOOP) =IDEG(8)
       ENDIF
       RETURN
       END
```

APPENDIX 3

```
      PROGRAM LAND
C     **************
      CHARACTER * 20 FILENAME
      CHARACTER * 80 HEADER,DUMMY
      INTEGER OARRAY(20,20)
      REAL NARRAY(1440,360),VMASK(4)
      DATA VMASK/-10.0,-10.,-10.,-10./

      FILENAME='NORTH DATA A'
      CALL NDOPEN (15,FILENAME,1,1,'OLD',110,0,ISTAT)
      IF (ISTAT.NE.0) THEN
         WRITE(6,*)'ERROR IN OPENING FILE ',FILENAME
         WRITE(6,*) 'ISTAT = ',ISTAT
         STOP
      ENDIF
      KUP=0
      DO 300 KF=1,16
        KR=KF
         IF(KF.GE.9) THEN
         KUP=180
         KR=KF-8
        ENDIF
        WRITE(6,*) 'READING FILE ',KF
        DO 100 Y=1,9
         DO 100 X=1,9
               READ(15,'(A80)')DUMMY
               READ(15,'(A80)')HEADER
C              READ(15,'(A80)')DUMMY
            DO 200 J=1,20
                  READ(15,'(20I5)') (OARRAY(II,J),II=1,20)
            DO 200 I=1,20
                 IF (OARRAY(I,J).EQ.-10) THEN
                    NARRAY((X-1)*20+I+(KR-1)*180,(Y-1)*20+J+KUP) = -10.0
                  ELSE
         NARRAY((X-1)*20+I+(KR-1)*180,(Y-1)*20+J+KUP) = 1.0*OARRAY(I,J)
                  ENDIF
 200        CONTINUE
 100     CONTINUE
 300  CONTINUE
      WRITE(6,*) NARRAY(1,1),NARRAY(180,1),NARRAY(360,1)
      WRITE(6,*) NARRAY(1,45),NARRAY(180,45),NARRAY(360,45)
      WRITE(6,*) NARRAY(1,90),NARRAY(180,90),NARRAY(360,90)
C
      FILENAME='NORTH CARDS D'
      CALL NDOPEN (57,FILENAME,1,3,'UNKNOWN',80,0,ISTAT)
      CALL NICHEAD(57,'MEDIAN DEPTH','STREAM','CD','NICK''S DEPTHS')
      CALL ASCOUT(NARRAY,1440,1440,360,VMASK,2,57)
      CALL NCLOSE(15,ISTAT)
      CALL NCLOSE(57,ISTAT)
         IF (ISTAT.NE.0) THEN
            WRITE(6,*)'ERROR IN CLOSING FILE ',FILENAME
            WRITE(6,*)'ISTAT = ',ISTAT
         ENDIF
      STOP
      END
```

```
      SUBROUTINE NICHEAD(OP, TRAC, DEPVAR, OPFORM, NRUN)
C
C     ROUTINE TO WRITE HEADERS ON THE FILES
C
      CHARACTER TRAC*(*), OPFORM*(*), NRUN*(*)
      CHARACTER*9 DEPVAR, FROM(3), INCR(3), TO(3), QUAN(3)
      COMMON /TSTEP/ NDFIR, NDLAS, NDINC
      COMMON /TIME/  TTSEC
      INTEGER OP, NOP(3)
C
C     ESTABLISH DETAILS FOR HEADING
C
C
        NDFIR=0.0
        NDLAS=0.0
        NDINC=0.0
        QUAN(1) = 'LONGITUDE'
        QUAN(2) = ' LATITUDE'
        FROM(1) = '    0.    '
        FROM(2) = '  -89.500'
        INCR(1) = '    1.0   '
        INCR(2) = '    1.0   '
        TO(1)   = '  359.5   '
        TO(2)   = '   -0.50  '
        NOP(1)  = 360
        NOP(2)  = 90
      QUAN(3)=' TIMESTEP'
      NOP(3)=1
      WRITE(FROM(3),'(I9)')NDFIR
      WRITE(INCR(3),'(I9)')NDINC
      WRITE(TO(3),'(I9)')NDLAS
C
        WRITE(OP,5101)TRAC,OPFORM
C
      WRITE(OP,5102)NRUN
      WRITE(OP,5103)(I,I=1,3)
      WRITE(OP,5104)(QUAN(I),I=1,3)
      WRITE(OP,5105)(FROM(I),I=1,3)
      WRITE(OP,5106)(INCR(I),I=1,3)
      WRITE(OP,5107)(TO(I),I=1,3)
      WRITE(OP,5108)(NOP(I),I=1,3)
5100  FORMAT('VARIABLE :',A15,2X,A9,T41,'FORMAT   :',A2)
5101  FORMAT('VARIABLE :',A15,T41,'FORMAT   :',A2)
5102  FORMAT('MODEL : FAF      COMMENTS :',A50)
5103  FORMAT('INDEX ',9X,':',3('    ',I1,'     :'))
5104  FORMAT('QUANTITY ',6X,':',A9,':',A9,':',A9,':')
5105  FORMAT('FROM     ',6X,':',A9,':',A9,':',A9,':')
5106  FORMAT('INCREMENT',6X,':',A9,':',A9,':',A9,':')
5107  FORMAT('TO       ',6X,':',A9,':',A9,':',A9,':')
5108  FORMAT('NO.OF POINTS ',2X,':',I9,':',I9,':',I9,':')
      WRITE(57,*)
      WRITE(57,*)
      RETURN
      END
```

```
      SUBROUTINE ASCOUT (ARRAY,IDIM,ID,JD,VMASK,NCHAR,NOUT)
C
C     VERSION WITH SINGLE PRECISION REAL ARGUMENTS FOR 64-BIT MACHINE
C
C     SUBROUTINE TO ENCODE A SECTION OF AN ARRAY AS SETS OF 'NCHAR'
C     PRINTABLE CHARACTERS, AND WRITE AS A FORMATTED CARD-IMAGE DUMP.
C     (USES ASCII CHARACTERS 0-9 , A-Z , LOWER CASE A-Z AND BRACKETS)
C
C     ARRAY - 2-D ARRAY OF VALUES TO BE CONVERTED
C     IDIM  - DECLARED I-DIMENSION OF ARRAY IN CALLING PROGRAM
C     ID,JD - SPECIFY SECTION OF ARRAY TO BE CONVERTED
C     VMASK - 4-ELEMENT ARRAY WHOSE VALUES INDICATE 'MASKED' POINTS.
C             SUCH POINTS ARE DENOTED BY ONE OF THE FOUR POSSIBLE
C             COMBINATIONS OF FULL STOP AND COMMA, PADDED OUT TO NCHAR
C             CHARACTERS BY REPETITION OF THE LAST CHARACTER OF THE PA
C             THESE VALUES ARE IGNORED IN FINDING MAX AND MINS FOR SCA
C             THE VMASK VALUES ARE NORMALLY MUCH LARGER THAN OTHER VAL
C     NCHAR - NUMBER OF CHARACTERS TO BE USED TO REPRESENT AN ARRAY VA
C     NOUT  - FORTRAN CHANNEL NUMBER OF OUTPUT DATASET.
C
C     M. A. ROWE  SEPT. 1987 ( REWRITTEN J. R. BLUNDELL  07/07/1988 )
C     THIS VERSION (INTERNALLY DECLARED CHARACTER ARRAY) 14/12/1988
C     MODIFIED TO ALLOW FOR FOUR TYPES OF MASKED POINT   07/02/1989
C     N.B. INTERNAL ARITHMETIC ALWAYS DONE IN 64-BIT MODE
C
C     INTERNAL PARAMETERS:
C
C     LRECL - MAX. LENGTH OF DATA RECORD TO BE OUTPUT
C     NASCC - NUMBER OF DIFFERENT ASCII CHARACTERS USED IN
C             REPRESENTATION OF NUMBERS (AT UNMASKED POINTS)
C     NCMAX - MAX. NUMBER OF CHARACTERS WHICH CAN BE USED
C             TO REPRESENT AN ARRAY ELEMENT
C
      INTEGER LRECL,NASCC,NCMAX
C
      PARAMETER ( LRECL=80, NASCC=64, NCMAX=5 )
C
C     LOCAL VARIABLES
C
      INTEGER ICODE(NCMAX),IDIM,ID,JD,NCHAR,NOUT,
     *        I,J,NNUM,IC,INTEG,NCBUFF,LINLEN,MTYPE
      REAL ARRAY(IDIM,JD),VMASK(4)
      REAL FMIN,FMAX,RANGE,ARANG,SCALE
      CHARACTER*1 ASCARR(LRECL),LKUP(NASCC),CMASK(2),MASK(NCMAX,4)
      CHARACTER*(NASCC) CHAREP
C
      EQUIVALENCE (CHAREP(1:1),LKUP(1))
C
C     SPECIFY THE NASCC CHARACTERS TO BE USED IN THE NUMBER
C     REPRESENTATION, AND THE CHARACTERS DENOTING MASKED POINTS
C
      CHAREP( 1:10) = '0123456789'
      CHAREP(11:36) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
      CHAREP(37:62) = 'abcdefghijklmnopqrstuvwxyz'
      CHAREP(63:64) = '()'
      CMASK(1)='.'
      CMASK(2)=','
```

```
C
C     WRITE OUT CODING INFO IN FIRST DATA RECORD
C     (WRITE WARNING TO UNIT 6 IF IT WON'T FIT)
C
      IF ( NASCC.GT.72 ) WRITE(6,50) NASCC
   50 FORMAT(/,2X,'**ASCOUT WARNING: OVERLENGTH CODING RECORD,',
     *       ' NASCC =',I3)
      WRITE(NOUT,'(I4,1X,2A1,1X,72A1)') NASCC,CMASK,(LKUP(I),I=1,NASCC
C
C     CHECK INPUT VALUE OF NCHAR
C
      IF ( NCHAR.LT.2 .OR. NCHAR.GT.NCMAX ) THEN
        WRITE(6,100) NCHAR
  100   FORMAT(/,2X,'**ASCOUT WARNING: ROUTINE CALLED',
     *         ' WITH INCORRECT NCHAR =',I4)
        RETURN
      END IF
C
C     CHECK INPUT VALUES OF VMASK ARE ALL DIFFERENT,
C     OTHERWISE MASKING WILL BE AMBIGUOUS
C
      DO 110 J=1,3
        DO 110 I=J+1,4
          IF ( VMASK(I).EQ.VMASK(J) ) WRITE(6,120) I,J
  110 CONTINUE
  120 FORMAT(/,2X,'**ASCOUT WARNING: VMASK(',I1,') = VMASK(',I1,')',
     *       /,2X,'**MASKING PRODUCED WILL BE AMBIGUOUS')
C
C     CREATE THE 4 TYPES OF MASK, INCLUDING PADDING CHARACTERS
C
      DO 130 IC=1,NCHAR
        MASK(IC,1) = CMASK(1)
        MASK(IC,2) = CMASK(1)
        MASK(IC,3) = CMASK(2)
        MASK(IC,4) = CMASK(2)
  130 CONTINUE
      MASK(2,2) = CMASK(2)
      MASK(2,3) = CMASK(1)
C
C     ESTABLISH RANGE OF DATA AND SCALING FOR CONVERSION
C     ( TYPICAL SIZE OF VALUES ASSUMED O(10**5) )
C
      FMAX = -9999999.9
      FMIN =  9999999.9
      DO 150 I=1,ID
        DO 150 J=1,JD
          DO 140 MTYPE=1,4
            IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) GOTO 145
  140     CONTINUE
            FMIN = MIN( FMIN,ARRAY(I,J) )
            FMAX = MAX( FMAX,ARRAY(I,J) )
  145     CONTINUE
  150 CONTINUE
```

```
      IF ( FMAX.LT.-99999.9 .OR. FMIN.GT.99999.9 )
     *  WRITE(6,200) FMIN,FMAX
  200 FORMAT(/,2X,'**ASCOUT WARNING: LARGE +VE MINIMUM OR LARGE',
     *       ' -VE MAXIMUM VALUE',/,2X,'FMAX, FMIN = ',1P,2E16.5)
C
      NNUM = ID*JD
      WRITE(NOUT,'(1P,2E20.12,4I10)') FMIN,FMAX,ID,JD,NNUM,NCHAR
      ARANG = REAL( NASCC**NCHAR - 1 )
      RANGE = FMAX - FMIN
      SCALE = ARANG/RANGE
      IF ( INT(SCALE).LT.1 ) WRITE(6,220) SCALE
  220 FORMAT(/,2X,'**ASCOUT WARNING: SCALE = ',1P,E14.5)
C
        IF ( (RANGE*1.0E10 ).LT.1.0E0 ) THEN
C
      WRITE(NOUT,250)
  250 FORMAT('**ASCOUT WARNING: FIELD APPROX. CONSTANT,',
     *       ' NOT CHARACTER CODED')
C
        ELSE
C
C     SCALE ARRAY AND ENCODE AS NCHAR PRINTABLE CHARACTERS
C
      NCBUFF = 0
      IF ( NCHAR.EQ.3 ) LINLEN=78
      IF ( NCHAR.NE.3 ) LINLEN=80
      DO 500 J=1,JD
        DO 500 I=1,ID
C
          DO 350 MTYPE = 1,4
            IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) THEN
C             TYPE MTYPE MASKED POINT; COPY FROM MASK(NCMAX,MTYPE)
              DO 300 IC = 1,NCHAR
                ASCARR(NCBUFF+IC) = MASK(IC,MTYPE)
  300         CONTINUE
              GOTO 450
            END IF
  350     CONTINUE
C         NORMAL POINT; ENCODE AS NCHAR CHARACTERS
          INTEG = NINT( (ARRAY(I,J)-FMIN)*SCALE )
          DO 400 IC=NCHAR,1,-1
            ICODE(IC) = 1 + MOD( INTEG, NASCC )
            ASCARR(NCBUFF+IC) = LKUP( ICODE(IC) )
            INTEG = INTEG/NASCC
  400     CONTINUE
  450     CONTINUE
          NCBUFF = NCBUFF + NCHAR
C
          IF ( NCBUFF.EQ.LINLEN ) THEN
C           BUFFER ASCARR FULL; WRITE TO
C           CHANNEL NOUT (CARD-IMAGE FORMAT)
            IF ( NCHAR.NE.3 ) THEN
              WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
             ELSE
              WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
            END IF
            NCBUFF = 0
          END IF
```

```
C
  500 CONTINUE
C
C     FLUSH CHARACTER BUFFER IF NOT EMPTY
C
      IF ( NCBUFF.NE.0 ) THEN
        IF ( NCHAR.NE.3 ) THEN
          WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
         ELSE
          WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
        END IF
        NCBUFF = 0
      END IF
C
       END IF
C
      RETURN
      END
C
```

APPENDIX 4

```
        PROGRAM MAXMIN
C       ******************

       CHARACTER * 20 FILENAME,FLINE*80
       REAL FILE1(1440,360), FILE2(1440,360)
       REAL CARRAY(1440,720),VMASK(4)
       DATA VMASK/-10.0,-10.,-10.,-10./
C
C OPEN IP FILE 1
C
       FILENAME='NORTH CARDS D'
       OPEN (15,FILENAME)
C
C OPEN IP FILE 2
C
       FILENAME='NICKOS CARDS D'
       OPEN (16,FILENAME)
C
C OPEN OP FILE 1
C
       FILENAME='025DEGR CARDS D'
       OPEN (17,FILENAME,)
C
C READ ALL DATA FROM IP FILE 1
C
       NCOUNT=0
 407    READ(15,'(A)',END=409,ERR=409) FLINE
        NCOUNT=NCOUNT+1
        IF(FLINE(1:6).EQ.'  64 .') GOTO 408
        GOTO 407
 409    WRITE(6,*) 'TARGET LINE NOT FOUND IN FILE1'
        STOP
408    REWIND 15
        DO 401 I=1,NCOUNT-1
         READ(15,*)
 401    CONTINUE
C
      CALL ASCIIN(FILE1,FDMIN,FDMAX,1440,1440,360,15,NUMIN,VMASK,IFLAG)
C
C READ ALL DATA FROM IP FILE 2
C
        NCOUNT=0
 507    READ(16,'(A)',END=509,ERR=509) FLINE
        NCOUNT=NCOUNT+1
        IF(FLINE(1:6).EQ.'  64 .') GOTO 508
        GOTO 507
 509    WRITE(6,*) 'TARGET LINE NOT FOUND IN FILE2'
        STOP
 508    REWIND 16
        DO 501 I=1,NCOUNT-1
         READ(16,*)
 501    CONTINUE
C
```

```
      CALL ASCIIN(FILE2,FDMIN,FDMAX,1440,1440,360,16,NUMIN,VMASK,IFLAG)
C
C PUT FILE1 AND FILE2 INTO THE CARRAY
C
      DO 98 I=1,1440
        DO 99 J=1,360
          CARRAY(I,J)     = FILE1(I,J)
          CARRAY(I,360+J) = FILE2(I,J)
 99     CONTINUE
 98   CONTINUE
C
C OUTPUT ALL DATA TO OUTPUT FILE
C
      CALL NICHEAD(17,'MEDIAN DEPTH','STREAM','CD','NICK''S DEPTHS')
      CALL ASCOUT(CARRAY,1440,1440,720,VMASK,2,17)
C
C CLOSE ALL FILES
C
      CLOSE(15,ISTAT)
      CLOSE(16,ISTAT)
      CLOSE(17,ISTAT)
      STOP
      END
      SUBROUTINE ASCIIN (ARRAY,ARMIN,ARMAX,IDIM,M,NROW,
     *                   NIN,NUMIN,VMASK,IFLAG)
*
*     VERSION WITH DOUBLE PRECISION REAL ARGUMENTS FOR 32-BIT MACHINE
*
*     SUBROUTINE TO READ DATA WRITTEN AS SETS OF PRINTABLE
*     CHARACTERS AND CONVERT BACK TO FLOATING POINT FORM.
*
*     INPUT ARGUMENTS:
*     IDIM       -    FIRST DIMENSION OF ARRAY AS DECLARED
*     M          -    NUMBER OF I-VALUES TO BE FILLED
*     NROW       -    NUMBER OF ROWS OF DATA
*     NIN        -    FORTRAN CHANNEL NUMBER FROM WHICH DATA IS READ
*     VMASK      -    4-ELEMENT ARRAY OF VALUES TO
*                     BE ASSIGNED TO MASKED ELEMENTS
*     IFLAG      -    AS USED IN NAG ROUTINES:
*                             SET TO 0 ON ENTRY TO STOP IF AN ERROR
*                             SET TO 1 ON ENTRY TO CONTINUE   "
*
*     OUTPUT ARGUMENTS:
*     ARRAY      -    ARRAY OF DATA VALUES RECONSTRUCTED
*                     FROM THE CHARACTER DATA READ IN
*     ARMIN      -    MINIMUM DATA VALUE AT UNMASKED POINT
*     ARMAX      -    MAXIMUM DATA VALUE AT UNMASKED POINT
*     NUMIN      -    NUMBER OF DATA VALUES READ IN
*     IFLAG      -    USED AS IN NAG TO INDICATE ERRORS:
*                             SET TO 0 INDICATES NO ERRORS
*                             SET TO 1 IF VALUE OF NCHAR INCOMPATIBLE
*                             SET TO 2 IF M*NROW > NO. OF PTS IN DATA
*                             SET TO 3 IF M*NROW < NO. OF PTS IN DATA
*                             SET TO 4 IF UNACCEPTABLE DATA READ IN
*
```

```
*     M. A. ROWE  SEPT. 1987 ( rewritten J. R. BLUNDELL  07/07/1988 )
*     This version (internally declared character array) 15/12/1988
*     Modified to allow for four types of masked point   07/02/1989
*     N.B. All internal arithmetic done in 64-bit mode
*
*     INTERNAL PARAMETERS:
*
*     LRECL   - Max. length of data records to be processed
*     NCMAX   - Max. number of characters usable to represent each numbe
*     NPMAX   - Max. number of printable characters usable in decoding
*     IMSK1,2 - Entry in lookup table to signify masking character
*     IDUFF   - Entry in lookup table to signify invalid character
*     N.B. must choose each of IMSK1,2 and IDUFF > (NPMAX-1)*(NCMAX+1)

      INTEGER LRECL,NCMAX,NPMAX,IMSK1,IMSK2,IDUFF
*
      PARAMETER ( LRECL=80, NCMAX=5, NPMAX=72,

*           IMSK1=777, IMSK2=888, IDUFF=999 )
*
*     Local variables
*
      INTEGER ICVAL(0:255),NASCC,NCHAR,IDIM,M,NROW,NIN,NUMIN,I,J,
     *        ICODE(NCMAX),N,ICSUM,ICRIT,IC,INTEG,MSKVAL(4),NCREC,
     *        IFLAG,ID,JD,NGET,IFRET,NCTODO,NLINE,NCDONE,NCLEFT,MTYPE
      REAL FMIN,FMAX,RANGE,ARANG,SCALE
      CHARACTER*1 ASCARR(LRECL),CHAREP(NPMAX),CMASK(2)
*
**************   Change argument precision here   **************
*
C     DOUBLE PRECISION ARRAY(IDIM,NROW),ARMIN,ARMAX,VMASK(4)
      REAL             ARRAY(IDIM,NROW),ARMIN,ARMAX,VMASK(4)
*
*     Check input values of VMASK are all different,
*     otherwise masking information will be lost
*
      DO 20 J=1,3
        DO 20 I=J+1,4
          IF ( VMASK(I).EQ.VMASK(J) ) WRITE(6,30) I,J
   20 CONTINUE
   30 FORMAT(/,2X,'**ASCIIN WARNING: VMASK(',I1,') = VMASK(',I1,')',
     *       /,2X,'**MASKING INFORMATION WILL BE LOST')
*
*     Read in NASCC, the number of different characters used to
*     encode valid numbers, followed by the characters denoting
*     masked points, then the list of NASCC encoding characters.
*     Check validity of NASCC, and for repeated code characters.
*
      READ(NIN,'(I4,1X,2A1,1X,72A1)') NASCC,CMASK,(CHAREP(I),I=1,NASCC)
      IF ( NASCC.GT.72 .OR. NASCC.GT.NPMAX ) THEN
        WRITE(6,50) NASCC
      END IF
   50 FORMAT(/,2X,'**ASCIIN WARNING: READS IN EXCESSIVE NASCC =',I4)
      IF ( CMASK(1).EQ.CMASK(2) ) WRITE(6,150)
```

```
      DO 100 IC=1,NASCC
        IF ( CHAREP(IC).EQ.CMASK(1) ) WRITE(6,150)
        IF ( CHAREP(IC).EQ.CMASK(2) ) WRITE(6,150)
        DO 100 I=1,IC-1
          IF ( CHAREP(IC).EQ.CHAREP(I) ) WRITE(6,150)
  100 CONTINUE
  150 FORMAT(/,2X,'**ASCIIN WARNING: FINDS REPEATED',
     *            ' CHARACTER IN CODING LIST',
     *       /,2X,'**DATA PROBABLY CORRUPTED DURING TRANSFER',
     *       /,2X,'**RECONSTRUCTED FIELDS WILL BE INCORRECT')
*
*     Read the next data record
*
      READ(NIN,200) FMIN,FMAX,ID,JD,NUMIN,NCHAR
**    PRINT *,'FMIN=',FMIN,'  FMAX=',FMAX,'  NUMIN=',NUMIN
  200 FORMAT(1P,2E20.12,4I10)
*
*     Check value of NCHAR is valid
*
      IF ( NCHAR.LT.2 .OR. NCHAR.GT.NCMAX ) THEN
        WRITE(6,250) NCHAR
  250   FORMAT(/,2X,'**ASCIIN WARNING: READS INVALID NCHAR =',I4)
        IF ( IFLAG.EQ.0 ) THEN
          STOP 1
        ELSE
          IFRET = 1
        END IF
      END IF
*
*     Check value of NUMIN equals that expected in program
*
      IF ( NUMIN.GT.(M*NROW) ) THEN
        IF ( IFLAG.EQ.0 ) THEN
          STOP 3
        ELSE
          IFRET = 3
        END IF
      ELSE IF ( NUMIN.LT.(M*NROW) ) THEN
        IF ( IFLAG.EQ.0 ) THEN
          STOP 2
        ELSE
          IFRET = 2
        END IF
      END IF
*
*     N.B. Data are stored as sets of NCHAR printable characters.
*          ASCII codes in the range 32 to 122 are used (this
*          includes upper and lower case letters, the digits
*          0 to 9 and parentheses) but not control characters.
*
*     Set initial lookup table entries to invalid character value
*
      DO 300 I=0,255
        ICVAL(I) = IDUFF
  300 CONTINUE
*
```

```
*     Update lookup table entries indexed by the position in
*     the collating sequence of the printable characters used
*     Set MSKVAL, the numbers indicating NCHAR masking characters
*     Set ICRIT, the maximum valid value of ICSUM
*
      DO 350 I=1,NASCC
        ICVAL( ICHAR(CHAREP(I)) ) = I - 1
  350 CONTINUE
      ICVAL( ICHAR(CMASK(1)) ) = IMSK1
      ICVAL( ICHAR(CMASK(2)) ) = IMSK2
      MSKVAL(1) = NCHAR*IMSK1 + IMSK1
      MSKVAL(2) = NCHAR*IMSK1 + IMSK2
      MSKVAL(3) = NCHAR*IMSK2 + IMSK1
      MSKVAL(4) = NCHAR*IMSK2 + IMSK2
      ICRIT = (NASCC-1)*(NCHAR+1)
**    WRITE(6,'(4X,I3,4X,I3)') (I,ICVAL(I),I=0,255)
*
*     Read in the rest of the data, in character form, and
*     convert character form to reals, unless field is nearly
*     uniform, in which case set all array elements to FMAX.
*
      RANGE = FMAX - FMIN
      ARANG = DBLE( NASCC**NCHAR - 1 )
      SCALE = RANGE/ARANG
*****
      IF ( (RANGE*1.0D10) .LT. 1.0D0 ) THEN
***
        ARMIN = FMAX
        ARMAX = FMAX
        DO 400 J=1,NROW
          DO 400 I=1,M
            ARRAY(I,J) = FMAX
  400   CONTINUE
        READ(NIN,'(A1)') ASCARR(1)
***
        ELSE
***
        ARMIN = FMIN
        ARMAX = FMAX
        N = 0
        NLINE = 2
        NCLEFT = 0
        NCTODO = NCHAR*NUMIN
        IF ( NCHAR.EQ.3 ) NCREC=78
        IF ( NCHAR.NE.3 ) NCREC=80
*
        DO 600 J=1,NROW
          DO 600 I=1,M
            N = N + 1
            IF ( N.GT.NUMIN ) GOTO 600
*
```

```
            IF ( NCLEFT.EQ.0 ) THEN
*             Buffer ASCARR empty; read another line
              NGET = MIN( NCTODO,NCREC )
              IF ( NCHAR.NE.3 ) THEN
                READ(NIN,'(80A1)',END=700) (ASCARR(IC),IC=1,NGET)
               ELSE
                READ(NIN,'(1X,78A1,1X)',END=700) (ASCARR(IC),IC=1,NGET)
              END IF
              NCLEFT = NGET
              NCDONE = 0
              NLINE = NLINE + 1
            END IF
*
*           Convert set of NCHAR characters to integer
*           Compute ICSUM for checking if at masked point
            ICSUM = 0
            INTEG = 0
            DO 450 IC=1,NCHAR
              ICODE(IC) = ICVAL( ICHAR(ASCARR(NCDONE+IC)) )
              ICSUM = ICSUM + ICODE(IC)
              INTEG = INTEG*NASCC + ICODE(IC)
  450       CONTINUE
            ICSUM = ICSUM + ICODE(NCHAR)
*
*           Interpret characters according to ICSUM
            DO 480 MTYPE=1,4
              IF ( ICSUM.EQ.MSKVAL(MTYPE) ) THEN
*               Type MTYPE masked point; set ARRAY = VMASK(MTYPE)
                ARRAY(I,J) = VMASK(MTYPE)
                GOTO 550
              END IF
  480       CONTINUE
            IF ( ICSUM.GT.ICRIT ) THEN
*             Found a character not in coding list;
*             print warning & either quit or set ARRAY = VMASK(1)

              WRITE(6,500) NLINE,N,(ICODE(IC),IC=1,NCHAR)
  500         FORMAT(/,'  **ASCIIN ERROR: INVALID CHARACTER IN DATA',
     *               /,'  **OCCURS IN LINE',I6,' OF CHARACTER DUMP',
     *               /,'  **FOR N =',I8,' THE CODES ARE:',5I4)
              IF ( IFLAG.EQ.0 ) THEN
                STOP 4
               ELSE
                IFRET = 4
                ARRAY(I,J) = VMASK(1)
              END IF
            ELSE
*             Reconstruct real value
              ARRAY(I,J) = DBLE(INTEG)*SCALE + FMIN
            END IF
*
  550       CONTINUE
            NCDONE = NCDONE + NCHAR
            NCLEFT = NCLEFT - NCHAR
            NCTODO = NCTODO - NCHAR
*
  600   CONTINUE
***
      END IF
```

```
****
      IFLAG = IFRET
      RETURN
*
  700 CONTINUE
      WRITE(6,750) NIN,NLINE
  750 FORMAT(/,2X,'**ASCIIN WARNING: DATA FILE READ IN',
     *       /,2X,'**ON UNIT',I3,' HAS BEEN',
     *       /,2X,'**TRUNCATED; ONLY HAS',I8,' LINES')
      IFLAG = IFRET
      RETURN
*
      END
C
C

      SUBROUTINE NICHEAD(OP, TRAC, DEPVAR, OPFORM, NRUN)
C
C     ROUTINE TO WRITE HEADERS ON THE FILES
C
      CHARACTER TRAC*(*), OPFORM*(*), NRUN*(*)
      CHARACTER*9 DEPVAR, FROM(3), INCR(3), TO(3), QUAN(3)
      COMMON /TSTEP/ NDFIR, NDLAS, NDINC
      COMMON /TIME/  TTSEC
      INTEGER OP, NOP(3)
C
C     ESTABLISH DETAILS FOR HEADING
C
C
        NDFIR=0.0
        NDLAS=0.0
        NDINC=0.0
        QUAN(1) = 'LONGITUDE'
        QUAN(2) = ' LATITUDE'
        FROM(1) = '    0.    '
        FROM(2) = '  -89.500'
        INCR(1) = '    1.0   '
        INCR(2) = '    1.0   '
        TO(1)   = '  359.5   '
        TO(2)   = '   89.50  '
        NOP(1)  = 360
        NOP(2)  = 180
      QUAN(3)=' TIMESTEP'
      NOP(3)=1
      WRITE(FROM(3),'(I9)')NDFIR
      WRITE(INCR(3),'(I9)')NDINC
      WRITE(TO(3),'(I9)')NDLAS
C
        WRITE(OP,5101)TRAC,OPFORM
```

```
C
      WRITE(OP,5102)NRUN
      WRITE(OP,5103)(I,I=1,3)
      WRITE(OP,5104)(QUAN(I),I=1,3)
      WRITE(OP,5105)(FROM(I),I=1,3)
      WRITE(OP,5106)(INCR(I),I=1,3)
      WRITE(OP,5107)(TO(I),I=1,3)
      WRITE(OP,5108)(NOP(I),I=1,3)
5100  FORMAT('VARIABLE :',A15,2X,A9,T41,'FORMAT   :',A2)
5101  FORMAT('VARIABLE :',A15,T41,'FORMAT   :',A2)
5102  FORMAT('MODEL : FAF      COMMENTS :',A50)
5103  FORMAT('INDEX ',9X,':',3('  .',I1,'     :'))
5104  FORMAT('QUANTITY ',6X,':',A9,':',A9,':',A9,':')
5105  FORMAT('FROM     ',6X,':',A9,':',A9,':',A9,':')
5106  FORMAT('INCREMENT',6X,':',A9,':',A9,':',A9,':')
5107  FORMAT('TO       ',6X,':',A9,':',A9,':',A9,':')
5108  FORMAT('NO.OF POINTS ',2X,':',I9,':',I9,':',I9,':')
      WRITE(57,*)
      WRITE(57,*)
      RETURN
      END
c
c
c
      SUBROUTINE ASCOUT (ARRAY,IDIM,ID,JD,VMASK,NCHAR,NOUT)
C
C     VERSION WITH SINGLE PRECISION REAL ARGUMENTS FOR 64-BIT MACHINE
C
C     SUBROUTINE TO ENCODE A SECTION OF AN ARRAY AS SETS OF 'NCHAR'
C     PRINTABLE CHARACTERS, AND WRITE AS A FORMATTED CARD-IMAGE DUMP.
C     (USES ASCII CHARACTERS 0-9 , A-Z , LOWER CASE A-Z AND BRACKETS)
C
C     ARRAY - 2-D ARRAY OF VALUES TO BE CONVERTED
C     IDIM  - DECLARED I-DIMENSION OF ARRAY IN CALLING PROGRAM
C     ID,JD - SPECIFY SECTION OF ARRAY TO BE CONVERTED
C     VMASK - 4-ELEMENT ARRAY WHOSE VALUES INDICATE 'MASKED' POINTS.
C             SUCH POINTS ARE DENOTED BY ONE OF THE FOUR POSSIBLE
C             COMBINATIONS OF FULL STOP AND COMMA, PADDED OUT TO NCHAR
C             CHARACTERS BY REPETITION OF THE LAST CHARACTER OF THE PAIR
C             THESE VALUES ARE IGNORED IN FINDING MAX AND MINS FOR SCALI
C             THE VMASK VALUES ARE NORMALLY MUCH LARGER THAN OTHER VALUE
C     NCHAR - NUMBER OF CHARACTERS TO BE USED TO REPRESENT AN ARRAY VALU
C     NOUT  - FORTRAN CHANNEL NUMBER OF OUTPUT DATASET.
C
C     M. A. ROWE  SEPT. 1987 ( REWRITTEN J. R. BLUNDELL  07/07/1988 )
C     THIS VERSION (INTERNALLY DECLARED CHARACTER ARRAY) 14/12/1988
C     MODIFIED TO ALLOW FOR FOUR TYPES OF MASKED POINT   07/02/1989
C     N.B. INTERNAL ARITHMETIC ALWAYS DONE IN 64-BIT MODE
C
C     INTERNAL PARAMETERS:
C
C     LRECL - MAX. LENGTH OF DATA RECORD TO BE OUTPUT
C     NASCC - NUMBER OF DIFFERENT ASCII CHARACTERS USED IN
C             REPRESENTATION OF NUMBERS (AT UNMASKED POINTS)
C     NCMAX - MAX. NUMBER OF CHARACTERS WHICH CAN BE USED
C             TO REPRESENT AN ARRAY ELEMENT
C
```

```
      INTEGER LRECL,NASCC,NCMAX
C
      PARAMETER ( LRECL=80, NASCC=64, NCMAX=5 )
C
C     LOCAL VARIABLES
C
      INTEGER ICODE(NCMAX),IDIM,ID,JD,NCHAR,NOUT,
     *        I,J,NNUM,IC,INTEG,NCBUFF,LINLEN,MTYPE
      REAL ARRAY(IDIM,JD),VMASK(4)
      REAL FMIN,FMAX,RANGE,ARANG,SCALE
      CHARACTER*1 ASCARR(LRECL),LKUP(NASCC),CMASK(2),MASK(NCMAX,4)
      CHARACTER*(NASCC) CHAREP
C
      EQUIVALENCE (CHAREP(1:1),LKUP(1))
C
C     SPECIFY THE NASCC CHARACTERS TO BE USED IN THE NUMBER
C     REPRESENTATION, AND THE CHARACTERS DENOTING MASKED POINTS
C
      CHAREP( 1:10) = '0123456789'
      CHAREP(11:36) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
      CHAREP(37:62) = 'abcdefghijklmnopqrstuvwxyz'
      CHAREP(63:64) = '()'
      CMASK(1)='.'
      CMASK(2)=','
C
C     WRITE OUT CODING INFO IN FIRST DATA RECORD
C     (WRITE WARNING TO UNIT 6 IF IT WON'T FIT)
C
      IF ( NASCC.GT.72 ) WRITE(6,50) NASCC
   50 FORMAT(/,2X,'**ASCOUT WARNING: OVERLENGTH CODING RECORD,',
     *       ' NASCC =',I3)
      WRITE(NOUT,'(I4,1X,2A1,1X,72A1)') NASCC,CMASK,(LKUP(I),I=1,NASCC)
C
C     CHECK INPUT VALUE OF NCHAR
C
      IF ( NCHAR.LT.2 .OR. NCHAR.GT.NCMAX ) THEN
        WRITE(6,100) NCHAR
  100   FORMAT(/,2X,'**ASCOUT WARNING: ROUTINE CALLED',
     *         ' WITH INCORRECT NCHAR =',I4)
        RETURN
      END IF
C
C     CHECK INPUT VALUES OF VMASK ARE ALL DIFFERENT,
C     OTHERWISE MASKING WILL BE AMBIGUOUS
C
      DO 110 J=1,3
        DO 110 I=J+1,4
          IF ( VMASK(I).EQ.VMASK(J) ) WRITE(6,120) I,J
  110 CONTINUE
  120 FORMAT(/,2X,'**ASCOUT WARNING: VMASK(',I1,') = VMASK(',I1,')',
     *       /,2X,'**MASKING PRODUCED WILL BE AMBIGUOUS')
C
C     CREATE THE 4 TYPES OF MASK, INCLUDING PADDING CHARACTERS
C
```

```
      DO 130 IC=1,NCHAR
        MASK(IC,1) = CMASK(1)
        MASK(IC,2) = CMASK(1)
        MASK(IC,3) = CMASK(2)
        MASK(IC,4) = CMASK(2)
  130 CONTINUE
      MASK(2,2) = CMASK(2)
      MASK(2,3) = CMASK(1)
C
C     ESTABLISH RANGE OF DATA AND SCALING FOR CONVERSION
C     ( TYPICAL SIZE OF VALUES ASSUMED O(10**5) )
C
      FMAX = -9999999.9
      FMIN =  9999999.9
      DO 150 I=1,ID
        DO 150 J=1,JD
         DO 140 MTYPE=1,4
            IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) GOTO 145
  140     CONTINUE
            FMIN = MIN( FMIN,ARRAY(I,J) )
            FMAX = MAX( FMAX,ARRAY(I,J) )
  145     CONTINUE
  150 CONTINUE
      IF ( FMAX.LT.-99999.9 .OR. FMIN.GT.99999.9 )
     *  WRITE(6,200) FMIN,FMAX
  200 FORMAT(/,2X,'**ASCOUT WARNING: LARGE +VE MINIMUM OR LARGE',
     *       ' -VE MAXIMUM VALUE',/,2X,'FMAX, FMIN = ',1P,2E16.5)
C
      NNUM = ID*JD
      WRITE(NOUT,'(1P,2E20.12,4I10)') FMIN,FMAX,ID,JD,NNUM,NCHAR
      ARANG = REAL( NASCC**NCHAR - 1 )
      RANGE = FMAX - FMIN
      SCALE = ARANG/RANGE
      IF ( INT(SCALE).LT.1 ) WRITE(6,220) SCALE
  220 FORMAT(/,2X,'**ASCOUT WARNING: SCALE = ',1P,E14.5)
C
        IF ( (RANGE*1.0E10 ).LT.1.0E0 ) THEN
C
      WRITE(NOUT,250)
  250 FORMAT('**ASCOUT WARNING: FIELD APPROX. CONSTANT,',
     *       ' NOT CHARACTER CODED')
C
        ELSE
C
C     SCALE ARRAY AND ENCODE AS NCHAR PRINTABLE CHARACTERS
C
      NCBUFF = 0
      IF ( NCHAR.EQ.3 ) LINLEN=78
      IF ( NCHAR.NE.3 ) LINLEN=80
      DO 500 J=1,JD
        DO 500 I=1,ID
C
          DO 350 MTYPE = 1,4
            IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) THEN
C             TYPE MTYPE MASKED POINT; COPY FROM MASK(NCMAX,MTYPE)
              DO 300 IC = 1,NCHAR
                ASCARR(NCBUFF+IC) = MASK(IC,MTYPE)
  300         CONTINUE
              GOTO 450
            END IF
```

```
  350     CONTINUE
C         NORMAL POINT; ENCODE AS NCHAR CHARACTERS
          INTEG = NINT( (ARRAY(I,J)-FMIN)*SCALE )
          DO 400 IC=NCHAR,1,-1
            ICODE(IC) = 1 + MOD( INTEG, NASCC )
            ASCARR(NCBUFF+IC) = LKUP( ICODE(IC) )
            INTEG = INTEG/NASCC
  400     CONTINUE
  450     CONTINUE
          NCBUFF = NCBUFF + NCHAR
C
          IF ( NCBUFF.EQ.LINLEN ) THEN
C           BUFFER ASCARR FULL; WRITE TO
C           CHANNEL NOUT (CARD-IMAGE FORMAT)
            IF ( NCHAR.NE.3 ) THEN
              WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
             ELSE
              WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
            END IF
            NCBUFF = 0
          END IF
C
  500 CONTINUE
C
C     FLUSH CHARACTER BUFFER IF NOT EMPTY
C
      IF ( NCBUFF.NE.0 ) THEN
        IF ( NCHAR.NE.3 ) THEN
          WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
         ELSE
          WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
        END IF
        NCBUFF = 0
      END IF
C
        END IF
C
      RETURN
      END
C
C
```


Natural
Environment
Research
Council