



3 4456 0381890 1

ORNL/TM-12666

ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

Modernization of the
Graphics Post-Processors of the
Hamburg German Climate Computer
Center Carbon Cycle Codes

Emily J. Stevens
Gregory S. McNeilly

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
BOOK ROOM 101
LIBRARY LOAN COPY
DO NOT TRANSFER TO AN OTHER PERSON
If you wish to make a copy of this
report, send a request with a self-addressed
envelope with postage paid.

ORNL/TM-12666

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 620-8401

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

105

Computing Applications Division

**MODERNIZATION OF THE GRAPHICS POST-PROCESSORS OF
THE HAMBURG GERMAN CLIMATE COMPUTER CENTER
CARBON CYCLE CODES**

Emily J. Stevens
Gregory S. McNeilly

March 1994

Prepared in partial fulfillment of the requirements of the DOE Science and Engineering Research Semester, Illinois College, Jacksonville, Illinois, under the direction of Gregory S. McNeilly, research advisor, Oak Ridge National Laboratory, Computing Applications Division.

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U. S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



3 4456 0381890 1

CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENT	v
1. INTRODUCTION	1
2. DISCUSSION	1
3. SUMMARY	13
SELECTED BIBLIOGRAPHY	14
REFERENCES	16
APPENDIX A	17
APPENDIX B	22
APPENDIX C	82

ACKNOWLEDGMENT

Research sponsored by the U.S. Department of Energy, Carbon Dioxide Research Program, Atmospheric and Climate Research Division, Office of Health and Environmental Research and by the National Aeronautics and Space Administration, and by the Environmental Protection Agency.

ABSTRACT

The existing National Center for Atmospheric Research (NCAR) code in the Hamburg Oceanic Carbon Cycle Circulation Model and the Hamburg Large-Scale Geostrophic Ocean General Circulation Model was modernized and reduced in size while still producing an equivalent end result. A reduction in the size of the existing code from more than 50,000 lines to approximately 7,500 lines in the new code has made the new code much easier to maintain. The existing code in the Hamburg model uses legacy NCAR (including even emulated CALCOMP subroutines) graphics to display graphical output. The new code uses only current (version 3.1) NCAR subroutines.

1. INTRODUCTION

The Hamburg Large-Scale Geostrophic (LSG) ocean general circulation model and the Hamburg Oceanic Carbon Cycle circulation model calculate the motion of oceanic carbon cycle tracers.¹ The LSG model calculations are used as "fixed" input into the carbon cycle model.² With each of these models a post-processor capable of processing the data and displaying it graphically for the user in a variety of different ways was included. Both the LSG and the carbon cycle post-processors are capable of showing section plots, where the data are retrieved from a specific ocean, and map plots, where the data are retrieved from all the oceans. The LSG post-processor is also capable of generating a velocity map plot, a meridional section plot and a line plot of certain carbon cycle tracers in the ocean. The carbon cycle post-processor contains a plot that shows the model versus the geosec (measured) data in the form of a line graph. All these plots in the Hamburg post-processors use old National Center for Atmospheric Research (NCAR) calls and "CALCOMP" emulation, where every point is virtually plotted and the color is calculated for each point. These NCAR calls were to be removed from the existing code, and a new code developed with more modern NCAR calls. A significant reduction in the size of code was obtained, making the new post-processor much easier to maintain.

2. DISCUSSION

After examining the current NCAR utility set, it was decided that the new model would primarily use the following packages: CONPACK, PLOTCHAR, LABELBAR, AUTOGRAPH, EZMAPA, EZMAP, and VELVCT. CONPACK is a contour plotting package that constructs contour plots from rectangular arrays of data by a method of scanning, done by ARSCAM, and contour plotting, done by CONPACK. This package "provides a 'tool kit' of FORTRAN subroutines that may be called in various combinations in order to draw different styles of contour plots."³ PLOTCHAR is a character plotting package that allows the user three qualities of characters along with numerous fonts.³ LABELBAR is a plotting package that is used to "create a labeled, filled, rectangular bar to serve as a key for a filled plot."³ AUTOGRAPH is "a graphics package enabling the user to draw graphs, each with a labelled background and each displaying one or more curves."⁴ EZMAPA and EZMAP are both map-drawing packages: EZMAPA provides a color-fill interface while still using the built-in packages that EZMAP has to offer, such as continent drawing.⁴ VELVCT is a two-dimensional velocity package where arrows are plotted to show direction and magnitude at a particular point on the plot.⁴ These packages give the post-processor an interface that is easily maintainable and much shorter than the original Hamburg post-processor. The first goal was to create a carbon cycle post-processor using the new NCAR calls, while displaying the same information in a manner similar to the Hamburg post-processor.

All source code is archived on the Computing Applications Division workstation complex, and all source code listings for the new post-processors are included in the Appendixes of this paper.

The top level post-processor logical flow is outlined in Fig. 1. First, an overview (quicklook) of simulation data compared with measurements is generated. Second, a "map" plot series (horizontal cuts at various depths) is created. Third, a "section" plot series (vertical cuts within various oceans) is produced.

The quicklook section of the carbon cycle post-processor shows the model vs the geosec (measured) data. This code reads the needed data from PROFIL and quick.dat, two files created in the Hamburg model and used by the

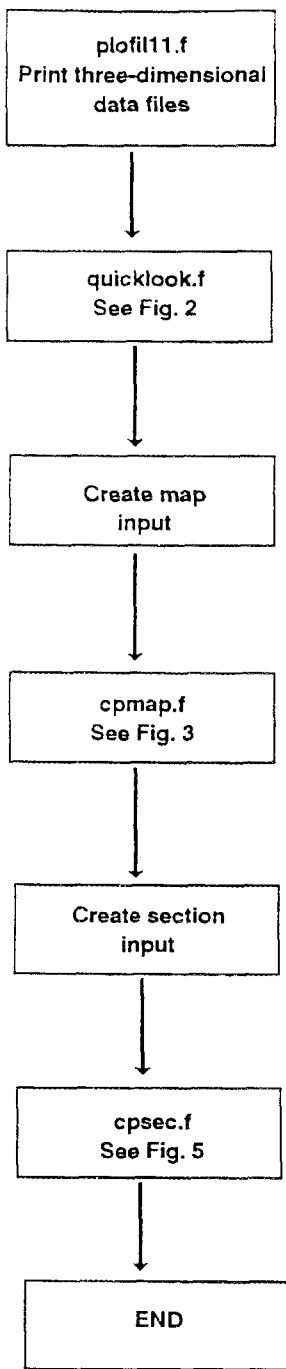


Fig. 1. Overview flow chart for carbon cycle model.

post-processors (see Fig. 2). The plot shows five windows, each plotting the model and geosecs data of different carbon cycle tracers in a specified ocean. The data are scanned, and the minimum and maximum of both lines for one window are found so that both lines are scaled the same. Then the AUTOGRAPH routine EZXY is called twice, once for the model data curve and once for the geosecs data curve, for each window. The informational labels are added by PLOTCHAR.

The map-plotting code is devised to read in map input data, and then enter a loop where the refinement and the plotting will take place (see Fig. 3). The map input is created by a csh-script and gives the user the choice of which maps are to be plotted and at which depth they are to be plotted. After retrieving the name of the plot file using subroutine READINPUT, the correct three-dimensional data file is read in by a series of two-dimensional reads. This step is done so that each layer is read in individually and either discarded because it was not selected or used in further manipulation. This procedure is all done in the subroutine CHOOSEMAP. CHOOSEMAP also calls the correct subroutines for refinement and plotting of the data. The refinement of the data has been disabled presently because unsmoothed and unrefined data are desired. This procedure is indicated in the flow charts by circled subroutines and dotted flow arrows. This process is used for all twenty-six possible maps, and any additional information needed on relevant subroutines can be found in the comments of the program itself.

The plotting section of the maps, PLOTMAP, uses CONPACK, LABELBAR, and EZMAPA calls (see Fig. 4). CONPACK is set up with many user-callable routines that allow the user to specify exactly how the map should look and what should be displayed on it. CONPACK must be set up by a series of initialization calls that give the user some choice in what is used and what is not used. These subroutines take the two-dimensional data array and place them into a work array, where the data are manipulated so that they are usable by CONPACK. The work array must have plenty of space and is always much larger than the data array because CONPACK must add the choices we have set earlier and any inherent choices that it needs to put the contours on the screen.

The contours are then scanned by the NCAR routine ARSCAM, where color is matched to each contour. The contours are divided evenly throughout the data value range, and continents are blocked out by use of special value numbers. These special value numbers are present to tell CONPACK not to plot those numbers. These have been introduced into the data before coming to CONPACK using the continent scheme used in the Hamburg model. Those gridpoints with these special values are left without color; then the subroutine MAPOVR, which uses EZMAPA and EZMAP calls, is called to fill the continents in with a patterned form. The mapping of contour colors to informational values is done by LABELBAR and displayed on the screen according to initialization values set by the program; all informational labels on the map are done by PLOTCHAR.

The section plot is handled much the same way as the map plot (see Fig. 5). This code is designed to read in the section data input, manipulate these data and call routines for plotting. As in the map plot, almost all the refinement process has been turned off for the time being. The plotting subroutine, PLOTSEC, is much the same as the map subroutine (see Fig. 6). The only difference is in the size of the viewing window, a few informational labels and the presence of a small continental map with a line showing the section of the ocean where the data are located. This map is produced with a smaller viewing window but uses the same EZMAPA and EZMAP subroutines used for the continents in the map plots. The section line is drawn with the AUTOGRAPH routine EZXY, which takes two one-dimensional arrays and uses them to find the line.

The last item to be done with the carbon cycle post-processor was to bring all three plot packages together and run them consecutively. This step was made easier because the carbon cycle side of the Hamburg model is executed from a csh-script in which each plot type is a separate entity. This aspect of the Hamburg model is followed in the new model and executed through a script also. A few slight differences are noted in these scripts. The new script contains a makefile, which is used to compile and build the programs.

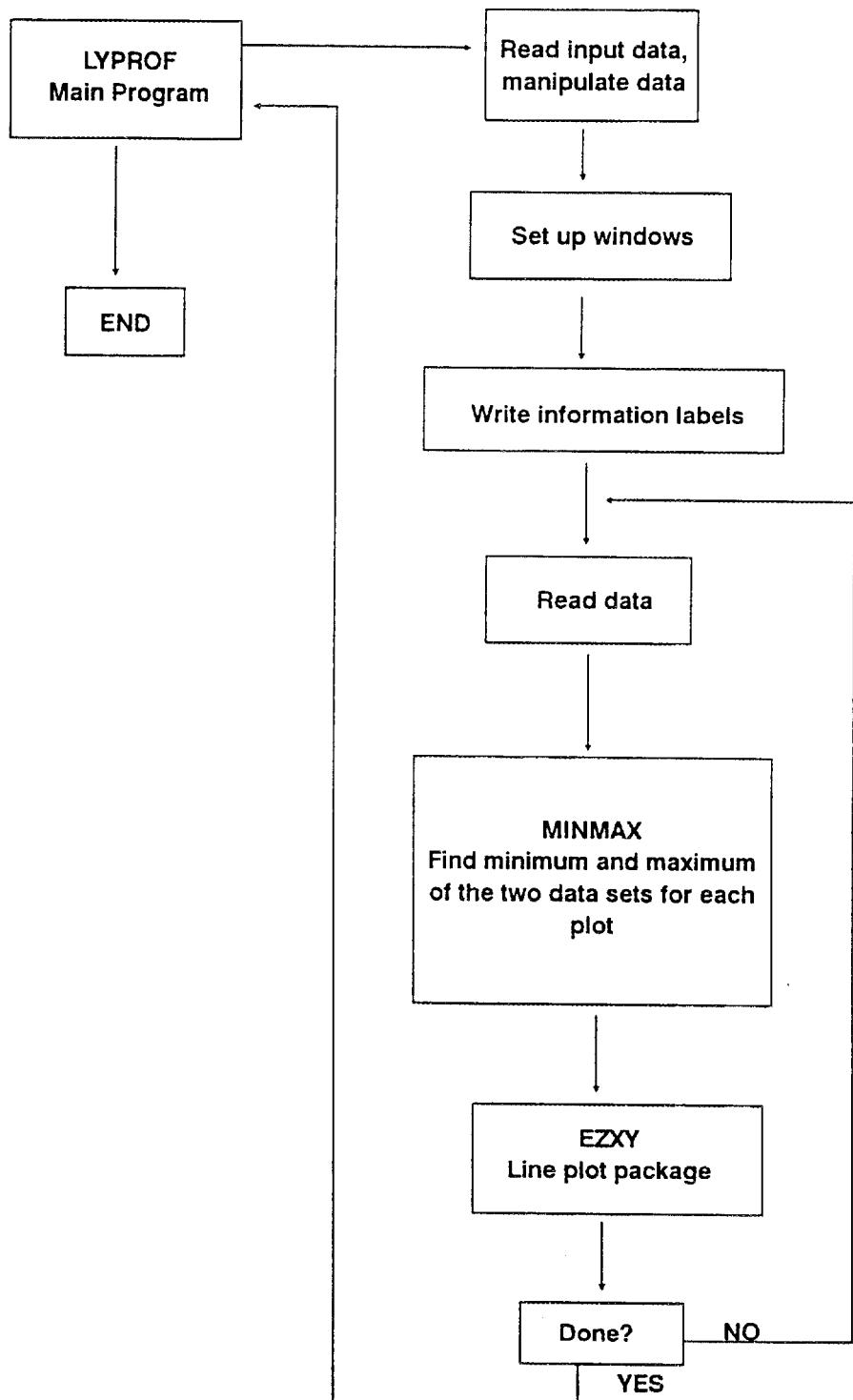


Fig. 2. Flow chart of quicklook.f program in carbon cycle model.

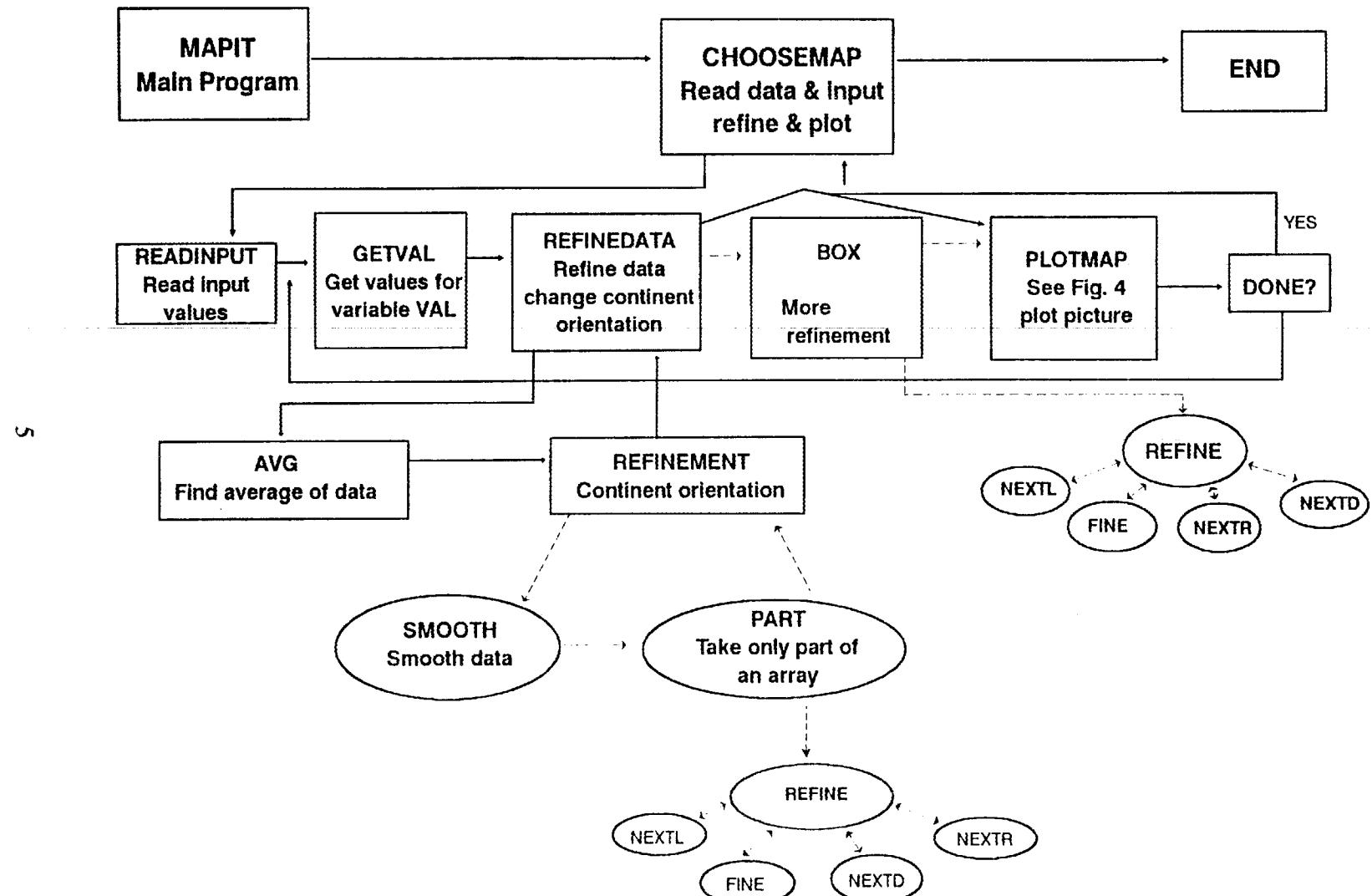


Fig. 3. Flow chart for map plot of carbon cycle model.

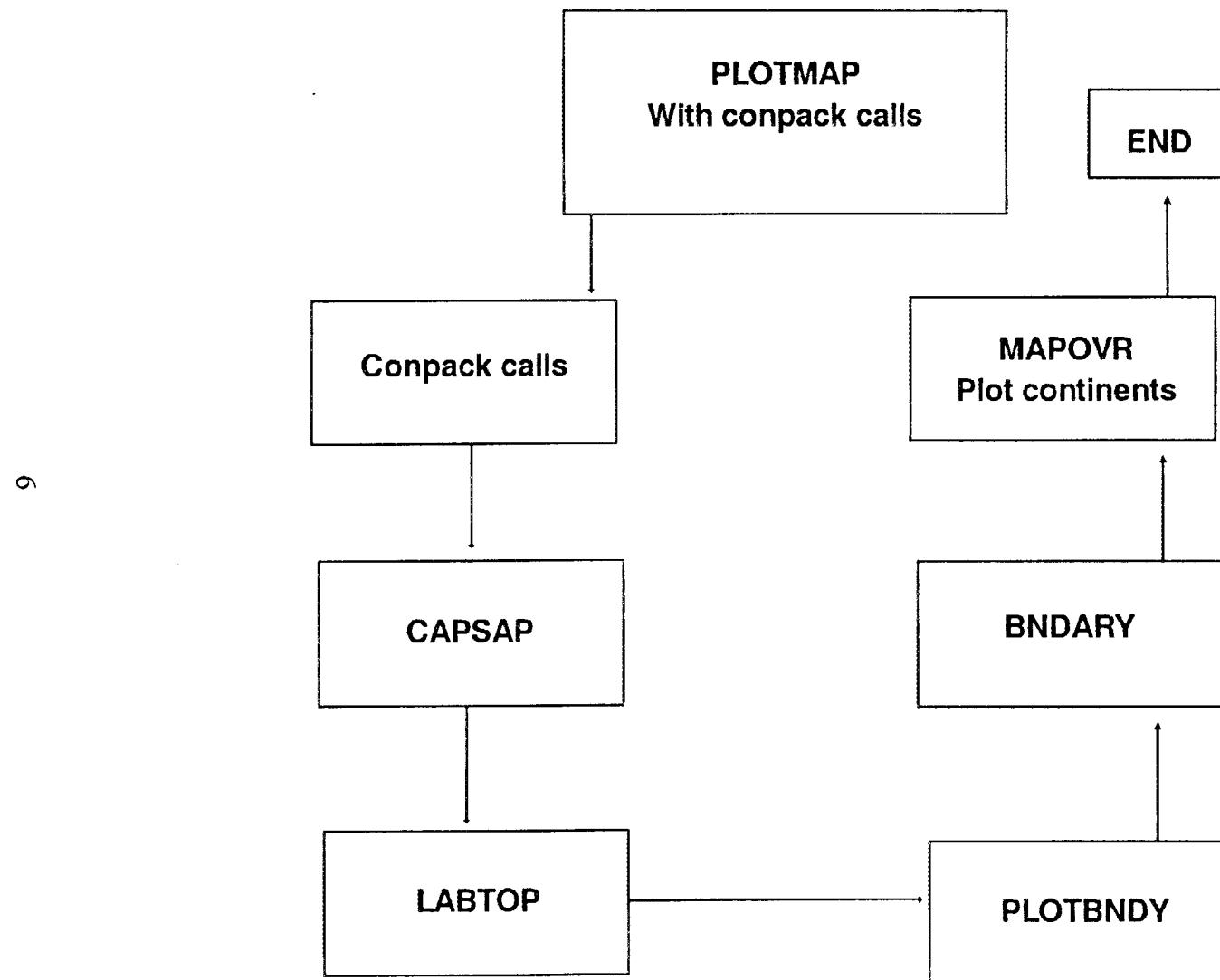


Fig. 4. Flow chart for the plot map.

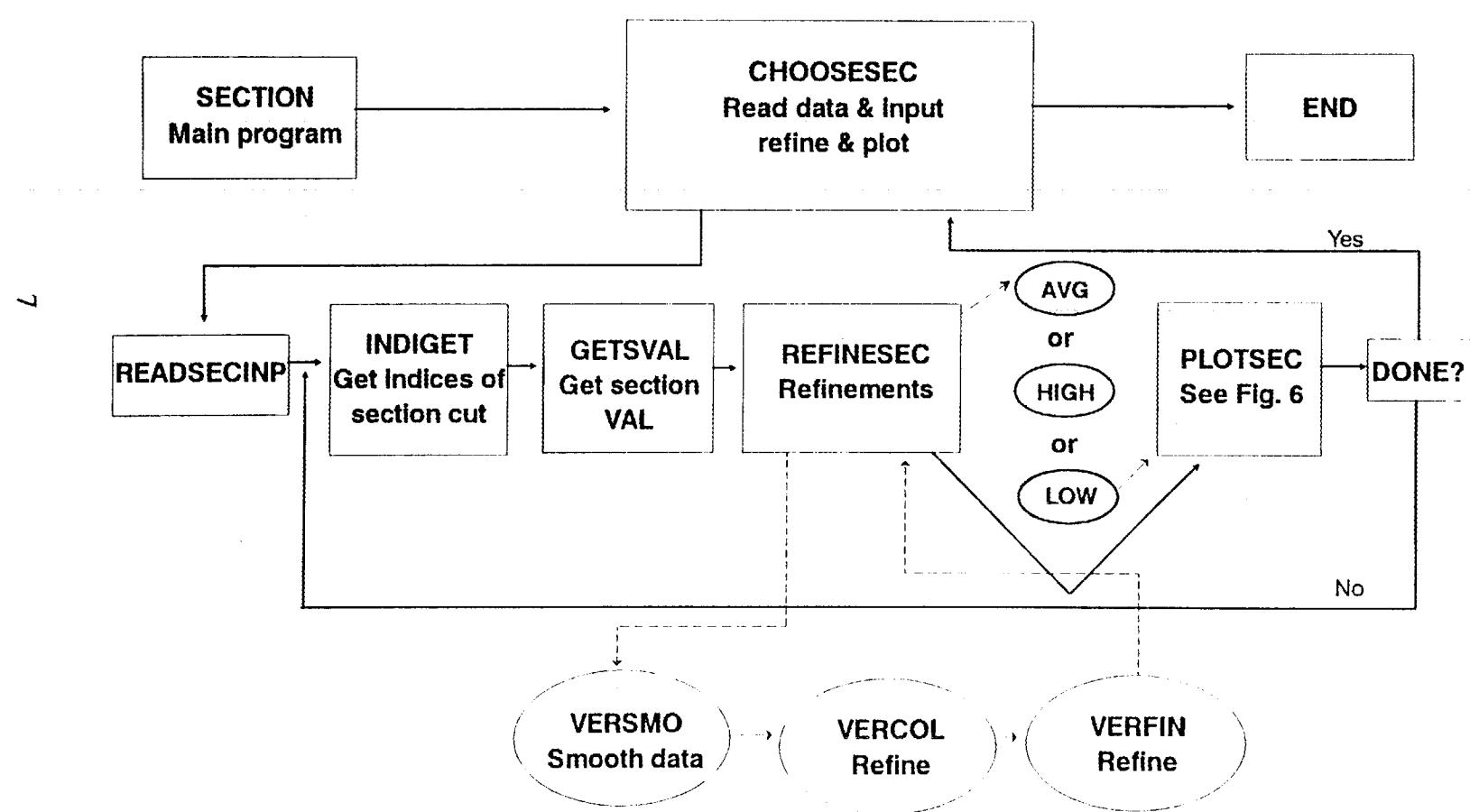


Fig.5. Flow chart for a section plot in the carbon cycle model.

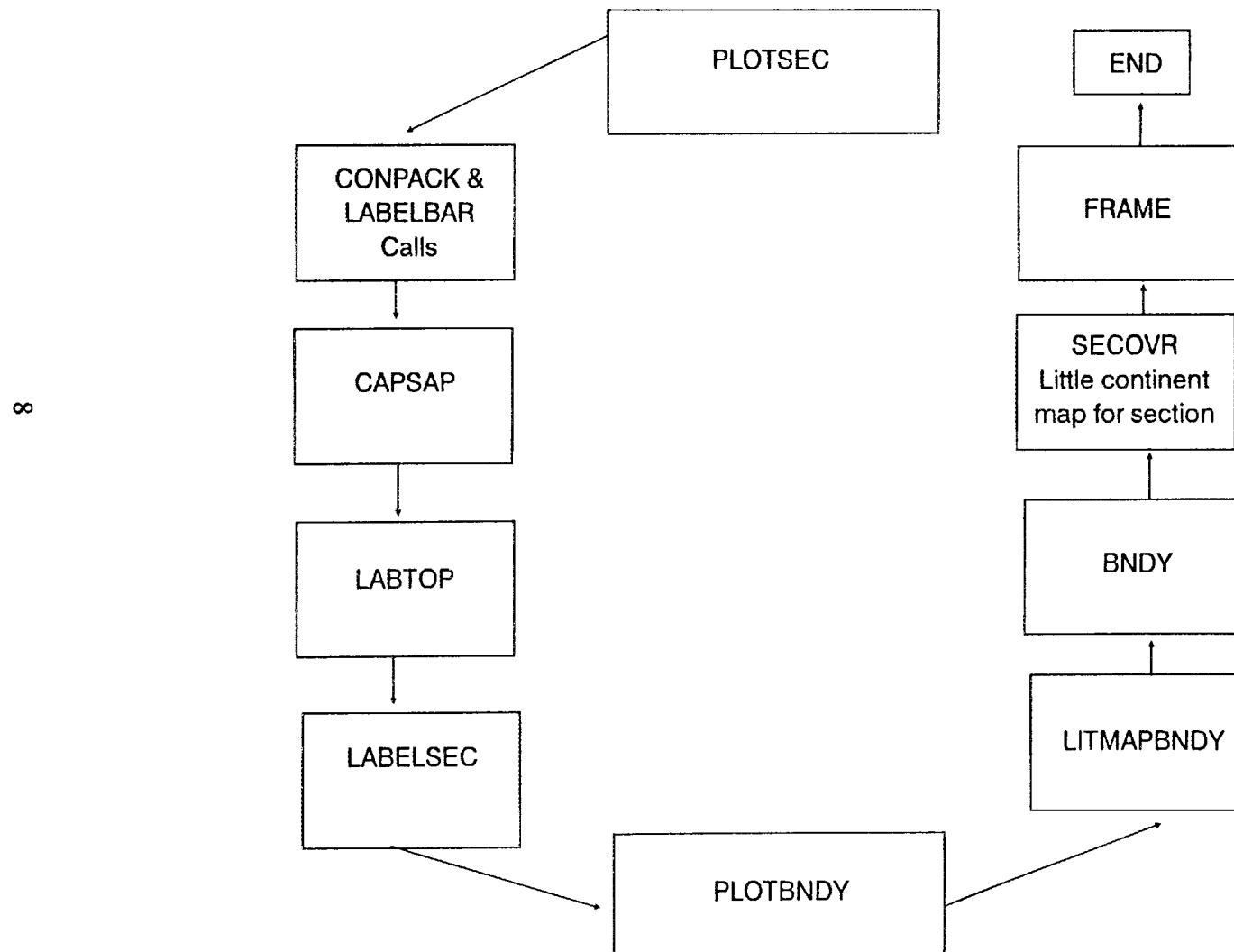


Fig.6. Flow chart for plotting a section.

The second task was to create a large-scale geostrophic (LSG) post-processor much the same as the carbon cycle post-processor (see Fig. 7). This goal was made much easier because the carbon cycle and LSG post-processors have two plots in common, the map and section plots. As previously mentioned, the LSG post-processor is capable of generating map and section plots, but it also generates a horizontal velocity plot, a meridional circulation plot and a zonally integrated flux plots. The map and section plots in the carbon cycle post-processor were reused as much as possible. The only modifications needed to these two plots were how the data were read and the point where the data were read; thus these two plot types will not be discussed again, but the code can be found in the Appendix of this paper. The secondary tasks were to create codes for a horizontal velocity plot, a meridional circulation plot, and zonally integrated flux plots and then a script to integrate everything.

The horizontal velocity plot is a basic map plot showing speeds (via color) of certain model velocities with an arrow overlay to show the direction of the velocity. This task was accomplished by using the map code and adding the velocity arrow plot (see Fig. 8). In order to do this the data arrays were saved in separate arrays and then sent to the plot package VELVCT. VELVCT takes the X and Y two-dimensional data arrays and looks at them in order to decide which direction the data are to point. The package takes these data and scales the arrows according to the velocity at that point. This effect was not exactly the one desired. The desired effect was achieved by a manipulation of the data. The data were manipulated so that when the velocity was calculated by the package it always equaled 1. In doing this the direction of the arrow was preserved, but the magnitude of every arrow was the same. All arrows with a magnitude less than 0.2 were omitted from the plot.

The meridional circulation plot (see Fig. 9) is very similar to the section plots used before. The difference between them lies entirely in the fact that the contours of this section are not color filled, they are simply line contours in which positive contours are red, negative contours are green and zero contours are blue. This plot was achieved by changing the section plotting code so that the ARSCAM subroutine, which color fills the contours, was removed and replaced by the simple contour part of CONPACK. Everything else in the meridional circulation is the same as the section plots used in both models. The zonally integrated flux plot is used for heat transport, salt transport, integrated fresh water flux, and integrated heat flux. Each of these plots shows the concentration of these tracers in the Atlantic ocean, the Pacific ocean, the Indian ocean and a global average. These plots use the plot package AUTOGRAPH (subroutine EZY) which takes a one-dimensional array and plots it using the number of positions in the array as the X dimension.

The last thing done to integrate the large-scale geostrophic post-processor was to write a csh-script that would run the program. This task was much easier to integrate than the carbon cycle post-processor because this post-processor begins after the Hamburg model has already created the data file. This data file has only the cuts of the data that are needed for the plots, and therefore no manipulation is needed to get the correct data. After this, all that had to be done in the csh-script was to compile and build the program and run it. The compile and build process is done with a makefile for the same reasons as the carbon cycle post-processor.

The third task was to create a portable plotting code in anticipation of the use of a different climate package. This step required an independent plotting code for each of the plots. The map, section, velocity, and meridional section plots all contain their own plotting code because of the slight appearance differences between them. They all use a generic two-dimensional array, where array size may be varied by a change in the relevant parameters. There was no need to have a generic quicklook line graph plotting code for the carbon cycle post-processor or a generic line graph plotting code for the LSG post-processor, since these plots are highly Hamburg-specific and easily replicated with the addition of a few lines of NCAR AUTOGRAPH routine calls.

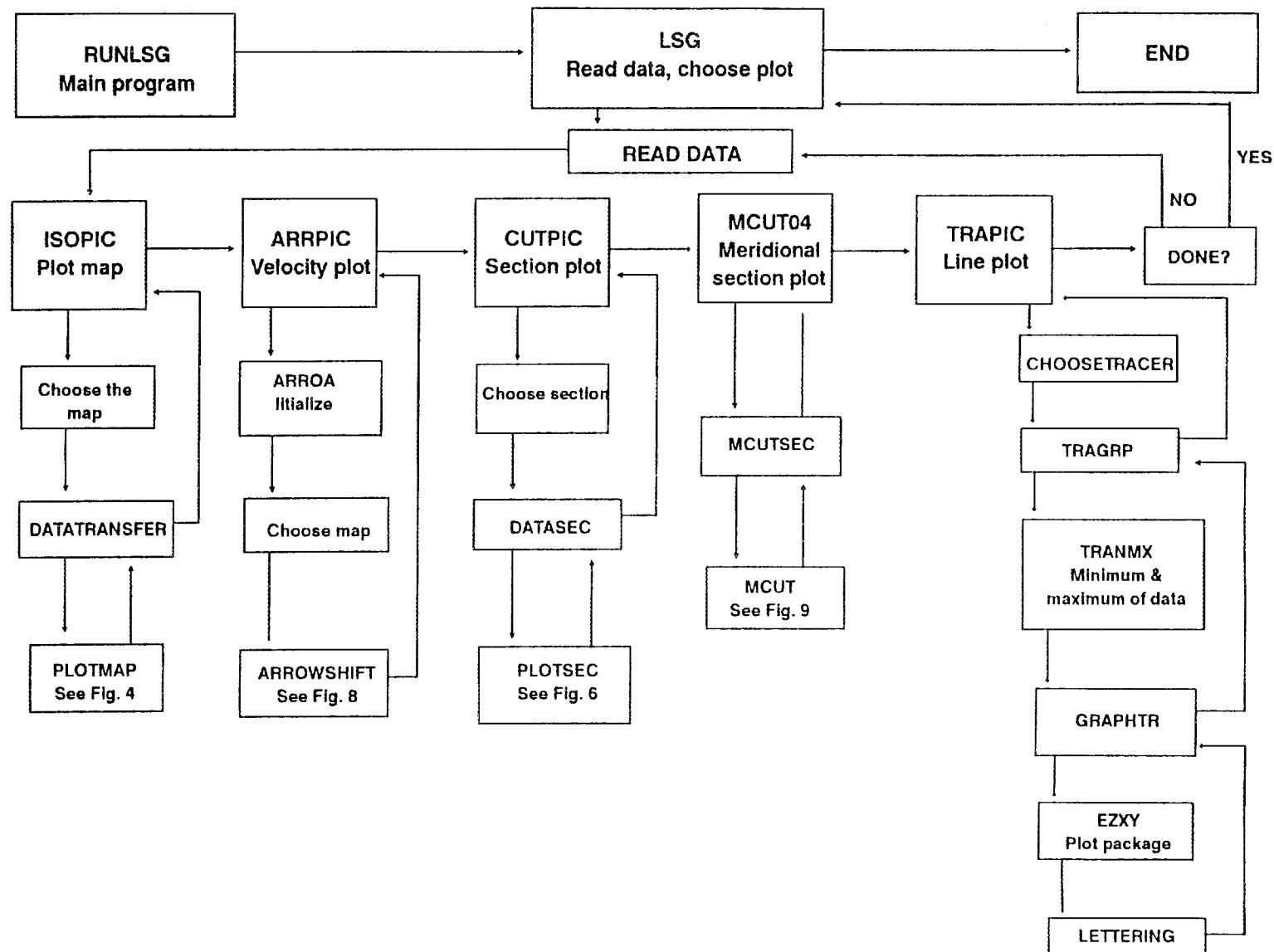


Fig. 7. Overview of LSG post-processor.

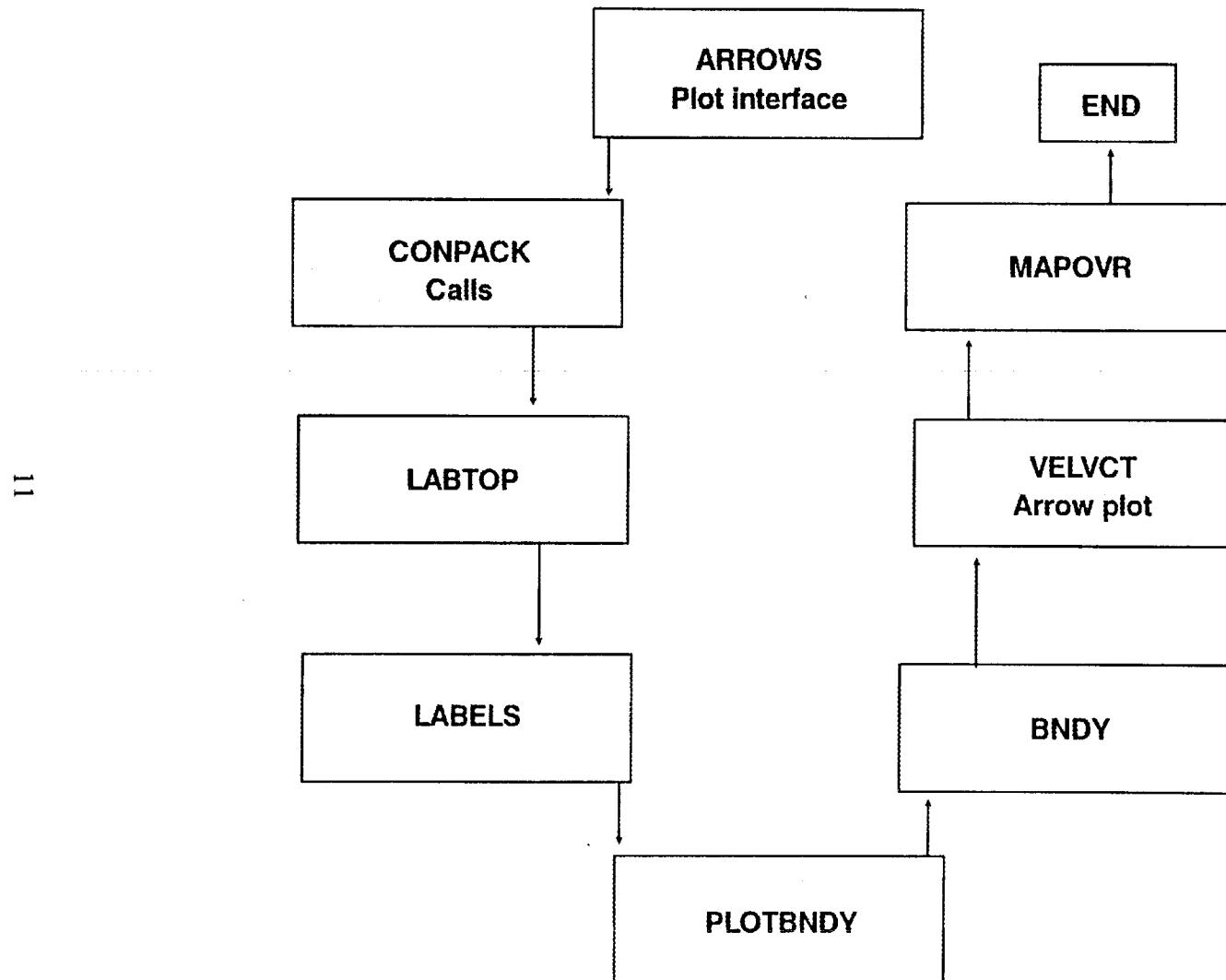


Fig. 8. Plot interface for arrow plots.

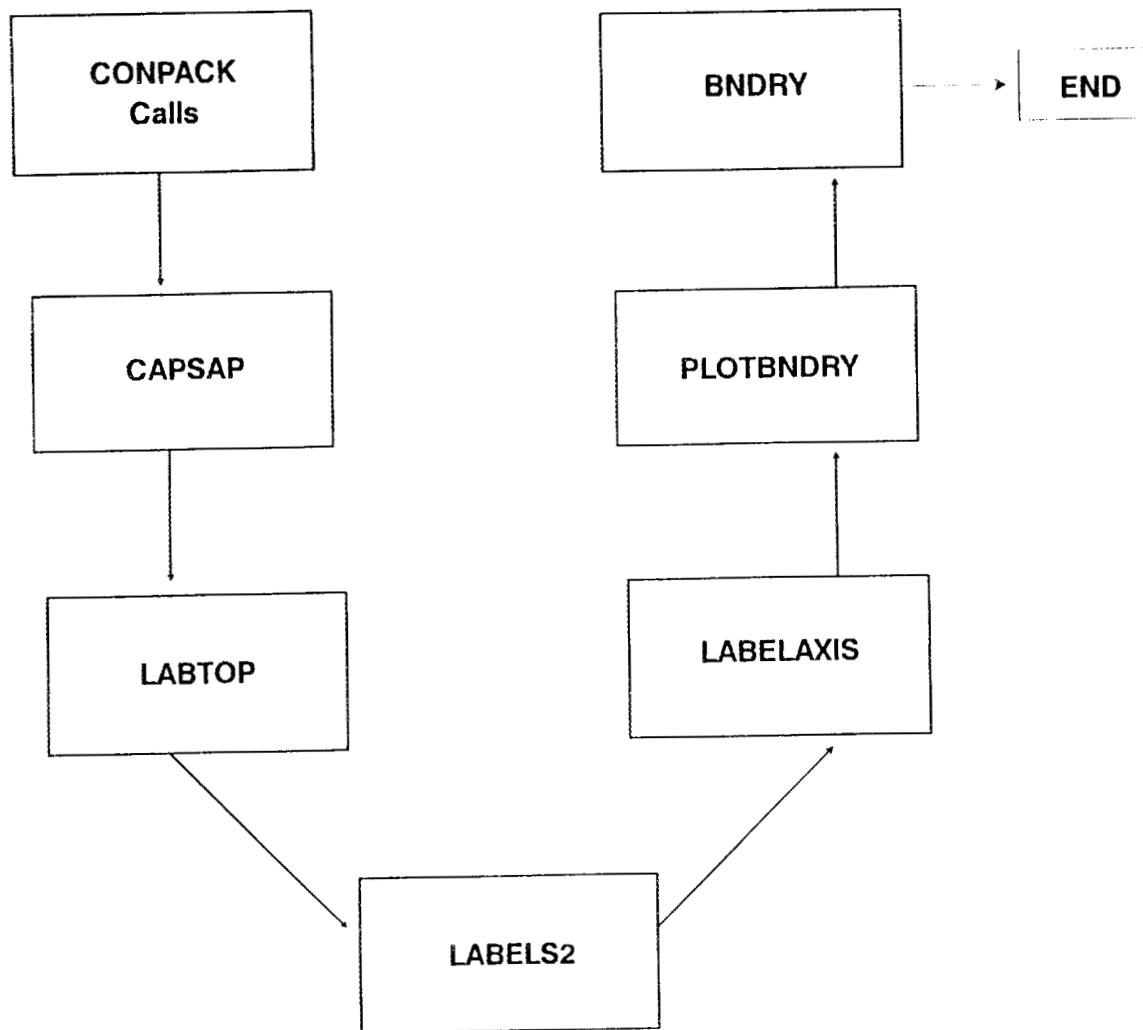


Fig. 9. Plot interface for meridional circulation plot.

3. SUMMARY

Figures 10 and 11 show the graph of a map plot and illustrate that the information from the new code and the information from the Hamburg code are equivalent. Figure 10 was generated by the Hamburg model, and Fig. 11 was generated by the new code. The most significant improvement with the new post-processor is a reduction in the maintenance required vis-a-vis the Hamburg post-processors. This reduction was achieved by the modernization of the NCAR code and a significant reduction in the size of the code, from more than 50,000 lines to approximately 7,500 lines.

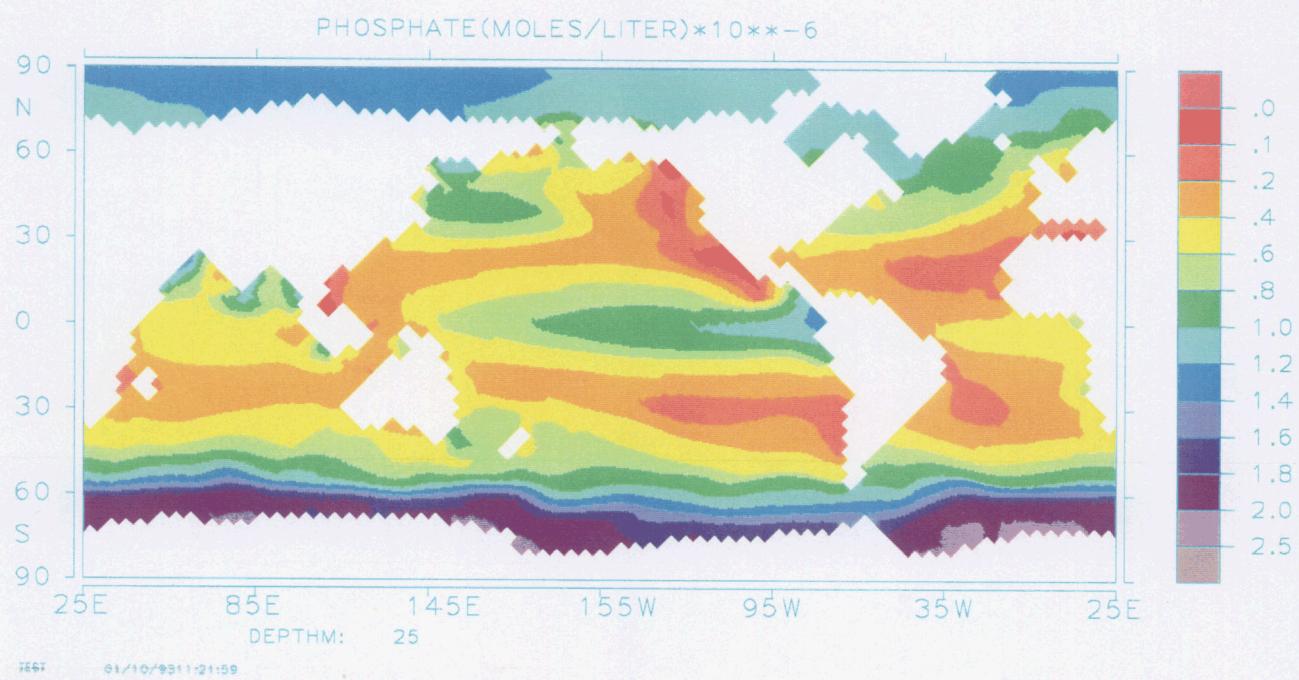


Fig. 10. Map plot from the Hamburg carbon cycle post-processor.

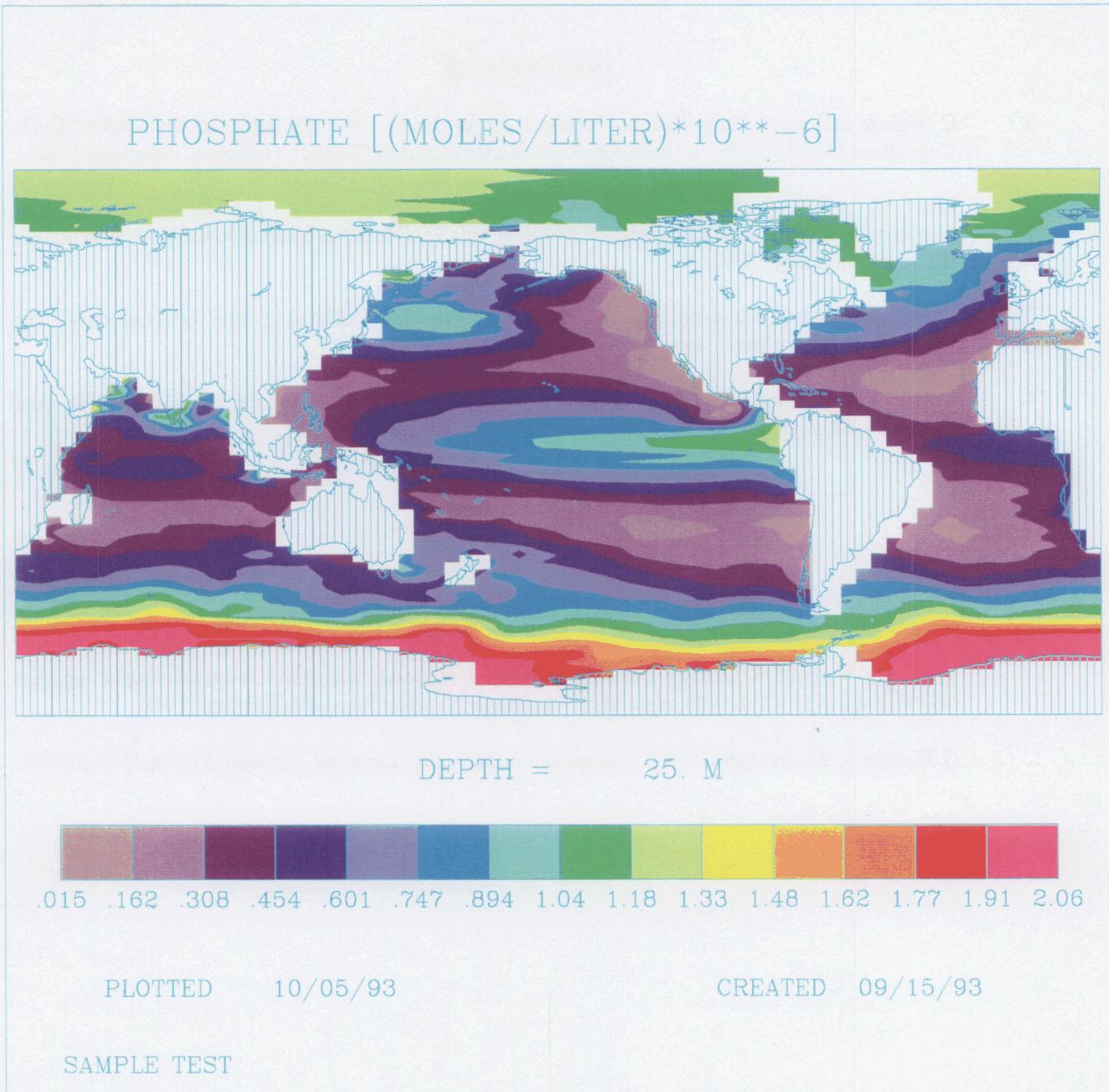


Fig. 11. Map plot from the new carbon cycle post-processor.

REFERENCES

1. C. Heinze and Ernst Maier-Reimer, Technical Report No. 5: The Hamburg Oceanic Carbon Cycle Circulation Model (Cycle 1), Deutsches KlimaRechenZentrum (DKRZ), Hamburg, Germany, 1992.
2. Ernst Maier-Reimer and Uwe Mikolajewics, Technical Report No. 2: The Hamburg Large-Scale Geostrophic Ocean General Circulation Model (Cycle 1), Deutsches KlimaRechenZentrum (DKRZ), Hamburg, Germany, 1992.
3. Fred Clare and Dave Kennison, NCAR Graphics Guide to New Utilities, *Version 3.0*, National Center for Atmospheric Research, Boulder, Colorado, 1989.
4. Robert Lackman and Fred Clare, Users Guide for NCAR GKS-0A Graphics, *Version 2.0*, National Center for Atmospheric Research, Boulder, Colorado, 1990.

SELECTED BIBLIOGRAPHY

American National Standards Institute, *American National Standard Programming Language FORTRAN*, ANSI X3.9-1978, New York: American National Standards Institute, 1978.

G. Enderle, K. Kansy, and G. Pfaff, Computer Graphics Programming, Springer-Verlag, Berlin, 1984.

APPENDIX A

Overview of Building and Execution of Post-processor

In building and executing the new model, the tar archive will first need to be extracted. After this has been completed there should be a directory hierarchy, where "postdir" would contain the carbon cycle post-processor files and "lsgmodel" would contain the LSG post-processor files. After this step the programs can be built and executed using the csh-script "carbproc.job" for the carbon cycle post-processor and the "readplodat.job" script for the LSG post-processor. After execution, three GKS meta files will have been created in the carbon cycle directory and one in the LSG directory. The carbon cycle files will be named `med map*.cgm`, `sec*.cgm`, and `quick*.cgm`, where * signifies the years that are being run, and the LSG file will be named `lsg.cgm`. In addition, when executing the new model there are a few data files that must be present. These files and their usage are explained in the respective script files.

FILE	SUBROUTINES
cont.f	COLRAX MAPOVR
cparr.f	ARROWS COLRAT LABELS
cpexcc.f	CAPSAP BNDARY DFCLRS LABTOP PLOTBNDRY LITMAPBNY
cpmap.f	PROGRAM MAPIT
cpmcut.f	DRAWCL LABELAXIS LABELS2 MCUT
cpsec.f	PROGRAM SECTION
datimx.c	
getval.f	BOX GETVAL
gksstate.f	CURRENTATTR SETATTR
ibm_times.f	DATE CLOCK DATMJB
litcont.f	COLRAZ SECOCR
plodin.f	ARROA ARROWSHIFT ARRPIC CUTPIC DATASEC DATATRANSFER ISOPIC MCUT04 MCUTSEC
plofilll.f	CALI10 TRAFER

plotters.f	COLRAM COLRAS LABELAXIS LABELMAP LABELSEC PLOTBNDY PLOTMAP PLOTSEC
quicklook.f	PROGRAM LYPROF MINMAX1
readati.f	AVG1 REFINEDATA
readinput.f	CHOOSEMAP READINP
refine.f	FINE NEXTD NEXTL NEXTR PART REFINE REFINEMENT SMOOTH
refinesec.f	MINMAX REFINESEC VERCOL VERFIN VERSMO
readtest.f	LSG
runlsg.f	PROGRAM RUNLSG
section.f	AVG CHOOSESEC GETSVAL HIGH INDIGET LOWS READSECINP
trapin.f	GRAPHTR INITRAP LETTERING TRAGRP TRAMNMX TRAPIC

APPENDIX B

Post Subdirectory Files


```
#!/bin/csh
#
# These links will have to be changed to get a different PROFIL or POSTIN or if
# the files in gsm directory are moved.
#
# The data from file POSTIN is used in the map and section plots. The open
# and format of this file can be found in plofilit.f. This file uses POSTIN
# to create the three-dimensional files that are used by the map and section
# data. For additional information refer to the Hamburg Oceanic Carbon
# Cycle Circulation Model (Cycle 1) Technical Report No. 5.
#
# Profil and quick.dat are used by the quicklook plot. The open of these
# data files can be found in the quicklook.f file and the format of the
# data from these files will be found shortly afterwards.
#
#
set number=40001-50000
ln -sf ./gsm/mpiobcm/output1/ca_nohuman/$number/PROFIL PROFIL
ln -sf ./gsm/mpiobcm/output1/ca_nohuman/$number/POSTIN POSTIN
ln -sf ./gsm/mpiobcm/post/ca/lyprof.dat quick.dat
#
# Reset the number if the above directories 40001-50000 are changed.
#
make plofilitrun
make carbonrun
make secrun
make quickrun
#
plofilitrun
#
#
`rm' mapinput
cat > mapinput<< EOF1
SAMPLE TEST
      LAYER(S) TO BE PLOTTED: YES=1, NO=0
          25 M LAYER 1: 1
          75 M LAYER 2: 0
         150 M LAYER 3: 0
         250 M LAYER 4: 0
         450 M LAYER 5: 0
         700 M LAYER 6: 0
        1000 M LAYER 7: 0
        2000 M LAYER 8: 1
        3000 M LAYER 9: 0
        4000 M LAYER10: 0
        5000 M LAYER11: 0
      PARAMETER: YES=1, NO=0
CNAME( 1)='TOTAL CO2 [(MOLES/LITER)*10**-6]' :: 0
CNAME( 2)='ALKALINITY [(EQUIVALENTS/LITER)*10**-6]' :: 0
CNAME( 3)='PHOSPHATE [(MOLES/LITER)*10**-6]' :: 1
CNAME( 4)='DISSOLVED OXYGEN [(MOLES/LITER)*10**-6]' :: 0
CNAME( 5)='POC [(MOLES/LITER)*10**-6]' :: 0
CNAME( 6)='CALCITE [(MOLES/LITER)*10**-6]' :: 0
CNAME( 7)='DELTA 13C' :: 1
CNAME( 8)='DELTA 14C' :: 1
CNAME( 9)='POC DELTA 13C' :: 0
CNAME(10)='POC DELTA 14C' :: 0
CNAME(11)='CALCITE DELTA 13C' :: 0
CNAME(12)='CALCITE DELTA 14C' :: 0
CNAME(13)='DEGREE OF [CO3--] SATURATION (PERCENT)' :: 0
CNAME(14)='PH-VALUE' :: 0
CNAME(15)='[CO3--] [(MOLES/LITER)*10**-6]' :: 0
CNAME(16)='SOLUBILITY PRODUCT' :: 0
CNAME(17)='PREFORMED PO4 [(MOLES/LITER)*10**-6]' :: 0
CNAME(18)='TEMPERATURE (DEG C)' :: 1
CNAME(19)='SALINITY (PSS78)' :: 0
CNAME(20)='SIO4 [(MOLES/LITER)*10**-6]' :: 0
EOF1
#
#
secrun
#
`mv' qmeta map$number.cgm
#
#
`rm' secinput
cat > secinput << EOF1
SAMPLE TEST
      LAYER(S) TO BE PLOTTED: YES=1, NO=0
          25 M LAYER 1: 1
          75 M LAYER 2: 0
         150 M LAYER 3: 0
         250 M LAYER 4: 0
         450 M LAYER 5: 0
         700 M LAYER 6: 0
        1000 M LAYER 7: 0
        2000 M LAYER 8: 1
        3000 M LAYER 9: 0
        4000 M LAYER10: 0
        5000 M LAYER11: 0
      PARAMETER: YES=1, NO=0
CNAME( 1)='TOTAL CO2 [(MOLES/LITER)*10**-6]' :: 1
CNAME( 2)='ALKALINITY [(EQUIVALENTS/LITER)*10**-6]' :: 0
CNAME( 3)='PHOSPHATE [(MOLES/LITER)*10**-6]' :: 1
CNAME( 4)='DISSOLVED OXYGEN [(MOLES/LITER)*10**-6]' :: 0
CNAME( 5)='POC [(MOLES/LITER)*10**-6]' :: 0
CNAME( 6)='CALCITE [(MOLES/LITER)*10**-6]' :: 0
CNAME( 7)='DELTA 13C' :: 1
CNAME( 8)='DELTA 14C' :: 1
CNAME( 9)='POC DELTA 13C' :: 0
CNAME(10)='POC DELTA 14C' :: 0
CNAME(11)='CALCITE DELTA 13C' :: 0
CNAME(12)='CALCITE DELTA 14C' :: 0
CNAME(13)='DEGREE OF [CO3--] SATURATION (PERCENT)' :: 0
CNAME(14)='PH-VALUE' :: 0
CNAME(15)='[CO3--] [(MOLES/LITER)*10**-6]' :: 0
CNAME(16)='SOLUBILITY PRODUCT' :: 0
CNAME(17)='PREFORMED PO4 [(MOLES/LITER)*10**-6]' :: 0
CNAME(18)='TEMPERATURE (DEG C)' :: 1
CNAME(19)='SALINITY (PSS78)' :: 0
CNAME(20)='SIO4 [(MOLES/LITER)*10**-6]' :: 0
EOF1
#
#
quickrun
`mv' qmeta quick$number.cgm
exit
```

carbproc.job

```
CNAME(19)='SALINITY (PSS78)' :: 0
CNAME(20)='DISSOLVED GASEOUS CO2 (SURFACE)' :: 0
CNAME(21)='OCEAN-ATMOSPHERE DIFFERENCE IN PCO2' :: 0
CNAME(22)='PRIMARY PRODUCTION (G/(M**2 MONTH))' :: 0
CNAME(23)='CALCITE PRODUCTION (G/(M**2 MONTH))' :: 0
CNAME(24)='ORG. C SEDIMENT POOL CONTENT (MOLE/M**2)' :: 0
CNAME(25)='CACO3 SEDIMENT POOL CONTENT (MOLE/M**2)' :: 0
CNAME(26)='SIO4 [(MOLES/LITER)*10**-6]' :: 0
```

EOF1

#

carbonrun

#

'mv' qmeta map\$number.cgm

#

'rm' secinput

cat > secinput << EOF1

SAMPLE TEST

TYPE OF CROSS SECTION: YES=1, NO=0
 ATLANTIC1: 1 (WESTERN ATLANTIC)
 ATLANTIC2: 0 (EASTERN ATLANTIC)
 PACIFIC1: 1 (WESTERN PACIFIC)
 PACIFIC2: 0 (EASTERN PACIFIC)
 PARAMETER: YES=1, NO=0

```
CNAME( 1)='TOTAL CO2 [(MOLES/LITER)*10**-6]' :: 1
CNAME( 2)='ALKALINITY [(EQUIVALENTS/LITER)*10**-6]' :: 0
CNAME( 3)='PHOSPHATE [(MOLES/LITER)*10**-6]' :: 1
CNAME( 4)='DISSOLVED OXYGEN [(MOLES/LITER)*10**-6]' :: 0
CNAME( 5)='POC [(MOLES/LITER)*10**-6]' :: 0
CNAME( 6)='CALCITE [(MOLES/LITER)*10**-6]' :: 0
CNAME( 7)='DELTA 13C' :: 1
CNAME( 8)='DELTA 14C' :: 1
CNAME( 9)='POC DELTA 13C' :: 0
CNAME(10)='POC DELTA 14C' :: 0
CNAME(11)='CALCITE DELTA 13C' :: 0
CNAME(12)='CALCITE DELTA 14C' :: 0
CNAME(13)='DEGREE OF [CO3--] SATURATION (PERCENT)' :: 0
CNAME(14)='PH-VALUE' :: 0
CNAME(15)='[CO3--] [(MOLES/LITER)*10**-6]' :: 0
CNAME(16)='SOLUBILITY PRODUCT' :: 0
CNAME(17)='PREFORMED PO4 [(MOLES/LITER)*10**-6]' :: 0
CNAME(18)='TEMPERATURE (DEG C)' :: 1
CNAME(19)='SALINITY (PSS78)' :: 0
CNAME(20)='SIO4 [(MOLES/LITER)*10**-6]' :: 0
```

EOF1

#

secrun

#

'mv' qmeta sec\$number.cgm

#

quickrun

'mv' qmeta quick\$number.cgm

exit

cont.f

```
SUBROUTINE MAPOVR(1TEXTCLR)
C THIS SUBROUTINE IS USED TO PLACE THE SMALL CONTINENTAL MAP ON THE
C PLOT OF THE SECTION. IT SERVES THE PURPOSE OF TELLING THE USER
C WHERE THE PARTICULAR SECTION THEY ARE LOOKING AT IS LOCATED IN THE
C OCEANS.
C
C EXTERNAL SUBROUTINES : DECLRS
C CURRENTATR
C SETATR
C ALL THE APPROPRIATE NCAR CALLS
C
C INCLUDE 'gksatr.txt'
C DIMENSION IAM(250000)
C DIMENSION XCS(10000),YCS(10000),IPAT(16)
C DIMENSION IAI(10),IAG(10)
C DIMENSION IF(13)
C CHARACTER*1 GLAB
C DATA IPAT/1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1/
C
C EXTERNAL COLRAX
C
C DEFINE COLOR TABLE AND RETRIEVE THE CURRENT ATTRIBUTES.
C
C CALL DECLRS
C CALL CURRENTATR
C
C CALL GSASF(IF,E,IF)
C IF(11) = 1
C IF(12) = 1
C CALL GSASF(IF)
C CALL GSFAIS()
C CALL MAPPOS(.01,.99,.35,.85)
C
C SET UP THE MAP BUT DON'T DRAW ANYTHING
C
C THE PARAMETERS ARE SET SO THAT THE MAP IS TO ONLY DO CONTINENTS IN THE
C CIRCULAR EQUIDISTANT METHOD. THIS METHOD ALLOWS EZMAPA TO BE ALTERED
C ALONG THE X AND THE Y AXIS TO SUPPORT DIFFERENT ASPECT RATIOS.
C
C CALL MAPSTC('OU','CO')
C CALL MAPROJ('CE',0.,205.,0.)
C CALL MAPSET('MA',0.,0.,0.,0.)
C CALL MAPSTI('VS',150)
C CALL MAPSTI('G1',1)
C CALL MAPSTI('G2',2)
C CALL MAPINT
C CALL ARINAM(IAM,250000)
C CALL MAPRLA(IAM)
C CALL ARPRAM(IAM,0,0,0)
C
C COMPUTE AND PRINT OUT AMOUNT OF SPACE USED IN THE AREA MAP
C
C ISU=250000*(IAM(6)-IAM(5)-1)
C WRITE(*,*) '
C WRITE(*,*) '
C WRITE(*,*) 'BEGINNING OF CONTIDENT DRAWING.'
C WRITE(*,*) 'SPACE USED IN AREA MAP IS ',ISU
C
C ARSCAM IS USED TO SCAN THE MAP AND PLOT THE COLORED AREAS. IT USES
C A USER DEFINED PROCEDURE COLRAX THAT WILL DETERMINE WHICH APFAS ARE
C TO BE COLOR FILLED, HOW THEY ARE FILLED, ETC
C
C CALL ARSCAM(IAM,XCS,YCS,10000,IAI,IAG,10,COLRAX)
C CALL SETATR
C CALL GSPLCI(1)
C CALL GCOLWR(1,0)
C
C RETURN
C END
C
C----- SUBROUTINE COLRAX(XCS,YCS,NCS,IAI,IAG,NAI)
C----- DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
C
C THIS SUBROUTINE IS A USER DEFINED SUBROUTINE THAT IS CALLED BY THE
C NCAR UTILITY ARSCAM. THIS PARTICULAR ROUTINE IS SET UP TO COLOR
C FILL ONLY CONTINENTS BECAUSE THIS MAP IS USED AS AN OVERLAY ONTO THE
C CONTOUR PLOTS. THE CONTINENTS ARE SET TO BE FILLED IN WITH CYAN
C AT THE PRESENT TIME BUT CAN BE CHANGED AT ANY TIME BY SIMPLY
C CHANGING THE PARAMETER ICNCLR IN 'litcon.txt'.
C
C CALL GSFAIS()
C CALL GSFACT(1)
C IF (IAI(1).GE.0.AND.IAI(2).GE.0) THEN
C   STM=MAX0(IAI(1),IAI(2))
C   IF (STM.GT.0) THEN
C     IF (NCS.GT.150) PRINT *, 'COLRAM - NCS TOO BIG',NCS
C     CALL GSFAIS()
C     ICI=MAPACI(1,1)
C     IF (ICL.NE.1) THEN
C       CALL GFA(HCS,1,XCS,YCS)
C     ENDIF
C   ENDIF
C ENDIF
C RETURN
C END
```

cpexec.f

```

C SUBROUTINE CAPSAP (LBL1,TIME,IAMA,LAMA,LAYER)
C
C      DIMENSION IAMA(*)
C
C      CHARACTER(*) LABL
C      CHARACTER CD*8, CT*10
C
C Compute and print the time required to draw the contour plot and how
C much space was used in the various arrays.
C
C      TIME=SECOND(DUMI)-TIME
C      CALL DATE(CD)
C      CALL CLOCK(CT)
C      PRINT *, ' '
C      PRINT *, ' '
C      PRINT *, ' PLOT TITLE WAS ',LBL
C      PRINT *, ' LAYER OF PLOT -- ',LAYER
C      PRINT *, ' TIME TO DRAW PLOT WAS ',TIME
C      PRINT *, ' DATE OF PLOTTED MAP -- ',CD
C      PRINT *, ' TIME OF PLOTTED MAP -- ',CT
C      CALL CGPGETI ('IWU - INTEGER WORKSPACE USAGE',IIWU)
C      CALL CGPGETI ('IRWU - REAL WORKSPACE USAGE',IRWU)
C      PRINT *, ' INTEGER WORKSPACE USED ',IIWU
C      PRINT *, ' REAL WORKSPACE USED ',IRWU
C      IF (LAMA.NE.0) THEN
C          IAMU=LAMA-(IAMA(6)-IAMA(5)-1)
C          PRINT *, ' AREA MAP SPACE USED ',IAMU
C      END IF
C
C Done.
C
C      RETURN
C
C      END
C-----+
C SUBROUTINE LABTOP (LBL1,SIZE)
C
C      CHARACTER(*) LABL
C
C Put a label just above the top of the plot. The SET call is re-done
C to allow for the use of fractional coordinates, and the text extent
C capabilities of the package PLOTCHAR are used to determine the label
C position.
C
C SET AND GETSET ARE SPPS CALLS THAT MAY NEED TO BE CHANGED AT A LATER DATE.
C
C      CALL GETSET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C      SZFS=SIZE*(XVPR-XVPL)
C      CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
C      CALL PCGETI ('QU - QUALITY FLAG',IQUA)
C      CALL PCSFTI ('QU - QUALITY FLAG',0)
C      CALL PCSFTI ('TE - TEXT EXTENT COMPUTATION FLAG',1)
C      CALL PLCHHQ (.5,.5,LBL1,SZFS,160,0.)
C      CALL PCGETR ('DR - DISTANCE TO BOTTOM OF STRING',DRBG)
C      CALL PLCHHQ (.5*(XVPL+XVPR),YVPT+SZFS,DBOS,LBL1,SZFS,0.,0.)
C      CALL PCSFTI ('QU - QUALITY FLAG',IQUA)
C      CALL SET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C
C Done.
C
C      RETURN
C
C      END
C-----+
C SUBROUTINE BNDARY

```

```

C Draw a line showing where the edge of the plotter frame is.
C
C PLOTIF IS A SPPS CALL AND MAY NEED TO BE CHANGED AT A LATER DATE.
C
        CALL GSPLCI (1)
        CALL PLOTIF (0.,0.,0)
        CALL PLOTIF (1.,0.,1)
        CALL PLOTIF (1.,1.,1)
        CALL PLOTIF (0.,1.,1)
        CALL PLOTIF (0.,0.,1)
        CALL PLOTIF (0.,0.,2)

C Done.
C
        RETURN
C
        END

C-----+
SUBROUTINE PLOTRNDRY(XTOP,XBOT,YLFT,YRGT)
INCLUDE 'litcon.txt'

C THIS SET OF CALLS PLACES A BOX AROUND THE PLOTTED PART IN THE SECTION
C PLOTS.
C
C PLOTIF IS AN SPPS CALL AND MAY NEED TO BE CHANGED AT A LATER DATE.
C
        CALL PLOTIF (YLFT,XTOP,0)
        CALL PLOTIF (YLFT,XBOT,1)
        CALL PLOTIF (YRGT,XBOT,1)
        CALL PLOTIF (YRGT,XTOP,1)
        CALL PLOTIF (YLFT,XTOP,1)
        CALL PLOTIF (YLFT,XTOP,2)

C
        RETURN
        END

C-----+
SUBROUTINE LITMAPBNY
INCLUDE 'litcon.txt'

C THIS SET OF CALLS PLACES A BOX AROUND THE LITTLE CONTINENT MAP IN THE
C SECTION PLOTS.
C
C PLOTIF IS A SPPS CALL AND MAY NEED TO BE CHANGED AT A LATER DATE.
C
        CALL PLOTIF (VRGHT,VTOP,0)
        CALL PLOTIF (VRGHT,VBOT,1)
        CALL PLOTIF (VLFT,VBOT,1)
        CALL PLOTIF (VLFT,VTOP,1)
        CALL PLOTIF (VRGHT,VTOP,1)
        CALL PLOTIF (VRGHT,VTOP,2)

        RETURN
        END

C-----+
SUBROUTINE DFCLRS
C Define a set of RGB color triples for colors 1 through 15.
C
        DIMENSION RGBV(3,15)
C Define the RGB color triples needed below.
C
        DATA RGBV / 0.00 , 1.00 , 1.00 ,

```

cpexcc.f

```
+      0.70 , 0.70 , 0.70 ,
+      0.75 , 0.50 , 1.00 ,
+      0.50 , 0.00 , 1.00 ,
+      0.00 , 0.00 , 1.00 ,
+      0.50 , 0.60 , 1.00 ,
+      0.00 , 1.00 , 1.00 ,
+      0.40 , 1.00 , 0.70 ,
+      0.00 , 1.00 , 0.00 ,
+      0.70 , 1.00 , 0.40 ,
+      1.00 , 1.00 , 0.00 ,
+      1.00 , 0.75 , 0.00 ,
+      1.00 , 0.38 , 0.38 ,
+      1.00 , 0.00 , 0.00 ,
+      1.00 , 0.20 , 1.00 /
```

C Define 16 different color indices, for indices 0 through 15. The
C color corresponding to index 0 is black and the color corresponding
C to index 1 is cyan. There is a sixteenth color index added which is
C white.

C CALL GSCR (1,0,0.,0.,0.)

C DO 101 I=1,15
101 CALL GSCR (1,I,RGBV(1,I),RGBV(2,I),RGBV(3,I))
101 CONTINUE
 CALL GSCR (1,16,1.,1.,1.)

C Done.
C RETURN
C

cpmap.f

```
PROGRAM MAPIT
C
C THIS PROGRAM IS DESIGNED TO TAKE AN ARRAY OF DATA AND THEN SEND IT
C THROUGH A SERIES OF PROCESSES IN WHICH TO PREPARE IT TO BE
C PLOTTED BY NCAR BY THE PLOT PACKAGE CONPACK.
C
C
C Declare required data arrays and workspace arrays.
C
INCLUDE 'data.txt'
C
OPEN GKS
C
CALL GOPKS(6, IDUM)
CALL GOPWK(1, 2, 1)
CALL GACWK(1)
C
C
C Retrieve the data from the subroutine CHOOSEMAP - which will read
C from a file along with all other necessary items and then it will
C call all needed subroutines to finish the plotting of all the maps.
C
CALL GSPLCI(IEXTCLR)
CALL CHOOSEMAP
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCLFS
END
```

cpsec.f

```
PROGRAM SECTION
C
C THIS PROGRAM IS DESIGNED TO TAKE AN ARRAY OF DATA AND THEN SEND IT
C THROUGH A SERIES OF PROCESSES IN WHICH TO PREPARE IT TO BE
C PLOTTED BY NCAR. -- Emily Stevens
C
C
C Declare required data arrays and workspace arrays
C
INCLUDE 'data.txt'
C
C OPEN GKS
C
CALL GOPKS(6,1DUM)
CALL GOPWK(1,2,1)
CALL GACWK(1)
C
C
C Retrieve the data from the subroutine CHOOSESEC - which will read
C from a file along with all other necessary items and then it will
C call all needed subroutines to finish the plotting of all the maps.
C
CALL CHOOSESEC
CALL GDAWK(1)
CALL GCLWK(1)
CALL GCIKS
END
```

data.txt

```

PARAMETER(IE=72, JE=72, KE=11, NMAP = 26, IMAX = 78,
*          JMAX = 78, KMAX = 25, IWA = 10000, IMA = 200000,
*          NV = 12, NS = 10, MSHIFT = 6, RMASK = -.1111E+11,
*          ITEXTCLR = 1, NUMCONT = -13)

C COMMON/MAPS/CNAME(NMAP),IPABL(NMAP),CFIL(NMAP),ILAY(KE),
*          VAL(100),HEADER,TEXT6,IVAL,IFINE,DEPTP(IE,JE),
*          CHOOSEN(IE,JE),TITLE
COMMON/PLOTTERS/RWRK(IWA), IWRK(IWA), IAMA(IMA),
1      XCRA(IWA), YCRA(IWA), IAIA(NS), IGIA(NS)
COMMON/DATA/ARRAY(ZDAT(IE,JE), ZDAT(IE,JE), MHZ(IMAX,JMAX),
1      ZLAYER(0:NV), LAYER, MHZIO(IMAX)
REAL*4 ZDAT,RWRK,XCRA,YCRA,ARRAY,ZLAYER
INTEGER*4 IWRK,IMA,IAIA,IGIA,LAYER
LOGICAL GLOMAP, MHZIO, MHZ
CHARACTER*40 HEADER,CNAME,TITLE
CHARACTER*6 CFIL
CHARACTER*8 TEXT6

C * PARAMETERS *
C IE    --- THE FIRST DIMENSION OF THE ARRAYS TO BE SENT TO CONPACK.
C JE    --- THE SECOND DIMENSION OF THE ARRAYS TO BE SENT TO CONPACK.
C KE    --- THE LAYER DIMENSION OF THOSE ARRAYS BEFORE THEY ARE
C        REDUCED TO TWO DIMENSIONAL ARRAYS.
C NMAP   --- NMAP IS THE NUMBER OF MAPS ALLOWED TO HAVE. THERE SHOULD
C        BE AN EQUAL NUMBER OF FILES OF WHICH THE NAMES ARE STORED
C        IN CFIL. AT THE PRESENT TIME IN THE PROCESSOR FOR THIS
C        FILE THERE ARE 26 ALLOWED CHOICE BUT AT THIS LEVEL ONLY
C        25 ARE ALLOWED TO BE CHOSEN BECAUSE THE SILICON FILE IS
C        NOT PROCESSED IN PLOFIL1.F.
C IMAX   --- MAXIMUM ALLOWABLE FIRST DIMENSION. THIS IS USED FOR
C        ARRAYS IN THE REFINEMENT PROCESS.
C JMAX   --- MAXIMUM ALLOWABLE SECOND DIMENSION. THIS IS USED FOR
C        ARRAYS IN THE REFINEMENT PROCESS.
C KMAX   --- MAXIMUM ALLOWABLE THIRD DIMENSION WHICH CORRESPONDS TO
C        THE LAYER DIMENSION.
C IWA    --- THE DIMENSION OF THE INTEGER WORK ARRAY AND THE REAL
C        WORK ARRAY, USED IN THE NCAR INTERFACE.
C IMA    --- THE DIMENSION OF THE DATA ARRAY THAT IS USED IN THE NCAR
C        INTERFACE.
C NV    --- THE NUMBER OF POSSIBLE DEPTH LAYERS.
C NS    --- DIMENSION OF THE VARIABLES USED BY THE USER DEFINED
C        SUBROUTINE COLRAM IN THE NCAR INTERFACE.
C MSHIFT  --- CONSTANT THAT IS USED FOR THE SMOOTH AND REFINEROUTINES.
C RMASK   --- CONSTANT USED TO GIVE NCAR A SPECIAL VALUE SO THAT IT
C        DOES NOT TRY AND CONTOUR THE CONTINENTS INTO THE DATA.
C ITEXTCLR--- THIS IS THE COLOR OF THE TEXT THAT IS PLACED ON THE MAP
C        AND ALSO THE COLOR OF THE BORDERS.

C * VARIABLES *
C CNAME   --- ARRAY THAT IS USED TO READ THE INPUT FILE THAT IS CREATED
C        BY THE PROCESSOR. IT HOLDS THE HEADER FOR THE PLOT.
C IPABL   --- ARRAY THAT IS USED TO READ THE INPUT FILE THAT IS CREATED
C        BY THE PROCESSOR. IT HOLDS ZERO FOR NOT PLOTTING A
C        PARTICULAR MAP AND ONE IF THAT MAP NEEDS TO BE PLOTTED.
C CFIL    --- ARRAY THAT HOLDS THE NAMES OF THE FILES THAT ARE TO BE
C        USED IN THE PROGRAM.
C ILAY    --- ARRAY THAT IS READ FROM THE MAP INPUT AND HOLDS A ZERO
C        IF A MAP IS NOT TO BE PLOTTED AND ONE IF IT IS.
C VAL     --- ARRAY THAT HOLDS DATA TO BE USED IN A REFINEMENT PROCESS.
C HEADER  --- HOLDS THE SPECIFIC HEADER FOR THE PARTICULAR MAP THAT IS
C        BEING PLOTTED.

C TEXT6   --- HOLDS THE CREATION DATE OF THE DATA FOR WHICH THE MAP IS
C        TO BE PLOTTED.
C IVAL    --- INTEGER VALUE THAT IS USED IN THE REFINEMENT PROCESS.
C IFINE   --- INTEGER VALUE THAT TELLS THE REFINEMENT HOW FINE IT NEEDS
C        TO BE.
C DEPTP   --- ARRAY THAT HOLDS THE DEPTH OF THE DATA. IT IS USED TO
C        DETERMINE IF THE POINT IS ON THE CONTINENTS OR IN THE
C        OCEAN.
C CHOOSEN --- ARRAY THAT HOLDS THE DATA THAT IS TO BE PLOTTED. IT IS
C        A TWO DIMENSIONAL ARRAY THAT IS READ FROM THE PARTICULAR
C        FILE ASKED FOR BY THE USER.
C TITLE   --- THE TITLE OF THIS SERIES OF MAPS
C RWRK   --- REAL WORK ARRAY USED BY THE NCAR SUBROUTINES.
C IWRK   --- INTEGER WORK ARRAY USED BY THE NCAR SUBROUTINES.
C IAMA   --- INTEGER DATA ARRAY USED BY THE NCAR SUBROUTINES.
C XCRA   --- REAL ARRAY USED BY ARSCAM AND SENT TO THE USER DEFINED
C        SUBROUTINE COLRAM TO HOLD THE X COORDINATES OF PART OF
C        THE DATA.
C YCRA   --- REAL ARRAY USED BY ARSCAM AND SENT TO THE USER DEFINED
C        SUBROUTINE COLRAM TO HOLD THE Y COORDINATES OF PART OF
C        THE DATA.
C IAIA   --- USED BY THE USER DEFINED SUBROUTINE COLRAM TO HOLD NAI
C        PAIRS OF IDENTIFIERS FOR THE AREA DEFINED BY XCS AND
C        YCS.
C IAGA   --- USED IN CONJUNCTION IAIA. FOR EACH VALUE OF I FROM 1 TO
C        NAI, IAIA(I) IS THE AREA IDENTIFIER FOR THE AREA WITH
C        RESPECT TO THE GROUP OF EDGES SPECIFIED BY THE GROUP
C        IDENTIFIER(IAGA).
C ARRAY   --- AN ARRAY USED TO HOLD THE DATA WHILE IT IS BEING
C        PROCESSED.
C ZDAT   --- THE DATA ARRAY THAT HOLDS THE FINAL DATA THAT IS SENT
C        TO THE NCAR INTERFACE. (IT IS TWO DIMENSIONAL SINGLE
C        PRECISION)
C MHZ    --- A LOGICAL DATA ARRAY THAT HOLDS WHETHER A POINT IS TRUE
C        OR FALSE DEPENDING ON THE DEPTH.
C ZLAYER  --- HOLDS THE POSSIBLE DEPTH LEVELS TO BE USED BY REFINER
C        AND THE NCAR INTERFACE.
C LAYER   --- HOLDS WHAT LEVEL(LAYER) THE PROGRAM IS RUNNING AT AND
C        USES IT IN THE NCAR INTERFACE.
C MHZIO   --- HOLDS ROW '0' OF THE MHZ ARRAY.

```

data2.txt

```
PARAMETER(IE=72, JE=72, KE=11, NMAP = 26, IMAX = 78,  
*          JMAX = 78, KMAX = 25, IWA = 10000, IMA = 200000,  
*          NV = 12, NS = 10, MSHIFT = 6, IS = 4, ISMAX = 9,  
*          RMASK=-.1111E+11, NUMCONT = -13, NUCONT = 20)  
C  
COMMON/MAPS/SNAME(NMAP),ISPAPL(NMAP),SFIL(NMAP),ILAY(KE),  
*          VALS(100),HEADER,TEXT6,IVAL,IFINE,DEPTP(IE,JE),  
*          CHOOSEN(IE,JE),TITLE,ISECT(IS),SECNAM,INDISO(JE)  
COMMON/PLOTTERS/RWRK(IWA),IWRK(IWA),[AMA(IMA),  
1      XCRA(IWA),YCRA(IWA),IAIA(NS),IGIA(NS)  
COMMON/DATA/ARRAY(IE,JE),ZDAT(JE,KE),MHZ(IMAX,JMAX),  
1      ZLAYER(0:NV), LAYER, MHZIO(IMAX),IISEC(JMAX,ISMAX)  
REAL*4 ZDAT,RWRK,XCPA,YCRA,ARRAY,ZLAYER  
INTEGER*4 IWPK,IMA,IAIA,IGIA,LAYER  
LOGICAL GLOMAP,MHZJO,MHZ  
CHARACTER*40 HEADER,SNAME,TITLE  
CHARACTER*20 SECNAM  
CHARACTER*6 SFIL  
CHARACTER*8 TEXT6
```

datimx.c

```
#include <stdio.h>
#include <time.h>
void datimx(now)           /* Simulate Fortran/VS DATIMX subroutine */
int now[13];
{
    int year,month,day;
    time_t ltime;
    struct tm *newtime;
    time(&ltime);
    newtime=localtime(&ltime);
    now[13]=newtime->tm_year;
    now[12]=newtime->tm_yday;
    now[11]=newtime->tm_wday;

    now[8]=-1;
    now[7]=1900+newtime->tm_year;
    now[6]=1+newtime->tm_mon;
    now[5]=newtime->tm_mday;
    now[4]=newtime->tm_hour;
    now[3]=newtime->tm_min;
    now[2]=newtime->tm_sec;
    now[1]=0;
    now[0]=-1;

    now[9]=now[4]+1;
    if (now[9] > 12) now[9]=now[9]-1;
    now[10]=1;
    if (now[9] == 12) now[10]=2;

    /* printf("Year : %d \n",year);      */
    /* printf("Month : %d \n",month);    */
    /* printf("Day : %d \n",newtime->tm_yday);      */
}
```

getval.f

```

SUBROUTINE GETVAL
C THIS SUBROUTINE IS USED IN THE MAPPING SECTION OF THE PROCESSOR AND
C IS USED TO GET THE NECESSARY VALUES FOR THE REFINEMENT PROCESS OF THE
C PACKAGE.
C INCLUDE 'data.txt'
DIMENSION VALX(30,100)

C
VAL(1) = 0.
VAL(2) = 5.
VAL(3) = 10.
VAL(4) = 15.
VAL(5) = 20.
VAL(6) = 25.
VAL(7) = 30.
VAL(8) = 40.
VAL(9) = 50.
VAL(10)= 60.
VAL(11)= 70.
VAL(12)= 80.
VAL(13)= 90.
VAL(14)= 100.

C *** ZUWEISUNG DER VAL-FELDER F. VERSCH. PARAMETER - BEGINN
IF(K.EQ.1)THEN
VALX( 1,1) = 1725.
VALX( 1,2) = 1750.
VALX( 1,3) = 1800.
VALX( 1,4) = 1825.
VALX( 1,5) = 1850.
VALX( 1,6) = 1875.
VALX( 1,7) = 1900.
VALX( 1,8) = 1925.
VALX( 1,9) = 1950.
VALX( 1,10)= 1975.
VALX( 1,11)= 2000.
VALX( 1,12)= 2050.
VALX( 1,13)= 2100.
VALX( 1,14)= 2150.
ELSE
VALX( 1,1) = 1950.
VALX( 1,2) = 2000.
VALX( 1,3) = 2050.
VALX( 1,4) = 2100.
VALX( 1,5) = 2150.
VALX( 1,6) = 2200.
VALX( 1,7) = 2250.
VALX( 1,8) = 2300.
VALX( 1,9) = 2350.
VALX( 1,10)= 2400.
VALX( 1,11)= 2450.
VALX( 1,12)= 2500.
VALX( 1,13)= 2550.
VALX( 1,14)= 2600.
ENDIF
IF(K.EQ.1)THEN
VALX( 2,1) = 2140.
VALX( 2,2) = 2160.
VALX( 2,3) = 2180.
VALX( 2,4) = 2200.
VALX( 2,5) = 2220.
VALX( 2,6) = 2240.
VALX( 2,7) = 2260.
VALX( 2,8) = 2280.
VALX( 2,9) = 2300.
VALX( 2,10)= 2320.
VALX( 2,11)= 2340.
VALX( 2,12)= 2360.
VALX( 2,13)= 2380.
VALX( 2,14)= 2400.
ELSE
VALX( 2,1) = 2150.
VALX( 2,2) = 2200.
VALX( 2,3) = 2250.
VALX( 2,4) = 2300.
VALX( 2,5) = 2325.
VALX( 2,6) = 2350.
VALX( 2,7) = 2375.
VALX( 2,8) = 2400.
VALX( 2,9) = 2425.
VALX( 2,10)= 2450.
VALX( 2,11)= 2475.
VALX( 2,12)= 2500.
VALX( 2,13)= 2550.
VALX( 2,14)= 2600.
ENDIF
IF(K.EQ.1)THEN
VALX( 3,1) = 0.
VALX( 3,2) = 0.1.
VALX( 3,3) = 0.2.
VALX( 3,4) = 0.4.
VALX( 3,5) = 0.6.
VALX( 3,6) = 0.8.
VALX( 3,7) = 1.0.
VALX( 3,8) = 1.2.
VALX( 3,9) = 1.4.
VALX( 3,10)= 1.6.
VALX( 3,11)= 1.8.
VALX( 3,12)= 2.0.
VALX( 3,13)= 2.5.
VALX( 3,14)= 3.
ELSE
VALX( 3,1) = 0.75.
VALX( 3,2) = 1.0.
VALX( 3,3) = 1.25.
VALX( 3,4) = 1.5.
VALX( 3,5) = 1.75.
VALX( 3,6) = 2.0.
VALX( 3,7) = 2.25.
VALX( 3,8) = 2.5.
VALX( 3,9) = 2.75.
VALX( 3,10)= 3.0.
VALX( 3,11)= 3.25.
VALX( 3,12)= 3.5.
VALX( 3,13)= 3.75.
VALX( 3,14)= 4.
ENDIF
IF(K.EQ.1)THEN
VALX( 4,1) = 160.
VALX( 4,2) = 180.
VALX( 4,3) = 200.
VALX( 4,4) = 220.
VALX( 4,5) = 240.
VALX( 4,6) = 260.
VALX( 4,7) = 280.
VALX( 4,8) = 300.
VALX( 4,9) = 320.
VALX( 4,10)= 340.
VALX( 4,11)= 360.

```

getval.f

```

VALX( 4,12)= 380.
VALX( 4,13)= 400.
VALX( 4,14)= 420.
ELSE
VALX( 4,1) = 50.
VALX( 4,2) = 75.
VALX( 4,3) = 100.
VALX( 4,4) = 125.
VALX( 4,5) = 150.
VALX( 4,6) = 175.
VALX( 4,7) = 200.
VALX( 4,8) = 225.
VALX( 4,9) = 250.
VALX( 4,10)= 275.
VALX( 4,11)= 300.
VALX( 4,12)= 325.
VALX( 4,13)= 350.
VALX( 4,14)= 375.
ENDIF
IF(K.EQ.1)THEN
VALX( 5,1) = 0.
VALX( 5,2) = 0.1
VALX( 5,3) = 0.5
VALX( 5,4) = 1.
VALX( 5,5) = 2.
VALX( 5,6) = 4.
VALX( 5,7) = 6.
VALX( 5,8) = 8.
VALX( 5,9) = 10.
VALX( 5,10)= 12.5
VALX( 5,11)= 15.
VALX( 5,12)= 20.
VALX( 5,13)= 25.
VALX( 5,14)= 30.
ELSE
VALX( 5,1) = 0.
VALX( 5,2) = 0.1
VALX( 5,3) = 0.5
VALX( 5,4) = 1.
VALX( 5,5) = 2.
VALX( 5,6) = 4.
VALX( 5,7) = 6.
VALX( 5,8) = 8.
VALX( 5,9) = 10.
VALX( 5,10)= 12.5
VALX( 5,11)= 15.
VALX( 5,12)= 20.
VALX( 5,13)= 25.
VALX( 5,14)= 30.
ENDIF
IF(K.EQ.1)THEN
VALX( 6,1) = 0.
VALX( 6,2) = 0.1
VALX( 6,3) = 0.5
VALX( 6,4) = 1.
VALX( 6,5) = 1.5
VALX( 6,6) = 2.
VALX( 6,7) = 2.5
VALX( 6,8) = 3.
VALX( 6,9) = 3.5
VALX( 6,10)= 4.
VALX( 6,11)= 4.5
VALX( 6,12)= 5.
VALX( 6,13)= 10.
VALX( 6,14)= 15.
ENDIF
ELSE
VALX( 6,1) = 0.
VALX( 6,2) = 0.1
VALX( 6,3) = 0.5
VALX( 6,4) = 1.
VALX( 6,5) = 1.5
VALX( 6,6) = 2.
VALX( 6,7) = 2.5
VALX( 6,8) = 3.
VALX( 6,9) = 3.5
VALX( 6,10)= 4.
VALX( 6,11)= 4.5
VALX( 6,12)= 5.
VALX( 6,13)= 10.
VALX( 6,14)= 15.
ENDIF
IF(K.EQ.1)THEN
VALX( 7,1) = 0.
VALX( 7,2) = 0.25
VALX( 7,3) = 0.5
VALX( 7,4) = 0.75
VALX( 7,5) = 1.
VALX( 7,6) = 1.25
VALX( 7,7) = 1.5.
VALX( 7,8) = 1.75
VALX( 7,9) = 2.
VALX( 7,10)= 2.25
VALX( 7,11)= 2.5
VALX( 7,12)= 3.
VALX( 7,13)= 4.
VALX( 7,14)= 5.
ELSE
VALX( 7,1) = -0.5
VALX( 7,2) = -0.4
VALX( 7,3) = -0.3
VALX( 7,4) = -0.2
VALX( 7,5) = -0.1
VALX( 7,6) = 0.
VALX( 7,7) = 0.1
VALX( 7,8) = 0.2
VALX( 7,9) = 0.3
VALX( 7,10)= 0.4
VALX( 7,11)= 0.5
VALX( 7,12)= 0.6
VALX( 7,13)= 0.7
VALX( 7,14)= 0.8
ENDIF
IF(K.EQ.1)THEN
VALX( 8,1) = -130.
VALX( 8,2) = -120.
VALX( 8,3) = -110.
VALX( 8,4) = -100.
VALX( 8,5) = -90.
VALX( 8,6) = -80.
VALX( 8,7) = -70.
VALX( 8,8) = -60.
VALX( 8,9) = -50.
VALX( 8,10)= -40.
VALX( 8,11)= -30.
VALX( 8,12)= -20.
VALX( 8,13)= -10.
VALX( 8,14)= 0.
ELSE
VALX( 8,1) = -230.
VALX( 8,2) = -220.

```

getval.f

```

VALX( 8,3) = -210.
VALX( 8,4) = -200.
VALX( 8,5) = -190.
VALX( 8,6) = -180.
VALX( 8,7) = -170.
VALX( 8,8) = -160.
VALX( 8,9) = -150.
VALX( 8,10)= -140.
VALX( 8,11)= -130.
VALX( 8,12)= -120.
VALX( 8,13) = -110.
VALX( 8,14)= -100.
ENDIF
VALX( 9,1) = -19.5
VALX( 9,2) = -19.2
VALX( 9,3) = -19.1
VALX( 9,4) = -19.0
VALX( 9,5) = -18.9
VALX( 9,6) = -18.8
VALX( 9,7) = -18.7
VALX( 9,8) = -18.6
VALX( 9,9) = -18.5
VALX( 9,10)= -18.4
VALX( 9,11)= -18.3
VALX( 9,12)= -18.2
VALX( 9,13)= -18.1
VALX( 9,14)= -17.8
VALX(10,1) = -130.
VALX(10,2) = -120.
VALX(10,3) = -110.
VALX(10,4) = -100.
VALX(10,5) = -90.
VALX(10,6) = -80.
VALX(10,7) = -70.
VALX(10,8) = -60.
VALX(10,9) = -50.
VALX(10,10)= -40.
VALX(10,11)= -30.
VALX(10,12)= -20.
VALX(10,13) = -10.
VALX(10,14)= 0.
VALX(11,1) = 0.
VALX(11,2) = 0.25
VALX(11,3) = 0.5
VALX(11,4) = 0.75
VALX(11,5) = 1.
VALX(11,6) = 1.25
VALX(11,7) = 1.5
VALX(11,8) = 1.75
VALX(11,9) = 2.
VALX(11,10)= 2.25
VALX(11,11)= 2.5
VALX(11,12)= 3.
VALX(11,13) = 4.
VALX(11,14)= 5.
VALX(12,1) = -130.
VALX(12,2) = -120.
VALX(12,3) = -110.
VALX(12,4) = -100.
VALX(12,5) = -90.
VALX(12,6) = -80.
VALX(12,7) = -70.
VALX(12,8) = -60.
VALX(12,9) = -50.
VALX(12,10)= -40.
VALX(12,11)= -30.
VALX(12,12)= -20.
VALX(12,13) = -10.
VALX(12,14)= 0.
IF(K.EQ.1)THEN
VALX(13,1) = -50.
VALX(13,2) = 0.
VALX(13,3) = 50.
VALX(13,4) = 100.
VALX(13,5) = 150.
VALX(13,6) = 200.
VALX(13,7) = 250.
VALX(13,8) = 300.
VALX(13,9) = 350.
VALX(13,10)= 400.
VALX(13,11)= 450.
VALX(13,12)= 500.
VALX(13,13)= 600.
VALX(13,14)= 650.
ELSE
VALX(13,1) = 80.
VALX(13,2) = 85.
VALX(13,3) = 90.
VALX(13,4) = 95.
VALX(13,5) = 100.
VALX(13,6) = 105.
VALX(13,7) = 110.
VALX(13,8) = 115.
VALX(13,9) = 120.
VALX(13,10)= 130.
VALX(13,11)= 140.
VALX(13,12)= 150.
VALX(13,13)= 200.
VALX(13,14)= 250.
ENDIF
VALX(14,1) = 7.75
VALX(14,2) = 7.8
VALX(14,3) = 7.85
VALX(14,4) = 7.9
VALX(14,5) = 7.95
VALX(14,6) = 8.00
VALX(14,7) = 8.05
VALX(14,8) = 8.1
VALX(14,9) = 8.15
VALX(14,10)= 8.2
VALX(14,11)= 8.25
VALX(14,12)= 8.3
VALX(14,13) = 8.4
VALX(14,14) = 8.5
VALX(15,1) = 100.
VALX(15,2) = 101.
VALX(15,3) = 102.
VALX(15,4) = 103.
VALX(15,5) = 104.
VALX(15,6) = 105.
VALX(15,7) = 106.
VALX(15,8) = 107.
VALX(15,9) = 108.
VALX(15,10)= 109.
VALX(15,11)= 110.
VALX(15,12)= 111.
VALX(15,13)= 112.
VALX(15,14)= 113.
VALX(16,1) = 2.5
VALX(16,2) = 3.

```

```

VALX(16,3) = 3.5
VALX(16,4) = 4.
VALX(16,5) = 4.5
VALX(16,6) = 5.
VALX(16,7) = 5.5
VALX(16,8) = 6.
VALX(16,9) = 6.5
VALX(16,10)= 7.
VALX(16,11)= 7.5
VALX(16,12)= 8.
VALX(16,13)= 8.5
VALX(16,14)= 9.
VALX(17,1) = 0.
VALX(17,2) = 0.1
VALX(17,3) = 0.25
VALX(17,4) = 0.5
VALX(17,5) = 0.75
VALX(17,6) = 1.0
VALX(17,7) = 1.25
VALX(17,8) = 1.50
VALX(17,9) = 1.75
VALX(17,10)= 2.
VALX(17,11)= 2.25
VALX(17,12)= 2.5
VALX(17,13)= 2.75
VALX(17,14)= 3.
IF(K.EQ.1)THEN
VALX(18,1) = -1.
VALX(18,2) = 0.
VALX(18,3) = 2.5
VALX(18,4) = 5.
VALX(18,5) = 7.5
VALX(18,6) = 10.
VALX(18,7) = 12.5
VALX(18,8) = 15.
VALX(18,9) = 17.5
VALX(18,10)= 20.
VALX(18,11)= 22.5
VALX(18,12)= 25.
VALX(18,13)= 30.
VALX(18,14)= 35.
ELSE
VALX(18,1) = -1.5
VALX(18,2) = -1.
VALX(18,3) = -0.5
VALX(18,4) = 0.
VALX(18,5) = 0.5
VALX(18,6) = 1.
VALX(18,7) = 1.5
VALX(18,8) = 2.
VALX(18,9) = 2.5
VALX(18,10)= 3.
VALX(18,11)= 3.5
VALX(18,12)= 4.
VALX(18,13)= 5.
VALX(18,14)= 10.
ENDIF
IF(K.EQ.1)THEN
VALX(19,1) = 30.
VALX(19,2) = 31.
VALX(19,3) = 32.
VALX(19,4) = 33.
VALX(19,5) = 34.
VALX(19,6) = 34.25
VALX(19,7) = 34.5

```

getval.f

```

VALX(19,8) = 34.75
VALX(19,9) = 35.
VALX(19,10)= 35.25
VALX(19,11)= 35.5
VALX(19,12)= 36.
VALX(19,13)= 36.5
VALX(19,14)= 37.
ELSE
VALX(19,1) = 34.6
VALX(19,2) = 34.65
VALX(19,3) = 34.7
VALX(19,4) = 34.75
VALX(19,5) = 34.8
VALX(19,6) = 34.85
VALX(19,7) = 34.9
VALX(19,8) = 34.95
VALX(19,9) = 35.0
VALX(19,10)= 35.05
VALX(19,11)= 35.1
VALX(19,12)= 35.15
VALX(19,13)= 35.2
VALX(19,14)= 35.25
ENDIF
VALX(20,1) = 0.
VALX(20,2) = 5.
VALX(20,3) = 10.
VALX(20,4) = 15.
VALX(20,5) = 20.
VALX(20,6) = 25.
VALX(20,7) = 30.
VALX(20,8) = 35.
VALX(20,9) = 40.
VALX(20,10)= 45.
VALX(20,11)= 50.
VALX(20,12)= 55.
VALX(20,13)= 60.
VALX(20,14)= 65.
VALX(21,1) = -50.
VALX(21,2) = -40.
VALX(21,3) = -30.
VALX(21,4) = -20.
VALX(21,5) = -10.
VALX(21,6) = -5.
VALX(21,7) = 0.
VALX(21,8) = 5.
VALX(21,9) = 10.
VALX(21,10)= 20.
VALX(21,11)= 30.
VALX(21,12)= 40.
VALX(21,13)= 50.
VALX(21,14)= 60.
VALX(22,1) = 0.
VALX(22,2) = 5.
VALX(22,3) = 10.
VALX(22,4) = 15.
VALX(22,5) = 20.
VALX(22,6) = 25.
VALX(22,7) = 30.
VALX(22,8) = 40.
VALX(22,9) = 50.
VALX(22,10)= 60.
VALX(22,11)= 70.
VALX(22,12)= 80.
VALX(22,13)= 90.
VALX(22,14)= 100.

```

```

getval.f

VALX(23,1) = 0.
VALX(23,2) = 2.
VALX(23,3) = 4.
VALX(23,4) = 6.
VALX(23,5) = 8.
VALX(23,6) = 10.
VALX(23,7) = 15.
VALX(23,8) = 20.
VALX(23,9) = 25.
VALX(23,10)= 30.
VALX(23,11)= 40.
VALX(23,12)= 50.
VALX(23,13)= 60.
VALX(23,14)= 70.
VALX(24,1) = 0.
VALX(24,2) = 1.
VALX(24,3) = 5.
VALX(24,4) = 10.
VALX(24,5) = 15.
VALX(24,6) = 20.
VALX(24,7) = 25.
VALX(24,8) = 30.
VALX(24,9) = 35.
VALX(24,10)= 40.
VALX(24,11)= 45.
VALX(24,12)= 50.
VALX(24,13)= 55.
VALX(24,14)= 60.
VALX(25,1) = 0.
VALX(25,2) = 100.
VALX(25,3) = 200.
VALX(25,4) = 300.
VALX(25,5) = 400.
VALX(25,6) = 500.
VALX(25,7) = 1000.
VALX(25,8) = 2000.
VALX(25,9) = 5000.
VALX(25,10)= 10000.
VALX(25,11)= 20000.
VALX(25,12)= 30000.
VALX(25,13)= 40000.
VALX(25,14)= 50000.
IF(K.EQ.1)THEN
VALX(26,1) = 0.
VALX(26,2) = 0.1
VALX(26,3) = 0.25
VALX(26,4) = 0.5
VALX(26,5) = 0.75
VALX(26,6) = 1.0
VALX(26,7) = 1.25
VALX(26,8) = 1.50
VALX(26,9) = 1.75
VALX(26,10)= 2.
VALX(26,11)= 2.25
VALX(26,12)= 2.5
VALX(26,13)= 2.75
VALX(26,14)= 3.
ELSE
VALX(26,1) = 0.
VALX(26,2) = 0.25
VALX(26,3) = 0.5
VALX(26,4) = 0.75
VALX(26,5) = 1.
VALX(26,6) = 5.
VALX(26,7) = 10.
VALX(26,8) = 25.
VALX(26,9) = 50.
VALX(26,10)= 75.
VALX(26,11)= 100.
VALX(26,12)= 150.
VALX(26,13)= 200.
VALX(26,14)= 250.
ENDIF
C
DO 7854 IVX=1,IVAL
VAL(IVX)=VALX(IFR,IVX)
C      PRINT*, 'IFR IVX VALX(IFR,IVX) VAL(IVX) '
C      IFR,IVX,VALX(IFR,IVX),VAL(IVX)
7854 CONTINUE
RETURN
END

SUBROUTINE BOX
C THIS SUBROUTINE IS AN ADDITIONAL REFINEMENT PROCESS TAKEN FROM THE
C HAMBURG MODEL
C
PARAMETER (IMAXHR = 546, JMAXHR = 546)
INCLUDE 'data.txt'
DIMENSION FARRAY(0:IMAXHR,JMAXHR)
LOGICAL FMHZ(IE,JE),MHDMU(IE,JE)

ICON = 1000
100 CONTINUE
IF(ICON.LT.0) GOTO 130
IF(IVAL.LE.0) THEN
SCHUM = 1.E-6*DELTA
ELSE
XIN=1.E50
XAX=XIN
DO 110 I=1,IVAL
IF(VAL(I).LT.XIN) XIN=VAL(I)
IF(VAL(I).GT.XAX) XAX=VAL(I)
110 CONTINUE
SCHUM = 1.E 9*(XAX-XIN)
ENDIF
C
DO 120 J=1,JE
DO 120 I=1,IE
ARRAY(I,J) = ARRAY(I,J) + SCHUM
120 CONTINUE
C
130 CONTINUE
IF(ICON.EQ.0) THEN
DO 135 J=1,JE
DO 135 I=1,IE
MHDMU(I,J) = TRUE
135 CONTINUE
CALL REFINER(ARRAY,FARRAY,MHDMU,FMHZ,IE,JE,IFINE)
ELSE
CALL REFINER(ARRAY,FARRAY,MHZ,FMHZ,IE,JE,IFINE)
ENDIF
XFAC = XMAP/FLOAT(IE)
YFAC = YMAP/FLOAT(JE-1)
C
RETURN
END

```

gksatr.txt

```
COMMON/GKSSTATE/POLYLI,LNTYPE,LNWDSF,PLCOLI,CTNR,WINDOW(4),
          VIEWPT(4),CLIP
          INTEGER POLYLI,LNTYPE,LNWDSF,PLCOLI,CTNR,CLIP
C
C * VARIABLES *
C
C POLYLI   -- HOLDS THE POLYLINE INDEX NUMBER
C LNTYPE   -- HOLDS THE TYPE OF THE LINE NUMBER
C LNWDSF   -- HOLDS THE WIDTH OF THE LINE
C PLCOLI   -- HOLDS THE COLOR OF THE POLYLINE
C CTNR     -- HOLDS THE CURRENT NORMALIZATION NUMBER
C WINDOW   -- HOLDS THE CURRENT COORDINATES OF THE WINDOW
C VIEWPT   -- HOLDS THE CURRENT COORDINATES OF THE VIEWPORT
C CLIP     -- HOLDS WHETHER CLIPPING IS TURNED ON OR NOT
C
```

gksstate.f

```
SUBROUTINE CURRENTATR
C
C THIS SUBROUTINE WILL RETRIEVE SOME OF THE CURRENT SETTINGS THAT
C NEED TO BE SAVED BEFORE PUTTING ON THE CONTINENTAL OVERLAY.
C
INCLUDE 'gksatr.txt'
C
CALL GQPLI(ERRIND,POLYLI)
CALL GQLN(ERRIND,LNTYPE)
CALL GQLWSC(ERRIND,LNWDSF)
CALL GQPLCI(ERRIND,PLCOLI)
CALL QCNTN(ERRIND,CTNR)
CALL GONT(NTNR,ERRIND,WINDOW,VIEWPT)
CALL GQCLIP(ERRIND,CLIP)

RETURN
END

SUBROUTINE SETATR
C
C THIS SUBROUTINE WILL SET THE ATTRIBUTES THAT WERE SAVED BY
C THE SUBROUTINE CURRENTATR BACK TO THEIR ORIGINAL STATE.
C
INCLUDE 'gksatr.txt'
C
CALL GSIN(LNTYPE)
CALL GSLWSC(LNWDSF)
CALL GSPLCI(PLCOLI)
CALL GSPLI(POLYLI)
CALL GSCLIP(CLIP)

RETURN
END
```

ibm_times.f

```

c subroutine date (mmddyy)
c This routine will return mmddyy as character*9 variable
c
c character*9 hhmmss,mmddyy
c call datmjb (kh,km,ks,kmo,kdy,kyr,kdow,kdoy,hhmmss,mmddyy)
c return
c end
c subroutine clock (hhmmss)
c
c This routine will return hhmmss as character*9 variable
c
c character*9 hhmmss,mmddyy
c call datmjb (kh,km,ks,kmo,kdy,kyr,kdow,kdoy,hhmmss,mmddyy)
c return
c end
c subroutine datmjb (kh,km,ks,kmo,kdy,kyr,kdow,kdoy,hhmmss,mmddyy)
c
c character*9 hhmmss,mmddyy
c
c integer now(0:13)
c
c call datimx (now) ! C-language routine; calls IRM's "localtime"
c
c kh=now(4) ! hr (0-24) from calling datim.c or datimx.c
c km=now(3) ! min from calling datim.c or datimx.c
c ks=now(2) ! sec from calling datim.c or datimx.c
c
c kmo=now(6) ! month from calling datim.c or datimx.c
c kdy=now(5) ! day of month from calling datim.c or datimx.c
c iyr=now(7) ! year (4-digit) from calling datim.c or datimx.c
c
c kyr=iyr-100*(iyr/100) !(2-digit)
c
c kdow=now(11) ! day of week Sun=0, Mon=1, ... Sat=6
c kdow=now(11) ! day of week from calling datim.c (not in datim.c)
c kdoy=now(12) ! day of year from calling datimx.c (not in datim.c)
c
c correct day-of-year for leap years, except at end of century
c if (4*(iyr/4).eq.iyr .and. kmo.ge.3 .and. kyr.ne.0) kdoy=kdoy+1
c
c write (hhmmss,10) kh,km,ks
c format (i2.2,'.',i2.2,'.',i2.2)
c
c write (mmddyy,11) kmo,kdy,kyr
c format (i2.2,'/',i2.2,'/',i2.2)
c
c return
c end
c
c set up timing functions for the workstation
c
c function timeff()
c integer time
c
c wall clock time in milliseconds to suit ocean.f
c
c timeff = time() * 1000
c return
c end
c function second()
c dimension tarray(2)
c
c tarray(1) is cpu time of user in seconds
c tarray(2) is cpu time of system in seconds
c
c total has user+system
c
c total = dtime(tarray)
c second = tarray(1)
c ibm style is 1/100's
c second = mclock()
c second = second / 100
c return
c end

```

litcon.txt

```
PARAMETER (MPACSH = -15, MALTSH = 68, XFAC = 5.0, YFAC = 2.5,  
*          XMAX = 360., XMIN = 0., YMAX = 180., YMIN = 0.,  
*          ICNCLR = 1, VTOP = .120, VBOT = .050, VLFT = .41,  
*          VRGHT = .57, NUM = 72)  
  
C * PARAMETERS FOR LITTLE CONTINENTS IN SECTION *  
C  
C MPACSH --- THE DEGREE OF SHIFTING NEEDED FOR THE SECTION LINE  
C OF THE PACIFIC OCEAN.  
C MALTSH --- THE DEGREE OF SHIFTING NEEDED FOR THE SECTION LINE  
C OF THE ATLANTIC OCEAN.  
C XFAC --- X COORDINATE FACTOR USED IN MULTIPLYING TO GET THE  
C CORRECT LINE ORIENTATION.  
C YFAC --- Y COORDINATE FACTOR USED IN MULTIPLYING TO GET THE  
C CORRECT LINE ORIENTATION.  
C XMAX --- MAXIMUM ALLOWABLE DEGREE IN X DIRECTION  
C YMAX --- MAXIMUM ALLOWABLE DEGREE IN Y DIRECTION  
C XMIN --- MINIMUM ALLOWABLE DEGREE IN X DIRECTION  
C YMIN --- MINIMUM ALLOWABLE DEGREE IN Y DIRECTION  
C ICNCLR --- COLOR INDEX OF THE LITTLE CONTINENTS  
C VTOP --- COORDINATE OF THE TOP OF THE VIEWPORT FOR THE  
C SMALL CONTINENTAL MAP.  
C VBOT --- COORDINATE OF THE BOTTOM OF THE VIEWPORT FOR THE  
C SMALL CONTINENTAL MAP.  
C VLFT --- COORDINATE OF THE LEFT SIDE OF THE VIEWPORT FOR THE  
C SMALL CONTINENTAL MAP.  
C VRGHT --- COORDINATE OF THE RIGHT SIDE OF THE VIEWPORT FOR THE  
C SMALL CONTINENTAL MAP.  
C NUM --- THE INDICE VALUE OF INDISU -- SHOULD EQUAL JE
```

litcont.f

```

2 CONTINUE
DO 3 J=1,NUM
  PY(J) = TEMP(J)
3 CONTINUE
CALL EZXY(PX,PY,NUM,GLAB)
C
C END HAMBURG DEPENDANT
C

CALL SQASF(IE,IF)
IF(IF1) = 1
IF(IF2) = 1
CALL GSASF(IF)
CALL GSFAIS(1)
CALL MAPPOS(VLEFT,VRGHT,VBOT,VTOP)

C
C SET UP THE MAP BUT DON'T DRAW ANYTHING
C

C THE PARAMETERS ARE SET SO THAT THE MAP IS TO ONLY DO CONTINENTS IN THE
C CIRCULAR EQUIDISTANT METHOD. THIS METHOD ALLOWS EZMAPA TO BE ALTERED
C ALONG THE X AND THE Y AXIS TO SUPPORT DIFFERENT ASPECT RATIOS.
C

CALL MAPSTC('OU','CO')
CALL MAPROJ('CE',0.,205.,0.)
CALL MAPSET('MA',0.,0.,0.,0.)
CALL MAPSTI('VS',150)
CALL MAPSTI('G1',1)
CALL MAPSTI('G2',2)
CALL MAPINT
CALL ARINAM(IAM,250000)
CALL MAPBLA(IAM)
CALL ARPRAM(IAM,0,0,0)

C COMPUTE AND PRINT OUT AMOUNT OF SPACE USED IN THE AREA MAP
C
  ISU=250000-(IAM(6)-IAM(5)-1)
  WRITE(*,*) '
  WRITE(*,*) '
  WRITE(*,*) 'BEGINNING OF CONTINENT DRAWING.'
  WRITE(*,*) 'SPACE USED IN AREA MAP IS ',ISU

C ARSCAM IS USED TO SCAN THE MAP AND PLOT THE COLORED AREAS. IT USES
C A USER DEFINED PROCEDURE COLPA THAT WILL DETERMINE WHICH AREAS ARE
C TO BE COLOR FILLED, HOW THEY ARE FILLED, ETC.
C
C
  CALL ARSCAM(IAM,XCS,YCS,10000,IAT,IAC,10,COLRAZ)
  CALL SETATR
  CALL GSPLCI(1)
  CALL FRAME

C
C
C
  RETURN
END

SUBROUTINE COLRAZ(XCS,YCS,IUS,IAT,IAC,IATI)
INCLUDE 'lironcl.txt'
DIMENSION XCS(*), YCS(*), IAT(*), IAC(*)

C THIS SUBROUTINE IS A USER DEFINED SUBROUTINE THAT IS CALLED BY THE
C NCAR UTILITY ARSCAM. THIS PARTICULAR ROUTINE IS SET UP TO COLOR
C FILL ONLY CONTINENTS BECAUSE THIS MAP IS USED AS AN OVERLAY ONTO THE
C CONTOUR PLOTS. THE CONTINENTS ARE SET TO BE FILLED IN WITH CYAN

```

litcont.f

```
C AT THE PRESENT TIME BUT CAN BE CHANGED AT ANY TIME BY SIMPLY
C CHANGING THE PARAMETER ICNCLR IN 'litcon.txt'.
C
C      IF (IAI(1).GE.0.AND.IAI(2).GE.0) THEN
C          ITM=MAX0(IAI(1),IAI(2))
C          IF (ITM.GT.0) THEN
C              IF (NCS.GT.150) PRINT *, 'COLPAM - NCS TOO BIG',NCS
C              CALL GSFACT(1)
C              ICI=MAPACI(ITM)
C              IF (ICI.NE.1) THEN
C                  CALL GFA(NCS-1,XCS,YCS)
C              ENDIF
C          ENDIF
C          RETURN
C      END
```

makefile for carbon cycle post-processor

```
# This is the makefile to run the carbon side of the model.
#
# SHELL = /bin/sh
#
#
plofillrun : plofill.o

carbonrun : cpexcc.o gksstate.o cont.o readat1.o refine.o\
           getval.o readinput.o cpmap.o plotters.o litcont.o\
           ibm_times.o datimx.o
           xlf -g -qxref=full -o $@ cpexcc.o gksstate.o cont.o\
           readat1.o refine.o getval.o readinput.o cpmap.o plotters.o\
           litcont.o ibm_times.o\
           datimx.o -L/usr/local/lib -lncarg_gks -lncarg -lncarg_ras -lncargv\
           -lncarg_loc

secrun : cpexcc.o gksstate.o litcont.o refinesec.o plotters.o cpsec.o\
         section.o ibm_times.o datimx.o
         xlf -qxref=full -o $@ cpexcc.o gksstate.o litcont.o\
         refinesec.o plotters.o cpsec.o section.o ibm_times.o\
         datimx.o -L/usr/local/lib -lncarg_gks -lncarg\
         -lncarg_ras -lncargv -lncarg -lncarg_loc

quickrun : quicklook.o cpexcc.o ibm_times.o datimx.o
           xlf -qxref=full -o $@ quicklook.o cpexcc.o ibm_times.o\
           datimx.o -L/usr/local/lib -lncarg_gks -lncarg\
           -lncarg_ras -lncargv -lncarg -lncarg_loc

plofill.o : plofill.f
           xlf -qautodbl-dblpad -qxref=full -o plofillrun plofill.f
cpexcc.o : cpexcc.f
           xlf -c cpexcc.f
gksstate.o : gksstate.f
           xlf -c gksstate.f
readat1.o : readat1.f
           xlf -c readat1.f
cont.o : cont.f
           xlf -c cont.f
refine.o : refine.f
           xlf -c refine.f
getval.o : getval.f
           xlf -c getval.f
quicklook.o : quicklook.f
           xlf -c quicklook.f
readinput.o : readinput.f
           xlf -c readinput.f
plotters.o : plotters.f
           xlf -c plotters.f
litcont.o : litcont.f
           xlf -c litcont.f
refinesec.o : refinesec.f
           xlf -c refinesec.f
section.o : section.f
           xlf -c section.f
cpmap.o : cpmap.f
           xlf -c cpmap.f
cpsec.o : cpsec.f
           xlf -c cpsec.f
ibm_times.o : ibm_times.f
           xlf -c ibm_times.f
datimx.o : datimx.c
           cc -c datimx.c
```

plotfill.f

```

C THIS PROGRAM IS TAKEN DIRECTLY FROM THE CARBON SIDE OF THE HAMBURG
C MODEL IN ITS ENTIRITY ONLY SLIGHTLY MODIFIED. IT IS USED TO PRODUCE THE
C FORMATTED FILES THAT THE SECTION AND MAP PARTS OF THE CODE READ IN.
C
C block data
C implicit real*8 (a-h,o-z)
C PARAMETER(IE=72,JE=72,KE=11,NV=12,IPARA=42,DRY=-.1111E+11,
C 1 ONE=1,E0)
C CHARACTER PARNAM(IPARA)*50,CONNAM(IPARA)*50,CUNIT(IPARA)*23
C CHARACTER FILNA*7
C COMMON/TRACER/ ATCQ12,ATCQ13,ATCQ14,DGCO2(IE,JE),
C 1 PCO2OA(IE,JE),PHSURF(IE,JE),PRORCA(IE,JE),PRCAC(IE,JE),
C 2 OCSD12(IE,JE),DEPTP(IE,JE),DEPTU(IE,JE),
C 3 SCO212(IE,JE,KE),ALKALI(IE,JE,KE),PHOSPH(IE,JE,KE),
C 4 OXYGEN(IE,JE,KE),POC12(IE,JE,KE),CALC12(IE,JE,KE),
C 5 SCO213(IE,JE,KE),SCO214(IE,JE,KE),POC13(IE,JE,KE),
C 6 POC14(IE,JE,KE),CALC13(IE,JE,KE),CALC14(IE,JE,KE),
C 7 SUPSAT(IE,JE,KE),PHVALU(IE,JE,KE),CARION(IE,JE,KE),
C 8 SOLPRO(IE,JE,KE),PREPHO(IE,JE,KE),RWAT(IE,JE),DDZ(IE,JE,KE),
C 9 U(IE,JE,KE),V(IE,JE,KE),T(IE,JE,KE),S(IE,JE,KE),
C * PM(IE,JE,KE),C14AGE(IE,JE,KE),SILICA(IE,JE,KE)
C DATA ATCQ12,ATCQ13,ATCQ14 /3*RR.RR/
C end
C PROGRAM TRAVER
C implicit real*8 (a-h,o-z)
C PARAMETER(IE=72,JE=72,KE=11,NV=12,IPARA=42,DRY=-.1111E+11,
C 1 ONE=1,E0)
C CHARACTER PARNAM(IPARA)*50,CONNAM(IPARA)*50,CUNIT(IPARA)*23
C CHARACTER FILNA*7
C COMMON/TRACER/ ATCQ12,ATCQ13,ATCQ14,DGCO2(IE,JE),
C 1 PCO2OA(IE,JE),PHSURF(IE,JE),PRORCA(IE,JE),PRCAC(IE,JE),
C 2 OCSD12(IE,JE),CCSD12(IE,JE),DEPTP(IE,JE),DEPTU(IE,JE),
C 3 SCO212(IE,JE,KE),ALKALI(IE,JE,KE),PHOSPH(IE,JE,KE),
C 4 OXYGEN(IE,JE,KE),POC12(IE,JE,KE),CALC12(IE,JE,KE),
C 5 SCO213(IE,JE,KE),SCO214(IE,JE,KE),POC13(IE,JE,KE),
C 6 POC14(IE,JE,KE),CALC13(IE,JE,KE),CALC14(IE,JE,KE),
C 7 SUPSAT(IE,JE,KE),PHVALU(IE,JE,KE),CARION(IE,JE,KE),
C 8 SOLPRO(IE,JE,KE),PREPHO(IE,JE,KE),RWAT(IE,JE),DDZ(IE,JE,KE),
C 9 U(IE,JE,KE),V(IE,JE,KE),T(IE,JE,KE),S(IE,JE,KE),
C * PM(IE,JE,KE),C14AGE(IE,JE,KE),SILICA(IE,JE,KE)
C DIMENSION WIE(JE,KE),DMIN1(JE,JE,KE),DISC1(JE,JE,KE),
C * DISS3(JE,JE,KE)
C DIMENSION FACTOR(IPARA)
C DIMENSION IDDR(512),ODDR(512),ZPREL(6),ICONUM(IPARA)
C DIMENSION RMINTU(KE),
C * RMDZW(KE),PLALI(JE)
C CHARACTER*CTEMP
C CHARACTER*TEXT6
C DATA IDR,ODDR,ZPREL /512*RR,512*RR,RR,6*RR,RR/
C DATA ICONUM /41*0./ etc - qsm
C DATA ICONUM /ipara*0./
C DATA ATCQ12,ATCQ13,ATCQ14 /3*RR.RR/ put in block data - qsm
C DATA PARNAM /ipara*'RR,RR'/
C DATA CONNAM /ipara*'RR,RR'/
C DATA CUNIT /ipara*'RR,RR'/
C DATA FACTOR /ipara*0. /
C DATA FILNA /'P000101'/
C
C -----
C*          0. CODE NUMBERS OF ARRAYS TO BE READ FROM UNIT RR
C-----+
C CHANNEL 1      1 ATMOSPHERIC 12C VALUE
C               2 ATMOSPHERIC 13C VALUE
C               3 ATMOSPHERIC 14C VALUE
C               4   TOTAL CO2 12C
C               5   ALKALINITY
C
C               6   PO4
C               7   OXYGEN
C               8   POC 12C
C               9   CACO3 12C
C              10  TOTAL CO2 13C
C              11  TOTAL CO2 14C
C              12  POC 13C
C              13  POC 14C
C              14  CACO3 13C
C              15  CACO3 14C
C              16  SILICA
C
C               ICONUM(1)=1
C               ICONUM(2)=2
C               ICONUM(3)=3
C               ICONUM(4)=4
C
C               OPEN(8,FILE='POSTIN',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
C               OPEN(1,FILE='ATMOS',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(4,FILE='SCO2',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(39,FILE='ALKALI',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(40,FILE='PO4',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(7,FILE='O2',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(8,FILE='POC',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(9,FILE='CALCIT',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(10,FILE='DC13',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(11,FILE='DC14',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(12,FILE='POC13',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(13,FILE='POC14',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(14,FILE='CALC13',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(15,FILE='CALC14',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(16,FILE='CO2SUR',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(17,FILE='QADIFF',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(18,FILE='PHOURE',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(19,FILE='SOPRO1',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(20,FILE='CALPRO1',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(21,FILE='ORGSED',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(22,FILE='CALGED',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(24,FILE='SATC03',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(25,FILE='PH',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(26,FILE='C03',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')
C               OPEN(27,FILE='KSP',STATUS='UNKNOWN',
C *                  ACCESS='SEQUENTIAL',FORM='FORMATTED')

```

plofil1.f

```

    ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(28,FILE:'PO40',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(23,FILE:'BOTOPP',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(29,FILE:'SEALAN',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(30,FILE:'ODZINT',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(31,FILE:'U',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(32,FILE:'V',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(33,FILE:'T',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(34,FILE:'S',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(35,FILE:'BOTOPU',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(36,FILE:'W',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(37,FILE:'DMIN3',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(38,FILE:'DISC3',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(41,FILE:'PM',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(42,FILE:'C14AGE',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(43,FILE:'S104',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')
OPEN(44,FILE:'DISS3',STATUS='UNKNOWN',
     ACCESS='SEQUENTIAL',FORM='FORMATTED')

C
DO 1 I=1,IE
DO 1 J=1,IE
  DGC02(I,J)=88.88
  PCG020A(I,J)=88.88
  PHSURF(I,J)=88.88
  PRORCA(I,J)=88.88
  PRCACA(I,J)=88.88
  OCSD12(I,J)=88.88
  CCSD12(I,J)=88.88
  DEPTP(I,J)=88.88
  DEPTU(I,J)=88.88
  RWAT(I,J)=88.88
DO 1 K=1,KE
  SC0212(I,J,K)=88.88
  ALKALI(I,J,K)=88.88
  PHOSPH(I,J,K)=88.88
  OKYGEN(I,J,K)=88.88
  POC12(I,J,K)=88.88
  CALC12(I,J,K)=88.88
  SC0213(I,J,K)=88.88
  SC0214(I,J,K)=88.88
  POC13(I,J,K)=88.88
  POC14(I,J,K)=88.88
  CALC13(I,J,K)=88.88
  CALC14(I,J,K)=88.88
  SUPSAT1(I,J,K)=88.88
  PHVALU1(I,J,K)=88.88
  CARION(I,J,K)=88.88
  SOLPRO1(IE,JE,KE)=88.88
  PREPHO1(IE,JE,KE)=88.88
  PDZ(IE,JE,KE)=88.88
  U(IE,JE,KE)=88.88
  V(IE,JE,KE)=88.88
  T(IE,JE,KE)=88.88
  S(IE,JE,KE)=88.88
  W(IE,JE,KE)=88.88
  DMIN1(IE,JE,KE)=88.88
  DISC1(IE,JE,KE)=88.88
  DISS1(IE,JE,KE)=88.88
  DM(IE,JE,KE)=88.88
  C14AGE(IE,JE,KE)=88.88
  SILICA(IE,JE,KE)=88.88
1 CONTINUE
CONNAM( 1)='12CO2 ATMOSPHERE 1D
CUNIT( 1)='PPM
FACTOR( 1)=1.E0
CONNAM( 2)='13CO2 ATMOSPHERE REL. TO PDB STANDARD 1D
CUNIT( 2)='PER MILLE
FACTOR( 2)=1.E0
CONNAM( 3)='14CO2 ATMOSPHERE REL. TO NBS STANDARD 1D
CUNIT( 3)='PER MILLE
FACTOR( 3)=1.E0
CONNAM( 4)='TOTAL CO2 3D
CUNIT( 4)='MICROMOLE/L
FACTOR( 4)=1.E6
CONNAM( 5)='ALKALINITY 3D
CUNIT( 5)='MICROEQUIVALENTS/L
FACTOR( 5)=1.E6
CONNAM( 6)='PO4 3D
CUNIT( 6)='MICROMOLE/L
PO4R=1./122.
C REDFIELD RATIO AENDERUNG BEGIN
C PO4R=PO4H-PO4R*30./100.
C REDFIELD RATIO AENDERUNG BEGIN
FACTOR( 6)=1.E6*PO4R
CONNAM( 7)='PO2 3D
CUNIT( 7)='MICROMOLE/L
FACTOR( 7)=1.E6
CONNAM( 8)='POC 3D
CUNIT( 8)='MICROMOLE/L
FACTOR( 8)=1.E6
CONNAM( 9)='CALCITE 3D
CUNIT( 9)='MICROMOLE/L
FACTOR( 9)=1.E6
CONNAM(10)='TOTAL DELTA 13CO2 REL. TO PDB STANDARD 3D
CUNIT(10)='PER MILLE
FACTOR(10)=1.E0
CONNAM(11)='TOTAL BIG DELTA 14CO2 REL. TO NBS STANDARD 3D
CUNIT(11)='PER MILLE
FACTOR(11)=1.E0
CONNAM(12)='POC DELTA 13C REL. TO PDB STANDARD 3D
CUNIT(12)='PER MILLE
FACTOR(12)=1.E0
CONNAM(13)='POC DELTA 14C REL. TO NBS STANDARD 3D
CUNIT(13)='PER MILLE
FACTOR(13)=1.E0
CONNAM(14)='CALCITE DELTA 13C REL. TO PDB STANDARD 3D
CUNIT(14)='PER MILLE
FACTOR(14)=1.E0
CONNAM(15)='CALCITE DELTA 14C REL. TO NBS STANDARD 3D
CUNIT(15)='PER MILLE
FACTOR(15)=1.E0
CONNAM(16)='DISSOLVED GASEOUS CO2 SURFACE 2D
CUNIT(16)='MICROMOLE/L
FACTOR(16)=1.E6
CONNAM(17)='POCO2 DIFFERENCE OCEAN/ATMOSPHERE 2D

```

plofil11.f

```

CUNIT(17)='PPM
FACTOR(17)=1.E0
CONNAM(18)='PH SURFACE 2D
CUNIT(18)='
FACTOR(18)=1.E0
CONNAM(19)='SOFT TISSUE PRODUCTION 2D
CUNIT(19)='G/(M**2 Y)
FACTOR(19)=12.*50.*1.E3
CONNAM(20)='CALCITE HARD PARTS PRODUCTION 2D
CUNIT(20)='G/(M**2 Y)
FACTOR(20)=12.*50.*1.E3
CONNAM(21)='POC C12 SEDIMENTATION 2D
CUNIT(21)='G/(M**2 Y)
CUNIT(21)='MOL/M**2
C FACTOR(21)=1.E0
FACTOR(21)=10./120000.
CONNAM(22)='CALCITE C12 SEDIMENTATION 2D
CUNIT(22)='G/(M**2 Y)
CUNIT(22)='MOL/M**2
C FACTOR(22)=1.E0
FACTOR(22)=10./120000.
CONNAM(23)='BOTTOM TOPOGRAPHY P-POINTS 2D
CUNIT(23)='M
FACTOR(23)=1.E-2
CONNAM(24)='DEG. OF SATURATION OF CACO3 3D
CUNIT(24)='PERCENT
FACTOR(24)=1.E2
CONNAM(25)='PH 3D
CUNIT(25)='
FACTOR(25)=1.E0
CONNAM(26)='IONIC CO3 CONCENTRATION 3D
CUNIT(26)='MICROMOLES/L
FACTOR(26)=1.E6
CONNAM(27)='SOLUBILITY PRODUCT OF CALCITE 3D
CUNIT(27)=10**7
FACTOR(27)=1.E7
CONNAM(28)='PREFORMED PO4 3D
CUNIT(28)='MICROMOLES/L
FACTOR(28)=1.E6/133.
CONNAM(29)='SEA/LAND DISTRIBUTION 2D
CUNIT(29)='
FACTOR(29)=1.E0
CONNAM(30)='DDZ INTERVAL OF BOX, IF CELL WET DDZ GT 0..3D
CUNIT(30)='M
FACTOR(30)=1.E-2
CONNAM(31)='U COMPONENT OF VELOCITY 3D
CUNIT(31)='CM/S
FACTOR(31)=1.E0
CONNAM(32)='V COMPONENT OF VELOCITY 3D
CUNIT(32)='CM/S
FACTOR(32)=1.E0
CONNAM(33)='TEMPERATURE 3D
CUNIT(33)='DEG C
FACTOR(33)=1.E0
CONNAM(34)='SALINITY 3D
CUNIT(34)='
FACTOR(34)=1.E0
CONNAM(35)='BOTTOM TOPOGRAPHY U-POINTS 2D
CUNIT(35)='M
FACTOR(35)=1.E-2
CONNAM(36)='W COMPONENT OF VELOCITY 3D
CUNIT(36)='CM/S
FACTOR(36)=1.E0
CONNAM(37)='DMIN3 PROD. SOFT TISS. PART FALLEN AND REMAINING 3D'
CUNIT(37)='PERCENT
FACTOR(37)=1.E2
CONNAM(38)='DISC3 PROD. CALC. SHELL PART FALLEN AND REMAINING 3D'
CUNIT(38)='PERCENT
FACTOR(38)=1.E2
CONNAM(39)='PERCENT MODERN OF TOTAL CO2 14C
CUNIT(39)='PERCENT
FACTOR(39)=1.
CONNAM(40)='CONV. C14 AGE OF TOTAL CO2
CUNIT(40)='YR
FACTOR(40)=1.
CONNAM(41)='SI04
CUNIT(41)='MICROMOLES/L
FACTOR(41)=1.E6/122.
CONNAM(42)='DISS3 PROD. SI04 SHELL PART FALLEN AND REMAINING 3D'
CUNIT(42)='PERCENT
FACTOR(42)=1.E2
READ(88)PARNAM(1)
C READ(88)ATCQ12
C READ(88)PARNAM(2)
C READ(88)ATCQ13
C READ(88)PARNAM(3)
C READ(88)ATCQ14
C
C* ----- 1. READ MAIN HEADERS -----
C
C READ(88)IDDR
C ICREDAT=IDDP(R)
C WRITE(CTEMP,(16)) ICREDAT
C WRITE(CTEMP,(A2,A1,A2,A1,A2)) (CTEMP(3:4),'/',
C CTEMP(5:6),'/',CTEMP(1:2))
C WRITE(6,(''IDDR''))
C WRITE(6,(''I20''))IDDR
C
C READ(88)ODDR
C WRITE(6,(''ODDR''))
C WRITE(6,(''E20.5''))ODDR
C
C* ----- 2. SET ESSENTIAL PARAMETERS FROM MAIN HEADER INFO -----
C
C* 2.1 DATE AND YEAR OF DATA
IDDAT=IDDR(11)
IDAYEA=IDDR(12)
C
C* 2.2 NUMBER OF POINTS ON LATITUDE LINE
IEINP=IDDR(19)
C
C* 2.3 NUMBER OF LATITUDE LINES
JEINP=IDDR(20)
C
C* 2.4 NUMBER OF LAYERS
KEINP=IDDR(21)
PRINT*, '***INP IE JE KE ',IEINP,JEINP,KEINP
C
C* 2.5 CHECK DIMENSIONS OF INPUT ARRAYS
IF(IE.NE.IEINP)STOP '**ELOFIL* IE NE IEINP'
IF(JE.NE.JEINP)STOP '**ELOFIL* JE NE JEINP'
IF(KE.NE.KEINP)STOP '**ELOFIL* KE NE KEINP'
C
C* 2.6 SET DEPTH LEVELs IN M
DO 4001 K=1,KE
  RMINTU(K)=ODDP(9+K)
  PRINT*, '***INP RMINTU(1,Y,1)=',RMINTU(K)

```

plotfil11.f

```

4001 CONTINUE
C
C*      2.8 ZONAL AND MERIDIONAL GRID STEP IN DEGREES
C      ZONGRI=ODDR(50)
C      RMEGRI=ODDR(51)
C      PRINT*, '***INP_ZONGRI RMEGRI ',ZONGRI,RMGRGI
C
C*      2.8 LATITUDES OF LAT. LINES (DEGREES)
DO 4003 J=1,JE
    RLALI(J)=ODDR(51+J)
C    PRINT*, '***INP_RLALI(''J,'')=',RLALI(J)
4003 CONTINUE
C
C-----X. READ PARAMETER DATA ARRAYS
C-----ISTOP=ODDR(16)
C      PRINT*, 'NUMBER OF DATA RECORDS ISTOP='',ISTOP
C
C*      X.X READ ATMOSPHERIC 12C
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)ATCQ12
C      WRITE(6,(''ATCQ12='',F10.3))ATCQ12
C
C*      X.X READ ATMOSPHERIC 13C
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)ATCQ13
C      WRITE(6,(''ATCQ13='',F10.3))ATCQ13
C
C*      X.X READ ATMOSPHERIC 14C
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)ATCQ14
C      WRITE(6,(''ATCQ14='',F10.3))ATCQ14
C
C      READ(RR)PARNAME(4)
C      READ(RR)SCO212
C
C*      X.X READ SCO212
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)SCO212
C      WRITE(6,(''SCO212 READ FROM INPUTFILE''))
C
C*      X.X READ ALKALI
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)ALKALI
C      WRITE(6,(''ALKALI READ FROM INPUTFILE''))
C
C*      X.X READ PO4
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)PHOSPH
C      WRITE(6,(''PO4 READ FROM INPUTFILE''))
C
C*      X.X READ OXYGEN
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)OXYGEN
C      WRITE(6,(''OXYGEN READ FROM INPUTFILE''))
C
C*      X.X READ POC12
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)POC12
C      WRITE(6,(''POC12 READ FROM INPUTFILE''))
C
C*      X.X READ CALC12
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)CALC12
C      WRITE(6,(''CALC12 READ FROM INPUTFILE''))
C
C*      X.X READ SCO213
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)SCO213
C      WRITE(6,(''SCO213 READ FROM INPUTFILE''))
C
C*      X.X READ SCO214
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)SCO214
C      WRITE(6,(''SCO214 READ FROM INPUTFILE''))
C
C*      X.X READ POC13
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)POC13
C      WRITE(6,(''POC13 READ FROM INPUTFILE''))
C
C*      X.X READ POC14
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)POC14
C      WRITE(6,(''POC14 READ FROM INPUTFILE''))
C
C*      X.X READ CALC13
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)CALC13
C      WRITE(6,(''CALC13 READ FROM INPUTFILE''))
C
C*      X.X READ CALC14
READ(RR)ZPREL
C      WRITE(6,(''ZPREL''))
C      WRITE(6,(F12.2)'ZPREL
READ(RR)CALC14
C      WRITE(6,(''CALC14 READ FROM INPUTFILE''))
C
C*      X.X READ SILICA
READ(RR)ZPREL

```

```

C      WRITE(6,'("ZPREL"))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)SILICA
C      WRITE(6,'("SILICA READ FROM INPUTFILE"))'
C
C*      X.X READ SURFACE CO2 GAG
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL"))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)DGCO2
C      WRITE(6,'("DGCO2 READ FROM INPUTFILE"))'
C
C*      X.X READ CO2-DIFFERENCE OC./ATM.
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL"))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)PCO2OA
C      WRITE(6,'("PCO2OA READ FROM INPUTFILE"))'
C
C*      X.X READ SURFACE PH VALUE
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL"))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)PHSURF
C      WRITE(6,'("PHSURF READ FROM INPUTFILE"))'
C
C*      X.X READ PRIMARY PRODUCTION (ORG. C)
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL"))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)PRORCA
C      WRITE(6,'("PRORCA READ FROM INPUTFILE"))'
C
C*      X.X READ CACO3 PRODUCTION
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)PRCACA
C      WRITE(6,'("PRCACA READ FROM INPUTFILE"))'
C
C*      X.X READ ORG. C SEDIMENT POOL
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)OCSD12
C      WRITE(6,'("OCSD12 READ FROM INPUTFILE"))'
C
C*      X.X READ CACO3 C SEDIMENT POOL
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)CCSD12
C      WRITE(6,'("CCSD12 READ FROM INPUTFILE"))'
C
C*      X.X READ CACO3 SATURATION
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)SUPSAT
C      WRITE(6,'("SUPSAT READ FROM INPUTFILE"))'
C
C*      X.X READ PH VALUE ALL LEVELS
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C
C      READ(8)PHVALU
C      WRITE(6,'("PHVALU READ FROM INPUTFILE"))'
C
C*      X.X READ CARBON CONCENTRATION
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)CARION
C      WRITE(6,'("CARION READ FROM INPUTFILE"))'
C
C*      X.X READ SOLUBILITY PRODUCT (CACO3)
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)SOLPRO
C      WRITE(6,'("SOLPRO READ FROM INPUTFILE"))'
C
C*      X.X READ PREFORMED PO4
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)PREFPO
C      WRITE(6,'("PREFPO READ FROM INPUTFILE"))'
C
C*      X.X READ P POINT TOPOGRAPHY
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)DEPTP
C      WRITE(6,'("DEPTP READ FROM INPUTFILE"))'
C
C*      X.X READ LAND/SEA DISTRIBUTION
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)RWAT
C      WRITE(6,'("RWAT READ FROM INPUTFILE"))'
C
C*      X.X READ ACTUAL LAYER THICKNESSES
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)DDZ
C      WRITE(6,'("DDZ READ FROM INPUTFILE"))'
C
C*      X.X READ ZONAL VELOCITY
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)V
C      WRITE(6,'("V READ FROM INPUTFILE"))'
C
C*      X.X READ MERIDIONAL VELOCITY
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)V
C      WRITE(6,'("V READ FROM INPUTFILE"))'
C
C*      X.X READ WATER TEMPERATURE
C      READ(8)ZPREL
C      WRITE(6,'("ZPREL'))'
C      WRITE(6,'(F12.2)' )ZPREL
C      READ(8)T
C      WRITE(6,'("T READ FROM INPUTFILE"))'

```

plofil11.f

plotfil1.f

```

C
C*      X.X READ SALINITY
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)S
C      WRITE(6,'(''S READ FROM INPUTFILE'')')
C
C*      X.X READ VECTOR POINT TOPOGRAPHY
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)DEPTU
C      WRITE(6,'(''DEPTU READ FROM INPUTFILE'')')
C
C*      X.X READ VERTICAL VELOCITY
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)W
C      WRITE(6,'(''W READ FROM INPUTFILE'')')
C
C*      X.X READ DMIN3
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)DMIN3
C      WRITE(6,'(''DMIN3 READ FROM INPUTFILE'')')
C
C*      X.X READ DISC3
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)DISC3
C      WRITE(6,'(''DISC3 READ FROM INPUTFILE'')')
C
C*      X.X READ DISS3
      READ(88)ZPREL
C      WRITE(6,'(''ZPREL'')')
C      WRITE(6,'(F12.2)'')ZPREL
      READ(88)DISS3
C      WRITE(6,'(''DISS3 READ FROM INPUTFILE'')')
      GOTO 4711
      READ(88)PARNAME(5)
      READ(88)ALKALI
      READ(88)PARNAME(6)
      READ(88)PHOSPH
      READ(88)PARNAME(7)
      READ(88)OXYGEN
      READ(88)PARNAME(8)
      READ(88)POC12
      READ(88)PARNAME(9)
      READ(88)CALC12
      READ(88)PARNAME(10)
      READ(88)SCO213
      READ(88)PARNAME(11)
      READ(88)SCO214
      READ(88)PARNAME(12)
      READ(88)POC13
      READ(88)PARNAME(13)
      READ(88)POC14
      READ(88)PARNAME(14)
      READ(88)CALC13
      READ(88)PARNAME(15)
      READ(88)CALC14
      READ(88)PARNAME(16)
      READ(88)DSC02
      READ(88)PARNAME(17)
      READ(88)PCO20A
      READ(88)PARNAME(18)
      READ(88)DHSURF
      READ(88)PARNAME(19)
      READ(88)PRORCA
      READ(88)PARNAME(20)
      READ(88)DRCACA
      READ(88)PARNAME(21)
      READ(88)CCSD12
      READ(88)PARNAME(22)
      READ(88)CCSD12
      READ(88)PARNAME(24)
      READ(88)SUPSAT
      READ(88)PARNAME(25)
      READ(88)PHVALU
      READ(88)PARNAME(26)
      READ(88)CARLON
      READ(88)PARNAME(27)
      READ(88)SOLPRO
      READ(88)PARNAME(28)
      READ(88)PREPHO
      READ(88)PARNAME(29)
      READ(88)DEPTP
      READ(88)PARNAME(29)
      READ(88)RNAT
      READ(88)PARNAME(30)
      READ(88)DDZ
      READ(88)PARNAME(31)
      READ(88)U
      READ(88)PARNAME(32)
      READ(88)V
      READ(88)PARNAME(33)
      READ(88)IT
      READ(88)PARNAME(34)
      READ(88)S
      READ(88)PARNAME(35)
      READ(88)DEPTU
      READ(88)PARNAME(36)
      READ(88)W
      READ(88)PARNAME(37)
      READ(88)DMIN3
      READ(88)PARNAME(18)
      READ(88)DISC3
      4711 CONTINUE
C      PRINT*, '4711 ERREICH'
C
C      EVFR13=0.992
C      EVFR14=EVFR13**2
C      AN013-EVFR13/ATCQ13
C      AN014-EVFR14/ATCQ14
C      AN013=1./ATCQ13
C      AN014=1./ATCQ14
C      ATCQ13=ATCQ13*ATCQ12
C      ATCQ14=ATCQ14*ATCQ12
      DO 1371 I=1,IE
      DO 1371 J=1,JE
      DO 1371 K=1,KE
          SCO213(I,J,K)=(SCO213(I,J,K)+1.E0)*SCO212(I,J,K)/AN013
          SCO214(I,J,K)=(SCO214(I,J,K)+1.E0)*SCO212(I,J,K)/AN014
          IF(I.EQ.50)PRINT*, '***SCO213(I,J,K)= '
          SCO213(I,J,K)
          IF(I.EQ.50)PRINT*, '***SCO214(I,J,K)= '

```

plofil11.f

```

C      *      SCO214(I,J,K)
C      POC13(I,J,K)=(POC13(I,J,K)+1_E0)*POC12(I,J,K)/ANO13
C      POC14(I,J,K)=(POC14(I,J,K)+1_E0)*POC12(I,J,K)/ANO14
C      CALC13(I,J,K)=(CALC13(I,J,K)+1_E0)*CALC12(I,J,K)/ANO13
C      CALC14(I,J,K)=(CALC14(I,J,K)+1_E0)*CALC12(I,J,K)/ANO14
1371 CONTINUE
      CALL CALI10
C
C      C14MIN=1_E20
C      C14MAX=-1_E20
      DO 31654 K=1,KE
      DO 31654 J=1,JE
      IF(DDZ(I,J,K).LT.1.E-1)GOTO 31654
      IF(SCO214(I,J,K).LT.C14MIN)THEN
          C14MIN=SCO214(I,J,K)
          IMIN=I
          JMIN=J
          KMIN=K
      ELSE IF(SCO214(I,J,K).GT.C14MAX)THEN
          C14MAX=SCO214(I,J,K)
          IMAX=I
          JMAX=J
          KMAX=K
      ENDIF
      C      IF(SCO214(I,J,K).GE.0.)THEN
      C          PRINT*,***I J K SCO214(I,J,K),SCO214(I,J,K)
      C      ENDIF
31654 CONTINUE
C      PRINT*,***C14MIN-,C14MIN
C      PRINT*,***C14MAX-,C14MAX
C
      LAUMON=-1
C
      WRITE(1,'(A50,A23,A7)')CONNAM(1),CUNIT(1),FILNA
      WRITE(1,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(1,'(E10.4)')ATCQ12
      WRITE(1,'(A50,A23,A7)')CONNAM(2),CUNIT(2),FILNA
      WRITE(1,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(1,'(E10.4)')ATCQ13
      WRITE(1,'(A50,A23,A7)')CONNAM(3),CUNIT(3),FILNA
      WRITE(1,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(1,'(E10.4)')ATCQ14
C      PRINT*,104 ERREICHT*
      DO 104 I=1,IE
      DO 104 J=1,JE
      DO 104 K=1,KE
          SCO212(I,J,K)=SCO212(I,J,K)*FACTOR(4)
104     IF(DDZ(I,J,K).LT.ONE)SCO212(I,J,K)=DRY
      WRITE(4,'(A50,A23,A7)')CONNAM(4),CUNIT(4),FILNA
      WRITE(4,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(4,'(R(IX,E10.4))')SCO212
C
C      CREATE A DDZ FILE FOR USE IN THE LSQMODEL SIDE OF THE PROGRAM.
C
      OPEN (55,FILE='ddz',FORM='FORMATTED',STATUS='UNKNOWN')
      WRITE(55,'(SF15.5)') (((DDZ(I,J,K),I=1,IE),J=1,JE),K=1,KE)
      CLOSE(55)
C
      PRINT*,105 ERREICHT*
      DO 105 I=1,IE
      DO 105 J=1,JE
      DO 105 K=1,KE
          ALKALI(I,J,K)=ALKALI(I,J,K)*FACTOR(5)
105     IF(DDZ(I,J,K).LT.ONE)ALKALI(I,J,K)=DRY
C
      WRITE(39,'(A50,A23,A7)')CONNAM(5),CUNIT(5),FILNA
      WRITE(39,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(39,'(R(IX,E10.4))')ALKALI
      PRINT*,106 ERREICHT*
      DO 106 I=1,IE
      DO 106 J=1,JE
      DO 106 K=1,KE
          PHOSPH(I,J,K)=PHOSPH(I,J,K)*FACTOR(6)
106     IF(DDZ(I,J,K).LT.ONE)PHOSPH(I,J,K)=DRY
      WRITE(40,'(A50,A23,A7)')CONNAM(6),CUNIT(6),FILNA
      WRITE(40,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(40,'(R(IX,E10.4))')PHOSPH
      PRINT*,107 ERREICHT*
      DO 107 I=1,IE
      DO 107 J=1,JE
      DO 107 K=1,KE
          OXYGEN(I,J,K)=OXYGEN(I,J,K)*FACTOR(7)
107     IF(DDZ(I,J,K).LT.ONE)OXYGEN(I,J,K)=DRY
      WRITE(7,'(A50,A23,A7)')CONNAM(7),CUNIT(7),FILNA
      WRITE(7,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(7,'(R(IX,E10.4))')OXYGEN
      PRINT*,108 ERREICHT*
      DO 108 I=1,IE
      DO 108 J=1,JE
      DO 108 K=1,KE
          PO12(I,J,K)=POC12(I,J,K)*FACTOR(8)
108     IF(DDZ(I,J,K).LT.ONE)POC12(I,J,K)=DRY
      WRITE(8,'(A50,A23,A7)')CONNAM(8),CUNIT(8),FILNA
      WRITE(8,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(8,'(R(IX,E10.4))')POC12
      PRINT*,109 ERREICHT*
      DO 109 I=1,IE
      DO 109 J=1,JE
      DO 109 K=1,KE
          CALC12(I,J,K)=CALC12(I,J,K)*FACTOR(9)
109     IF(DDZ(I,J,K).LT.ONE)CALC12(I,J,K)=DRY
      WRITE(9,'(A50,A23,A7)')CONNAM(9),CUNIT(9),FILNA
      WRITE(9,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(9,'(R(IX,E10.4))')CALC12
      PRINT*,110 ERREICHT*
      DO 110 I=1,IE
      DO 110 J=1,JE
      DO 110 K=1,KE
          SCO213(I,J,K)=SCO213(I,J,K)*FACTOR(10)
110     IF(DDZ(I,J,K).LT.ONE)SCO213(I,J,K)=DRY
      WRITE(10,'(A50,A23,A7)')CONNAM(10),CUNIT(10),FILNA
      WRITE(10,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(10,'(R(IX,E10.4))')SCO213
      PRINT*,111 ERREICHT*
      DO 111 I=1,IE
      DO 111 J=1,JE
      DO 111 K=1,KE
          SCO214(I,J,K)=SCO214(I,J,K)*FACTOR(11)
111     IF(DDZ(I,J,K).LT.ONE)SCO214(I,J,K)=DRY
      WRITE(11,'(A50,A23,A7)')CONNAM(11),CUNIT(11),FILNA
      WRITE(11,'(''MONTH '',I2,AR)')LAUMON,TEXT6
      WRITE(11,'(R(IX,E10.4))')SCO214
      PRINT*,112 ERREICHT*
      DO 112 I=1,IE
      DO 112 J=1,JE
      DO 112 K=1,KE
          PO13(I,J,K)=POC13(I,J,K)*FACTOR(12)
112     IF(DDZ(I,J,K).LT.ONE)POC13(I,J,K)=DRY
      WRITE(12,'(A50,A23,A7)')CONNAM(12),CUNIT(12),FILNA
      WRITE(12,'(''MONTH '',I2,AR)')LAUMON,TEXT6

```

plofil11.f

```

      WRITE(12,'(R(IX,E10.4))')POC1
      C PRINT*, '113 ERREICHT'
      DO 113 I=1,IE
      DO 113 J=1,JE
      DO 113 K=1,KE
         POC14(I,J,K)=POC14(I,J,K)*FACTOR(13)
      113 IF(DDZ(I,J,K).LT.ONE)POC14(I,J,K)=DRY
      WRITE(13,'(A50,A23,A7)')CONNAM(13),CUNIT(13),FILNA
      WRITE(13,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(13,'(R(IX,E10.4))')POC14
      C PRINT*, '114 ERREICHT'
      DO 114 I=1,IE
      DO 114 J=1,JE
      DO 114 K=1,KE
         CALC13(I,J,K)=CALC13(I,J,K)*FACTOR(14)
      114 IF(DDZ(I,J,K).LT.ONE)CALC13(I,J,K)=DRY
      WRITE(14,'(A50,A23,A7)')CONNAM(14),CUNIT(14),FILNA
      WRITE(14,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(14,'(R(IX,E10.4))')CALC13
      C PRINT*, '115 ERREICHT'
      DO 115 I=1,IE
      DO 115 J=1,JE
      DO 115 K=1,KE
         CALC14(I,J,K)=CALC14(I,J,K)*FACTOR(15)
      115 IF(DDZ(I,J,K).LT.ONE)CALC14(I,J,K)=DRY
      WRITE(15,'(A50,A23,A7)')CONNAM(15),CUNIT(15),FILNA
      WRITE(15,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(15,'(R(IX,E10.4))')CALC14
      C PRINT*, '116 ERREICHT'
      C DO 143 I=1,IE
      C DO 143 J=1,JE
      C DO 143 K=1,KE
      C SILICA(I,J,K)=SILICA(I,J,K)*FACTOR(41)
      C 143 IF(DDZ(I,J,K).LT.ONE)PM(I,J,K)=DRY
      C WRITE(43,'(A50,A23,A7)')CONNAM(41),CUNIT(41),FILNA
      C WRITE(43,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      C WRITE(43,'(R(IX,E10.4))')SILICA
      C PRINT*, '116 ERREICHT'
      DO 116 I=1,IE
      DO 116 J=1,JE
         DGC02(I,J)=DGC02(I,J)*FACTOR(16)
      116 IF(DDZ(I,J,1).LT.ONE)DGC02(I,J)=DRY
      WRITE(16,'(A50,A23,A7)')CONNAM(16),CUNIT(16),FILNA
      WRITE(16,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(16,'(R(IX,E10.4))')DGC02
      C PRINT*, '117 ERREICHT'
      DO 117 I=1,IE
      DO 117 J=1,JE
         PCO20A(I,J)=PCO20A(I,J)*FACTOR(17)
      117 IF(DDZ(I,J,1).LT.ONE)PCO20A(I,J)=DRY
      WRITE(17,'(A50,A23,A7)')CONNAM(17),CUNIT(17),FILNA
      WRITE(17,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(17,'(R(IX,E10.4))')PCO20A
      C PRINT*, '118 ERREICHT'
      DO 118 I=1,IE
      DO 118 J=1,JE
         PHSURF(I,J)=PHSURF(I,J)*FACTOR(18)
      118 IF(DDZ(I,J,1).LT.ONE)PHSURF(I,J)=DRY
      WRITE(18,'(A50,A23,A7)')CONNAM(18),CUNIT(18),FILNA
      WRITE(18,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(18,'(R(IX,E10.4))')PHSURF
      C PRINT*, '119 ERREICHT'
      DO 119 I=1,IE
      DO 119 J=1,JE
         PRORCA(I,J)=PRORCA(I,J)*FACTOR(19)
      119 IF(DDZ(I,J,1).LT.ONE)PRORCA(I,J)=DRY
      WRITE(19,'(A50,A23,A7)')CONNAM(19),CUNIT(19),FILNA
      WRITE(19,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(19,'(R(IX,E10.4))')PRORCA
      C PRINT*, '120 ERREICHT'
      DO 120 I=1,IE
      DO 120 J=1,JE
         PRCAC(A,I,J)=PRCAC(A,I,J)*FACTOR(20)
      120 IF(DDZ(I,J,1).LT.ONE)PRCAC(A,I,J)=DRY
      WRITE(20,'(A50,A23,A7)')CONNAM(20),CUNIT(20),FILNA
      WRITE(20,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(20,'(R(IX,E10.4))')PRCAC(A
      C PRINT*, '121 ERREICHT'
      DO 121 I=1,IE
      DO 121 J=1,JE
         OCSD12(I,J)=OCSD12(I,J)*FACTOR(21)
      121 IF(DDZ(I,J,1).LT.ONE)OCSD12(I,J)=DRY
      WRITE(21,'(A50,A23,A7)')CONNAM(21),CUNIT(21),FILNA
      WRITE(21,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(21,'(R(IX,E10.4))')OCSD12
      C PRINT*, '122 ERREICHT'
      DO 122 I=1,IE
      DO 122 J=1,JE
         CCSD12(I,J)=CCSD12(I,J)*FACTOR(22)
      122 IF(DDZ(I,J,1).LT.ONE)CCSD12(I,J)=DRY
      WRITE(22,'(A50,A23,A7)')CONNAM(22),CUNIT(22),FILNA
      WRITE(22,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(22,'(R(IX,E10.4))')CCSD12
      C PRINT*, '123 ERREICHT'
      DO 123 I=1,IE
      DO 123 J=1,JE
         SUPSAT(I,J,K)=SUPSAT(I,J,K)*FACTOR(24)
      124 IF(DDZ(I,J,K).LT.ONE)SUPSAT(I,J,K)=DRY
      WRITE(24,'(A50,A23,A7)')CONNAM(24),CUNIT(24),FILNA
      WRITE(24,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(24,'(R(IX,E10.4))')SUPSAT
      C PRINT*, '125 ERREICHT'
      DO 125 I=1,IE
      DO 125 J=1,JE
      DO 125 K=1,KE
         PVALU(I,J,K)=PVALU(I,J,K)*FACTOR(25)
      125 IF(DDZ(I,J,K).LT.ONE)PVALU(I,J,K)=DRY
      WRITE(25,'(A50,A23,A7)')CONNAM(25),CUNIT(25),FILNA
      WRITE(25,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(25,'(R(IX,E10.4))')PVALU
      C PRINT*, '126 ERREICHT'
      DO 126 I=1,IE
      DO 126 J=1,JE
      DO 126 K=1,KE
         CARION(I,J,K)=CARION(I,J,K)*FACTOR(26)
      126 IF(DDZ(I,J,K).LT.ONE)CARION(I,J,K)=DRY
      WRITE(26,'(A50,A23,A7)')CONNAM(26),CUNIT(26),FILNA
      WRITE(26,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(26,'(R(IX,E10.4))')CARION
      C PRINT*, '127 ERREICHT'
      DO 127 I=1,IE
      DO 127 J=1,JE
      DO 127 K=1,KE
         SOLPRO(I,J,K)=SOLPRO(I,J,K)*FACTOR(27)
      127 IF(DDZ(I,J,K).LT.ONE)SOLPRO(I,J,K)=DRY
      WRITE(27,'(A50,A23,A7)')CONNAM(27),CUNIT(27),FILNA
      WRITE(27,'(''MONTH '',I2,AB)')LAUMON,TEXT6
      WRITE(27,'(R(IX,E10.4))')SOLPRO
      C PRINT*, '128 ERREICHT'
      DO 128 I=1,IE
      DO 128 J=1,JE

```

plofil11.f

```

DO 128 K=1,KE
  PREPHO(I,J,K)=PREPHO(I,J,K)*FACTOR(28)
128  IF(DDZ(I,J,K).LT.ONE)PREPHO(I,J,K)=DRY
  WRITE(28,'(A50,A23,A7)')CONNAM(28),CUNIT(28),FILNA
  WRITE(28,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(28,'(8(IX,E10.4))')TREPHO
  DO 129 I=1,IE
  DO 129 J=1,JE
129  DEPTP(I,J)=DEPTP(I,J)*FACTOR(23)
  WRITE(23,'(A50,A23,A7)')CONNAM(23),CUNIT(23),FILNA
  WRITE(23,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(23,'(8(IX,E10.4))')DEPTP
  DO 129 I=1,IE
  DO 129 J=1,JE
129  RWAT(I,J)=RWAT(I,J)*FACTOR(29)
  WRITE(29,'(A50,A23,A7)')CONNAM(29),CUNIT(29),FILNA
  WRITE(29,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(29,'(8(IX,E10.4))')RWAT
  DO 130 I=1,IE
  DO 130 J=1,JE
  DO 130 K=1,KE
130  DDZ(I,J,K)=DDZ(I,J,K)*FACTOR(30)
  WRITE(30,'(A50,A23,A7)')CONNAM(30),CUNIT(30),FILNA
  WRITE(30,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(30,'(8(IX,E10.4))')DDZ
  DO 131 I=1,IE
  DO 131 J=1,JE
  DO 131 K=1,KE
131  U(I,J,K)=U(I,J,K)*FACTOR(31)
  WRITE(31,'(A50,A23,A7)')CONNAM(31),CUNIT(31),FILNA
  WRITE(31,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(31,'(8(IX,E10.4))')U
  DO 132 I=1,IE
  DO 132 J=1,JE
  DO 132 K=1,KE
132  V(I,J,K)=V(I,J,K)*FACTOR(32)
  WRITE(32,'(A50,A23,A7)')CONNAM(32),CUNIT(32),FILNA
  WRITE(32,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(32,'(8(IX,E10.4))')V
  DO 133 I=1,IE
  DO 133 J=1,JE
  DO 133 K=1,KE
    T(I,J,K)=T(I,J,K)*FACTOR(33)
133  IF(DDZ(I,J,K).LT.ONE)T(I,J,K)=DRY
  WRITE(33,'(A50,A23,A7)')CONNAM(33),CUNIT(33),FILNA
  WRITE(33,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(33,'(8(IX,E10.4))')T
  DO 134 I=1,IE
  DO 134 J=1,JE
  DO 134 K=1,KE
    S(I,J,K)=S(I,J,K)*FACTOR(34)
134  IF(DDZ(I,J,K).LT.ONE)S(I,J,K)=DRY
  WRITE(34,'(A50,A23,A7)')CONNAM(34),CUNIT(34),FILNA
  WRITE(34,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(34,'(8(IX,E10.4))')S
  DO 135 I=1,IE
  DO 135 J=1,JE
135  DEPTU(I,J)=DEPTU(I,J)*FACTOR(35)
  WRITE(35,'(A50,A23,A7)')CONNAM(35),CUNIT(35),FILNA
  WRITE(35,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(35,'(8(IX,E10.4))')DEPTU
  DO 136 I=1,IE
  DO 136 J=1,JE
  DO 136 K=1,KE
136  W(I,J,K)=W(I,J,K)*FACTOR(36)

      DMIN3(I,J,K)=DMIN3(I,J,K)*FACTOR(37)
137  IF(DDZ(I,J,K).LT.ONE)DMIN3(I,J,K)=DRY
  WRITE(37,'(A50,A23,A7)')CONNAM(37),CUNIT(37),FILNA
  WRITE(37,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(37,'(8(IX,E10.4))')DMIN3
  DO 138 I=1,IE
  DO 138 J=1,JE
  DO 138 K=1,KE
    DISC3(I,J,K)=DISC3(I,J,K)*FACTOR(38)
138  IF(DDZ(I,J,K).LT.ONE)DISC3(I,J,K)=DRY
  WRITE(38,'(A50,A23,A7)')CONNAM(38),CUNIT(38),FILNA
  WRITE(38,'(''MONTH '',I2,AB)')LAUMON,TEXT6
  WRITE(38,'(8(IX,E10.4))')DISC3
  DO 141 I=1,IE
  DO 141 J=1,JE
  DO 141 K=1,KE
    PM(I,J,K)=(ISCO214(I,J,K)/1000.+1.)/100.
    PM(I,J,K)=(ISCO214(I,J,K)/1000.+1.)
    PM(I,J,K)=(ISCO214(I,J,K)/1000.+1.)*100.
    IF(PM(I,J,K).GT.100.)PRINT*,***I,J,K PM *,I,J,K,PM(I,J,K)
141  IF(DDZ(I,J,K).LT.ONE)PM(I,J,K)=DRY
    PMATE=(ATCO14/1000.+1.)*100.
    PRINT*,***PMATE*,PMAT
    WRITE(41,'(A50,A23,A7)')CONNAM(39),CUNIT(39),FILNA
    WRITE(41,'(''MONTH '',I2,AB)')LAUMON,TEXT6
    WRITE(41,'(8(IX,E10.4))')PMAT
    AGEMIN=1.E20
    AGEMAX=-1.E20
    DO 142 I=1,IE
    DO 142 J=1,JE
    DO 142 K=1,KE
      IF(DDZ(I,J,K).LT.ONE)THEN
        C14AGE(I,J,K)=DRY
      ELSE
        IF(PM(I,J,K).LE.0.)THEN
          C14AGE(I,J,K)=99999.
        ELSE
          C14AGE(I,J,K)=803.* ALOG(PM(I,J,K)/100.)
        ENDIF
        IF(C14AGE(I,J,K).GT.AGEMAX)AGEMAX=C14AGE(I,J,K)
        IF(C14AGE(I,J,K).LT.AGEMIN)AGEMIN=C14AGE(I,J,K)
      ENDIF
142  CONTINUE
    C PRINT*,***AGEMIN= ,AGEMIN
    DO 1421 I=1,IE
    DO 1421 J=1,JE
    DO 1421 K=1,JE
      IF(DDZ(I,J,K).LT.ONE)THEN
        C14AGE(I,J,K)=DRY
      ELSE
        C14AGE(I,J,K)=C14AGE(I,J,K)-AGEMIN
      ENDIF
1421 CONTINUE
    C AGEMIN=1.E20
    C AGEMAX= -1.E20
    C DO 1422 I=1,IE
    C DO 1422 J=1,JE
    C DO 1422 K=1,KE
    C IF(DDZ(I,J,K).LT.ONE)THEN

```

plofil11.f

```

C      C14AGE(I,J,K)=DRY
C      ELSE
C          IF(C14AGE(I,J,K).GT.AGEMAX)THEN
C              AGEMAX=C14AGE(I,J,K)
C              IMAX=I
C              JMAX=J
C              KMAX=K
C          ELSE IF(C14AGE(I,J,K).LT.AGEMIN)THEN
C              IMIN=I
C              JMIN=J
C              KMIN=K
C              AGEMIN=C14AGE(I,J,K)
C          ENDIF
C      ENDIF
C1422 CONTINUE
C      PRINT*, ' I J K AGEMAX DDZ SC0214 ',
C      * IMAX,JMAX,KMAX,AGEMAX,DDZ(IMAX,JMAX,KMAX),
C      * SC0214(IMAX,JMAX,KMAX)
C      PRINT*, ' I J K AGEMIN DDZ SC0214 ',
C      * IMIN,JMIN,KMIN,AGEMIN,DDZ(IMIN,JMIN,KMIN),
C      * SC0214(IMIN,JMIN,KMIN)
C      DO 1423 J=1,JE
C      DO 1423 I=1,IE
C      - DO 1423 I=1,IE
C      - C14AGE(I,J,K)=C14AGE(I,J,K)-C14AGE(I,J,1)
C14231 CONTINUE
C1423 CONTINUE
      WRITE(42,'(A50,A23,A7)')CONNAM(40),CUNIT(40),FILNA
      WRITE(42,'(''MONTH '',12,8)')LAUMON,TEXT6
      WRITE(42,'(B(1X,E10.4))')C14AGE
      DO 144 I=1,IE
      DO 144 J=1,JE
      DO 144 K=1,KE
          DISS3(I,J,K)=DISS3(I,J,K)*FACTOR(38)
144     IF(DDZ(I,J,K).LT.ONE)DISS3(I,J,K)=DRY
      WRITE(44,'(A50,A23,A7)')CONNAM(38),CUNIT(38),FILNA
      WRITE(44,'(''MONTH '',12,8)')LAUMON,TEXT6
      WRITE(44,'(B(1X,E10.4))')DISS3
      CLOSE(88,STATUS='KEEP')
      CLOSE(1,STATUS='KEEP')
      CLOSE(139,STATUS='KEEP')
      CLOSE(40,STATUS='KEEP')
      CLOSE(7,STATUS='KEEP')
      CLOSE(8,STATUS='KEEP')
      CLOSE(9,STATUS='KEEP')
      CLOSE(10,STATUS='KEEP')
      CLOSE(11,STATUS='KEEP')
      CLOSE(12,STATUS='KEEP')
      CLOSE(13,STATUS='KEEP')
      CLOSE(14,STATUS='KEEP')
      CLOSE(15,STATUS='KEEP')
      CLOSE(16,STATUS='KEEP')
      CLOSE(17,STATUS='KEEP')
      CLOSE(18,STATUS='KEEP')
      CLOSE(19,STATUS='KEEP')
      CLOSE(20,STATUS='KEEP')
      CLOSE(21,STATUS='KEEP')
      CLOSE(22,STATUS='KEEP')
      CLOSE(24,STATUS='KEEP')
      CLOSE(25,STATUS='KEEP')
      CLOSE(26,STATUS='KEEP')
      CLOSE(27,STATUS='KEEP')
      CLOSE(28,STATUS='KEEP')
      CLOSE(29,STATUS='KEEP')
      CLOSE(30,STATUS='KEEP')

      CLOSE(31,STATUS='KEEP')
      CLOSE(32,STATUS='KEEP')
      CLOSE(33,STATUS='KEEP')
      CLOSE(34,STATUS='KEEP')
      CLOSE(35,STATUS='KEEP')
      CLOSE(36,STATUS='KEEP')
      CLOSE(37,STATUS='KEEP')
      CLOSE(38,STATUS='KEEP')
      CLOSE(41,STATUS='KEEP')
      CLOSE(42,STATUS='KEEP')
      CLOSE(43,STATUS='KEEP')
      CLOSE(44,STATUS='KEEP')
      STOP
      END
      SUBROUTINE CALI10
C
C***** *CALIBR* - CALIBRATES C13-, C14 VALUES
C
C      CHRISTOPH HEINZE      MPI HAMBURG      88/01/29.
C
C      PURPOSE.
C
C      -----
C      *CALI10* CALIBRATES C-13-, C-14 VALUES OF OCEAN AND ATMOSPHERE;
C      THUS FIXES ABSOL. C13- & C-14 INVENTORY.
C      FINAL C13-, C-14 VALUES ARE GIVEN IN CONVENTIONAL
C      NOTATIONS AS D13C RELATIVE TO PDB STANDARD AND AS
C      BIG D14C WITH FRACTIONATION CORRECTION FOR C13
C
C      INTERFACE.
C
C      -----
C      *CALL* *CALI10*
C
C      METHOD.
C
C      -----
C      D13C (RELATIVE TO PDB STANDARD, KEELING (1981), CF. KEELING
C      (1981)) IN DEEP EASTERN PACIFIC BELOW 3000 M IS SET TO 0
C      ACCORDING TO MEASUREMENTS (KRIGPICK ET AL. (1970), CF.
C      BROECKER AND PENG (1982)) YIELDING FACTOR FOR CONVERSION
C      OF MODEL C13/C12 RATIO INTO ABSOLUTE C13 CONCENTRATIONS.
C      THE DEEP PACIFIC VALUES ARE CHOSEN AS REFERENCE SINCE THEY
C      ARE THE BEST APPROXIMATE PRE-ANTHROPOGENIC VALUES.
C      WITH THE ABSOLUTE C13 CONCENTRATIONS C13 IS REPORTED
C      AS D13C RELATIVE TO PDB STANDARD.
C
C      BIG D14C (RELATIVE TO NBS CHALIC ACID STANDARD WITH
C      FRACTIONATION CORRECTION FOR C13, SHIVIER AND POLLACH (1977),
C      BROECKER AND OLSON (1961), CF. KEELING (1981)) IN NORTH EAST
C      ATLANTIC SURFACE WATER IS SET TO -50 ACCORDING TO PRE-RMPC
C      C14 VALUES (CF. BROECKER AND PENG (1982)) YIELDING FACTOR
C      FOR CONVERSION OF MODEL C14/(C12/C13) RATIO INTO ABSOLUTE
C      C14 CONCENTRATIONS.
C
C      THE NORTH ATLANTIC SURFACE VALUES ARE USED AS REFERENCE,
C      SINCE THEY ARE THE BEST APPROXIMATELY KNOWN PRE ANTHROPOGENIC
C      SURFACE WATER VALUES. SURFACE WATER VALUES INSTEAD
C      OF DEEP WATER VALUES ARE USED AS
C      REFERENCE IN ORDER TO HAVE A BETTER CONTROL OF THE
C      SIMULATION OF DEEP WATER PRODUCTION AND GAS EXCHANGE
C      BETWEEN OCEAN AND ATMOSPHERE.
C
C      WITH THE ABSOLUTE C14 CONCENTRATIONS C14 IS
C      REPORTED AS BIG D14C ACCORDING TO THE SIMPLIFIED
C      FORMULA SUGGESTED BY BROECKER AND OLSON (1961).
C
C      EXTERNALS.
C
C      -----
C      NONE.

```

plofill1.f

```

C      REFERENCE.
C
C      BROECKER, W.S., AND E.A. OLSON (1961)
C      LAMONT RADIOCARBON MEASUREMENTS VIII.
C      RADIOCARBON, 3, 176-204.
C
C      BROECKER, W.S., AND T. H. PENG (1982)
C      TRACERS IN THE SEA.
C      ELDIGIO PRESS, LAMONT-DOHRY GEOLOGICAL OBSERVATORY,
C      PALISADES, N.Y., 690 PP.
C
C      KEELING, C.D. (1981)
C      THE MODELING OF RARE ISOTOPIC CARBON WITH REGARD TO
C      NOTATIONS. IN: CARBON CYCLE MODELING, B. BOLIN, ED.,
C      J. WILEY AND SONS, 89-99.
C
C      KROOPNICK, P., W.G. DEUSER, AND H. CRAIG (1970)
C      CARBON-13 MEASUREMENTS ON DISSOLVED INORGANIC CARBON AT
C      THE NORTH PACIFIC (1969) GESECS STATION.
C      J. GEOPHYS. RES., 75, 7668-7671.
C
C      STUIVER, M., AND H.A. POLLACH (1977)
C      DISCUSSION REPORTING OF  $^{14}\text{C}$  DATA.
C      RADIOCARBON, 19, NO.3, 355-363
C
C      UREY, H.C. (1947)
C      THE THERMODYNAMIC PROPERTIES OF ISOTOPIC SUBSTANCES.
C      J. AM. CHEM. SOC., 562-581.
C
C*     VARIABLE      TYPE      PURPOSE.
C
C      *OPSMA*      REAL      PREVENTS CALC12, POC12 FROM GETTING
C                           ZERO
C      *RE1312*      REAL      PDR STANDARD C13/C12 RATIO (CF.
C                           KEELING (1981))
C      *RE14TO*      REAL      NBS STANDARD C14/(TOTAL C) RATIO
C                           (CF. KEELING (1981))
C      *C12PAC*      REAL      MEAN C12 CONCENTRATION IN DEEP
C                           EASTERN PACIFIC BELOW 3000M
C      *C13PAC*      REAL      MEAN C13 CONCENTRATION IN DEEP EASTERN
C                           PACIFIC BELOW 3000M (UNCALIBRATED)
C      *NCOUNT*      INTEGER   COUNTS CARBON VALUES ADDED FOR MEAN VALUE
C      *C13FAC*      REAL      CONVERSION FACTOR C13 UNCALIBRATED INTO
C                           C13 ABSOLUTE
C      *C12ATL*      REAL      MEAN C12 CONCENTRATION IN NORTH EAST
C                           ATLANTIC SURFACE WATER
C      *C13ATL*      REAL      MEAN C13 CONCENTRATION (UNCALIBRATED)
C                           IN NORTH EAST ATLANTIC SURFACE WATER
C      *C14ATL*      REAL      MEAN C14 CONCENTRATION (UNCALIBRATED)
C                           IN NORTH EAST ATLANTIC SURFACE WATER
C                           (PRE ANTHROPOGENIC)
C      *D13ATL*      REAL      D13C VALUE FOR MEAN NORTH EAST ATLANTIC
C                           SURFACE WATER REL. TO PDR STANDARD
C      *D14ATL*      REAL      LITTLE D14 VALUE FOR MEAN NORTH EAST
C                           ATLANTIC SURFACE WATER REL. TO NBS
C                           STANDARD
C      *C14FAC*      REAL      CONVERSION FACTOR C14 UNCALIBRATED INTO
C                           C14 ABSOLUTE
C      *SUMM*        REAL      DUMMY ARGUMENT FOR SUMMATION OF
C                           ATMOSPHERIC CARBON VALUES
C
C      implicit real*8 (a,b,c,d)
C
C      PARAMETER(IE=72,JE=72,KE=11,NV=12,IPARA=42,DRY=1,LL11E=11,
C      1, ONE=1,FO=1,EJEKE=1,EJE*KE)
C      IMPLICIT HALF PRECISION (A-H,O-Z)
C      COMMON/TRACER/ ATCQ12,ATCQ13,ATCQ14,DGCO2(IE,JE),
C      1, POCO2A(IE,JE),PHSURF(IE,JE),PRORCA(IE,JE),PRC'ACA(IE,JE),
C      2, OCSD12(IE,JE),CCSD12(IE,JE),DFPTP(IE,JE),DFPTU(IE,JE),
C      3, SCO212(IE,JE,KE),ALKALI(IE,JE,KE),PHOSPH(IE,JE,KE),
C      4, OXYGEN(IE,JE,KE),POC12(IE,JE,KE),CALC12(IE,JE,KE),
C      5, SCO213(IE,JE,KE),SCO214(IE,JE,KE),POC13(IE,JE,KE),
C      6, POC14(IE,JE,KE),CALC13(IE,JE,KE),CALC14(IE,JE,KE),
C      7, SUPSAT(IE,JE,KE),RHVALU(IE,JE,KE),CARION(IE,JE,KE),
C      8, SOLPRO(IE,JE,KE),PRPHO(IE,JE,KE),RWAT(IE,JE),DDZ(IE,JE,KE),
C      9, UI(IE,JE,KE),VII(IE,JE,KE),T(IE,JE,KE),S(IE,JE,KE),
C      * PM(IE,JE,KE),C14AGE(IE,JE,KE),SILICA(IE,JE,KE)
C
C      X. OPSMA PREVENTS CALC12, POC12 FROM GETTING ZERO
C
C      OPSMA=1. E-14
C      DO 9999 K=1,KE
C      DO 9999 J=1,JE
C      DO 9999 I=1,IE
C          CALC12(I,J,K)=CALC12(I,J,K)*OPSMA
C          POC12(I,J,K)=POC12(I,J,K)*OPSMA
C 9999 CONTINUE
C
C      X. SET STANDARD CARBON ISOTOPE RATIOS
C
C      RE1312=0.0112372
C      RE14TO=1.176E-12
C
C      X. SET PREINDUSTR. D13C AND BIG D14C IN ATMOSPHERE
C
C      PRE1313=6.5
C      PRE14=0.
C
C      X. ABSOLUTE ATM. D13 CONCENTRATION FOR D13C-PRE13
C
C      BETA=(PRE1313/1000.)+1.
C      C13ATM=BETA*RE1312*ATCQ12
C      * / (1.+BETA*PRE1312)
C
C      X. FACTOR FOR CALIBRATING C13
C
C      C13FAC=C13ATM/ATCQ13
C      PRINT*, 'CALIBRATION FACTOR FOR C13:'
C      PRINT*, 'C13FAC=',C13FAC
C
C      X. LITTLE D14C IN ATM. FOR BIG DELTA 14C IN ATM. = -50
C
C      ALPHA=2.* (PRE14+25.1)
C      D14CAT=(PRE14+ALPHA)/(1.-ALPHA/1000.)
C
C      X. ABSOLUTE 14C CONCENTRATION IN PREINDUSTR. ATMOSPHERE
C
C      C14ATM=((D14CAT/1000.)+1.)*RE14TO*ATCQ12
C
C      X. FACTOR FOR CALIBRATING C14
C
C      C14FAC=C14ATM/ATCQ14
C      PRINT*, 'CALIBRATION FACTOR FOR C14:'
C      PRINT*, 'C14FAC=',C14FAC
C
C      X. ABSOLUTE 14C (OCEAN)
C
C      END OF FILE

```

plofil11.f

```

DO 3 J=1,JE
DO 3 I=1,IE
SC0214(I,J,K)=SC0214(I,J,K)*C14FAC
CALC14(I,J,K)=CALC14(I,J,K)*C14FAC
POC14(I,J,K)=POC14(I,J,K)*C14FAC
3 CONTINUE
C -----
C*      X. LITTLE D14C CALIBRATED (OCEAN)
C
DO 4 K=1,KE
DO 4 J=1,JE
DO 4 I=1,IE
SC0214(I,J,K)=((SC0214(I,J,K)/
1   (SC0212(I,J,K))/
2   RE14TO-1.E0)*1000.E0
CALC14(I,J,K)=((CALC14(I,J,K)/
1   (CALC12(I,J,K))/
2   RE14TO-1.E0)*1000.E0
POC14(I,J,K)=((POC14(I,J,K)/
1   (POC12(I,J,K))/
2   RE14TO-1.E0)*1000.E0
4 CONTINUE
C -----
C*      X. ABSOLUTE 13C (OCEAN)
C
DO 5 K=1,KE
DO 5 J=1,JE
DO 5 I=1,IE
SC0213(I,J,K)=SC0213(I,J,K)*C13FAC
CALC13(I,J,K)=CALC13(I,J,K)*C13FAC
POC13(I,J,K)=POC13(I,J,K)*C13FAC
5 CONTINUE
C -----
C*      X. D13C CALIBRATED (OCEAN)
C
DO 6 K=1,KE
DO 6 J=1,JE
DO 6 I=1,IE
SC0213(I,J,K)=
1   ((SC0213(I,J,K)/(SC0212(I,J,K)-SC0213(I,J,K)))/RE1312-
2   1.E0)*1000.E0
CALC13(I,J,K)=
1   ((CALC13(I,J,K)/(CALC12(I,J,K)-CALC13(I,J,K)))/RE1312-
2   1.E0)*1000.E0
POC13(I,J,K)=
1   ((POC13(I,J,K)/(POC12(I,J,K)-POC13(I,J,K)))/RE1312-
2   1.E0)*1000.E0
6 CONTINUE
C -----
C*      X. BIG D14C CALIBRATED (OCEAN)
C
DO 7 K=1,KE
DO 7 J=1,JE
DO 7 I=1,IE
SC0214(I,J,K)=SC0214(I,J,K)-2.E0*
1   (SC0213(I,J,K)+25.E0)*
2   (1.E0+SC0214(I,J,K)/1000.E0)
CALC14(I,J,K)=CALC14(I,J,K)-2.E0*
1   (CALC13(I,J,K)+25.E0)*
2   (1.E0-CALC14(I,J,K)/1000.E0)
POC14(I,J,K)=POC14(I,J,K)-2.E0*
1   (POC13(I,J,K)+25.E0)*
2   (1.E0-POC14(I,J,K)/1000.E0)
7 CONTINUE
C -----
C*      X. MEAN CARBON VALUES FOR ATMOSPHERE
C
PRINT*, 'ATMOSPHERE:'
PRINT*, '    C12= ', ATCQ12
PRINT*, '    C13 UNCAL. = ', ATCQ13
PRINT*, '    C14 UNCAL. = ', ATCQ14
C -----
C*      X. ABSOLUTE 14C (ATMOSPHERE)
C
ATCQ14=ATCQ14*C14FAC
PRINT*, '    C14 CAL. = ', ATCQ14
C -----
C*      X. LITTLE D14C CALIBRATED (ATMOSPHERE)
C
ATCQ14=((ATCQ14/(ATCQ12))/RE14TO-1.E0)*
1   1000.E0
PRINT*, '    LITTLE DC14 = ', ATCQ14, ' CALIB.'
C -----
C*      X. ABSOLUTE 13C (ATMOSPHERE)
C
ATCQ13=ATCQ13*C13FAC
PRINT*, '    C13 CAL. = ', ATCQ13
C -----
C*      X. D13C CALIBRATED (ATMOSPHERE)
C
ATCQ13=((ATCQ13/(ATCQ12-ATCQ13))/RE1312-1.E0)*1000.E0
PRINT*, '    DC13 CAL. = ', ATCQ13
C -----
C*      X. BIG D14C CALIBRATED (ATMOSPHERE)
C
ATCQ14=ATCQ14*2.E0*(ATCQ13/25.E0)*(1.E0+ATCQ14/1000.E0)
PRINT*, '    BIG DC14 CAL. = ', ATCQ14
C
RETURN
END

```

plotters.f

```

SUBROUTINE PLOTMAP(IFR)
C
C Declare required data arrays and workspace arrays.
C
INCLUDE 'data.txt'
DIMENSION IASF(13), LIND(14)
DATA IASF / 13*1 /
DATA LIND / 2,3,4,5,6,7,8,9,10,11,12,13,14,15 /
C
C Declare arrays to hold the list of indices and the list of labels
C required by the label-bar routine.
C
C LLBS CONTROLS THE NUMBER OF CHARACTERS PRINTED OUT FOR THE LABEL BAR.
C AT THE PRESENT TIME IT IS SET TO SHOW ONLY FOUR CHARACTERS. THIS MAY
C HAVE TO BE CHANGED FOR DIFFERENT DATA.
C
CHARACTER*4 LLRG(15)
CHARACTER CD*8, CT*10, TEXT3*12
EXTERNAL COLRAM
C
CALL DATE(CDI)
C
C Turn off the clipping indicator.
C
CALL GSCLIP (0)
C
C Set all aspect source flags to "individual".
C
CALL GSASF (IASF)
C
C Force solid fill.
C
CALL GSFAIS (1)
C
C Define color indices.
C
CALL DFCLRS
C
C Get the current elapsed time, in seconds.
C
TIME=SECOND(DUMI)
C
C Force the plot into the portion of the frame explicitly defined
C by the calls to set the bottom,top,left and right of the
C viewport.
C
CALL CPSETR('VPS - VIEWPORT SHAPE',0.)
CALL CPSETR('VPT - VIEWPORT TOP',.85)
CALL CPSETR('VPB - VIEWPORT BOTTOM',.35)
CALL CPSETR('VPL - VIEWPORT LEFT',.01)
CALL CPSETR('VPR - VIEWPORT RIGHT',.99)
C
C Disallow the trimming of trailing zeroes.
C
CALL CPSETI ('NOF - NUMERIC OMISSION FLAGS',0)
CALL CPSETR ('SPV - SPECIAL VALUE FLAG',RMASK)
C
C Tell CONPACK to use 13 contour levels, splitting the range into 14
C equal bands, one for each of the 14 colors available.
C
CALL CPSETI ('CLS - CONTOUR LEVEL SELECTOR',NUMCONT)
CALL CPSETR ('T2D - TWO-DIMENSIONAL SMOOTHING',2.5)
C
C Initialize the drawing of the contour plot.
C
C
C CALL CPRECT (ZDAT, IE, IE, JE, RWRK, IWA, IWRK, IWA)
C
C Initialize the area map and put the contour lines into it.
C
CALL ARINAM (IAMA,IMA)
CALL CPCLM (ZDAT,RWRK,IWRK,IAMA)
C
C Color the map.
C
CALL ARSCAM (IAMA,XCRA,YCRA,IWA,IATA,IGIA,NS,COLRAM)
C
C Put black contour lines over the colored map.
C
CALL GSPLCI (0)
CALL CPCLDR (ZDAT,RWRK,IWRK)
CALL GSPLCI (1)
C
C Draw a label bar for the plot, relating colors to values.
C
CALL CPGETR ('ZMN',ZMIN)
CALL CPGETR ('ZMX',ZMAX)
C
DO 102 I=1,15
    CALL CPSETR ('ZDV - Z DATA VALUE',
                 ZMIN+REAL(I-1)*(ZMAX-ZMIN)/14.)
    CALL CPGETC ('ZDV - Z DATA VALUE',LLRS(I))
102  CONTINUE
C
CALL LBSETI ('CBL - COLOR OF BOX LINES',1)
CALL LBSETI ('CLB - COLOR OF LABELS',ITEXTCLR)
CALL LBMBAR (0,.05,.95,.15,.25,14,1..5,LIND,0,LLRG,15,1)
C
C Compute and print statistics for the plot, label it, and put a
C boundary line at the edge of the plotter frame.
C
CALL CAPSAP (HEADER,TIME,IAMA,IMA,LAYFR)
CALL LABTOP (HEADER,.017)
WRITE(TEXT3,'(F10.0,A2)') ZLAYER(LAYER),' M'
CALL LABELMAP(CD,'DEPTH = ',TEXT3,'PLOTTED ','CREATED ',
              TEXT6,.011,.013,.011,.011,IFR,TITLE,.011)
CALL PLOTNDY
CALL BNDRY
CALL MAPOVR(ITEXTCLR)
C
C WRITING A TEXT FILE FOR THE QUICK.F PROGRAM TO USE. THIS WILL
C GIVE UNIFORM LABELING ON THE PLOTS OF ALL THREE TYPES.
C
OPEN(22,FILE='QUICKLAB',STATUS='UNKNOWN',FORM='FORMATTED')
WRITE(22,'(A8,A8)') 'CREATED ',TEXT6
WRITE(22,'(A40)') TITLE
CLOSE(22)
RETURN
END
C
C
C SUBROUTINE COLPAM (XCRA,YCRA,NCPA,IATA,IGIA,NATA)
C
DIMENSION XCRA(*),YCRA(*),IATA(*),IGIA(*)
C
C The arrays XCRA and YCRA, for indices 1 to NCPA, contain the X and Y
C coordinates of points defining a polygon. The area identifiers in
C the array IATA, each with an associated group identifier in the array
C IGIA, tell us whether the polygon is to be color-filled or not.

```

plotters.f

```

C Assume the polygon will be filled until we find otherwise.
C
C IFM,-1
C
C If any of the area identifiers is negative, don't fill the polygon.
C
DO 101 I=1,NAIA
  IF (IAIA(I).LT.0) IFLL=0
101  CONTINUE
C
C Otherwise, fill the polygon in the color implied by its area
C identifier relative to edge group 3 (the contour-line group).
C
IF (IFLL.NE.0) THEN
  IFLL=0
  DO 102 I=1,NAIA
    IF (IGIA(I).EQ.3) IFLL=IAIA(I)
102  CONTINUE
    IF (IFLL.GT.0.AND.IFLL.LT.15) THEN
      CALL GSPACI (IFLL+1)
      CALL GFA (NCRA,1,XCPA,YCPA)
    END IF
  END IF
C
C Done.
C
      RETURN
C
      END
C-----C
C-----C
C-----C
      SUBROUTINE LABELMAP(TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6,SIZE1,
     *      SIZE2,SIZE3,SIZE4,SIZE5,SIZE6,IFR,TEXT7,SIZE7)
C
C THIS PROCEDURE IS USED TO PLACE THE DEPTH AND THE DATES ON THE
C MAP. IT IS SIMILAR TO THE PROCEDURE TO PLOT THE HEADER ON THE MAP.
C
CHARACTER*(*) TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6,TEXT7
C
CALL GETSET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDR,YWDT,LNLC)
CALL      SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PCGETI ('QU - QUALITY FLAG',IQUA)
CALL PCSETI ('QU - QUALITY FLAG',0)
CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
DO 300 I=1,?
  GOTO (1,2,3,4,5,6,7) I
1   CALL PLCHHQ(.380,.100,TEXT1,SIZE1,0.,+1.5)
2   CONTINUE
    IF(IFR.LT.20) THEN
      CALL PLCHHQ(.540,.100,TEXT2,SIZE2,0.,+1.5)
    ENDIF
  GOTO 300
3   CONTINUE
  IF(IFR.LT.20) THEN
    CALL PLCHHQ(.690,.300,TEXT3,SIZE3,0.,+1.5)
  ENDIF
  GOTO 300
4   CALL PLCHHQ(.225,.100,TEXT4,SIZE4,0.,+1.5)
5   CALL PLCHHQ(.780,.100,TEXT5,SIZE5,0.,+1.5)
6   CALL PLCHHQ(.910,.100,TEXT6,SIZE6,0.,+1.5)
7   CALL PLCHHQ(.540,.030,TEXT7,SIZE7,0.,+1.5)
300  CONTINUE

```

```

CALL PCSETI ('QU - QUALITY FLAG', IQUA)
CALL      SET (XVPL, XVPR, YVPR, YVPT, XWDL, XWDR, YWDR, YWDT, LNUGI)

RETURN
END

SUBROUTINE PLOTBNDY

THIS SUBROUTINE PUTS A CYAN BORDER AROUND THE MAP PART OF THE PLOT.

PLOTIF IS A SPPS CALL AND MAY NEED TO BE CHANGED AT A LATER DATE.

CALL GSFLCI(1)
CALL PLOTIF (.99,.85,0)
CALL PLOTIF (.99,.35,1)
CALL PLOTIF (.01,.35,1)
CALL PLOTIF (.01,.85,1)
CALL PLOTIF (.99,.85,1)
CALL PLOTIF (.99,.85,2)

RETURN
END

SUBROUTINE PLOTSEC(XTOP,XBOT,YLFT,YRGT,ICR,DEPTH)

Declare required data arrays and workspace arrays.

INCLUDE 'data2.txt'
DIMENSION IASF(13), LIND(14), DEPTH(IMAX,JMAX)
DATA IASF / 13*1 /
DATA LIND / 2,3,4,5,6,7,8,9,10,11,12,13,14,15 /

Declare arrays to hold the list of indices and the list of labels
required by the label-bar routine.

CHARACTER*10 LLBS(15)
CHARACTER CD*8, CT*10, TEXT3*12
EXTERNAL COLPAS

CALL DATE(CD)

Turn off the clipping indicator.

CALL GSCLIP (0)

Set all aspect source flags to "individual".

CALL GSASF (IASP)

Force solid fill.

CALL GSFAIS (1)

Define color indices.

CALL DFCLRS

Get the current elapsed time, in seconds.

TIME=SECOND(DUMI)

```

```

C Force the plot into the portion of the frame explicitly defined
C by the calls to set the bottom,top,left and right of the
C viewport.
C
CALL CPSETR('VPS - VIEWPORT SHAPE',0.)
CALL CPSETR('VPT - VIEWPORT TOP',XTOP)
CALL CPSETR('VPB - VIEWPORT BOTTOM',XBOT)
CALL CPSETR('VPL - VIEWPORT LEFT',YLFT)
CALL CPSETR('VPR - VIEWPORT RIGHT',YRGT)
C
C Disallow the trimming of trailing zeroes.
C
CALL CPSETI ('NOF - NUMERIC OMISSION FLAGS',0)
CALL CPSETR ('SPV - SPECIAL VALUE FLAG',RMASK)
C
C Tell CONPACK to use 13 contour levels, splitting the range into 14
C equal bands, one for each of the 14 colors available.
C
CALL CPSETI('CLS - CONTOUR LEVEL SELECTOR',NUMCONT)
CALL CPSETR('T2D - TWO-DIMENSIONAL SMOOTHING',2.5)
C
C Initialize the drawing of the contour plot.
C
CALL CPRECT (ZDAT,JE,JE,KE,RWRK,IWA,IWRK,IWA)
C
C Initialize the area map and put the contour lines into it.
C
CALL ARTNAM (IAMA,IMA)
CALL CPCLM (ZDAT,PWRK,IWRK,IAMA)
C
C Color the map.
C
CALL ARSCAM (IAMA,XCRA,YCRA,IWA,IAIA,IGIA,NS,COLRAC)
C
C Put black contour lines over the colored map.
C
CALL GSPLCI (0)
CALL CPCLDR (ZDAT,RWRK,IWRK)
CALL GSPLCI (1)
C
C Draw a label bar for the plot, relating colors to values, putting the
C values into the array LLBS
C
CALL CPGETR ('ZMN',ZMIN)
CALL CPGETR ('ZMX',ZMAX)
C
DO 102 I=1,15
CALL CPSETI ('ZDV - Z DATA VALUE',
              ZMIN+REAL(I-1)*(ZMAX-ZMIN)/14.)
CALL CPGETR ('ZDV - Z DATA VALUE',HOLD)
IF (HOLD.GT.10.OR.HOLD.LT.-10.) THEN
  WRITE(LLBS(I),'(F6.1)') HOLD
ELSE IF (HOLD.GT.0) THEN
  WRITE(LLBS(I),'(F4.2)') HOLD
ELSE
  WRITE(LLBS(I),'(F4.1)') HOLD
ENDIF
102 CONTINUE
C
CALL LBSETI ('CBL - COLOR OF BOX LINES',1)
CALL LBLBAR (0,.05,.95,.15, 25,14,1,, 5,LIND,0,LLBS,15,1)
C
C Compute and print statistics for the plot, label it, and put a
C boundary line at the edge of the plotter frame.
C

```

```

CALL CAPSAP (HEADER,TIME,IAMA,IMA,LAYER)
CALL LABTOP (HEADER,.017)
CALL LABELSEC(CD,'PLOTTED ','CREATED ',TEXT6, 011, 011,
             .011,.011,TITLE, 011,SECNAM,.013)
CALL LABELAXIS(ZLAYER,KE,XTOP,XBOT,YLFT,YRGT)
CALL PLOTNDRY(XTOP,XBOT,YLFT,YRGT)
CALL LITMAPNY
CALL BNDARY
CALL SECovR(INDISU,ICR)
CALL GCLRWK(1,0)
RETURN
END

C
C -----
C
SUBROUTINE COLRAS (XCRA,YCRA,NCPA,IAIA,IGIA,NAIA)
C
DIMENSION XCRA(*),YCRA(*),IAIA(*),IGIA(*)
C
C The arrays XCRA and YCRA, for indices 1 to NCPA, contain the X and Y
C coordinates of points defining a polygon. The area identifiers in
C the array IAIA, each with an associated group identifier in the array
C IGIA, tell us whether the polygon is to be color-filled or not.
C
C
C Assume the polygon will be filled until we find otherwise.
C
IFLL=1
C
C If any of the area identifiers is negative, don't fill the polygon
C
DO 101 I=1,NAIA
  IF (IAIA(I).LT.0) IFLL=0
101  CONTINUE
C
C Otherwise, fill the polygon in the color implied by its area
C identifier relative to edge group 3 (the contour-line group).
C
IF (IFLL.NE.0) THEN
  IFLL=0
  DO 102 I=1,NAIA
    IF (IGIA(I).EQ.3) IFLL=IAIA(I)
102  CONTINUE
    IF (IFLL.GT.0.AND.IFLL.LT.15) THEN
      CALL GSFACT (IFLL+1)
      CALL GFA (INCRA-1,XCRA,YCRA)
    END IF
  END IF
C
C Done.
C
RETURN
C
C -----
C
SUBROUTINE LABELSEC(TEXT1,TEXT4,TEXT5,TEXT6,SIZE1,
                     SIZE4,SIZE5,SIZE6,TEXT7,SIZE7,TEXT8,SIZE8)
C
C THIS PROCEDURE IS USED TO PLACE THE SECTION TITLE AND THE DATES ON
C THE MAP. IT IS SIMILAR TO THE PROCEDURE TO PLOT THE HEADER ON THE
C MAP.
C
CHARACTER*1 TEXT1,TEXT4,TEXT5,TEXT6,TEXT7,TEXT8

```

plotters.f

```

C
      CALL GETSET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
      CALL SET (0..1..0..1..0..1..0..1..1)
      CALL PCGETI ('QU - QUALITY FLAG',IQUA)
      CALL PCSETI ('QU - QUALITY FLAG',0)
      CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
      DO 300 I=1,6
        GOTO (1,2,3,4,5,6) I
        1 CALL PLCHHQ(.380,.100,TEXT1,SIZE1,0.,+1.5)
        GOTO 300
        2 CALL PLCHHQ(.225,.100,TEXT4,SIZE4,0.,+1.5)
        GOTO 300
        3 CALL PLCHHQ(.780,.100,TEXT5,SIZE5,0.,+1.5)
        GOTO 300
        4 CALL PLCHHQ(.910,.100,TEXT6,SIZE6,0.,+1.5)
        GOTO 300
        5 CALL PLCHHQ(.640,.030,TEXT7,SIZE7,0.,+1.5)
        GOTO 300
        6 CALL PLCHHQ(.780,.100,TEXT8,SIZE8,0.,+1.5)
300  CONTINUE
      CALL PCSETI ('QU - QUALITY FLAG',IQUA)
      CALL SET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C
      RETURN
      END
C
C-----.
C
      SUBROUTINE LABELAXIS(ZLAYER,KE,XTOP,XBOT,YLFT,YRHT)
C
C THIS SUBROUTINE IS USED TO LABEL THE X AND Y AXIS OF THE SECTON PLOTS
C PUTTING THE DEPTH ON THE Y AXIS.
C
C THIS IS HAMBURG DEPENDANT.
C
      DIMENSION ZLAYER(0:12)
      CHARACTER*4 XAXISLAB(7)
      CHARACTER*9 DEPTH(11)
      CHARACTER*2 DASH
C
      CALL GETSET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
      CALL SET (0..1..0..1..0..1..0..1..1)
      CALL PCGETI ('QU - QUALITY FLAG',IQUA)
      CALL PCSETI ('QU - QUALITY FLAG',0)
      CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
      DO 1 L=1,11
        WRITE(DEPTH(L),'(F6.0,A3)') ZLAYER(L), ' --'
1     CONTINUE
      YCOR = XTOP
      XCOR = YLFT + .035
      DO 100 K=1,KE
        CALL PLCHHQ(XCOR,YCOR,DEPTH(K),.008,0.,+1.5)
        YCOR = YCOR - .04985
100   CONTINUE
      CALL PLCHHQ(.050,.700,'DEPTH M',.009,90.,+1.5)
      XAXISLAB(1) = '90 S'
      XAXISLAB(2) = '60 S'
      XAXISLAB(3) = '30 S'
      XAXISLAB(4) = ' EQ '
      XAXISLAB(5) = '30 N'
      XAXISLAB(6) = '60 N'
      XAXISLAB(7) = '90 N'
      DASH = '--'
      XCORD = YLFT
      YCORD = XBOT+.02

```

quicklook.f

```

PROGRAM LYPROF
C THIS PROGRAM IS DESIGNED TO TAKE THE INFORMATION FROM THE HAMBURG
C OCEANIC CARBON CYCLE CIRCULATION MODEL AND COMPARE THE MODEL DATA
C TO THE GEOSCS DATA. THIS PROGRAM IS BASICALLY HAMBURG DEPENDENT.
C
C THIS PROGRAM USES THE NCAR PLOT PACKAGE AUTOGRAPH FOR THE PLOTTING
C OF THE PHYSICAL DATA AND THE NCAR PACKAGE PLOTCHAR FOR THE LABELS
C NOT LOCATED ON THE PLOTS.
C
PARAMETER ( KE = 11, MODCOLOR = 5, IGEOCOLOR = 14,
*           ITEXTCOLOR = 1, RMODLNWD = 1., GEOLNWD = 3.,
*           TEXTLNWD = 1., LNTYPE = 1)
DIMENSION BNDRY(5,4)
DIMENSION CPROF(12,KE,11),SPROF(12,KE,11)
CHARACTER*40 TITLE
CHARACTER*16 TEXT10
CHARACTER*20 TEXBEC(10), TEXVAR(5), BLA
CHARACTER*15 TEXT
CHARACTER*10 TEXT2
CHARACTER*1 GLAB
DIMENSION DEPMO(KE), GEODEP(20), KBGTR(10), KBRTG(11), FIRVA(5),
*           DVAL(5), XMOD(100), YMOD(100), CONC(20,5,10), IVGTR(5),
*           IVRTG(5), XGEO(100), YGEO(100)
DATA KBGTR/5,11,1,3,7,8,9,2,6,4/
DATA FIRVA/2000.,1900.,0.,0.,0./
DATA DVAL/100.,100.,50.,0.5,50./
DATA KBRTG/3,8,4,8,1,8,5,6,7,2,8/
DATA IVGTR/2,1,4,3,8/
DATA IVRTG/2,5,1,4,3/
DATA DEPMO/25.,75.,150.,250.,450.,700.,
1 1000.,2000.,3000.,4000.,5000./
DATA GEODEP/12.,37.,63.,87.,150.,250.,350.,450.,
1 550.,650.,750.,850.,950.,1250.,1750.,2500.,3500.,4500.,5500.,
2 6500./
C -- OPEN GKS AND SET UP THE COLOR TABLE --
C
CALL GOPKS(6, IDUM)
CALL GOPWK(1,2,1)
CALL GACWK(1)
CALL DFCLRS
OPEN(10,FILE='quick.dat',STATUS='UNKNOWN',FORM='FORMATTED')
OPEN(11,FILE='PROFIL',STATUS='UNKNOWN',FORM='FORMATTED')
CALL GSPLCI(TEXTCOLOR)
CALL GSLWSC(TEXTLNWD)
CALL AGSETF('FRAME.',2.)
C -- SET UP THE FIVE WINDOWS POSITIONS IN THE PLOTTER FRAME --
C
BNDRY(1,1) = .67
BNDRY(1,2) = .97
BNDRY(1,3) = .45
BNDRY(1,4) = .15
BNDRY(2,1) = .35
BNDRY(2,2) = .63
BNDRY(2,3) = .45
BNDRY(2,4) = .15
BNDRY(3,1) = .03
BNDRY(3,2) = .33
BNDRY(3,3) = .45
BNDRY(3,4) = .15
BNDRY(4,1) = .67
BNDRY(4,2) = .97
BNDRY(4,3) = .85
BNDRY(4,4) = .55
BNDRY(5,1) = .35
BNDRY(5,2) = .65
BNDRY(5,3) = .85
BNDRY(5,4) = .55
C
C -- READ FROM LYPROF DATA (quick.dat) --
C
READ(10,501) TEXBEC
READ(10,501) TEXVAR
501 FORMAT(A20)
C
C -- READ FROM PROFIL --
C
READ(11,7100,ERR=3000) CPROF,SPROF
7100 FORMAT(6E12.5)
C
C -- MANIPULATE DATA ACCORDING TO THE ORIGINAL MODEL.
C
DO 33 KB=1,11
DO 33 KT=1,KE
DO 33 IV=1,12
IF(IV.NE.8) THEN
SPROF(IV,KT,KB)=SPROF(IV,KT,KB)*1.E6
CPROF(IV,KT,KB)=CPROF(IV,KT,KB)*1.E6
ELSE
CPROF(IV,KT,KB)=1000.*CPROF(IV,KT,KB)
SPROF(IV,KT,KB)=1000.*SPROF(IV,KT,KB)
ENDIF
IF(IV.EQ.3) CPROF(IV,KT,KB)=CPROF(IV,KT,KB)/122
IF(IV.EQ.3) SPROF(IV,KT,KB)=SPROF(IV,KT,KB)/122
CONTINUE
33
C
C START OF PLOTTING PART
C
DO 10 KB=1,10
C
C PLOTTING OF THE LABEL FOR ALL THE PLOTS USING *PLOTCHAR*
C
OPEN(22,FILE='QUICKLAB',STATUS='UNKNOWN',FORM='FORMATTED')
READ(22,'(A16)') TEXT10
READ(22,'(A40)') TITLE
CLOSE(22)
CALL GSPLCI(TEXTCOLOR)
CALL GSLSN(LNTYPE)
CALL GSLSNC(TEXTLNWD)
CALL GETSET (XVPL,XVPR,YVPL,YVPR,XWDL,XWDR,YWDL,YWDR,TMDT,LMDT)
CALL SET (0.,1.,0.,1.,0.,1.,1.,1.)
CALL PGSETI ('QU' QUALITY FLAG,IQUA)
CALL PGSETI ('QU' QUALITY FLAG,IQU)
CALL PGSETI ('TE' TEXT EXTENT COMPUTATION FLAG,9)
CALL PLCHIQ (1000, 800,'MEAN PROFILE', 013,0, -1,1)
CALL PLCHIQ (1000, 720,'GEODESIS THICK RIDGE', 013,0, -1,1)
CALL PLCHIQ (1000, 720,'MODEL THIN BLUE', 014,0, -1,1)
CALL PLCHIQ (1000, 680,TEXBEC(PBEC), 010,0, -1,1)
CALL PLCHIQ (1000, 680,TEXT10, 010,0, -1,1)
CALL PLCHIQ (1000, 650,TITLE, 008,0, -1,1)
CALL PGSETI ('QU' QUALITY FLAG,IQU)
CALL SET (XVPL,XVPR,YVPL,YVPR,XWDL,XWDR,YWDL,YWDR,TMDT,LMDT)
C
C READ GEOSCS DATA FROM THE LYPROF INPUT (quick.dat)
C
READ(11,501) BLA
501 FT=1,10
READ(11,501) CPROF(1,FT,1)

```

quicklook.f

```

1      CONTINUE
C
C      IRB = KBGTR(KBEC)
C
C -- START OF EACH FRAME --
C
DO 20 ISEC=1,5
NUM = 0
C
C -- SET UP THE BACKGROUND TO BE HALF AXIS AND DIMENSION THE WINDOW --
C
CALL AGSETF('BACKGROUND.',3.)
CALL AGSETF('GRAPH/LEFT.',BNDRY(ISEC,1))
CALL AGSETF('GRAPH/RIGHT.',BNDRY(ISEC,2))
CALL AGSETF('GRAPH/TOP.',BNDRY(ISEC,3))
CALL AGSETF('GRAPH/BOTTOM.',BNDRY(ISEC,4))
C
C -- SET UP THE LEFT LABEL --
C
      WRITE(TEXT2,'(A10)') 'DEPTH (KM)'
CALL AGSETC('LABEL/NAME.','L')
CALL AGSETI('LINE/NUMBER.',100)
CALL AGSETC('LINE/TEXT.',TEXT2)
C
C -- SET UP THE BOTTOM LABEL --
C
CALL AGSETC('LABEL/NAME.','B')
CALL AGSETI('LINE/NUMBER.',-100)
WRITE(TEXT,'(A15)') TEXVAR(ISEC)
CALL AGSETC('LINE/TEXT.',TEXT)
C
C -- BEGIN PREPARATION FOR PLOTTING OF MODEL DATA --
C
X00 = 18.-MOD(ISEC-1,3)*8
Y00 = 2.+8.*((ISEC-1)/3)
BEGI = FIRVA(ISEC)
DISTA = DVAL(ISEC)
IVR = IVGTR(ISEC)
IP = 3
NUM = NUM + 1
XMOD(NUM) = CPROF(IVR,1,IRB)
YMOD(NUM) = -DEPMO(1)*1E-3
DO 43 K= 2,KE
IF (CPROF(IVR,K,IRB).EQ.0.) GOTO 43
NUM = NUM + 1
XMOD(NUM) = CPROF(IVR,K,IRB)
YMOD(NUM) = -DEPMO(K)*1E-3
NUM = NUM + 1
XMOD(NUM) = CPROF(IVR,K,IRB)
YMOD(NUM) = -DEPMO(K)*1E-3
43  CONTINUE
C
C -- BEGIN PREPARATION FOR PLOTTING OF GESECS DATA --
C
NDAT = 1
XGEO(NDAT) = CONC(1,ISEC,KBEC)
YGEO(NDAT) = -GEODEP(1)*1E-3
DO 44 K=2,19
IP = 3
K= X00+1.+(CONC(K,ISEC,KBEC)-BEGI)/DISTA
IF(X.GT.X00+1.) IP = 2
IF (X.LT.10000.) THEN
NDAT = NDAT + 1
XGEO(NDAT) = CONC(K,ISEC,KBEC)
YGEO(NDAT) = -GEODEP(K)*1E-3
ENDIF
CONTINUE
44  CONTINUE
ENDIF
C
C -- PLOT MODEL DATA AND SET APPROPRIATE GKS POLYLINES ATTRIBUTES
C
YMN = -6000.*1E 3
YMX = 0.
CALL GSPLCI(MODCOLOR)
CALL GSWSFC(RMOD,NN)
CALL GSIN(LNTYPE)
CALL MINMAX1(XMOD,XGEO,NDAT,NUM,RMAX,RMIN)
CALL AGSEFF('X/MINIMUM.',RMIN)
CALL AGSEFF('X/MAXIMUM.',RMAX)
CALL AGSEFF('Y/MINIMUM.',YMN)
CALL AGSEFF('Y/MAXIMUM.',YMX)
CALL EZXY(XMOD,YMOD,NUM,GLAB)
C
C -- PLOT GESECS DATA AND SET APPROPRIATE GKS POLYLINES ATTRIBUTES
C
CALL GSPLCI(IGEOCOLOR)
CALL GSIN(LNTYPE)
CALL GSWSFC(GEOINWD)
CALL AGSEFF('BACKGROUND.',4.)
CALL AGSEFF('X/MINIMUM.',RMIN)
CALL AGSEFF('X/MAXIMUM.',RMAX)
CALL AGSEFF('Y/MINIMUM.',YMN)
CALL AGSEFF('Y/MAXIMUM.',YMX)
CALL EZXY(XGEO,YGEO,NDAT,GLAB)
20  CONTINUE
C
C Advance to a new frame.
C
CALL GCLRWK(1,0)
10  CONTINUE
CLOSE(10)
CLOSE(11)
C CLOSE(2)
GOTO J010
3000 WRITE(*,*) 'ERROR IN READING PROFILE'
STOP
3010 CONTINUE
CALL GUAWK(1)
CALL GCIAWK(1)
CALL GCLRS
END
SUBROUTINE MINMAX1(X,XG,NDAT,NUM,RMAX,RMIN)
DIMENSION X(NDAT), XG(NDAT)
RMIN = 1.11E+15
RMAX = 1.1111E+15
DO 1001 J=1,NDAT
IF (RMIN.GT.XG(J)) THEN
RMIN = XG(J)
ENDIF
IF (RMAX.LT.XG(J)) THEN
RMAX = XG(J)
ENDIF
1001 CONTINUE
DO 1002 I = 1,NUM
C
IF (RMIN.GT.X(I)) THEN
RMIN = X(I)
ENDIF

```

quicklook.f

```
ENDIF
IF (RMAX.LT.X(I)) THEN
    RMAX = X(I)
ENDIF
1002 CONTINUE
DELTA = ABS(RMAX-RMIN)*0.10
RMIN = RMIN - DELTA
RMAX = RMAX + DELTA
RETURN
END
```

readat.f

```

SUBROUTINE REFINEDATA(AVERAGE, IT)
C
C THIS SUBROUTINE WILL TAKE THE DATA READ FROM THE READINFO SUBROUTINE
C SEND IT THROUGH THE SUBROUTINE REFINEMENT TO MAKE THE
C ARRAY SHIFT SO THAT THE CONTINENTS ARE NOT SKewed AND THE MAP IS CUT
C AT THE PRIME MERIDIAN.
C
C EXTERNAL SUBROUTINES INCLUDE : REFINEMENT
C                               AVG1
C
C THE INCLUDE FILE *DATA.TXT* IS USED FOR CERTAIN VARIABLES THAT WERE
C NEEDED FOR THE REFINEMENT PROCESS AND THE NCAR INTERFACE.
C
C ALL REFINEMENTS ARE PRESENTLY DISABLED.
C AFFECTED SUBROUTINES (FROM THIS SUBROUTINE) :
C     AVG1
C
C include 'data.txt'
DIMENSION VALUES(IMAX,JMAX),ATEMP(IMAX),CHECK(IMAX,JMAX)
DIMENSION_DEPTH(IMAX,JMAX),TEMP1(IE,JE),DEPTP(IE,JE)
REAL*4 VALUES,DEPTH,TEMP1,AVERAGE
C
C DO 1 I=1,IE
DO 1 J=1,JE
    TEMP1(I,J) = 0.
    ZDAT(I,J) = 0.
1 CONTINUE
DO 2 I=1,IE
DO 2 J=1,JE
    VALUES(I,J) = 0.
    DEPTH(I,J) = 0.
2 CONTINUE
C CALL AVG1(CHOOSEN, IE, JE, AVERAGE, RMASK)
DO 24 I=1,IE
DO 24 J=1,JE
    IF(DEPTP(I,J).EQ.0.) THEN
        TEMP1(I,J) = AVERAGE
    ELSE
        TEMP1(I,J) = CHOSSEN(I,J)
    ENDIF
24 CONTINUE
DO 25 I=1,IE
DO 25 J=1,JE
    DEPTH(I,J) = DEPTP(I,J)
    VALUES(I,J) = TEMP1(I,J)
25 CONTINUE
CALL REFINEMENT(VALUES,DEPTH,CHECK,AVERAGE)
DO 201 I=1,IE
JP = JE
DO 202 J=1,JE
    IF(CHECK(I,JP).EQ.RMASK) THEN
        IF (CHECK(I,JP).EQ.VALUES(I,JP)) THEN
            ZDAT(I,J) = VALUES(I,JP)
        ELSE
            ZDAT(I,J) = CHECK(I,JP)
        ENDIF
    ELSE
        ZDAT(I,J) = VALUES(I,JP)
    ENDIF
    JP = JP - 1
202 CONTINUE
201 CONTINUE
C
C CLOSE(87)
RETURN
END
C
C SUBROUTINE AVG1(SARRAY, IFRST, ISEC, AVERAGE, RMASK)
C
C THIS SUBROUTINE IS USED TO FIND THE AVERAGE OF ANY TWO DIMENSIONAL,
C ARRAY WITH DIMENSIONS OF IFRST,ISEC. IT WILL RETURN THE AVERAGE
C IN THE VARIABLE 'AVERAGE'. IT IS SET TO NOT INCLUDE ANY ITEMS THAT
C ARE PRESENTLY SET TO ZERO IN THE AVERAGE. THIS MAY NEED TO BE
C CHANGED WITH A DIFFERENT SET OF DATA.
C
C DIMENSION SARRAY(IFRST,ISEC)
REAL*4 SARRAY,AVERAGE,TEMP
C
C TEMP = 0.
AVERAGE = 0.
ITEM = 0
ITEM = IFRST*ISEC
DO 10 I = 1,IFRST
DO 10 J = 1,ISEC
    IF (SARRAY(I,J).EQ.RMASK) THEN
        ITEM = ITEM + 1
    ELSE
        TEMP = TEMP + SARRAY(I,J)
    ENDIF
10 CONTINUE
AVERAGE = (TEMP / ITEM)
RETURN
END

```

readinput.f

```

SUBROUTINE READINP
C
C THIS SUBROUTINE IS USED TO READ THE DATA FROM THE INPUT FILE MAPINPUT
C AND STORE THE DATA INTO THE COMMON BLOCKS TO BE USED LATER
C
C CDUM IS A TEMPORARY VARIABLE USED TO STORE THE FIRST LINE OF THE DATA
C FILE mapinput.
C
INCLUDE 'data.txt'
CHARACTER*40 CDUM
OPEN(51,FILE='mapinput',STATUS='UNKNOWN',FORM='FORMATTED')

READ(51,'(A40)')TITLE
READ(51,'(A40)')CDUM
DO 510 K=1,KE
  READ(51,'(54X,I1)')ILAY(K)
510 CONTINUE
READ(51,'(A40)')CDUM
DO 511 I=1,NMAP
  READ(51,'(11X,A40,3X,I1)')CNAME(I),IPAPI(I)
511 CONTINUE

CLOSE(51)
RETURN
END

SUBROUTINE CHOOSEMAP
C
C THIS SUBROUTINE HOLDS THE MAJOR LOOP WHICH WILL ALLOW UP TO TWENTY-
C SIX MAP PLOTS TO BE GENERATED. THIS LOOP SELECTS WHICH FILE TO READ
C THE DATA FROM AND STORES IT IN THE VARIABLE CHOOSEN AND IT IS THEN
C SENT THROUGH A SERIES OF MANIPULATION BEFORE IT IS FINALLY PLOTTED BY
C NCAR. THE CALL TO MAPOVR ALLOWS CONTINENTS TO BE PLOTTED OVER TOP OF
C THE CONTOUR DATA.
C
C CDUM,CTEMP -- BOTH TEMPORARY CHARACTER VARIABLES.
C
INCLUDE 'data.txt'
CHARACTER*40 CDUM
CHARACTER*8 CTEMP
C
CALL READINP
C
C FILL THE ARRAY THAT HOLDS THE NAMES OF THE FILES TO BE READ IN DURING
C THE FOLLOWING LOOP. (HAMBURG DEPENDENT)
C
CFIL(1)='SCO2'
CFIL(2)='ALKA'
CFIL(3)='PO4'
CFIL(4)='O2'
CFIL(5)='POC'
CFIL(6)='CALC1T'
CFIL(7)='DC13'
CFIL(8)='DC14'
CFIL(9)='POCC13'
CFIL(10)='POCC14'
CFIL(11)='CALC13'
CFIL(12)='CALC14'
CFIL(13)='SATCO3'
CFIL(14)='PH'
CFIL(15)='CO3'
CFIL(16)='KSP'
CFIL(17)='PO40'
CFIL(18)='T'
CFIL(19)='S'

CFIL(20)='CO2SUR'
CFIL(21)='OADIFF'
CFIL(22)='SOFFPRO'
CFIL(23)='CALBRO'
CFIL(24)='ORGSED'
CFIL(25)='CALSED'
CFIL(26)='S104'

C BEGINNING OF THE DATA LOOP.
C
DO 513 IFR=1,NMAP
  IF(IPAPI(IFR).EQ.0)GOTO 513
  C
  C THE SILICON FILE HAS BEEN DEACTIVATED AND THUS CAN NOT BE ALLOWED
  C TO PROCESS THROUGH THE LOOP. (HAMBURG DEPENDENT)
  C
  IF(CFIL(IFR).EQ.'S104') GOTO 512
  IFRTAP=20
  C
  DO 11 I=1,IE
    DO 11 J=1,JE
      DEPTP(I,J)= 0.
      CHOOSEN(I,J)= 0.
  11 CONTINUE
  C
  ZLAYER(0)= 0.
  ZLAYER(1)= 25.
  ZLAYER(2)= 75.
  ZLAYER(3)= 150.
  ZLAYER(4)= 250.
  ZLAYER(5)= 450.
  ZLAYER(6)= 700.
  ZLAYER(7)= 1000.
  ZLAYER(8)= 2000.
  ZLAYER(9)= 3000.
  ZLAYER(10)= 4000.
  ZLAYER(11)= 5000.
  ZLAYER(12)= 6000.
  C
  OPEN(59,FILE='BOTOPP',STATUS='OLD',FORM='FORMATTED')
  REWIND(59)
  C
  C READ THE DEPTH ARRAY INTO THE VARIABLE DEPTP
  C
  READ(59,'(A40)',END=3000)CDUM
  READ(59,'(6X,12,AB)')JTEMP,CTEMP
  DO 100 J=1,12
    READ(59,3111,END=3000)(DEPTP(I,J),I=1,IE)
  100 CONTINUE
  C
  OPEN(IFRTAP,FILE=CFIL(IFR),STATUS='OLD',FORM='FORMATTED')
  REWIND(IFRTAP)
  C
  READ(IFRTAP,'(A40)',END=4799)TDM
  READ(IFRTAP,'(6X,12,AB)',END=4799)BDMTH,TBXD
  C
  DO 2000 PLAY= 1,IE
    DO 330 I=1,IE
      DO 330 J=1,JE
        CHOOSEN(I,J)= 6
  330 CONTINUE
  C
  C FOPEN FILE 20 THE NAME IS THE FIRST LETTER OF THE FILE NAME THAT IS ADDED
  C BASED FOR PROBLEMS - CHANGING DEPENDENCY

```

readinput.f

```
      IF(IFR.GE.20.AND.IFR.LE.25.AND.KLAY.GT.1)GOTO 2000
C
C READ THE DATA FROM THE SPECIFIED FILE INTO THE VARIABLE CHOOSEN.
C
      DO 3110 J=1,JE
         READ(IFRTAP,3111,END=4799)(CHOOSEN(I,J),I=1,IE)
3110  CONTINUE
3111  FORMAT(8(1K,E10.4))
      IF (ILAY(KLAY).EQ.0)GOTO 2000
      LAYER = KLAY
      HEADER = CNAME(IFR)
      IHEAD = 40
C
C BEGIN DATA REFINEMENT (HAMBURG DEPENDENT)
C
      CALL GETVAL
      CALL REFINEDATA(AVERAGE)
      CALL BOX
C
C END DATA REFINEMENT (HAMBURG DEPENDENT)
C
      CALL PLOTMAP(IFR)
C
2000  CONTINUE
      CLOSE(IFRTAP,STATUS='KEEP')
      CLOSE(59,STATUS='KEEP')
      GOTO 513
512  WRITE(*,*) 'SIO4 IS DEACTIVATED AT THIS TIME'
513  CONTINUE
      GOTO 4800
4799  WRITE(*,*) 'ERROR READING IFRTAP'
      STOP
3000  WRITE(*,*) 'ERROR READING DEPTH'
      STOP
4800  CONTINUE
      RETURN
      END
```

refine.f

```

SUBROUTINE REFINEMENT(VALU,DEPTH,CHECK,AVERAGE)
C THIS SUBROUTINE WILL SMOOTH THE ARRAY SO THAT THE CONTINENTS ARE
C NOT SKewed AND THE WORLD IS CUT AT THE PRIME MERIDIAN INSTEAD OF
C ACROSS SOUTH AMERICA. AT THE PRESENT TIME THE REFINER SUBROUTINE
C AND THE SMOOTH SUBROUTINE ARE NOT USED BECAUSE UNREFINED AND
C UNSMOOTHED DATA WAS WANTED.
C
C   AFFECTED SUBROUTINES:
C     SMOOTH
C     REFINER
C     PART
C     NEXTR
C     NEXTD
C     NEXTL
C     FINE
C
C   THE REFINEMENTS HELD IN THIS FILE ARE TAKEN FROM THE
C   CARBON SIDE OF THE HAMBURG MODEL.
C
C   include 'data.txt'
DIMENSION ATEMP(IMAX),VALU(IMAX,JMAX),CHECK(IMAX,JMAX)
DIMENSION DEPTH(IMAX,JMAX)
DIMENSION ARRID(JMAX)
LOGICAL MTEMP(IMAX)

C-----  

C
      IMOOTH = 1
      IPART = 1
      IFINE = 5
      IEND = IE
      JEND = JE

C ACTUAL SHIFTING OF THE ARRAY SO THAT IT IS CUT AT THE PRIME MERIDIAN
C ----- BEGIN -----
DO 1000 J=1,JEND
  DO 1010 I=1,IEND
    ATEMP(I) = VALU(I,J)
1010  CONTINUE
  DO 1020 ITEMP=1,IEND
    I = ITEMP-MSHIFT-(J-1)/2
    IF(I.LE.0) I=I+IEND
    IF(I.GT.IEND) I=I-IEND
    VALU(I,J) = ATEMP(ITEMP)
    CHECK(I,J) = VALU(I,J)
    IF(VALU(I,J).EQ.RMASK)
    *      VALU(I,J) = AVERAGE
1020  CONTINUE
1000  CONTINUE
C ----- END -----
C
      IF(K.GE.0) THEN
        DO 1030 J=1,JEND
          DO 1030 I=1,IEND
            IF(DEPTH(I,J).LE.ZLAYER(K)) THEN
              MHZ(I,J) = .FALSE.
            ELSE
              MHZ(I,J) = .TRUE.
            END IF
1030  CONTINUE
      ENDIF
      DO 1400 J=1,JEND

```

```

DO 1040 I=1,IEND
  MTEMP(I) = MHZ(I,J)
CONTINUE
1040  DO 1050 ITEMP=1,IEND
    I = ITEMP-MSHIFT-(J-1)/2
    IF(I.LE.0) I=I+IEND
    IF(I.GT.IEND) I=I-IEND
    MHZ(I,J) = MTEMP(ITEMP)
1050  CONTINUE
1400  CONTINUE
C
      IF(IMOOTH.GT.0) THEN
        DO 2020 I=1,IMOOTH
          CALL SMOOTH(IE,JE,VALU,ZLAYER,DEPTH)
2020  CONTINUE
      ENDIF
      IF(IPART.GT.2) THEN
        CALL PART(VALU,MHZ,IE,IEND,ZLAYER,ARRID,MHZ10,THAX,JMAX,
                 GLOMAP)
      ENDIF
      RETURN
      END
C
C-----  

C
      SUBROUTINE SMOOTH(IE,JE,VALU,E,ZLAYER,DEPTH)
C
C   THIS SUBROUTINE IS USED TO SMOOTH OUT THE DATA.
C
      PARAMETER(IMAX=78, JMAX=78,IMAXHR=546,IMAXLR=546)
DIMENSION IDUM(IMAX,JMAX),JDUM(IMAX,JMAX),MDUM(IMAX,JMAX),
         NDUM(IMAXHR,JMAXHR),XDUM(IMAX,JMAX),YDUM(IMAX,JMAX)
DIMENSION VALU(IMAX,JMAX),DEPTH(IMAX,JMAX)
DIMENSION ZLAYER(0:12)

C
C
C*   E: ODD ZONAL GRID LINES
C*   JDUM: JOURNAL FOR DEPTH
C
1000  CONTINUE
DO 1000 J=2,JE-1,2
  DO 1000 I=1,IE
    ANUM = 0.
    XDUM(I,J) = 0.
    IF(DEPTH(I,J).LE.ZLAYER(E)) GO TO 1010
    IL = I+1
    IF(IL.LT.I) IL = IE
    IF(DEPTH(IL,J-1).GT.ZLAYER(E)) THEN
      ANUM = ANUM + 1.
      XDUM(IL,J) = XDUM(IL,J) + VALU(IL,J-1)
    END IF
    IF(DEPTH(I,J+1).GT.ZLAYER(E)) THEN
      ANUM = ANUM + 1.
      XDUM(I,J) = XDUM(I,J) + VALU(I,J+1)
    END IF
    IF(DEPTH(IL,J+1).GT.ZLAYER(E)) THEN
      ANUM = ANUM + 1.
      XDUM(IL,J) = XDUM(IL,J) + VALU(IL,J+1)
    END IF
1010  CONTINUE
DO 1030 J=1,JE
  DO 1030 I=1,IE
    ANUM = ANUM + 1.
    XDUM(I,J) = XDUM(I,J) + VALU(I,J)
  END IF
1030  CONTINUE

```

refine.f

```

1010 IF (NINT(ANUM).EQ.0) THEN
      XDUM(I,J) = VALU(I,J)
    ELSE
      XDUM(I,J) = (XDUM(I,J)+ANUM*VALU(I,J)) / (2.*ANUM)
    ENDIF
1000 CONTINUE
C
C*   2. EVEN ZONAL GRID LINES.
C*   ===== ===== =====
C
200 CONTINUE
DO 2000 J=3,JE-1,2
DO 2000 I=1,IE
  ANUM = 0.
  XDUM(I,J) = 0.
  IF (DEPTH(I,J).LE.ZLAYER(K)) GO TO 2020
  IR = I+1
  IF (IR.GT.IE) IR = 1
  IF (DEPTH(IR,J-1).GT.ZLAYER(K)) THEN
    ANUM = ANUM + 1.
    XDUM(I,J) = XDUM(I,J) + VALU(IR,J-1)
  END IF
  IF (DEPTH(I,J-1).GT.ZLAYER(K)) THEN
    ANUM = ANUM + 1.
    XDUM(I,J) = XDUM(I,J) + VALU(I,J-1)
  END IF
  IF (DEPTH(IR,J+1).GT.ZLAYER(K)) THEN
    ANUM = ANUM + 1.
    XDUM(I,J) = XDUM(I,J) + VALU(IR,J+1)
  END IF
  IF (DEPTH(IR,J+1).GT.ZLAYER(K)) THEN
    ANUM = ANUM + 1.
    XDUM(I,J) = XDUM(I,J) + VALU(IR,J+1)
  END IF
2020 IF (NINT(ANUM).EQ.0) THEN
      XDUM(I,J) = VALU(I,J)
    ELSE
      XDUM(I,J) = (XDUM(I,J)+ANUM*VALU(I,J)) / (2.*ANUM)
    ENDIF
2000 CONTINUE
C
C*   3. COPY RESULTS.
C*   ===== =====
C
300 CONTINUE
DO 3000 J=2,JE-1
DO 3000 I=1,IE
  VALU(I,J) = XDUM(I,J)
3000 CONTINUE
C
RETURN
END
C
C-----SUBROUTINE PART(VALU,MHZ,IEND,JEND,K,ARRIO,MHZ10,IMAX,JMAX,
*                      GLOMAP)
C
THIS ROUTINE, AT THE PRESENT TIME, IS UNUSED BUT WAS ADDED FOR EASE
IN FUTURE ALTERATIONS.
C
PARAMETER (IMAXHR=546, JMAXHR=546)
C
DIMENSION VALU(IMAX,JMAX), FARRAY(0:IMAXHR,JMAXHR)
DIMENSION ARRIO(JMAX)
LOGICAL MHZ10(JMAX)
LOGICAL MHZ(IMAX,JMAX), FMHZ(IMAXHR,JMAXHR), GLOMAP
C
C
GLOMAP=.TRUE.
IEND=IEND
JH=JEND
IFRST = ILAST*IPART*(IEND-1)
JFRST = JFIRST*IPART*(IPART-1)
C
C*   2. REFINES *ARRAY* TO *FARRAY*.
C*   ===== ===== =====
C
200 CONTINUE
CALL REFINE(VALU,FARRAY,MHZ,FMHZ,IH,JH,IPART,K,GLOMAP,MHZ10)
C
C>>>> SCHUTZABFRAGE UNTERER RAND ...
IF(JFRST+JEND-1.GT.JH) JFRST=JH-(JEND-1)
C
C*   3. COPY PART OF REFINED DATA TO *ARRAY*.
C*   ===== ===== =====
C
300 CONTINUE
DO 320 J=1,JEND
DO 320 I=1,IPART
  VALU(I,J) = FARRAY(I+IFRST-1,J+JFRST-1)
  MHZ(I,J) = FMHZ(I+IFRST-1,J+JFRST-1)
320 CONTINUE
C
C*   4. SET ROW "0" OF *ARRAY* AND *MHZ*: *ARRIO*, *MHZ10*.
C*   ===== ===== =====
C
400 CONTINUE
I1=IFRST+1
I2=I1
IF(I1.EQ.0) I2=IEND*IPART
DO 420 J=1,JEND
  ARRIO(IJ) = FARRAY(IJ,J+JFRST-1)
  MHZ10(IJ) = FMHZ(IJ,J+JFRST-1)
420 CONTINUE
GLOMAP=.FALSE.
C
C-----SUBROUTINE REFINE(VALU,FARRAY,MHZ,FMHZ,IH,JH,IPART,K,GLOMAP,
*                      MHZ10,ARRIO)
C
PARAMETER (IMAXHR=546, JMAXHR=546, IMAX=79, JMAX=79)
DIMENSION VALU(IMAX,JMAX), FARRAY(0:IMAXHR,JMAXHR)
ARRIO(JMAX)
LOGICAL MHZ10(IMAX,JMAX), FMHZ10(IMAXHR,JMAXHR)

```

```

refine.f

LOGICAL GLOMAP, MHZIO(JMAX)
C* 0. MAKE COPY OF DATA IN CASE OF (IFINE.EQ.1).
C*      ====== ====== ====== ====== ====== ======
C
C      WRITE(*,*)' SR *REFINE* : IFINE=',IFINE
C
IF(IFINE.LE.1) THEN
  DO 50 J=1,JE
    DO 50 I=1,IE
      FARRAY(I,J)=VALU(I,J)
      FMHZ(I,J)=MHZ(I,J)
50   CONTINUE
      RETURN
    END IF
C
C      >>> SOME PREPARATIONS ....
IE1=IE+1
JE1=JE+1
KFIN = IE*IFINE
LFIN = (JE-1)*IFINE+1
FFINE = FLOAT(IFINE)
EPS = 1.E-10
CC
C* 1. COMPUTE REAL ARRAY "WET".
C*      ====== ====== ======
C
DO 100 J=1,JE
DO 100 I=1,IE
  FELD(I,J)=VALU(I,J)
  IF(MHZ(I,J)) THEN
    WET(I,J)=1.
  ELSE
    WET(I,J)=0.
  END IF
100 CONTINUE

DO 110 I=0,IE1
  FELD(1,I) = 0.0
  FELD(1,JE1)= 0.0
  WET(1,0) = 0.0
  WET(1,JE1) = 0.0
110 CONTINUE

IF(GLOMAP) THEN
  DO 120 J=1,JE
    FELD(0,J) = FELD(IE,J)
    FELD(IE1,J) = FELD(1,J)
    WET(0,J) = WET(IE,J)
    WET(IE1,J) = WET(1,J)
120 CONTINUE
ELSE
  >>> PART MAP: ROW "0" NOT EQUAL TO ROW "IE" ! <<<
  DO 130 J=1,JE
    FELD(0,J) = ARRI0(J)
    IF(MHZIO(J)) THEN
      WET(0,J) = 1.0
    ELSE
      WET(0,J) = 0.0
    END IF
    FELD(IE1,J) = FELD(1,J)
    WET(IE1,J) = 0.0
130 CONTINUE
END IF
C*      ====== ====== ====== ====== ====== ======
C
C* 2. REFINE *ARRAY(I,J)* TO *FARRAY(K,L)*.
C*      ====== ====== ====== ====== ====== ======
C
DO 250 J1=1,JE
DO 250 I1=0,IE
  I2 = I1
  J2 = J1
  CALL NEXTR(I2,J2)
  I3 = I2
  J3 = J2
  CALL NEXTD(I3,J3)
  I4 = I1
  J4 = J1
  CALL NEXTD(I4,J4)
  CALL FINE(I1,J1,I1,J1,IFINE)
C
  DO 240 IR=1,IFINE
    L = I1
    K = I1
C
    DO 210 LL=1,IR-1
      CALL NEXTR(K,L)
210   CONTINUE
C
    DO 230 ID=1,IFINE
      IF(L.LT.1) GOTO 220
      IF(L.GT.LFIN) GOTO 220
      IF(E.GT.KFIN) GOTO 220
      ALPH = FLOAT(IR-1)/FFINE
      BETA = FLOAT(ID-1)/FFINE
      WERT = ((1.-ALPH)*(1.-BETA))*WET(11,J1)*FELD(11,I1)
      # + (ALPH*(1.-BETA))*WET(12,J1)*FELD(12,I1)
      # + (ALPH*BETA)*WET(13,J1)*FELD(13,I1)
      # + (BETA*(1.-ALPH))*WET(14,J1)*FELD(14,I1)
      # / ((1.-ALPH)*(1-BETA)*WET(11,J1))
      # + (ALPH*(1.-BETA))*WET(12,J1)
      # + (ALPH*BETA)*WET(13,J1)
      # + (BETA*(1.-ALPH))*WET(14,J1)
      # + EPS )
C
      KNEW = MOD(K+KFIN-1,FFINE)+1
      FARRAY(KNEW,I1) = WERT
C
220   CONTINUE
      CALL NEXTR(K,L)
C
230   CONTINUE
240   CONTINUE
250 CONTINUE
C
C* 3. ENLARGE FLAG ARRAY *WET* TO *FMHZ*.
C*      ====== ====== ====== ====== ====== ======
C
IE = 0.0,I1,IE

```

refine.f

```

DO 350 I=1,IE
      CALL FINE(I,J,K,L,IFINE)
      KSTART = K
      LSTART = L-(IFINE-1)

      DO 340 IL=1,IFINE
            K = KSTART
            L = LSTART

            DO 330 ID=1,IFINE
                  IF(L.LT.1) GOTO 320
                  IF(L.GT.LFIN) GOTO 320
                  KNEU = MOD(K+KFIN-1,KFIN) +1
                  FMHZ(KNEU,L) = MHZ(I,J)

320      CONTINUE
            CALL NEXTD(K,L)

330      CONTINUE
            CALL NEXTL(KSTART,LSTART)

340      CONTINUE

350 CONTINUE
      RETURN
      END
C-----SUBROUTINE NEXTD(I,J)
C      I=I+MOD(J+100000,2)
C      J=J+1
C      RETURN
C      END
C-----SUBROUTINE NEXTL(I,J)
C      I=I-1+MOD(J+100000,2)
C      J=J+1
C      RETURN
C      END
C-----SUBROUTINE NEXTR(I,J)
C      I=I+MOD(J+100000,2)
C      J=J-1
C      RETURN
C      END
C-----SUBROUTINE FINE(I,J,K,L,IFINE)

      IF(MOD(J,2).EQ.0) THEN
C        >>> J : EVEN <<<
        K = (IFINE+1)/2 + (I-1)*IFINE
      ELSE
C        >>> J : ODD <<<
        K = IFINE * I
      END IF
      L = 1 + (J-1)*IFINE
C
      RETURN
      END

```

refinesec.f

refinesec.f

```

        XDUM(J,K) = XDUM(J,K) + ARRAY(JR,K)
    ENDIF
    IF (NINT(ANUM) EQ 0) THEN
        XDUM(J,K) = ARRAY(J,K)
    ELSE
        XDUM(J,K) = (XDUM(J,K)+ANUM*ARRAY(J,K)) / (2.*ANUM)
    ENDIF
    ENDIF
1000 CONTINUE
200 CONTINUE
    DO 2000 K=1,KE
    DO 2000 J=1,JE
        ARRAY(J,K) = XDUM(J,K)
2000 CONTINUE
C
    RETURN
END

SUBROUTINE VERFIN(ARRAY,FARRAY,MHZ,FMHZ,JE,KE,JFINE,KKF,DEPMAX,
     *           INDI,ZLAYER,DEPTH)
C
PARAMETER (IMAX=78,JMAX=78,KMAX=25,JMAXHR=JMAX*7,IMAXHR=IMAX*7)
DIMENSION ARRAY(IMAX,JMAX),FARRAY(0:IMAXHR,JMAXHR),
     *           FELD(0:JMAX+1,0:KMAX+1),WET(0:JMAX+1,0:KMAX+1),
     *           DEP(0:JMAXHR),INDI(JE),ZLAYER(0:12),DEPTH(IMAX,JMAX)
LOGICAL MHZ(IMAX,JMAX),FMHZ(IMAXHR,JMAXHR)
JE1=JE+1
KE1=KE+1
EPS = 0.000001

KKFIN = KKF
JJFIN = JE*JFINE
DEX = XMAP/FLOAT(KKFIN)
DEY = YMAP/FLOAT(KKFIN)

FJFINE= FLOAT(JFINE)
JFINEH=IFIX(FJFINE*0.5+EPS)
FINKK = FLOAT(KKFIN)

WRITE(7,*)
SR *VERFIN* : JJFIN='JJFIN,' KKFIN='KKFIN,' KKFIN,
*      ' DEX='DEX,' DEY='DEY
C
DO 10 K=1,KE
DO 10 J=1,JE
    FELD(J,K)=ARRAY(J,K)
10 CONTINUE

DO 20 J=1,JE
    FELD(J,0) = FELD(J,1)
    FELD(J,KE1) = 0.0
20 CONTINUE

DO 30 K=0,KE1
    FELD(0,K) = 0.0
    FELD(JE1,K) = 0.0
30 CONTINUE

C
C =====
C* 1. COMPUTE REAL ARRAY *WET*.
C
DO 100 K=1,KE
    DO 100 J=1,JE
        IF(MHZ(J,K)) THEN
            WET(J,K)=1.
        ELSE
            WET(J,K)=0.
        END IF
100 CONTINUE
C
    DO 110 K=0,KE1
        WET(0,K)=0.
        WET(JE1,K)=0.
110 CONTINUE
C
    DO 120 J=1,JE
        WET(J,KE1)=0.
        WET(J,0)=WET(J,1)
120 CONTINUE
C
C >>>> COMPUTE DEPTH STEPS OF NEW GRID.
C
DELY=DEPMAX/FINKK
DELDP=DELY
DELYH=DELY*0.5
ZLKE1=ZLAYER(KE1)
ZLAYER(KE+1)=DEPMAX

DEP(0)=0.0
DO 130 KN=1,KKFIN
    DEP(KN)=KN*DELY-DELYH
130 CONTINUE
C
C* 2. INTERPOLATE ON NEW GRID.
C
DO 250 JN=1,JFINEH
DO 250 JN=1,JFINEH,JFIN+JFINEH
    >>>> SELECT THE FOUR SURROUNDING POINTS ON OLD GRID.
    JL=IFIX(FLOAT(JN)/FJFINE+EPS)
    JR=JL+1
    KO=0
    DO 210 K=KE,0,-1
        IF (DEP(KN).GE.ZLAYER(K)) THEN
            KO=K
            GOTO 215
        END IF
210 CONTINUE
215 CONTINUE
    KO=KO+1
C
    JNR = JN-JL*JFINE
    ALPH= FLOAT(JNR)/JFINE
    BETA= (DEP(KN)-ZLAYER(KO))/(ZLAYER(KO)-ZLAYER(K))
C
    FARRAY(JN,JFINEH,FN)
        (1.-ALPH)*(1.-BETA)*WET(JL,FN)*FELD(JL,FN)
        +(ALPH*(1.-BETA))*WET(JP,FN)*FELD(JP,FN)
        +(ALPH*BETA)*WET(JP,FN)*FELD(JP,FN)
        +(BETA*(1.-ALPH))*WET(JL,FN)*FELD(JL,FN)
        -(1.-ALPH)*(1-BETA)*WET(JL,FN)
        +(ALPH*(1.-BETA))*WET(JP,FN)
        +(ALPH*BETA)*WET(JP,FN)
        +(BETA*(1.-ALPH))*WET(JL,FN)
        EPS )
C

```

refinesec.f

```

250 CONTINUE
      ZLAYER(KE+1)=ZLKE1

C     =====
C*   3. COMPUTE ARRAY *FMHZ*.
C     =====
C
DO 320 J=1,JE
  I=IND1(J)-MSHIFT-(J-1)/2
  IF (I.LE.0) I=I+IEND
  DO 310 KN=1,KKFIN
  DO 310 JN=(J-1)*JFINE+1,J*JFINE
    IF(KN*DELY.LE.DEPTH(I,J)) THEN
      FMHZ(JN,KN)=.TRUE.
    ELSE
      FMHZ(JN,KN)=.FALSE.
    END IF
310  CONTINUE
320 CONTINUE

      RETURN
END

SUBROUTINE VERCOL(ARRAY,FARRAY,MHZ,FMHZ,JE,KE,JFINE,KKFIN,DEPTHMAX,
*                      ZLAYER,DEPTH,ICON,IVAL,DELTA,VALS)
PARAMETER(IMAX=78,JMAX=78,IMAXHR=IMAX*7,JMAXHR=JMAX*7,KMAX=25)
LOGICAL MHZ(IMAX,JMAX),FMHZ(IMAXHR,JMAXHR)
DIMENSION ARRAY(IMAX,JMAX),FARRAY(0:IMAXHR,JMAXHR)
DIMENSION ZLAYER(0:12),DEPTH(IMAX,JMAX),VALS(100)

IF(ICON.LT.0) GOTO 130
IF(IVAL.LE.0) THEN
  SCHUM = 1.E-6*DELTA
ELSE
  XIN=1.E50
  XAX=-XIN
  DO 110 I=1,IVAL
    IF(VALS(I).LT.XIN) XIN=VALS(I)
    IF(VALS(I).GT.XAX) XAX=VALS(I)
110  CONTINUE
  SCHUM = 1.E-9*(XAX-XIN)
END IF
130  CONTINUE
C
  DO 120 K=1,KE
  DO 120 J=1,JE
    ARRAY(J,K) = ARRAY(J,K) + SCHUM
120  CONTINUE
C
220 CONTINUE
      RETURN
END

SUBROUTINE MINMAX(JE,KE,ARRAY,MHZ,AMIN,AMAX)
PARAMETER(IMAX=78,JMAX=78)
DIMENSION ARRAY(IMAX,JMAX)
LOGICAL MHZ(IMAX,JMAX)
100 CONTINUE
  ICHECK = 0
  DO 1000 J=1,JE
    DO 1000 I=1,IE
C
      IF (MHZ(I,J)) THEN
        IF (ICHECK.EQ.1) THEN
          C
            IF (ARRAY(I,J).LT.AMIN) THEN
              AMIN = ARRAY(I,J)
              INIM = I
              JNIM = J
            ENDIF
          C
            IF (ARRAY(I,J).GT.AMAX) THEN
              AMAX = ARRAY(I,J)
              IXAM = I
              JXAM = J
            ENDIF
          C
            ELSE
              AMIN = ARRAY(I,J)
              INIM = I
              JNIM = J
              AMAX = ARRAY(I,J)
              IXAM = I
              JXAM = J
              ICHECK = 1
            ENDIF
          C
        END IF
      END IF
1000 CONTINUE
      RETURN
END

```

section.f

```

SUBROUTINE READSECINP
C THIS SUBROUTINE IS USED TO READ IN THE SECTION INPUT FROM THE SECINPUT FILE.
C INCLUDE 'data2.txt'
CHARACTER*40 CDUM
OPEN(51,FILE='secinput',STATUS='UNKNOWN',FORM='FORMATTED')
C
READ(51,'(A40)')TITLE
READ(51,'(A40)')CDUM
READ(51,'(54X,I11)')ISECT(1)
READ(51,'(54X,I11)')ISECT(2)
READ(51,'(54X,I11)')ISECT(3)
READ(51,'(54X,I11)')ISECT(4)
READ(51,'(A40)')CDUM
DO 511 I = 1,20
    READ(51,'(1IX,A40,3X,I11)')SNAME(I),ISPAPL(I)
511 CONTINUE
CLOSE(51)
RETURN
END

SUBROUTINE CHOOSESSEC
C THIS SUBROUTINE IS USED TO CHOOSE THE SECTION CUT, MANIPULATE THE DATA, AND THEN
C PLOT THE DATA. ALL REFINEMENTS OF THE DATA HAVE BEEN DISABLED BECAUSE UNREFINED
C AND UNSMOOTHED DATA WAS DESIRED.
C
C AFFECTED SUBROUTINES (AT THIS LEVEL) :
C   AVG
C   HIGH
C   LOWS
C
INCLUDE 'data2.txt'
DIMENSION TEM(IE,JE),DEPTH(IMAX,JMAX)
CHARACTER*40 CDUM
CHARACTER*8 CTEMP
C
SFIL(1) = 'SCO2 '
SFIL(2) = 'ALKA '
SFIL(3) = 'PO4 '
SFIL(4) = 'O2 '
SFIL(5) = 'POC '
SFIL(6) = 'CALCIT'
SFIL(7) = 'DC13 '
SFIL(8) = 'DC14 '
SFIL(9) = 'POCC13'
SFIL(10)= 'POCC14'
SFIL(11)= 'CALC13'
SFIL(12)= 'CALC14'
SFIL(13)= 'SATCO3'
SFIL(14)= 'PH '
SFIL(15)= 'CO3 '
SFIL(16)= 'KSP '
SFIL(17)= 'PO40 '
SFIL(18)= 'T '
SFIL(19)= 'S '
SFIL(20)= 'SIO4 '
C
CALL READSECINP
C
ZAYER(0)= 0.
ZAYER(1)= 25.
ZAYER(2)= 75.
ZAYER(3)= 150.
ZAYER(4)= 250.
ZAYER(5)= 450.
ZAYER(6)= 700.
ZAYER(7)= 1000.
ZAYER(8)= 2000.
ZAYER(9)= 3000.
ZAYER(10)= 4000.
ZAYER(11)= 5000.
ZAYER(12)= 6000.
C
OPEN(59,FILE='BOTOPP',STATUS='OLD',FORM='FORMATTED')
REWIND(59)
C
C READ THE DEPTH ARRAY INTO THE VARIABLE DEPTP
C
READ(59,'(A40)',END=5000)CDUM
READ(59,'(6X,I2,AB)') ITMP,CTEMP
DO 100 J=1,JE
    READ(59,3111)(DEPTP(I,J),I=1,IE)
100 CONTINUE
DO 512 ICR=1,4
    IF(ISECT(ICR).EQ.0)GOTO 512
    CALL INDIGET(ICR)
    DO 513 IFR=1,20
        IF(ISPAPl(IFR).EQ.0)GOTO 513
        IFRTAP=IFR+20
        OPEN(IFRTAP,FILE=SFIL(IFR),STATUS='UNKNOWN',FORM='FORMATTED')
        REWIND(IFRTAP)
        READ(IFRTAP,'(A40)',END=4799)CDUM
        READ(IFRTAP,'(6X,I2,AB)')MONTH,TEXT5
        DO 3050 K=1,KE
            READ(IFRTAP,3111)TEM
            FORMAT(1(IX,B10.4))
            IF(K.EQ.9) THEN
                DO 6003 I=1,IE
                    DO 6003 J=1,JE
                        CONTINUE
                    ENDIF
                    DO 3120 J=1,JE
                        CHODEN(J,K)= TEM(J,I+1,K)
3120 CONTINUE
3050 CONTINUE
        IF(ISPAPl(IFR).EQ.0)GOTO 513
        READER(SNAME(IFR))
        DO 3009 I=1,IE
            DO 3009 J=1,JE
                DEPTH(I,J) = DEPTP(I,J)
3009 CONTINUE
        CALL GETSVAL
        CALL REFINESEC(IE,JE,KE,DEPTH,TODISU,CHODEN,ZAYER,VALV,ZDAT,
                      RMASK)
        C   AVERAGE = 0.
        C   CALL AVG(ZDAT,JE,KE,AVERAGE)
        C   CALL HIGH(ZDAT,JE,KE,HIGH)
        C   CALL LOWS(ZDAT,JE,KE,LOW)
        C   DO 3011 J=1,JE
        C   DO 3011 K=1,KE
        C       IF(ZDAT(J,K).EQ.0.) THEN
        C           ZDAT(J,K) = HIGH + (HIGH/6)
        C       ENDIF
3011 CONTINUE
        XTOP = 1.0
        XBOT = -1.0

```

section.f

```

YLFT = .15
YRGTE = .85
CALL PLOTSEC(XTOP,XBOT,YLFT,YRGTE,ICR,DEPTH)
CLOSE(IFRTAP,STATUS='KEEP')
513  CONTINUE
4799 CONTINUE
512  CONTINUE
5000 CONTINUE
RETURN
END

SUBROUTINE INDIGET(ICR)
INCLUDE 'data2.txt'
IF(ICR.EQ.1)THEN
SECNAM='ATLANTIC OCEAN
INDISU( 1)= 1
INDISU( 2)= 1
INDISU( 3)= 1
INDISU( 4)= 1
INDISU( 5)= 1
INDISU( 6)= 1
INDISU( 7)= 1
INDISU( 8)= 1
INDISU( 9)= 1
INDISU(10)= 1
INDISU(11)= 1
INDISU(12)= 1
INDISU(13)= 1
INDISU(14)= 1
INDISU(15)= 1
INDISU(16)= 1
INDISU(17)= 1
INDISU(18)= 1
INDISU(19)= 1
INDISU(20)= 1
INDISU(21)= 1
INDISU(22)= 1
INDISU(23)= 1
INDISU(24)= 1
INDISU(25)= 1
INDISU(26)= 1
INDISU(27)= 2
INDISU(28)= 3
INDISU(29)= 4
INDISU(30)= 5
INDISU(31)= 6
INDISU(32)= 7
INDISU(33)= 8
INDISU(34)= 9
INDISU(35)=10
INDISU(36)=11
INDISU(37)=12
INDISU(38)=13
INDISU(39)=14
INDISU(40)=15
INDISU(41)=16
INDISU(42)=17
INDISU(43)=17
INDISU(44)=18
INDISU(45)=18
INDISU(46)=19
INDISU(47)=19
INDISU(48)=20
INDISU(49)=20
INDISU(50)=20
INDISU(51)=20
INDISU(52)=20
INDISU(53)=20
INDISU(54)=20
INDISU(55)=20
INDISU(56)=20
INDISU(57)=20
INDISU(58)=20
INDISU(59)=20
INDISU(60)=20
INDISU(61)=21
INDISU(62)=21
INDISU(63)=22
INDISU(64)=22
INDISU(65)=23
INDISU(66)=23
INDISU(67)=24
INDISU(68)=24
INDISU(69)=25
INDISU(70)=25
INDISU(71)=26
INDISU(72)=26
ELSE IF(ICR.EQ.2)THEN
SECNAM='ATLANTIC OCEAN
INDISU( 1)= 6
INDISU( 2)= 6
INDISU( 3)= 6
INDISU( 4)= 6
INDISU( 5)= 6
INDISU( 6)= 6
INDISU( 7)= 6
INDISU( 8)= 6
INDISU( 9)= 6
INDISU(10)= 6
INDISU(11)= 6
INDISU(12)= 6
INDISU(13)= 6
INDISU(14)= 6
INDISU(15)= 6
INDISU(16)= 6
INDISU(17)= 6
INDISU(18)= 6
INDISU(19)= 7
INDISU(20)= 7
INDISU(21)= 8
INDISU(22)= 8
INDISU(23)= 9
INDISU(24)= 9
INDISU(25)= 9
INDISU(26)= 10
INDISU(27)= 10
INDISU(28)= 10
INDISU(29)= 11
INDISU(30)= 12
INDISU(31)= 12
INDISU(32)= 12
INDISU(33)= 13
INDISU(34)= 14
INDISU(35)= 14
INDISU(36)= 15
INDISU(37)= 16
INDISU(38)= 17
INDISU(39)= 18
INDISU(40)= 19
INDISU(41)= 20

```

section.f

```

INDISU(42)= 21
INDISU(43)= 22
INDISU(44)= 23
INDISU(45)= 24
INDISU(46)= 25
INDISU(47)= 26
INDISU(48)= 27
INDISU(49)= 27
INDISU(50)= 28
INDISU(51)= 28
INDISU(52)= 28
INDISU(53)= 28
INDISU(54)= 28
INDISU(55)= 28
INDISU(56)= 28
INDISU(57)= 28
INDISU(58)= 28
INDISU(59)= 28
INDISU(60)= 28
INDISU(61)= 28
INDISU(62)= 28
INDISU(63)= 28
INDISU(64)= 28
INDISU(65)= 28
INDISU(66)= 28
INDISU(67)= 28
INDISU(68)= 28
INDISU(69)= 28
INDISU(70)= 28
INDISU(71)= 28
INDISU(72)= 28
ELSE IF(ICR.EQ.3)THEN
SECNAM='PACIFIC OCEAN
    INDISU( 1)=39
    INDISU( 2)=39
    INDISU( 3)=40
    INDISU( 4)=40
    INDISU( 5)=41
    INDISU( 6)=41
    INDISU( 7)=42
    INDISU( 8)=42
    INDISU( 9)=43
    INDISU(10)=43
    INDISU(11)=44
    INDISU(12)=44
    INDISU(13)=45
    INDISU(14)=45
    INDISU(15)=46
    INDISU(16)=46
    INDISU(17)=47
    INDISU(18)=47
    INDISU(19)=48
    INDISU(20)=48
    INDISU(21)=48
    INDISU(22)=48
    INDISU(23)=48
    INDISU(24)=48
    INDISU(25)=48
    INDISU(26)=48
    INDISU(27)=49
    INDISU(28)=49
    INDISU(29)=50
    INDISU(30)=50
    INDISU(31)=51
    INDISU(32)=52
    INDISU(33)=53
    INDISU(34)=54
    INDISU(35)=55
    INDISU(36)=56
    INDISU(37)=57
    INDISU(38)=57
    INDISU(39)=58
    INDISU(40)=58
    INDISU(41)=59
    INDISU(42)=59
    INDISU(43)=60
    INDISU(44)=60
    INDISU(45)=61
    INDISU(46)=61
    INDISU(47)=62
    INDISU(48)=62
    INDISU(49)=63
    INDISU(50)=63
    INDISU(51)=64
    INDISU(52)=64
    INDISU(53)=65
    INDISU(54)=65
    INDISU(55)=66
    INDISU(56)=66
    INDISU(57)=67
    INDISU(58)=67
    INDISU(59)=68
    INDISU(60)=68
    INDISU(61)=69
    INDISU(62)=69
    INDISU(63)=70
    INDISU(64)=70
    INDISU(65)=71
    INDISU(66)=71
    INDISU(67)=72
    INDISU(68)=72
    INDISU(69)=72
    INDISU(70)=72
    INDISU(71)=72
    INDISU(72)=72
ELSE IF(ICR.EQ.4)THEN
SECNAM='PACIFIC OCEAN
    INDISU( 1)= 46
    INDISU( 2)= 47
    INDISU( 3)= 47
    INDISU( 4)= 48
    INDISU( 5)= 48
    INDISU( 6)= 49
    INDISU( 7)= 49
    INDISU( 8)= 50
    INDISU( 9)= 50
    INDISU(10)= 51
    INDISU(11)= 51
    INDISU(12)= 52
    INDISU(13)= 52
    INDISU(14)= 53
    INDISU(15)= 53
    INDISU(16)= 54
    INDISU(17)= 54
    INDISU(18)= 55
    INDISU(19)= 55
    INDISU(20)= 56
    INDISU(21)= 56
    INDISU(22)= 57
    INDISU(23)= 57

```

section.f

```

INDISU(24)= 58
INDISU(25)= 58
INDISU(26)= 59
INDISU(27)= 59
INDISU(28)= 60
INDISU(29)= 60
INDISU(30)= 61
INDISU(31)= 61
INDISU(32)= 62
INDISU(33)= 62
INDISU(34)= 63
INDISU(35)= 63
INDISU(36)= 64
INDISU(37)= 64
INDISU(38)= 65
INDISU(39)= 65
INDISU(40)= 66
INDISU(41)= 66
INDISU(42)= 67
INDISU(43)= 67
INDISU(44)= 68
INDISU(45)= 68
INDISU(46)= 69
INDISU(47)= 69
INDISU(48)= 70
INDISU(49)= 70
INDISU(50)= 71
INDISU(51)= 71
INDISU(52)= 72
INDISU(53)= 72
INDISU(54)= 1
INDISU(55)= 1
INDISU(56)= 2
INDISU(57)= 2
INDISU(58)= 3
INDISU(59)= 3
INDISU(60)= 4
INDISU(61)= 4
INDISU(62)= 5
INDISU(63)= 5
INDISU(64)= 6
INDISU(65)= 6
INDISU(66)= 7
INDISU(67)= 7
INDISU(68)= 8
INDISU(69)= 8
INDISU(70)= 9
INDISU(71)= 9
INDISU(72)= 10
ENDIF

RETURN
END

SUBROUTINE GETSVAL
INCLUDE 'data2.txt'
DIMENSION VALX(20,100)
C
VALS(1) = 3.
VALS(2) = 2.8
VALS(3) = 2.
DO 10 M=4,100
  VALS(M) = 0.
10 CONTINUE
VALX( 1,1) = 1950.

```

```

VALX( 1,2) = 2000.
VALX( 1,3) = 2050.
VALX( 1,4) = 2100.
VALX( 1,5) = 2150.
VALX( 1,6) = 2200.
VALX( 1,7) = 2250.
VALX( 1,8) = 2275.
VALX( 1,9) = 2300.
VALX( 1,10)= 2325.
VALX( 1,11)= 2350.
VALX( 1,12)= 2400.
VALX( 1,13)= 2450.
VALX( 1,14)= 2500.
VALX( 2,1) = 2000.
VALX( 2,2) = 2050.
VALX( 2,3) = 2100.
VALX( 2,4) = 2150.
VALX( 2,5) = 2200.
VALX( 2,6) = 2250.
VALX( 2,7) = 2300.
VALX( 2,8) = 2350.
VALX( 2,9) = 2375.
VALX( 2,10)= 2400.
VALX( 2,11)= 2425.
VALX( 2,12)= 2450.
VALX( 2,13)= 2500.
VALX( 2,14)= 2550.
VALX( 3,1) = 0.
VALX( 3,2) = 0.25
VALX( 3,3) = 0.5
VALX( 3,4) = 0.75
VALX( 3,5) = 1.
VALX( 3,6) = 1.25
VALX( 3,7) = 1.5
VALX( 3,8) = 1.75
VALX( 3,9) = 2.
VALX( 3,10)= 2.5
VALX( 3,11)= 3.
VALX( 3,12)= 3.5
VALX( 3,13)= 4.
VALX( 3,14)= 4.5
VALX( 4,1) = 20.
VALX( 4,2) = 40.
VALX( 4,3) = 60.
VALX( 4,4) = 80.
VALX( 4,5) = 100.
VALX( 4,6) = 125.
VALX( 4,7) = 150.
VALX( 4,8) = 175.
VALX( 4,9) = 200.
VALX( 4,10)= 225.
VALX( 4,11)= 250.
VALX( 4,12)= 300.
VALX( 4,13)= 450.
VALX( 4,14)= 400.
VALX( 5,1) = 0.
VALX( 5,2) = 0.1
VALX( 5,3) = 0.5
VALX( 5,4) = 1.
VALX( 5,5) = 2.
VALX( 5,6) = 4.
VALX( 5,7) = 6.
VALX( 5,8) = 8.
VALX( 5,9) = 10.
VALX( 5,10)= 12.5

```

section.f

```

VALX( 5,11)= 15.
VALX( 5,12)= 20.
VALX( 5,13)= 25.
VALX( 5,14)= 30.
VALX( 6,1) = 0.
VALX( 6,2) = 0.1
VALX( 6,3) = 0.5
VALX( 6,4) = 1.
VALX( 6,5) = 1.5
VALX( 6,6) = 2.
VALX( 6,7) = 2.5
VALX( 6,8) = 3.
VALX( 6,9) = 3.5
VALX( 6,10)= 4.
VALX( 6,11)= 4.5
VALX( 6,12)= 5.
VALX( 6,13)= 10.
VALX( 6,14)= 15.
VALX( 7,1) = -1.0
VALX( 7,2) = -0.75
VALX( 7,3) = -0.5
VALX( 7,4) = -0.25
VALX( 7,5) = 0.
VALX( 7,6) = .25
VALX( 7,7) = .5
VALX( 7,8) = .75
VALX( 7,9) = 1.
VALX( 7,10)= 1.25
VALX( 7,11)= 1.5
VALX( 7,12)= 2.
VALX( 7,13) = 3.
VALX( 7,14)= 4.
VALX( 8,1) = -220.
VALX( 8,2) = -200.
VALX( 8,3) = -180.
VALX( 8,4) = -160.
VALX( 8,5) = -140.
VALX( 8,6) = -130.
VALX( 8,7) = -120.
VALX( 8,8) = -110.
VALX( 8,9) = -100.
VALX( 8,10)= -90.
VALX( 8,11)= -80.
VALX( 8,12)= -70.
VALX( 8,13)= -60.
VALX( 8,14)= -50.
VALX( 9,1) = -18.8
VALX( 9,2) = -18.7
VALX( 9,3) = -18.6
VALX( 9,4) = -18.5
VALX( 9,5) = -18.4
VALX( 9,6) = -18.3
VALX( 9,7) = -18.2
VALX( 9,8) = -18.1
VALX( 9,9) = -18.0
VALX( 9,10)= -17.9
VALX( 9,11)= -17.8
VALX( 9,12)= -17.7
VALX( 9,13)= -17.6
VALX( 9,14)= -17.5
VALX(10,1) = -110.
VALX(10,2) = -100.
VALX(10,3) = -90.
VALX(10,4) = -80.
VALX(10,5) = -75.

```

77

```

VALX(10,6) = -70.
VALX(10,7) = -65.
VALX(10,8) = -60.
VALX(10,9) = -55.
VALX(10,10)= -50.
VALX(10,11)= -45.
VALX(10,12)= -40.
VALX(10,13)= -35.
VALX(10,14)= -30.
VALX(11,1) = 0.
VALX(11,2) = 0.25
VALX(11,3) = 0.5
VALX(11,4) = 0.75
VALX(11,5) = 1.
VALX(11,6) = 1.25
VALX(11,7) = 1.5
VALX(11,8) = 1.75
VALX(11,9) = 2.
VALX(11,10)= 2.25
VALX(11,11)= 2.5
VALX(11,12)= 3.
VALX(11,13)= 4.
VALX(11,14)= 5.
VALX(12,1) = 160.
VALX(12,2) = 150.
VALX(12,3) = -140.
VALX(12,4) = -130.
VALX(12,5) = -120.
VALX(12,6) = -110.
VALX(12,7) = -100.
VALX(12,8) = -90.
VALX(12,9) = -80.
VALX(12,10)= -70.
VALX(12,11)= -60.
VALX(12,12)= -50.
VALX(12,13) = -40.
VALX(12,14)= -30.
VALX(13,1) = 0.
VALX(13,2) = 50.
VALX(13,3) = 70.
VALX(13,4) = 80.
VALX(13,5) = 95.
VALX(13,6) = 100.
VALX(13,7) = 105.
VALX(13,8) = 110.
VALX(13,9) = 120.
VALX(13,10)= 150.
VALX(13,11)= 200.
VALX(13,12)= 300.
VALX(13,13)= 400.
VALX(14,1) = 7.75
VALX(14,2) = 7.8
VALX(14,3) = 7.85
VALX(14,4) = 7.9
VALX(14,5) = 7.95
VALX(14,6) = 8.00
VALX(14,7) = 8.05
VALX(14,8) = 8.1
VALX(14,9) = 8.15
VALX(14,10)= 8.2
VALX(14,11)= 8.25
VALX(14,12)= 8.3
VALX(14,13)= 8.4
VALX(14,14)= 8.5

```

section.f

```

VALX(15,1) = 75.
VALX(15,2) = 80.
VALX(15,3) = 85.
VALX(15,4) = 90.
VALX(15,5) = 95.
VALX(15,6) = 100.
VALX(15,7) = 105.
VALX(15,8) = 110.
VALX(15,9) = 115.
VALX(15,10)= 120.
VALX(15,11)= 125.
VALX(15,12)= 130.
VALX(15,13)= 135.
VALX(15,14)= 140.
VALX(16,1) = 2.5
VALX(16,2) = 3.
VALX(16,3) = 3.5
VALX(16,4) = 4.
VALX(16,5) = 4.5
VALX(16,6) = 5.
VALX(16,7) = 5.5
VALX(16,8) = 6.
VALX(16,9) = 6.5
VALX(16,10)= 7.
VALX(16,11)= 7.5
VALX(16,12)= 8.
VALX(16,13)= 8.5
VALX(16,14)= 9.
VALX(17,1) = 0.
VALX(17,2) = 0.25
VALX(17,3) = 0.5
VALX(17,4) = 0.75
VALX(17,5) = 1.
VALX(17,6) = 1.25
VALX(17,7) = 1.5
VALX(17,8) = 1.75
VALX(17,9) = 2.
VALX(17,10)= 2.5
VALX(17,11)= 3.
VALX(17,12)= 3.5
VALX(17,13)= 4.
VALX(17,14)= 4.5
VALX(18,1) = -1.
VALX(18,2) = 0.
VALX(18,3) = 2.
VALX(18,4) = 4.
VALX(18,5) = 6.
VALX(18,6) = 8.
VALX(18,7) = 10.
VALX(18,8) = 12.5
VALX(18,9) = 15.
VALX(18,10)= 17.5
VALX(18,11)= 20.
VALX(18,12)= 25.
VALX(18,13)= 30.
VALX(18,14)= 35.
VALX(19,1) = 30.
VALX(19,2) = 31.
VALX(19,3) = 32.
VALX(19,4) = 33.
VALX(19,5) = 34.
VALX(19,6) = 34.25
VALX(19,7) = 34.5
VALX(19,8) = 34.75
VALX(19,9) = 35.

VALX(19,10)= 35.25
VALX(19,11)= 35.5
VALX(19,12)= 36.
VALX(19,13)= 36.5
VALX(19,14)= 37.
VALX(20,1) = 0.
VALX(20,2) = 0.25
VALX(20,3) = 0.5
VALX(20,4) = 0.75
VALX(20,5) = 1.
VALX(20,6) = 5.
VALX(20,7) = 10.
VALX(20,8) = 25.
VALX(20,9) = 50.
VALX(20,10)= 75.
VALX(20,11)= 100.
VALX(20,12)= 150.
VALX(20,13)= 200.
VALX(20,14)= 250.

1VAL=14
DO 7854 IVX=1,1VAL
    VALS(IVX)=VALX(IFR,IVX)
7854 CONTINUE

RETURN
END

C      SUBROUTINE AVG(SARRAY,IFRST,ISEC,AVERAGE)
C
C      THIS SUBROUTINE IS USED TO FIND THE AVERAGE OF ANY TWO DIMENSIONAL
C      ARRAY WITH DIMENSIONS OF IFRST,ISEC. IT WILL RETURN THE AVERAGE
C      IN THE VARIABLE 'AVERAGE'. IT IS SET TO NOT INCLUDE ANY ITEMS THAT
C      ARE PRESENTLY SET TO ZERO IN THE AVERAGE. THIS MAY NEED TO BE
C      CHANGED WITH A DIFFERENT SET OF DATA
C
C      DIMENSION SARRAY(IFRST,ISEC)
REAL*4 SARRAY,AVERAGE,TEMP
C
C      TEMP = 0.
AVERAGE = 0.
ITEM = 0
ITEM = IFRST*ISEC
DO 10 I = 1,IFRST
DO 10 J = 1,ISEC
    IF (SARRAY(I,J) EQ .0.) THEN
        ITEM = ITEM + 1
    ELSE
        TEMP = TEMP + SARRAY(I,J)
    ENDIF
10   CONTINUE
AVERAGE = (TEMP / ITEM)
RETURN
END

SUBROUTINE HIGH(SARRAY,IFRST,ISEC,HIGH)
C
DIMENSION SARRAY(IFRST,ISEC)
REAL*4 SARRAY,HIGH
C
HIGH = -11E+11
DO 10 I 1,IFRST
DO 10 J 1,ISEC
    IF (SARRAY(I,J) EQ .0.) THEN
        IF (SARRAY(I,J) GT HIGH) THEN

```

section.f

```
      HGH = SARRAY(I,J)
      ENDIF
8     CONTINUE
10    CONTINUE
C
      RETURN
END

SUBROUTINE LOWS(SARRAY,IFRST,ISEC,ALOW)
C
      DIMENSION SARRAY(IFRST,ISEC)
      REAL*4 SARRAY,ALOW
C
      ALOW = .11E+11
      DO 10 I=1,IFRST
      DO 10 J=1,ISEC
         IF(SARRAY(I,J).EQ.0.) GOTO 8
         IF(SARRAY(I,J).LT.ALOW) THEN
            ALOW = SARRAY(I,J)
         ENDIF
8        CONTINUE
10    CONTINUE
C
      RETURN
END
```


APPENDIX C

Lsgmodel Subdirectory Files

cparr.f

```

SUBROUTINE ARROWS(IFR,ARROWX,ARROWY)
C
C Declare required data arrays and workspace arrays.
C
C =====
C INCLUDE 'data.txt'
PARAMETER (IED2 = IE/2, JED2 = JE/2, SMVECT = 0.2)
DIMENSION IASF(13), LIND(14), SPV(2), ARROWX(IE,JE),
*           ARROWY(IE,JE), XARR(IED2,JED2), YARR(IED2,JED2),
*           TEMPX(IED2,JED2), TEMPY(IED2,JED2)
DATA IASF / 13*1 /
DATA LIND / 2,3,4,5,6,7,8,9,10,11,12,13,14,15 /
C
C Declare arrays to hold the list of indices and the list of labels
C required by the label-bar routine.
C
C LLBS CONTROLS THE NUMBER OF CHARACTERS PRINTED OUT FOR THE LABEL BAR.
C AT THE PRESENT TIME IT IS SET TO SHOW ONLY FOUR CHARACTERS. THIS MAY
C HAVE TO BE CHANGED FOR DIFFERENT DATA.
C
CHARACTER*4 LLBS(15)
CHARACTER CD*8, CT*10, TEXT3*12
C
C =====
C EXTERNAL COLRAT
C
C CALL DATE(CD)
C
C Turn off the clipping indicator.
C
CALL GSCLIP (1)
C
C Set all aspect source flags to "individual".
C
CALL GSASF (IASF)
C
C Force solid fill.
C
CALL GSFAIS (1)
C
C Define color indices.
C
CALL DFCLRS
C
C Get the current elapsed time, in seconds.
C
TIME=SECOND(DUMI)
C
C Force the plot into the portion of the frame explicitly defined
C by the calls to set the bottom,top,left and right of the
C viewport.
C
CALL CPSETR('VPS - VIEWPORT SHAPE',0.)
CALL CPSETR('VPT - VIEWPORT TOP',.85)
CALL CPSETR('VPB - VIEWPORT BOTTOM',.35)
CALL CPSETR('VPL - VIEWPORT LEFT',.01)
CALL CPSETR('VPR - VIEWPORT RIGHT',.99)
C
C Disallow the trimming of trailing zeroes.
C
CALL CPSETI ('NOF - NUMERIC OMISSION FLAGS',0)

```

83

```

C
C CALL CPSETR ('SPV - SPECIAL VALUE FLAG',RMASK)
C Tell CONPACK to use 13 contour levels, splitting the range into 14
C equal bands, one for each of the 14 colors available.
C
CALL CPSETI('CLS - CONTOUR LEVEL SELECTOR',NUMCONT)
CALL CPSETR('T2D - TWO-DIMENSIONAL SMOOTHING',2.5)
C Initialize the drawing of the contour plot.
C
CALL CPRECT (ZDAT,IE,IE,JE,RWRK,IWA,IWRK,IWA)
C Initialize the area map and put the contour lines into it.
C
CALL ARINAM (IAMA,IMA)
CALL CPCLAM (ZDAT,RWRK,IWRK,IMA)
C Color the map.
C
CALL ARSCAM (IAMA,XCRA,YCRA,IWA,IMA,IGIA,NL,COLRAT)
C Draw a label bar for the plot, relating colors to values.
C
CALL CPGETR ('ZMN',ZMIN)
CALL CPGETR ('ZMX',ZMAX)
DO 102 I=1,15
    CALL CPSETR ('ZDV - Z DATA VALUE',
     *             ZMIN+REAL(I-1)*(ZMAX-ZMIN)/14.)
    CALL CPGETC ('ZDV - Z DATA VALUE',LLBS(I))
102 CONTINUE
C
CALL LBSETI ('CBL - COLOR OF BOX LINES',1)
CALL LBSETI ('CLB - COLOR OF LABELS ',TEXTCLR)
CALL LBLBAR (.05,.95,.15,.25,14,1.,.5,LIND,0,LIND,15,1)
C
C Compute and print statistics for the plot, label it, and put a
C boundary line at the edge of the plotter frame.
C
CALL CAPSAP (HEADER,TIME,IAMA,IMA,LAYER)
CALL LABTOP (HEADER,.017)
WRITE(TEXT3,'(F10.0,A2)') ZLAYER(LAYER), ' H'
CALL LABELS(CD,'DEPTH = ',TEXT3,'PLOTTED ','CREATED ',
     * TEXT6,.011,.013,.011,.011,.011,IFR,TITLE,.011)
SPV(1) = 0.
SPV(2) = 0.
C
C THIN OUT THE ARRAY SO THAT THERE IS NOT AN ARROW AT EVERY GRIDPOINT
C
II = 1
DO 100 J = 1,JE,2
JJ = 1
DO 99 J = 1,JE,2
    XAPP(IJ,JJ) = ARROWX(I,J)
    YAPP(IJ,JJ) = ARROWY(I,J)
    JJ = JJ + 1
99 CONTINUE
    II = II + 1
100 CONTINUE
C
C NORMALIZE THE ARROW LENGTH
DO 101 I = 1,IED2
DO 101 J = 1,JED2
    ZAPP = SQRT((XAPP(I,J)**2)+(YAPP(I,J)**2))

```

```

IF (SQXY.GT.SMVECT) THEN
  TEMPX(I,J) = XARR(I,J) / SQXY
  TEMPY(I,J) = YARR(I,J) / SQXY
ELSE IF (SQXY.NE.0.) THEN
  TEMPX(I,J) = 0.
  TEMPY(I,J) = 0.
ELSE
  TEMPX(I,J) = XARR(I,J)
  TEMPY(I,J) = YARR(I,J)
ENDIF
101  CONTINUE
C
C CALL THE PLOT PACKAGE VELVCT TO PLOT THE ARROWS.
C
C --- MAKE THE ARROWS BLACK & SET LINE WIDTH TO 2.5 ---
C
CALL GSPLCI()
CALL GSLWSC(2.5)
CALL VELVCT(TEMPX, IED2, TEMPY, IED2, IED2, JED2, 0., 0., -1, 16, 4, SPV)
C
C --- RESET THE COLOR TO CYAN AND LINE WIDTH TO 1.0 ---
C
CALL GSPLCI(1)
CALL GSLWSC(1.0)
CALL PLOTBNDY

C
C CALL THE SUBROUTINE TO PLOT THE CONTINENTS ON THE MAP.
C
CALL MAPOVR
CALL GSPLCI(1)
C
=====
C
C      RETURN
END
C
C-----+
C
C      SUBROUTINE COLRAT (XCRA,YCRA,NCRA,IAIA,IGIA,NAIA)
C
C      DIMENSION XCRA(*),YCRA(*),IAIA(*),IGIA(*)
C
C The arrays XCRA and YCRA, for indices 1 to NCRA, contain the X and Y
C coordinates of points defining a polygon. The area identifiers in
C the array IAIA, each with an associated group identifier in the array
C IGIA, tell us whether the polygon is to be color-filled or not.
C
C
C Assume the polygon will be filled until we find otherwise.
C
C      IFLL=1
C
C If any of the area identifiers is negative, don't fill the polygon.
C
DO 101 I=1,NAIA
  IF (IAIA(I).LT.0) IFLL=0
101  CONTINUE
C
C Otherwise, fill the polygon in the color implied by its area
C identifier relative to edge group 3 (the contour line group).
C
IF (IFLL.NE.0) THEN
  IFLL=0
  DO 102 I=1,NAIA

```

```

    IF (IGIA(I).EQ.3) IFLL=IAIA(I)
102  CONTINUE
    IF (IFLL.GT.0.AND.IFLL.LT.15) THEN
      CALL GSFAC (IFLL,1)
      CALL GFA (NCRA-1,XCRA,YCRA)
    END IF
    C
    C Done.
    C
    RETURN
    C
    END
C
C -----
C
C      SUBROUTINE LABELS(TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6,SIZE1,
C      SIZE2,SIZE3,SIZE4,SIZE5,SIZE6,IFR,TEXT7,SIZE7)
C
C THIS PROCEDURE IS USED TO PLACE THE DEPTH AND THE DATES ON THE
C MAP. IT IS SIMILAR TO THE PROCEDURE TO PLOT THE HEADER ON THE MAP.
C
C THE FOLLOWING LOOP IS USED TO PLACE THE LABELS ON THE MAP.
C PLCHHQ IS A PLOTCHAR CALL THAT POSITIONS THE LABEL, DETERMINES THE SIZE,
C DETERMINES THE ORIENTATION (HORIZONTAL OR VERTICAL) AND WHETHER TO
C RIGHT JUSTIFY, LEFT JUSTIFY, OR CENTER THE LABEL.
C
C      CHARACTER*(*) TEXT1,TEXT2,TEXT3,TEXT4,TEXT5,TEXT6,TEXT7
C
C SET AND GETSET ARE SPPS CALLS AND MAY NEED TO BE CHANGED AT A LATER DATE
C
CALL GETSET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDP,YWDR,YWDT,LHID)
CALL   SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PGSETI ('QU'  QUALITY FLAG,1QUA)
CALL PCSETI ('QU'  QUALITY FLAG,0)
CALL PCSETI ('TE'  TEXT EXTENT COMPUTATION FLAG,0)
DO 300 I=1,7
  GOTO (1,2,3,4,5,6,7), I
  1  CALL PLCHHQ(.380,.100,TEXT1,SIZE1,0.,+1.5)
  2  CONTINUE
    IF(IFR.LT.20) THEN
      CALL PLCHHQ(.540,.300,TEXT2,SIZE2,0.,+1.5)
    ENDIF
  GOTO 300
  3  CONTINUE
    IF(IFR.LT.20) THEN
      CALL PLCHHQ(.690,.300,TEXT3,SIZE3,0.,+1.5)
    ENDIF
  GOTO 300
  4  CALL PLCHHQ(.225,.100,TEXT4,SIZE4,0.,+1.5)
  5  CALL PLCHHQ(.780,.100,TEXT5,SIZE5,0.,+1.5)
  6  CALL PLCHHQ(.910,.100,TEXT6,SIZE6,0.,+1.5)
  7  CALL PLCHHQ(.640,.030,TEXT7,SIZE7,0.,+1.5)
300  CONTINUE
CALL PCSETI ('QU'  QUALITY FLAG,1QUA)
CALL   SET (XVPL,XVPR,YVPR,YVPT,XWDL,XWDP,YWDR,YWDT,LHID)
C
RETURN
END

```

```

SUBROUTINE MCUT(XTOP,XBOT,YLFT,YRGT,ICR,DEPTH)
C
C Declare required data arrays and workspace arrays.
C
INCLUDE 'data2.txt'
DIMENSION IASF(13), LIND(14), DEPTH(IMAX,JMAX)
DATA IASF / 13*1 /
DATA LIND / 2,3,4,5,6,7,8,9,10,11,12,13,14,15 /
C
C Declare arrays to hold the list of indices and the list of labels
C required by the label-bar routine.
C
CHARACTER*10 LLBS(15)
CHARACTER CD*8, CT*10, TEXT3*12
EXTERNAL DRAWCL
C
CALL DATE(CD)
C
C Turn off the clipping indicator.
C
CALL GSCLIP(0)
C
C Set all aspect source flags to "individual".
C
CALL GSASF(IASF)
C
C Define color indices.
C
CALL DFCLRS
C
C Get the current elapsed time, in seconds.
C
TIME=SECOND(DUMI)
C
C Force the plot into the portion of the frame explicitly defined
C by the calls to set the bottom,top,left and right of the
C viewport.
C
CALL CPSETR('VPS - VIEWPORT SHAPE',0.)
CALL CPSETR('VPT - VIEWPORT TOP',XTOP)
CALL CPSETR('VPB - VIEWPORT BOTTOM',XBOT)
CALL CPSETR('VPL - VIEWPORT LEFT',YLFT)
CALL CPSETR('VPR - VIEWPORT RIGHT',YRGT)
C
C Disallow the trimming of trailing zeroes.
C
CALL CPSETI('NOF - NUMERIC OMISSION FLAGS',0)
CALL CPSETR('SPV - SPECIAL VALUE FLAG',RMASK)
C
C Tell CONPACK to use 20 contour levels and set up the labels.
C
CALL CPSETI('CLS - CONTOUR LEVEL SELECTOR',NUCONT)
C
C Turn on the positioning of labels by the penalty scheme.
C
CALL CPSETI('LLP - LINE LABEL POSITIONING',3)
C
C Label highs and lows with just the number, boxed and in purple.
C
CALL CPSETC('HLT - HIGH/LOW TEXT','SZDVS')
CALL CPSETI('HLB - HIGH/LOW LABEL BOX FLAG',1)
CALL CPSETI('HLC - HIGH/LOW LABEL COLOR INDEX',15)
C
C Delete high/low labels which overlap the informational label, another

```

```

C high/low label, or the edge.
C
CALL CPSETI('HLO - HIGH/LOW LABEL OVERLAP FLAG',2)
C
C Initialize the drawing of the contour plot.
C
CALL CPRECT(ZDAT,JE,JE,KE,RWRK,IWA,IWRK,IWA)
CALL CPPKCL(ZDAT,RWRK,IWRK)
C
C Move the informational label down away from the plot.
C
CALL CPSETI('ILA - INFORMATIONAL LABEL ANGLE',0.)
CALL CPSETR('ILX - INFORMATIONAL LABEL X POSITION', 20)
CALL CPSETR('ILY - INFORMATIONAL LABEL Y POSITION', -30)
CALL CPSETI('ILP - INFORMATIONAL LABEL POSITIONING', 4)
CALL CPSETI('ILB - INFORMATIONAL LABEL BOX',0)
CALL CPSETR('ILS - INFORMATIONAL LABEL SIZE',.015)
CALL CPSETR('ILW - INFORMATIONAL LABEL LINE WIDTH',2.)
CALL CGPSETI('NCL - NUMBER OF CONTOUR LEVELS',NCLV)
C
C MAKE THE NEGATIVE VALUES GREEN, POSITIVE VALUES RED, ZERO VALUES
C CYAN. IF A LABELED CONTOUR MAKE IT A DARKER SHADE AND THE LABEL TO
C MATCH THE LINE.
C
DO 102 ICLV=1,NCLV
CALL CPSETI('PAI - PARAMETER ARRAY INDEX',ICLV)
CALL CPGETR('CLV - CONTOUR LEVEL',CLEV)
CALL CPGETI('CLU - CONTOUR LEVEL USE',ICLU)
IF (CLEV.LT.0.) THEN
  IF (ICLU.EQ.1) THEN
    CALL CPSETR('CLC - CONTOUR LINE COLOR',8)
  ELSE
    CALL CPSETI('CLC - CONTOUR LINE COLOR',9)
    CALL CPSETR('LLC - LINE LABEL COLOR',9)
  ENDIF
ELSE IF (CLEV.EQ.0.) THEN
  CALL CPSETR('CLC - CONTOUR LINE COLOR',1)
  CALL CPSETI('LLC - LINE LABEL COLOR',1)
ELSE
  IF (ICLU.EQ.1) THEN
    CALL CPSETR('CLC - CONTOUR LINE COLOR',13)
  ELSE
    CALL CPSETI('CLC - CONTOUR LINE COLOR',14)
    CALL CPSETR('LLC - LINE LABEL COLOR',14)
  ENDIF
ENDIF
102  CONTINUE
C
C Initialize the area map.
C
CALL APINAM(IAMA,IBA)
C
C Put label boxes in the area map.
C
CALL CPGBAM(ZDAT,PWPK,IPPK,IAMA)
C
C Draw contour lines, avoiding drawing through label boxes.
C
CALL CIRCLEM(ZDAT,PWPK,IPPK,IAMA,IPRFL)
C
C Fill in the labels.
C
CALL CPLBDR(ZDAT,PWPK,IPPK)
C
CALL GSPLCI(1)

```

```

C Compute and print statistics for the plot, label it, and put a
C boundary line at the edge of the plotter frame.
C
C     CALL CAPSAP (HEADER,TIME,IAMA,IMA,LAYER)
C     CALL LABTOP (HEADER,.017)
C     CALL LABELS2(CD,'PLOTTED ','CREATED ',TEXT6,.011,.011,
C     .011,.011,TITLE,.011,SECNAM,.013)
C     CALL LABELAXIS(ZLAYER,KE,XTOP,XBOT,YLFT,YRGT)
C     CALL PLOTBNDRY(XTOP,XHOT,YLFT,YRGT)
C     CALL BNDARY
C     CALL GCLRWK(1.0)
C     RETURN
C     END
C
C-----SUBROUTINE DRAWCL(XCS,YCS,NCS,IAI,IAG,NAI)
C
C This version of DRAWCL draws the polyline defined by the points
C ((XCS(I),YCS(I)),I=1,NCS) if and only if none of the area identifiers
C for the area containing the polyline are negative. The dash package
C routine CURVED is called to do the drawing.
C
C     DIMENSION XCS(*), YCS(*), IAI(*), IAG(*)
C
C     IDR=1
C     DO 101 I=1,NAI
C       IF (IAI(I).LT.0) IDR = 0
C 101  CONTINUE
C     IF (IDR.NE.0) CALL CURVED (XCS,YCS,NCS)
C
C     RETURN
C     END
C-----SUBROUTINE LABELS2(TEXT1,TEXT4,TEXT5,TEXT6,SIZE1,
C     *           SIZE4,SIZE5,SIZE6,TEXT7,SIZE7,TEXT8,SIZE8)
C
C THIS PROCEDURE IS USED TO PLACE THE SECTION TITLE AND THE DATES ON
C THE SEC. IT IS SIMILAR TO THE PROCEDURE TO PLOT THE HEADER ON THE
C SECTION.
C
C     CHARACTER*(*) TEXT1,TEXT4,TEXT5,TEXT6,TEXT7,TEXT8
C
C SET AND GETSET ARE SPPS CALLS AND AT A LATER DATE MAY NEED TO BE CHANGED.
C
C     CALL GETSET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C     CALL SET (0.,1.,0.,1.,0.,1.,1.)
C     CALL PCGETI ('QU - QUALITY FLAG',IQUA)
C     CALL PCSETI ('QU - QUALITY FLAG',0)
C     CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
C     CALL GSPLCI(1)
C     DO 300 I=1,6
C       GOTO (1,2,3,4,5,6) I
C 1     CALL PLCHHQ(.380,.100,TEXT1,SIZE1,0.,+1.5)
C     GOTO 300
C 2     CALL PLCHHQ(.225,.100,TEXT4,SIZE4,0.,+1.5)
C     GOTO 300
C 3     CALL PLCHHQ(.780,.100,TEXT5,SIZE5,0.,+1.5)
C     GOTO 300
C 4     CALL PLCHHQ(.910,.100,TEXT6,SIZE6,0.,+1.5)
C     GOTO 300
C 5     CALL PLCHHQ(.640,.030,TEXT7,SIZE7,0.,+1.5)

```

cpmeut.f

```

6          GOTO 300
300      CALL PLCHHQ(.780,.300,TEXT8,SIZE8,0.,+1.5)
CONTINUE
CALL PCSETI ('QU - QUALITY FLAG',IQUA)
CALL SET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
C
RETURN
END
C
C-----SUBROUTINE LABELAXIS(ZLAYER,EE,XTOP,XBOT,YLFT,YRGT)
C
C THIS SUBROUTINE IS USED TO LABEL THE X AND Y AXIS OF THE SECTION PLOTS
C PUTTING THE DEPTH ON THE Y AXIS.
C
C THIS IS HAMBURG DEPENDANT.
C
DIMENSION ZLAYER(0:12)
CHARACTER*4 XAXISLAB(7)
CHARACTER*9 DEPTH(11)
CHARACTER*2 DASH
C
CALL GETSET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
CALL SET (0.,1.,0.,1.,0.,1.,1.)
CALL PCGETI ('QU - QUALITY FLAG',IQUA)
CALL PCSETI ('QU - QUALITY FLAG',0)
CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
DO 1 L=1,11
  WRITE(DEPTH(L),'(F6.0,A3)') ZLAYER(L),'
1  CONTINUE
YCOR = XTOP
XCOR = YLFT + .035
DO 100 K=1,6
  CALL PLCHHQ(XCOR,YCOR,DEPTH(K),.008,0.,+1.5)
  YCOR = YCOR -.04985
100  CONTINUE
CALL PLCHHQ(.050,.700,'DEPTH M',.009,90.,+1.5)
XAXISLAB(1) = '90 S'
XAXISLAB(2) = '60 S'
XAXISLAB(3) = '30 S'
XAXISLAB(4) = ' EQ '
XAXISLAB(5) = '30 N'
XAXISLAB(6) = '60 N'
XAXISLAB(7) = '90 N'
DASH = ''
XCORD = YLFT
YCIRD = XBOT+.02
YCOT = YLFT+.030
YCOT = XBOT+.030
DO 102 N=1,7
  CALL PLCHHQ(XCORD,YCOT,DASH,.008,90.,+1.5)
  CALL PLCHHQ(XCOT,YCOT,XAXISLAB(N),.008,0.,+1.5)
  XCOT = XCOT + .1165
  XCOT = XCOT + .1165
102  CONTINUE
CALL PCGETI ('QU - QUALITY FLAG',IQUA)
CALL SET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
RETURN
END

```

makefile for lsg post-processor

```
# This is the makefile to compile the files for the lsg side of the
# model.
#
SHELL = /bin/sh
#
IBMTIMES = ../postdir/ibm_times.o
DATIMX = ../postdir/datimx.o
LITCONT = ../postdir/litcont.o
CONT = ../postdir/cont.o
GKSSTATE = ../postdir/gksstate.o
CPEXCC = ../postdir/cpexcc.o
PLOTTERS = ../postdir/plotters.o
#
readplo : $(PLOTTERS) $(LITCONT) $(CONT) $(GKSSTATE) readtest.o plodin.o\
          cparr.o cpmcut.o trapin.o runlsg.o $(CPEXCC) $(IBMTIMES) \
          $(DATIMX)
          xlf -g -qxref=full -bloadmap:loadmap -o readplo $(PLOTTERS) \
          $(LITCONT) $(CONT) $(GKSSTATE) readtest.o plodin.o cparr.o \
          cpmcut.o trapin.o runlsg.o $(CPEXCC) \
          $(IBMTIMES) $(DATIMX) -L/usr/local/lib -lincarg_gks \
          -lincarg -lincarg_ras -lincarv -lincarg_loc
#
plodin.o : plodin.f
          xlf -c plodin.f
runlsg.o : runlsg.f
          xlf -c runlsg.f
cparr.o : cparr.f
          xlf -c cparr.f
cpmcut.o : cpmcut.f
          xlf -c cpmcut.f
trapin.o : trapin.f
          xlf -c trapin.f
readtest.o : readtest.f
          xlf -c readtest.f
```

mcutdat.txt

```
PARAMETER(IE=72, JE=72, KE=11, NMAP = 26, IMAX = 78,  
*          JMAX = 78, KMAX = 25, IWA = 10000, IMA = 200000,  
*          NV = 12, NS = 10, IS = 4, ISMAX = 9,  
*          RMASK = -.1111E+11)  
  
C  
COMMON/MAPS/SNAME(NMAP),ISPAPL(NMAP),SFIL(NMAP),ILAY(KE),  
*          VALS(100),HEADER,TEXT6,IVAL,IFINE,DEPTP(IE,JE),  
*          CHOSEN(IE,JE),TITLE,ISECT(IS),SECNAM,INDISU(JE)  
COMMON/PLOTTERS/RWRK(IWA),IWRK(IWA),IAMA(IMA),  
1          XCRA(IWA),YCRA(IWA),IAIA(NS),IGIA(NS)  
COMMON/DATA/ARRAY(IE,JE),ZDAT(JE,KE+1),MHZ(IMAX,JMAX),  
1          ZLAYER(0:NV),LAYER,MHZIO(IMAX),IISEC(JMAX,ISMAX)  
REAL*4 ZDAT,RWRK,XCRA,YCRA,ARRAY,ZLAYER  
INTEGER*4 IWRK,IAMA,IAIA,IGIA,LAYER  
LOGICAL GLOMAP,MHZIO,MHZ  
CHARACTER*40 HEADER,SNAME,TITLE  
CHARACTER*20 SECNAM  
CHARACTER*6 SFIL  
CHARACTER*8 TEXT6
```

```

SUBROUTINE CUTPIC(ZPREL, IFIELD, ZARRAY, SNAM, IZPRNUM, IZNUM, XARRAY,
*                   IDDR, RDDR)
C
C THIS IS THE INTERFACE FOR THE SECTION DATA. IT SETS UP THE NECESSARY
C VARIABLES AND THEN SENDS THE DATA TO THE NECESSARY SUBROUTINE
C DEPENDING ON WHICH TYPE OF DATA IS BEING USED.
C
C INCLUDE 'data2.txt'
C
C =====
C
C      DIMENSION ZPREL(6,50), IFIELD(JE), ZARRAY(JE,KE,50), DUMMY(JE,KE),
C      *           DEPTH(IMAX,JMAX), XARRAY(IE,JE,50), IDDR(512), RDDR(512),
C      *           ZDEPTH(101), DDZ(IE,JE,KE)
C
C      REAL*8 ZARRAY, ZPREL, XARRAY, RDDR
C      CHARACTER*40 SNAM
C
C =====
C
C SET UP THE DEPTH LOCATER DATA
C
C      DO 216 K=1, IDDR(21)
C      ZDEPTH(K) = RDDR(K+9)
216  CONTINUE
C      ZLAYER(0) = 0.
C      DO 217 K=1, KE
C          ZLAYER(K) = ZDEPTH(K)
217  CONTINUE
C      ZLAYER(KE+1) = 6000.
C      DO 11 I=1, IE
C          DO 11 J=1, JE
C              DEPTH(I,J) = XARRAY(I,J,1)
11   CONTINUE
C
C READ DDZ FROM THE FILE DDZ
C
C THE FILE 'ddz' IS CREATED ON THE CARBON SIDE OF THE MODEL AND IS USED TO
C FIND THE BOTTOM ON THE SECTION MAP. DDZ IS USED BECAUSE IT GIVES A
C MORE PRECISE BOTTOM ON THE SECTION SIDE.
C
C      OPEN(22,FILE='ddz',FORM='FORMATTED', STATUS='OLD')
C
C      READ(22,'(5F15.5)') DDZ
C
C      CLOSE(22)
C
C CHOOSE THE CORRECT SECTION
C
C      GO TO (999, 2, 3,999, 5,999,999,999, 9, 10,
C      #         999,999,999,999,999,999,999,999,999,
C      #         999, 62,999,999,999,999,999,999,999,999,
C      #         999,999,999,999,999,999,999,999,999,999,
C      #         999,999,999,999,999,999,999,999,999,999,
C      #         999,999,999,999,999,999,999,999,999,999)
C      # NINT(ABS(ZPREL(3,IZPRNUM)))
C
C      RETURN
C
C SET THE CORRECT HEADER FOR THE PLOT AND MULTIPLY THE DATA BY A FACTOR
C DEPENDENT UPON THE SECTION WANTED.

```

plodin.f

```

C
C 2  CONTINUE
C      HEADER = 'POTENTIAL TEMPERATURE (DEG C)'
C      SECNAM = SNAM
C      DO 1000 K=1,KE
C      DO 1000 J=1,JE
C          IF (ZARRAY(J,K,IZNUM).NE.RMASK) THEN
C              ZARRAY(J,K,IZNUM) = ZARRAY(J,K,IZNUM) - 273.16
C          ENDIF
1000 CONTINUE
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 3  CONTINUE
C      HEADER = 'ZONAL VELOCITY (CM/S)'
C      SECNAM = SNAM
C      DO 50 50 J=1,JE
C      DO 50 50 K=1,KE
C          IF (ZARRAY(J,K,IZNUM).NE.RMASK)
C              *          ZARRAY(J,K,IZNUM)=ZARRAY(J,K,IZNUM)*100.
50   CONTINUE
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 5  CONTINUE
C      HEADER = 'SALINITY (‰)'
C      SECNAM = SNAM
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 9  CONTINUE
C      HEADER = 'SIGMA-THETA'
C      SECNAM = SNAM
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 10 CONTINUE
C      HEADER = 'DEPTH 14-C'
C      SECNAM = SNAM
C      DO 1400 K=1,KE
C      DO 1400 J=1,JE
C          IF (ZARRAY(J,K,IZNUM).NE.RMASK)
C              *          ZARRAY(J,K,IZNUM) = ZARRAY(J,K,IZNUM) *1000. - 1000.
1400 CONTINUE
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 62  CONTINUE
C      HEADER = 'TEMPERATURE (DEG C)'
C      SECNAM = SNAM
C      CALL DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C      RETURN
C
C 999 WRITE(*,*)
C      * THAT IS NOT A VALID SECTION CHOICE
C      * EXITEN
C      END
C
C SUBROUTINE DATASEC(ZARRAY, IZNUM, DEPTH, IDDR, RDDR, IFIELD, DDZ)
C
C THIS SUBROUTINE TAKES THE DATA AND PUTS IT INTO THE ZONAL ARRAY FOR
C ICAP AND MAKES SURE THAT ADDITIONAL VARIABLES ARE FILLED WITH
C APPROPRIATE ITEMS FOR THE PLOT.
C

```

```

C =====
C INCLUDE 'data2.txt'
C DIMENSION ZARRAY(JE,KE,50),DEPTH(IMAX,JMAX),IDDR(512),RDDR(512),
C   SECTNAM(ISMAX),DDZ(IE,JE,KE),IFIELD(JE)
C REAL*8 ZARRAY,RDDR
C CHARACTER*40 SECTNAM
C =====
C DO 1089 J=1,JE
C DO 1089 K=1,KE
C   MHZ(J,K) = .TRUE.
1089 CONTINUE
C XTOP = .88
C XBOT = .38
C YLFT = .15
C YRGT = .85
C
C IF (SECTNAM.EQ.'ATLANTIC'          ') THEN
C   ICR = 1
C ELSE IF (SECTNAM.EQ.'PACIFIC'      ') THEN
C   ICR = 3
C ELSE
C   ICR = 2
C ENDIF
DO 1987 J=1,JE
  INDISU(J) = IFIELD(J)
1987 CONTINUE
C
C MASK THE BOTTOM USING THE DDZ ARRAY.
C
C DO 1990 J=1,JE
C DO 1909 K=1,KE
C   IF (DDZ(INDISU(J),J,K).EQ.0.) THEN
C     KBOT = K
C     GOTO 2999
C   ENDIF
1909 CONTINUE
GOTO 1930
2999 CONTINUE
DO 1910 K=KBOT,KE
  ZARRAY(J,K,IZNUM) = RMASK
1910 CONTINUE
1990 CONTINUE
JP = JE
DO 1010 J=1,JE
KP = KE
DO 1009 K=1,KE
  ZDAT(J,K) = ZARRAY(JP,KP,IZNUM)
  KP = KP - 1
1009 CONTINUE
JP = JP - 1
1010 CONTINUE
C
C CALL THE INTERFACE FOR THE PLOT PACKAGE.
C
CALL PLOTSEC(XTOP,XBOT,YLFT,YRGT,ICR,DUMMY)
C =====
C RETURN
END

```

plodin.f

```

SUBROUTINE ARRPIC(ZPREL,XARR,YARR,I2PRNUM,IXNUM,IYNUM,RDDR,IPDR)
C THIS SUBROUTINE IS THE INTERFACE FOR THE VECTOR MAPS.
C INCLUDE 'data.txt'
C =====
C DIMENSION XARR(IE,JE,50),ZPREL(6,50),YARR(IE,JE,50),
C   XARRAY(IE,JE),YARRAY(IE,JE),DEPTH(IMAX,JMAX),
C   ARROWX(IE,JE),ARROWY(IE,JE),RDDR(512),IPDR(512)
C REAL*8 XARR,ZPREL,YARR,RDDR
C =====
C SET UP THE DEPTH ARRAY AND MANIPULATE THE DATA SOMEWHAT IN ACCORDANCE
C WITH THE ORIGINAL HAMBURG LSG MODEL.
C
DO 9001 I=1,IE
DO 9001 J=1,JE
  DEPTH(I,J) = XARR(I,J,1)
9001 CONTINUE
DO 9033 J=1,JE
DO 9033 I=1,IE
  YARRAY(I,J) = SQRT(XARR(I,J,IXNUM)*XARR(I,J,IYNUM))
  *           YARRAY(I,J,LYNUM)*YARR(I,J,IYNUM))
9033 CONTINUE
DO 9034 J=1,JE
DO 9034 I=1,IE
  ARROWX(I,J) = XARR(I,J,IXNUM)
  ARROWY(I,J) = YARR(I,J,IYNUM)
  IL=I-1
  IF(IL.LT.1) IL=IE
  JO = MAX0(J,1,1)
  JU = MIN0(J+1,JE)
  XARRAY(I,J) = 0.25*(YARRAY(I,J)+YARRAY(IL,J)+YARRAY(IE,J)
  *                   +YARRAY(I,JU))
9034 CONTINUE
C
CALL ARROA(XAPPAY,YARRAY,RDDR,IPDR,ZPREL,I2PRNUM,DEPTH)
C
SELECT THE CORRECT MAP
C
GO TO 1999,999, 3, -4,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999,
# 999,52,53,999,999,999,999,999,999,999,999,
# 999,999,63,64,999,999,999,999,999,999,999,
# 999,999,73,74,999,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999,
# 999,999,999,999,999,999,999,999,999,999
# NINT(ABS(ZPREL(3,I2PRNUM)))
C
RETURN
C
GET THE HEADER AND MULTIPLY BY A FACTOR CORRESPONDING TO THE DATA
C
3  CONTINUE
4  CONTINUE
HEADER = 'HORIZONTAL GRID SPACING (METERS)'
DO 5019 I=1,IE
DO 5019 J=1,JE
  IF ((XAPPAY(I,J).NE.0.0)) THEN

```

plodin.f

```

XARRAY(I,J) = XARRAY(I,J)*100.
YARRAY(I,J) = YARRAY(I,J)*100.
ENDIF
ARROWX(I,J) = ARROWX(I,J) * 100.
ARROWY(I,J) = ARROWY(I,J) * 100.
5019 CONTINUE
CALL ARROWSHIFT(IFR,XARRAY,YARRAY,ARROWX,ARROWY)
RETURN

C
37 CONTINUE
38 CONTINUE
HEADER = 'BAROTROPIC VELOCITY [CM/S]'
DO 5020 I=1,IE
DO 5020 J=1,JE
  IF (XARRAY(I,J).NE.RMASK) THEN
    XARRAY(I,J) = XARRAY(I,J)*100.
    YARRAY(I,J) = YARRAY(I,J)*100.
  ENDIF
  ARROWX(I,J) = ARROWX(I,J) * 100.
  ARROWY(I,J) = ARROWY(I,J) * 100.
5020 CONTINUE
SMVEC = 0.2
CALL ARROWSHIFT(IFR,XARRAY,YARRAY,ARROWX,ARROWY)
RETURN

C
52 CONTINUE
53 CONTINUE
HEADER = 'WINDSTRESS [MPA]'
DO 5021 I=1,IE
DO 5021 J=1,JE
  IF (XARRAY(I,J).NE.RMASK) THEN
    XARRAY(I,J) = XARRAY(I,J)*1000.
    YARRAY(I,J) = YARRAY(I,J)*1000.
  ENDIF
  ARROWX(I,J) = ARROWX(I,J) * 1000.
  ARROWY(I,J) = ARROWY(I,J) * 1000.
5021 CONTINUE
CALL ARROWSHIFT(IFR,XARRAY,YARRAY,ARROWX,ARROWY)
RETURN

C
63 CONTINUE
64 CONTINUE
RETURN

C
73 CONTINUE
74 CONTINUE
RETURN

C
C =====
C
999 WRITE(*,*) 'THIS IS NOT A VALID ARROW MAP CHOICE.'
RETURN
END

SUBROUTINE ARROA(XARRAY,YARRAY,RDDR,ZPREL,IZPPNUM,DEPTH)
C
C SET UP THE BOTTOM DATA
C
C =====
C
INCLUDE 'data.txt'
DIMENSION XARRAY(IE,JE),YARRAY(IE,JE),ZDEPTH(101),PDDP(512),
     *           IDDR(512),ZPREL(6,50),DEPTH(TMAX,JMAX)
REAL*8 RDDR,ZPREL

C
C =====
C
      DO 316 K=1,101
        ZDEPTH(K) = RDDR(K+9)
316 CONTINUE
ZAYER(0) = 0.
DO 317 K=1,KE
  ZAYER(K) = ZDEPTH(K)
317 CONTINUE
ZAYER(KE+1) = 6000.
DO 9999 I=0,KE+1
  IF (ZAYER(I).EQ.ZPREL(4,IZPPNUM)) THEN
    LAYER = I
  ENDIF
9999 CONTINUE
IF (LAYER.LT.1.OR.LAYER.GT.12) THEN
  IFR=21
ELSE
  IFR = 1
ENDIF
IF (LAYER.GE.0) THEN
  DO 2101 J=1,JE
    DO 2101 I=1,IE
      IF (DEPTH(I,J).LE.ZAYER(LAYER)) THEN
        MHZ(I,J) = .FALSE.
      ELSE
        MHZ(I,J) = .TRUE.
      ENDIF
2101 CONTINUE
ELSE
  DO 2102 I=1,IE
    DO 2102 J=1,JE
      MHZ(I,J) = .TRUE.
2102 CONTINUE
ENDIF
DO 2103 I=1,IE
DO 2103 J=1,JE
  IF(.NOT MHZ(I,J)) THEN
    XARRAY(I,J) = RMASK
    YARRAY(I,J) = RMASK
  ENDIF
2103 CONTINUE
C
C =====
C
RETURN
END

SUBROUTINE ARROWSHIFT(IFR,XARRAY,YARRAY,ARROWX,ARROWY)
C
C THIS SUBROUTINE GETS THE DATA INTO THE ARRAY ZDAT FOR THE PLOT
C PACKAGE TO USE AND INTERFACES WITH THE NCAR GRAPHICS FOR THE MAP AND
C FOR THE ARROW OVERLAY.
C
C =====
C
INCLUDE 'data.txt'
DIMENSION XARPAY(IE,JE),ATEMPY(JMAX),YARPAY(IE,JE),BTEMP(1MAX),
     *           TEMPY(IE,JE),ZARP(IE,JE,50),XARP(IE,JE),ARROWX(IE,JE),
     *           ARROWY(IE,JE),CTEMP(1MAX),DTEMP(1MAX),TEMPA(IE,JE),
     *           TEMPB(IE,JE)
C
C =====
C
END

```

plodin.f

```

JEND = JE
C ACTUAL SHIFTING OF THE ARRAY SO THAT IT IS CUT AT THE PRIME MERIDIAN
C ----- BEGIN -----
DO 1000 J=1,JEND
DO 1010 I=1,IEND
    ATEMP(I) = XARRAY(I,J)
    BTEMP(I) = YARRAY(I,J)
    CTEMP(I) = ARROWX(I,J)
    DTEMP(I) = ARROWY(I,J)
1010 CONTINUE
DO 1020 ITEMP=1,IEND
    I = ITEMP+MSHIFT*(J-1)/2
    IF(I.LE.0) I=I+IEND
    IF(I.GT.IEND) I=I-IEND
    XARRAY(I,J) = ATEMP(ITEMP)
    YARRAY(I,J) = BTEMP(ITEMP)
    ARROWX(I,J) = CTEMP(ITEMP)
    ARROWY(I,J) = DTEMP(ITEMP)
1020 CONTINUE
1000 CONTINUE
C
C ----- END -----
C FLIP THE INDICES SO THAT NCAR PLOTS THE DATA CORRECTLY.
C
DO 4270 I=1,IE
JP = JE
DO 4270 J=1,JE
    ZDAT(I,J) = XARRAY(I,JP)
    TEMPY(I,J) = YARRAY(I,JP)
    TEMPA(I,J) = ARROWX(I,JP)
    TEMPB(I,J) = ARROWY(I,JP)
    JP = JP - 1
4270 CONTINUE
DO 4277 I=1,IE
DO 4277 J=1,JE
    ARROWX(I,J) = TEMPA(I,J)
    ARROWY(I,J) = TEMPB(I,J)
4277 CONTINUE
TITLE = 'SAMPLE TEST'
CALL GSPLCI(1)
CALL ARROWS(IFR,ARROWX,ARROWY)
C
CALL GSPLCI(1)
C
C =====
C
C RETURN
END

SUBROUTINE ISOPIC(ZPREL,XARRAY,I2PRNUM,IXNUM,ICREDAT,RDDR,IDDR)
C THIS SUBROUTINE IS THE INTERFACE FOR THE MAP DATA
C
INCLUDE 'data.txt'
C
C =====
C
DIMENSION XARRAY(IE,JE,50),ZPREL(6,50),DEPTH(IMAX,JMAX),
* RDDR(512),IDDR(512),ZDEPTH(101)
REAL*8 XARRAY,ZPREL,RDDR
CHARACTER*6 CTEMP
C
C =====

```

plodin.f

```

C
C GET THE CORRECT HEADER AND MULTIPLY THE DATA WITH THE CORRECT FACTOR.
C
1 CONTINUE
HEADER = 'SURFACE ELEVATION (CM/S)'
DO 1500 I=1,JE
DO 1500 J=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 100.
1500 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
2 CONTINUE
HEADER = 'POTENTIAL TEMPERATURE (DEG C)'
DO 1510 J=1,JE
DO 1510 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) - 273.16
1510 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
3 CONTINUE
RETURN
C
4 CONTINUE
RETURN
C
5 CONTINUE
HEADER = 'SALINITY (0/00)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
6 CONTINUE
RETURN
C
7 CONTINUE
HEADER = 'VERTICAL VELOCITY (10**-6 M/S)'
DO 1520 J=1,JE
DO 1520 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1.E6
1520 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
8 CONTINUE
HEADER = 'SIGMA IN SITU'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
9 CONTINUE
HEADER = 'SIGMA THETA'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
10 CONTINUE
HEADER = 'DELTA 14-C'
DO 1530 J=1,JE
DO 1530 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1000. - 1000.
1530 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)

C
C RETURN
C
11 CONTINUE
HEADER = 'log SYNTHETIC TRACER'
DO 1710 J=1,JE
DO 1710 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK) THEN
IF(XARRAY(I,J,IXNUM).LE.1.E-50) THEN
XARRAY(I,J,IXNUM)=500.
ELSE
XARRAY(I,J,IXNUM) = DLOG10(XARRAY(I,J,IXNUM))
ENDIF
1710 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
13 CONTINUE
HEADER = 'ICE THICKNESS (M)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
15 CONTINUE
HEADER = 'ICE COMPACTNESS (0/01)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
18 CONTINUE
HEADER = 'HEAT FLUX (PREScribed) (W/M**2)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
27 CONTINUE
HEADER = 'STREAMFUNCTION (10**6 M**3/S)'
DO 1540 J=1,JE
DO 1540 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1.E 6
1540 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
37 CONTINUE
RETURN
C
38 CONTINUE
RETURN
C
52 CONTINUE
HEADER = 'WIND STRESS (ZONAL COMP.) (N/m)'
DO 1990 J=1,JE
DO 1990 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1000
1990 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM)
RETURN
C
53 CONTINUE
HEADER = 'WIND STRESS (MERID. COMP.) (N/m)'
DO 1980 J=1,JE
DO 1980 I=1,IE
IF (XARRAY(I,J,IXNUM).NE.RMASK)
* XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1000
1980 CONTINUE

```

plodin.f

```

CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
60 CONTINUE
HEADER = 'DENSITY DIFFERENCE (KG/M**3)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
61 CONTINUE
HEADER = 'GEOPOTENTIAL THICKNESS (10 M**2/S**2)'
DO 2000 J=1,JE
DO 2000 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM)=XARRAY(I,J,IXNUM)*0.1
2000 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
62 CONTINUE
HEADER = 'POTENTIAL TEMPERATURE (DEG C)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
63 CONTINUE
RETURN
C
64 CONTINUE
RETURN
C
65 CONTINUE
HEADER = 'FRESH WATER FLUX(PERSCRIBED) (MM/MONTH)'
DO 1550 J=1,JE
DO 1550 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 2.592E9
1550 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
66 CONTINUE
HEADER = 'POTENTIAL ENERGY LOSS BY CONVECTION (MW/M**2)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
67 CONTINUE
HEADER = 'FRESH WATER FLUX(DIAGNOSTIC) (MM/MONTH)'
DO 1560 J=1,JE
DO 1560 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.0.)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 2.592E9
1560 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
68 CONTINUE
HEADER = 'HEAT FLUX(DIAGNOSTIC) [W/M**2]'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
69 CONTINUE
HEADER = 'DEPTH OF CONVECTION (M)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
72 CONTINUE
C
93 CONTINUE
RETURN
C
73 CONTINUE
HEADER = 'WIND STRESS CORR. (ZONAL COMP.) (MPA)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
74 CONTINUE
HEADER = 'WIND STRESS CORR. (MERID. COMP.) (MPA)'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
75 CONTINUE
HEADER = '(P E)-CORRECTION (MM/MONTH)'
DO 1580 J=1,JE
DO 1580 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 2.592E9
1580 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
76 CONTINUE
RETURN
C
78 CONTINUE
HEADER = 'HEAT FLUX CORRECTION [W/M**2]'
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
90 CONTINUE
HEADER = 'ARGON 39 (%)'
DO 1000 J=1,JE
DO 1000 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 100.
1000 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
91 CONTINUE
HEADER = 'CONTRIBUTION OF NORTH ATLANTIC WATER (%)'
DO 1880 J=1,JE
DO 1880 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 100
1880 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
93 CONTINUE
HEADER = 'CONTRIBUTION OF NORTH PACIFIC WATER (%)'
DO 1860 J=1,JE
DO 1860 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 100
1860 CONTINUE
CALL DATATRANSFER(XARRAY,IXNUM,IFR)
RETURN
C
94 CONTINUE
HEADER = 'CONTRIBUTION OF ANTARCTIC WATER (%)'
DO 1850 J=1,JE
DO 1850 I=1,IE
  IF (XARRAY(I,J,IXNUM).NE.RMASK)
*
```

plodin.f

```

      * XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) + 100.
1850 CONTINUE
      CALL DATATRANSFER(XARRAY,IXNUM,IFR)
      RETURN
C
97  CONTINUE
      HEADER = 'dQ/dt Coupling coefficient [W/(m**2)K]'
      CALL DATATRANSFER(XARRAY,IXNUM,IFR)
      RETURN
C
98  CONTINUE
      CALL ISO98(IE,JE,ZPREL,XARRAY)
      HEADER = 'TOPOGRAPHY [KM]'
      DO 1590 J=1,JE
      DO 1590 I=1,IE
         IF (XARRAY(I,J,IXNUM).NE.0.)
      *   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1.E-3
1590 CONTINUE
      CALL DATATRANSFER(XARRAY,IXNUM,IFR)
      RETURN
C
99  CONTINUE
      HEADER = 'V-TOPOGRAPHY [KM]'
      DO 1600 J=1,JE
      DO 1600 I=1,IE
         IF (XARRAY(I,J,IXNUM).NE.RMASK)
      *   XARRAY(I,J,IXNUM) = XARRAY(I,J,IXNUM) * 1.E-3
1600 CONTINUE
      CALL DATATRANSFER(XARRAY,IXNUM,IFR)
      RETURN
C
C =====
C
999 WRITE(*,*) 'THIS IS NOT A VALID MAP CHOICE.
      RETURN
END

      SUBROUTINE DATATRANSFER(XARRAY,IXNUM,IFR)
C
C THIS IS THE INTERFACE TO THE NCAR GRAPHICS AND SETS THE NECESSARY
C VARIABLES TO THE CORRECT DATA.
C
C =====
C
INCLUDE 'data.txt'
DIMENSION XARRAY(IE,JE,50),ATEMP(IMAX)
REAL*8 XARRAY
C
C =====
C
IEND = IE
JEND = JE
C
C ACTUAL SHIFTING OF THE ARRAY SO THAT IT IS CUT AT THE PRIME MERIDIAN
C ----- BEGIN -----
DO 1000 J=1,JEND
      DO 1010 I=1,IEND
         ATEMP(I) = XARRAY(I,J,IXNUM)
1010 CONTINUE
      DO 1020 ITEMP=1,IEND
         I = ITEMP-MSHIFT-(J-1)/2
         IF(I.LE.0) I=1+IEND
         IF(I.GT.IEND) I=I-IEND
         XARRAY(I,J,IXNUM) = ATEMP(ITEMP)
1020 CONTINUE

```

56

```

1000  CONTINUE
C
C ----- END -----
C
      DO 4270 I=1,IE
      JP = JE
      DO 4270 J=1,JE
         ZDAT(I,J) = XARRAY(I,JP,IXNUM)
         JP = JP - 1
4270 CONTINUE
      TITLE = 'SAMPLE TEST
      CALL GSPLCI(1)
C
C INTERFACE THE NCAR GRAPHICS.
C
      CALL PLOTMAP(IFR)
      DO 4271 I=1,IMAX
         ATEMP(I) = 0.
4271 CONTINUE
C
      RETURN
END

      SUBROUTINE MCUT04(ZPREL,IFIELD,SNAM,ISEC,ZARRAY,KE,IN3D,IZPRNUM,
*                           IZNUM,RDDR,IDDR,XARRAY)
C
C AN INTERFACE FOR A SECTION PLOT THAT IS NOT COLOR FILLED.
C
C =====
C
INCLUDE 'data2.txt'
DIMENSION IDR(512),ZDEPTH(101),DEPTH(IMAX,IMAX),IFIELD(JE)
REAL*8 ZPREL(6,50),ZARRAY(JE,KE,IZNUM),RDDR(512),
*   XARRAY(IE,JE,50)
CHARACTER*40 SNAM
C
C =====
C
HEADER = 'MERIDIONAL CIRCULATION (SV)'
SECNAM = SNAM
DO 2160 K=1,1DDR(21)
      ZDEPTH(K) = RDDR(K+9)
2160 CONTINUE
ZLAYER(0) = 0.
DO 2170 K=1,KE
      ZLAYER(K) = ZDEPTH(K)
2170 CONTINUE
ZLAYER(KE+1) = 6000.
DO 1100 I=1,IE
      DO 1100 J=1,JE
         DEPTH(I,J) = XARRAY(I,J,1)
1100 CONTINUE
      CALL MCUTSEC(ZARRAY,IZNUM,DEPTH,DDR,RDDR,IFIELD)
C
C =====
C
      RETURN
END

      SUBROUTINE MCUTSEC(ZARRAY,IZNUM,DEPTH,DDR,RDDR,IFIELD)
C
C THIS SUBROUTINE TAKES THE DATA Puts IT INTO THE ZONT ARRAY FOR DEEP
C AND MAKES SURE THAT ADDITIONAL VARIABLES ARE FILLED WITH APPROPRIATE
C ITEMS FOR THE PLOT FOR THE SECTION PLOT THAT IS NOT COLOR FILLED
C

```

plodin.f

```

C =====
C INCLUDE 'data2.txt'
C DIMENSION ZARRAY(JE,KE,50),DEPTH(IMAX,JMAX),IDDR(512),RDDR(512),
C   SECTNAM(ISMAX),DDZ(IE,JE,KE),IFIELD(JE)
C REAL*8 ZARRAY,RDDR
C CHARACTER*40 SECTNAM
C =====
C READ DDZ FROM THE FILE DDZ
C
C THE FILE 'ddz' IS CREATED ON THE CARBON SIDE OF THE MODEL AND IS USED TO
C FIND THE BOTTOM ON THE SECTION MAP. DDZ IS USED BECAUSE IT GIVES A
C MORE PRECISE BOTTOM ON THE SECTION SIDE.
C
C OPEN(22,FILE='ddz',FORM='FORMATTED',STATUS='OLD')
C READ(22,'(5F15.5)') DDZ
C CLOSE(22)
C
C DELTA = 5.0
C DO 1089 J=1,JE
C DO 1089 K=1,KE
C   MH2(J,K) = .TRUE.
1089 CONTINUE
C XTOP = .88
C XBOT = .38
C YLFT = .15
C YRGT = .85
C SCHUM2 = 1.E-6*DELTA
C DO 1078 J=1,JE
C DO 1078 K=1,KE
C   ZARRAY(J,K,IZNUM) = ZARRAY(J,K,IZNUM) - SCHUM2
1078 CONTINUE
C
C OBTAIN THE WET DRY ARRAY USING THE BOTTOM ARRAY READ IN FROM THE DATA
C FILE.
C
C DO 1990 J=1,JE
C DO 1999 K=1,KE
C   IF (DDZ(INDISU(J),J,K).EQ.0.) THEN
C     KBOT = K
C     GOTO 2999
C   ENDIF
1999 CONTINUE
GOTO 1990
2999 CONTINUE
DO 1910 K=KBOT,KE
C   ZARRAY(J,K,IZNUM) = RMASK
1910 CONTINUE
1990 CONTINUE
C
C MASK THE BOTTOM OF THE SECTION AND FLIP THE ARRAY INDICES SO THAT
C NCAR WILL PROPERLY DISPLAY THE CONTENTS OF THE ARRAY.
C
C   JP = JE
C   DO 1010 J=1,JE
C     KP = KE
C     DO 1009 K=1,KE
C       ZDAT(J,K) = ZARRAY(JP,KP,IZNUM)
C       KP = KP - 1
1009 CONTINUE
C     JP = JP - 1
1010 CONTINUE
C
C INTERFACE THE NCAR GRAPHICS.
C
C CALL MCUT(XTOP,XBOT,YLFT,YRGT,ICR,DEPTH)
C
C
C RETURN
C END

```

readplodat.job

```
# a script to run readtest.f with subroutines in plodin.f and trapin.f
#
#
# THIS CODE IS HAMBURG DEPENDENT IN THE WAY THE FILE IS SET UP TO BE READ.
# THE DATA IS READ IN READTEST.F. THIS FILE READS FROM THE DATA FILE
# BEGINNING WITH HEADER ARRAYS IDR, RDR AND ZPREL. SEE THE HAMBURG
# LARGE SCALE GEOSTOPHIC OCEAN GENERAL CIRCULATION MODEL (CYCLE 1) TECHNICAL
# REPORT WHERE THE HEADERS ARE DEFINED OR GOTO THE READTEST CODE WHERE THE
# OPENING OF THE FILE PLODAT AND THE READING OF THIS FILE IS LOCATED.
#
#
ln -sf ~gsm/mpilogcm/WORKpostlsg/PLODAT PLODAT
ln -sf ..../postdir/gksatr.txt gksatr.txt
ln -sf ~gsm/mpilogcm/post/lsg/section_uw SECTION
ln -sf ..../postdir/litcon.txt litcon.txt
ln -sf ..../postdir/data.txt data.txt
ln -sf ..../postdir/data2.txt data2.txt
ln -sf ..../postdir/ddz ddz
#
#
make readplo
readplo
'mv' gmeta lsg.cgm
```

```

SUBROUTINE LSG
C
C THIS SUBROUTINE IS USED TO READ IN THE DATA FROM THE DATA FILE PLODAT
C AND IS MOSTLY TAKEN FROM PLOPRG.F ON THE LSG SIDE OF THE HAMBURG
C MODEL. THIS SUBROUTINE IS THE ONLY SUBROUTINE ON THE LSG SIDE
C THAT IS DOUBLE PRECISION. EVERYTHING AFTER THIS SUBROUTINE IS
C SINGLE PRECISION.
C
C HAMBURG DEPENDENT IN THE WAY THE FILE IS SET UP TO BE READ. THIS FILE READS
C FROM THE DATA FILE ALL THE HEADER FILES BEGINNING WITH IDDR AND RDDR. IT WILL
C THEN READ A DATA HEADER OF SIX ARRAY POSITIONS. THIS WILL BE USED TO CALCULATE
C WHICH DATA TYPE IS TO BE READ. SEARCH FOR *READ TYPE* TO SEE EXPLICIT
C STATEMENTS.
C
C =====
C IMPLICIT REAL*8(A-H,O-Z)
CHARACTER EXPID *7, SNAM *40, EXPDES *40
PARAMETER (IE= 72, JE= 72, KE= 11, ICM= 99, L=50,
#      EXPID= 'FWTE ', XACTOR= 0.254, KE1=KE+1, K=100,
#      IMAX=78, JMAX=78)
DIMENSION XARRAY(IE,JE,L), YARRAY(JE,JE,L), CARRAY(JE,KE),
#      IFIELD(JE), ZARRAY(JE,KE,L), FAC(ICM), ADD(ICM),
#      ZPRELX(6,L), ZPRELY(6,L), ZPRELT(6,L),
#      ZPREL(6,L), IDDR(512), RDDR(512), TARRAY(JE,L),
#      DDZ(IE,JE,KE)
COMMON /FACT/ FAC, ADD
LOGICAL NEWT
C
C =====
C
IOLDT=0
NUM = 0
EXPDES='
FAC( 5) = .500000E+00
ITAPE=88
IALNUM=22
OPEN(5,FILE='PLODAT',STATUS='OLD')
READ(5,'(29(10I8),918/,10(10(A8)/),11(10I8)/,318)
# ERR=3000,END=3000) IDDR
ICREDAT = IDDR(8)
READ(5,'(8E13.6)      ',ERR=3000,END=3000) RDDR
C
C INITIALIZE COUNTING VARIABLES SO THAT THE DATA CAN BE STORED IN
C ARRAYS FOR POSSIBLE FURTHER USE.
C
IZPRNUM = 0
IXPRNUM = 1
IYPRNUM = 1
IZZPRUM = 0
ITPRNUM = 0
C
READ(5,'(6E13.5)      ',ERR=2000,END=3000)
#      (ZPRELX(M,IXPRNUM),M=1,6)
READ(5,'(10E13.6)      ',ERR=2000,END=3000)
#      ((XARRAY(I,J,1),I=1,IE),J=1,JE)
READ(5,'(6E13.5)      ',ERR=2000,END=3000)
#      (2PREL(M,IYPRNUM),M=1,6)
READ(5,'(10E13.6)      ',ERR=2000,END=3000)
#      ((YARRAY(I,J,1),I=1,IE),J=1,JE)
NPICT=0
NEWT=.TRUE.
IXNUM = 1
IYNUM = 1
IZNUM = 0

```

```

ITNUM = 0
C
C BEGINING OF THE READ LOOP
C
1000 IF ((IXNUM.EQ.50).OR.(IYNUM.EQ.50).OR.(IZNUM.EQ.50).OR.
#      (ITNUM.EQ.50)) THEN
      WRITE(*,*)
      *      'ARRAY BOUNDS REACHED!!!!'
      WRITE(*,*)
      *      'INCREASE ARRAY COMPACITY BEFORE RUNNING AGAIN!!!!'
      STOP
ENDIF
IZPRNUM = IZPRNUM+1
C
C *READ TYPE* DATA HEADER
C
READ(5,'(6E13.5)      ',ERR=2000,END=3000)
#      (ZPREL(M,IZPRNUM),M=1,6)
NPICT=NPICT+1
IF(NPICT.GT.50.AND.NEWL) THEN
      write(*,*) 'gsm max pix 50 reached, shutdown'
      go to 2000
ENDIF
C
C *READ TYPE* MAP DATA
C
IF (NINT(ZPREL(2,IZPRNUM)).EQ.IE*JE) THEN
      IXNUM=IXNUM+1
      READ(5,'(10E13.6)      ',ERR=2000,END=3000)
#      ((XARRAY(I,J,IXNUM),I=1,IE),J=1,JE)
      IXPRNUM = IXPRNUM+1
      DO 10 N=1,5
      ZPRELX(N,IXPRNUM) = ZPREL(N,IZPRNUM)
CONTINUE
CALL ARPPIC(ZPREL,XARRAY,IZPRNUM,IXNUM,ICREDAT,RDDR,IDDR)
NEWL=.TRUE.
GO TO 1000
ENDIF
C
C *READ TYPE* MAP DATA WITH VELOCITY ARROWS OVERLAYED
C
IF (NINT(ZPREL(2,IZPRNUM)).EQ.2*IE*JE) THEN
      IXNUM = IXNUM+1
      IXPRNUM = IXPRNUM + 1
      DO 20 N=1,6
      ZPRELX(N,IXPRNUM) = ZPREL(N,IZPRNUM)
CONTINUE
READ(5,'(10E13.6)      ',ERR=2000,END=3000)
#      ((XARRAY(I,J,IXNUM),I=1,IE),J=1,JE)
      IXPRNUM=IXPRNUM+1
      IYNUM = IYNUM + 1
      READ(5,'(6E13.5)      ',ERR=2000,END=3000)
#      (ZPRELY(M,IYPRNUM),M=1,6)
      READ(5,'(10E13.6)      ',ERR=2000,END=3000)
#      ((YARRAY(I,J,IYNUM),I=1,IE),J=1,JE)
      CALL ARPPIC(ZPREL,XARRAY,YARRAY,IZPRNUM,IXNUM,IYPRNUM,RDDR,IDDR)
      NEWL=.TRUE.
      GO TO 1000
ENDIF
C
C *READ TYPE* SECTION DATA
C
IF (NINT(ZPREL(2,IZPRNUM)).EQ.JE*PF) THEN
      READ(5,'(12,A)      ',ERR=2000,END=3000) IFLD
      READ(5,'(2513)      ',ERR=2000,END=3000) IFLD
      IZNUM = IZNUM + 1
      IZZPRUM = IZZPRUM + 1

```

readtest.f

```
DO 25 N=1,6
      ZPRELZ(N,IZPRNUM) = ZPREL(N,IZPRNUM)
25    CONTINUE
      READ(5,'(10E13.6)      ',ERR=2000,END=3000)
      *     ((ZARRAY(J,M,I2NUM),J=1,JE),M=1,KE)
      IF (NINT(ZPREL(5,IZPRNUM)) .EQ. 0) THEN
          CALL CUTPIC(ZPREL,IFIELD,ZARRAY,SNAME,IZPRNUM,I2NUM,XARRAY,
      *             IDDR,RDDR)
      ELSE
          CALL MCUTO4(ZPREL,IFIELD,SNAME,ISEC,ZARRAY,KE1,49,IZPRNUM,
      *             I2NUM,RDDR,IDDR,XARRAY)
      ENDIF
      NEWT=.TRUE.
      GO TO 1000
      ENDIF

C
C *READ TYPE*   LINE GRAPH DATA
C
      IF (NINT(ZPREL(2,IZPRNUM)).EQ.JE) THEN
          IF (NUM.EQ.4) THEN
              NUM = 1
          ELSE
              NUM = NUM + 1
          ENDIF
          ITPRNUM = ITPRNUM + 1
          DO 30 N = 1,6
              ZPRELT(N,ITPRNUM) = ZPREL(N,IZPRNUM)
30    CONTINUE
          ITNUM = ITNUM + 1
          IF (NUM.EQ.1) ISTART = ITNUM
          READ(5,'(10E13.6)      ',ERR=2000,END=3000)
          *     (TARRAY(J,ITNUM),J=1,JE)
          IF (IOLDT.NE.NINT(ZPREL(5,IZPRNUM))) THEN
              NEWT=.TRUE.
          ELSE
              NPICT=NPICT-1
          END IF
          IF (NUM.EQ.4)
              CALL TRAPIC(TARRAY,ZPREL,49,NEWT,IZPRNUM,ITNUM,ISTART,DDR)
          IOLDT=NINT(ZPREL(5,IZPRNUM))
          GO TO 1000
      ENDIF
      GO TO 1000
2000 WRITE(*,*) 'ERROR IN READING'
      STOP
C
C =====
C
3000 RETURN
      END
```

runlsg.f

```
PROGRAM RUNLSG
C
C THIS PROGRAM IS DESIGNED TO TAKE AN ARRAY OF DATA AND THEN SEND IT
C THROUGH A SERIES OF PROCESSES IN WHICH TO PREPARE IT TO BE
C PLOTTED BY NCAR. -- Emily Stevens
C
C
C Declare required data arrays and workspace arrays.
C
C      INCLUDE 'data.txt'
C
C      OPEN GKS
C
C          CALL GOPKS(6, IDUM)
C          CALL GOPWK(1, 2, 1)
C          CALL GACWK(1)
C
C
C Retrieve the data from the subroutine CHOOSEC - which will read
C from a file along with all other necessary items and then it will
C call all needed subroutines to finish the plotting of all the maps.
C
C          CALL LSG
C          CALL GDAWK(1)
C          CALL GCLWK(1)
C          CALL GCLKS
C
C
END
```

trapic.txt

```
PARAMETER(JE=72, NUMGRAPH=4, RLFT=.01, RRGTR=.99, TOP=.85,  
*           BOT=.35)  
COMMON/GRAPHERS/ ARRAY(JE,NUMGRAPH), HEADER, RMIN, RMAX,  
*           ICOLOR(NUMGRAPH), ARRD1(JE)  
CHARACTER*40 HEADER
```

```

SUBROUTINE TRAPIC(TARRAY,ZPREL,JN30,NEWT,I2PRNUM,ITNUM,ISTART,
*                   IDDR)
C
C THIS IS THE SUBROUTINE THAT WILL INTERFACE THE NCAR GRAPHICS FOR THE
C LINE PLOT GIVEN IN THE LSG SIDE OF THE HAMBURG MODEL.
C
C =====
C
C INCLUDE 'trapic.txt'
DIMENSION ZPREL(6,50),TARRAY(JE,50),PY(4),IDDR(512)
REAL*8 ZPREL,TARRAY
C
LOGICAL NEWT
C
C =====
C
CALL INITRAP
C
SELECT THE APPROPRIATE TRACER TO BE USED.
C
GO TO (999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,19, 20,
#   21, 22,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999,
#   999,999,999,999,999,999,999,999,999,999)
#   NINT(ABS(ZPREL(3,I2PRNUM)))
C
RETURN
C
C OBTAIN THE HEADER AND THE FOUR ARRAYS THAT COMPOSE ONE GRAPH INTO THE
C VARIABLE ARRAY.
C
19 CONTINUE
HEADER = 'HEAT TRANSPORT [PW]'
IST = I START
DO 7760 I=1,NUMGRAPH
  DO 7770 J=1,JE
    ARRAY(J,I) = TARRAY(J,IST)*0.1E-8
7770 CONTINUE
IST = IST + 1
7760 CONTINUE
CALL TRAGRP(IDDR)
RETURN
C
20 CONTINUE
HEADER = 'SALT TRANSPORT [KT/S]'
IST = ISTART
DO 7761 I=1,NUMGRAPH
  DO 7771 J=1,JE
    ARRAY(J,I) = TARRAY(J,IST) * 1.0E-6
7771 CONTINUE
IST = IST + 1
7761 CONTINUE
CALL TRAGRP(IDDR)
RETURN
C
21 CONTINUE
HEADER = 'INTEGRATED FRESH WATER FLUX [SV]'
IST = ISTART

```

trapin.f

```

DO 7762 I=1,NUMGRAPH
  DO 7772 J=1,JE
    ARRAY(J,I) = TARRAY(J,IST) * (-1) * 1.0E-6
7772 CONTINUE
IST = IST + 1
7762 CONTINUE
CALL TRAGRP(IDDR)
RETURN
C
22 CONTINUE
HEADER = 'INTEGRATED HEAT FLUX [PW]'
IST = ISTART
DO 7763 I = 1,NUMGRAPH
  DO 7773 J=1,JE
    ARRAY(J,I) = TARRAY(J,IST) * (-1.E 15)
7773 CONTINUE
IST = IST + 1
7763 CONTINUE
CALL TRAGRP(IDDR)
RETURN
C
C -- TURN OFF CLIPPING --
C
999 WRITE(*,*) 'THIS IS NOT A VALID LINE GRAPH CHOICE'
RETURN
END
C
SUBROUTINE INITRAP
C
C THIS SUBROUTINE IS TO INITIALIZE THE PLOT PACKAGE AUTOGRAPH. THE
C EZY OPTION WILL BE USED IN THE PART.
C
INCLUDE 'trapic.txt'
C
DIMENSION IASF(13)
DATA IASF / 13*1 /
C
C -- SET THE WINDOW UP --
C
CALL AGSETF('GRAPH/LEFT.',RLFT)
CALL AGSETF('GRAPH/RIGHT.',RRGT)
CALL AGSETF('GRAPH/TOP.',TOP)
CALL AGSETF('GRAPH/BOTTOM.',BOT)
C
C -- TURN OFF CLIPPING --
C
CALL GSCLIP (0)
C
C Set all aspect source flags to "individual"
C
CALL GSASE (IASF)
C
C Force solid fill.
C
CALL GSFAIS (1)
C
C Define color indices.
C
CALL DECIPS
C
C SET POLYLINE COLOR TO CYAN, SET LINE WIDTH TO 1 AND LINE TYPE TO ONE
C
CALL GSPLCI(1)
CALL GSLW(1)
CALL GSLWSC(1.)

```

```

      RETURN
      END

      SUBROUTINE TRAGR(P>IDDR)
C CONTINUE TO SET UP THE INITIALIZATION FOR THE PLOT
C
C INCLUDE 'trapic.txt'
C DIMENSION IDDR(512)
C
C CALL GSPLCI(1)
C CALL TRAMMX
C
C SET THE TYPE OF BACKGROUND TO BE USED TO A PERIMETER FOR THE FOURTH
C ARRAY AND TO NO BACKGROUND FOR THE FIRST THREE. MAKE SURE THAT THE
C NUMERIC LABELS AND TICK MARKS ARE SUPPRESSED ON THE BOTTOM OF THE
C GRAPH BECAUSE THE LONGITUDES WILL BE PUT IN LATER.
C
      DO 5001 NUM = 1,NUMGRAPH
      IF (NUM.EQ.4) THEN
        BACK = -1.
        CALL AGSETF ('BOTTOM/CONTROL.',-1.)
        CALL AGSETF ('LEFT/CONTROL.',4.)
        CALL AGSETF ('LEFT/TYPE.',3.)
        CALL AGSETF ('LEFT/MAJOR/TYPE.',1.)
        CALL AGSETF ('AXIS/LEFT/TICKS/MAJOR/SPACING/TYPE.',1.)
        CALL AGSETF ('BOTTOM/TYPE.',0.)
        CALL AGSETF ('AXIS/BOTTOM/TICKS/MAJOR/SPACING/TYPE.',0.)
      ELSE
        BACK = 4.
        CALL AGSETF ('BOTTOM/CONTROL.',0.)
        CALL AGSETF ('LEFT/CONTROL.',0.)
      ENDIF
C SET THE MINIMUM AND MAXIMUM NUMBER ALLOWED FOR THE X AND THE Y AXIS.
C
      RJE = JE
      CALL AGSETF ('Y/MINIMUM.',RMIN)
      CALL AGSETF ('Y/MAXIMUM.',RMAX)
      CALL AGSETF ('X/MINIMUM.',0.)
      CALL AGSETF ('X/MAXIMUM.',RJE)
      CALL AGSETF ('BACKGROUND.',BACK)
C PUT THE CURRENT LINE DATA (OF THE FOUR) INTO THE VARIABLE TO BE
C PLOTTED.
C
      DO 5002 J=1,JE
        ARRD1(J) = ARRAY(J,NUM)
5002  CONTINUE
      CALL GRAPHTR(NUM, IDDR)
5001  CONTINUE
      CALL GCLRWK(1,0)
      RETURN
      END

      SUBROUTINE TRAMMX
C OBTAIN THE MINIMUM AND THE MAXIMUM NUMBER OF ALL FOUR DATA LINES SO
C THAT ALL FOUR ARE PLOTTED TO THE SAME SCALE.
C
C INCLUDE 'trapic.txt'
C
      RMIN = 1.E+34
      RMAX = -1.E+34

```

```

      DO 3999 I=1,NUMGRAPH
      DO 3999 J=1,JE
        IF (ARRAY(J,I).LT.RMIN) THEN
          RMIN = ARRAY(J,I)
        ENDIF
        IF (ARRAY(J,I).GT.RMAX) THEN
          RMAX = ARRAY(J,I)
        ENDIF
3999  CONTINUE
      RETURN
      END

      SUBROUTINE GRAPHTR(NUM, IDDR)
C CALL THE GRAPHING PACKAGE AND PUT ON THE EXTRA INFORMATIONAL LABELS
C USING THE PLOT PACKAGE PLOTCHAR.
C
C INCLUDE 'trapic.txt'
C DIMENSION PY(4),IDDR(512),BOTTET(7)
C CHARACTER*3 BOTTET
C CHARACTER*1 BLANK
C CHARACTER*8 CD
C CHARACTER*9 TEXT6
C CHARACTER*6 CTEMP
C
C BLANK = ''
C ICOLOR(1) = 11
C ICOLOR(2) = 14
C ICOLOR(3) = 9
C ICOLOR(4) = 5
C
C ICREDIT = IDDR(R)
C WRITE(CTEMP,(16)) ICREDIT
C WRITE(TEXT6,'(A1,A2,A1,A2,A1,A2)') ' ',(CTEMP(3:4),//,CTEMP(5:6),
C                                         //,CTEMP(1:2))
C
      CALL DATE (CD)
      CALL GSPLCI (ICOLOR(NUM))
      CALL GSIM (1)
      CALL GSLWSC (1)
      CALL AGSETC ('LABEL/NAME.', 'L')
      CALL AGSETI ('LINE/NUMBER.', 100)
      CALL AGSETC ('LINE/TEXT.', BLANK)
      CALL AGSETC ('LABEL/NAME.', 'R')
      CALL AGSETI ('LINE/NUMBER.', 100)
      CALL AGSETC ('LINE/TEXT.', BLANK)
      CALL AGSETF ('FRAME.',2.)
      CALL AGSETF ('X/ORDER.',1.)
C
C CALL THE PLOT PACKAGE AUTOGRAPH WITH EZY.
C
      CALL EZY (ARRD1,JE,BLANK)
      CALL GSPLCI (1)
      CALL LETTERING(CD,TEXT6)
C
      RETURN
      END

      SUBROUTINE LETTERING(CD,TEXT6)
C PLOT THE INFORMATIONAL LABELS USING THE PLOT PACKAGE PLOTCHAR
C
C INCLUDE 'trapic.txt'
C DIMENSION BOTTET(7)
C CHARACTER*3 BOTTET

```

trapin.f

```

CHARACTER*1 DASH
CHARACTER*8 CD
CHARACTER*9 TEXT6

C
DASH = '|'
BOTLET(1) = '90S'
BOTLET(2) = '60S'
BOTLET(3) = '30S'
BOTLET(4) = '0'
BOTLET(5) = '30N'
BOTLET(6) = '60N'
BOTLET(7) = '90N'

C
C WRITE THE DATES AND LABELS TO THE SCREEN, MAKING THE COLORS INDICATE
C THE COLORS ON THE GRAPH.
C
C SET AND GETSET ARE SPPS CALLS AND MAY NEED TO BE CHANGED AT A LATER DATE.
C
CALL GETSET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)
CALL SET (0.,1.,0.,1.,0.,1.,0.,1.,1.)
CALL PGGETI ('QU - QUALITY FLAG',IQUA)
CALL PCSETI ('QU - QUALITY FLAG',0)
CALL PCSETI ('TE - TEXT EXTENT COMPUTATION FLAG',0)
CALL PLCHHQ (.990,.890,HEADER,.013,0.,+1.5)
CALL PLCHHQ (.880,.350,'GLOBAL' -- ,.011,0.,+1.5)
CALL GSPLCI (11)
CALL PLCHHQ (.940,.350,'YELLOW',.011,0.,+1.5)
CALL GSPLCI (1)
CALL PLCHHQ (.890,.325,'ATLANTIC' -- ,.011,0.,+1.5)
CALL PLCHHQ (.330,.325,'AVERAGE',.011,0.,+1.5)
CALL GSPLCI (14)
CALL PLCHHQ (.890,.325,'RED',.011,0.,+1.5)
CALL GSPLCI (1)
CALL PLCHHQ (.890,.300,'PACIFIC' -- ,.011,0.,+1.5)
CALL GSPLCI (9)
CALL PLCHHQ (.930,.300,'GREEN',.011,0.,+1.5)
CALL GSPLCI (1)
CALL PLCHHQ (.890,.275,'INDIC' -- ,.011,0.,+1.5)
CALL GSPLCI (5)
CALL PLCHHQ (.905,.275,'BLUE',.011,0.,+1.5)
CALL GSPLCI (1)
XCOR = RLFT*.1435
XCORD = RLFT*.165

C
C PLACE THE LONGITUDE LINE NUMBERS ON THE GRAPH.
C
DO 59 L=1,7
  CALL GSPLCI(5)
  CALL PLCHHQ (XCOR,BOT*.075,DASH,.009,0.,+1.5)
  CALL GSPLCI(1)
  CALL PLCHHQ (XCORD,BOT*.045,BOTLET(L),.009,0.,+1.5)
  XCOR = XCOR + .131
  XCORD = XCORD + .131
59 CONTINUE
C
CALL PLCHHQ (.455,.200,C'D.,.011,0.,+1.5)
CALL PLCHHQ (.300,.200,'PLOTTED',.011,0.,+1.5)
CALL PLCHHQ (.855,.200,'CREATED',.011,0.,+1.5)
CALL PLCHHQ (.995,.200,TEXT6,.011,0.,+1.5)
CALL PCSETI ('QU - QUALITY FLAG',IQUA)
CALL SET (XVPL,XVPR,YVPB,YVPT,XWDL,XWDR,YWDB,YWDT,LNLG)

C
RETURN
END

```

INTERNAL DISTRIBUTION

- | | | | |
|-------|----------------|--------|-------------------------------|
| 1. | J. R. Drake | 14. | W. M. Post |
| 2. | W. R. Emanuel | 15-17. | C. H. Shappert |
| 3. | R. L. Ferguson | 18. | G. E. Whitesides |
| 4. | R. H. Fowler | 19. | Central Research Library |
| 5. | H. R. Hicks | 20. | ORNL Y-12 Research Library |
| 6. | A. W. King | | Document Reference Section |
| 7. | G. Marland | 21. | Laboratory Records Department |
| 8-12. | G. S. McNeilly | 22. | Laboratory Records, ORNL (RC) |
| 13. | T. H. Peng | 23. | ORNL Patent Office |

EXTERNAL DISTRIBUTION

- | | |
|--------|---|
| 24 | Office of the Deputy Assistant Manager for Energy, Research and Development, Department of Energy, Oak Ridge Operations (DOE-ORO), P. O. Box 2008, Oak Ridge, TN 37831-6269 |
| 25-26. | Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831 |
| 27. | Dr. Martin Dickey, Illinois College, 1101 W. College Avenue, Jacksonville, IL 62650 |
| 28. | Professor Michael Liljegren, Illinois College, 1101 W. College Avenue, Jacksonville, IL 62650 |
| 29-31. | Emily Stevens, 1920 Creighton Road, Springfield, IL 62703 |

