FILE

Institute of
Oceanographic Sciences
Deacon Laboratory

INTERNAL DOCUMENT No. 328

Constructing a 2° x 1° resolution model of the Southern Ocean using the GFDL Modular Ocean Model (MOM)

A C Coward

1994

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES
DEACON LABORATORY

INTERNAL DOCUMENT No. 328

Constructing a 2° x 1° resolution model of the Southern Ocean using the GFDL Modular Ocean Model (MOM)

A C Coward

1994

Wormley
Godalming
Surrey GU8 5UB UK
Tel +44-(0)428 684141
Telex 858833 OCEANS G
Telefax +44-(0)428 683066

DOCUMENT DATA SHEET

AUTHOR
COWARD, A C

PUBLICATION DATE 1994

TITLE

Constructing a 2° x 1° resolution model of the Southern Ocean using the GFDL Modular Ocean Model (MOM).

REFERENCE

Institute of Oceanographic Sciences Deacon Laboratory, Internal Document, No. 328, 84pp. (Unpublished manuscript)

ABSTRACT

The GFDL Modular Ocean Model (Pacanowski et al, 1990) has rapidly become established as both a teaching and a research tool. This document details the work carried out to construct and operate a physical model of the Southern Ocean using the published GFDL code. The work is intended as the first stage of the development of a combined, physical and biological model of the Southern Ocean but is also a useful illustration of the techniques and methods that modellers are advised to adopt when using the GFDL MOM code.

KEYWORDS

NUMERICAL MODELLING
PROJECT - FRAM

ISSUING ORGANISATION

Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB. UK.

Director: Colin Summerhayes DSc

Telephone Wormley (0428) 684141
Telex 858833 OCEANS G.
Facsimile (0428) 683066

Copies of this report are available from: **The Library,** *PRICE* *£0.00*

CONTENTS

1. INTRODUCTION

This document details the work carried out to construct and operate a physical model of the Southern Ocean based on the GFDL Modular Ocean Model (MOM). This work is intended as a precursor to the development of a biological model of the Southern Ocean but is also useful for illustrating the use of the MOM code and describing some of the techniques that modellers will need to become familiar with in order to make effective use of the code.

The actual model described here has a 2° longitude by 1° latitude resolution with 25 vertical levels. The domain coincides with that modelled by both the coarse- and fine-resolution Antarctic models developed by the FRAM team (de Cuevas, 1992; de Cuevas 1993). However, since most of the biological processes occur exclusively in the upper ocean, the vertical resolution is biased more towards the surface than was the case with the Fine Resolution Antarctic Model. The vertical distribution is the same as that used by Sarmiento (1986).

Model updates for the biological model have been obtained from the United States. These updates represent the code run successfully in the North Atlantic model of Fasham *et al* (1993). To prepare a physical model capable of hosting the biological model the following developments were necessary:

- Setting the model dimensions and grid distribution.
- *Constructing the model topography*: Bottom topography over the model domain can be obtained from the dbdb5 dataset. However care must be taken to filter any possible sources of topographic instabilities.
- Including open boundary conditions (Stevens, 1991)
- Applying annual mean winds read in from external data files.
- Acquiring Levitus temperature and salinity data (annual means) on the model grid. These are required for surface forcing, possible initial conditions and to allow for the possibility of running in a robust diagnostic mode

Separate sections of this report are devoted to each of these developments.

2. RECONFIGURING THE MODEL DOMAIN

The MOM code is supplied pre-configured as a 4° x 3° global model. Input fields such as topography, surface forcing fields and surface restoring fields are calculated from a limited number of data points held in the relevant routines. For example, the routine: bcest.F linearly interpolates global, zonal averages of sea-surface temperature, salinity and the wind-stress components onto the current MOM grid. The internally held data consists of 40 values for each variable giving a latitudinal resolution of 4.5 degrees. Similarly, the routine: topog.F will produce an idealised world topography which will map into any domain.

Hence, in order to produce a model of any region of the globe it is only necessary to set certain 'grid parameters' and then compile. The exact content of the model is determined by the choice of preprocessor directives which are set at compile-time (see MOM READ_ME file). Obviously, the resulting descriptions of bottom topography and the surface boundary conditions are too crude for most modelling efforts. However they do fulfil a useful role in the initial development of the model.

Temporarily accepting these 'internal' descriptions of topography, surface conditions and initial conditions (again zonal averages at 4.5° resolution) a model of the Southern Ocean was set up by making the following minimal changes:

Variable	Description	Old value	New value	Location
imt	no. of grid pts longitudinally	92	182	param.h
jmt	no. of grid pts latitudinally	60	56	param.h
km	no. of vertical levels	15	25	param.h
Iseg	max no. of longitudinal stream-fn. segments	5	6	param.h
nisle	number of islands	2	1	param.h
stlat	starting latitude (degrees)	-90.	-79.0	blkdta.F
stlon	starting longitude (degrees)	-4.	0.0	blkdta.F
xmax	max grid box width (degrees)	4.	2.0	blkdta.F
xmin	min grid box width (degrees)	4.	2.0	blkdta.F
xwid	longitudinal width (degrees)	368.	364.0	blkdta.F
ymax	max grid box ht. (degrees)	3.	1.0	blkdta.F
ymin	min grid box ht. (degrees)	3.	1.0	blkdta.F
ywid	latitudinal height (degrees)	180.	56.0	blkdta.F
rests	surf. restore time scale (day)	50.	360.0	blkdta.F

Changing the vertical resolution of the model (km) introduces an extra complication. Namely the need to re-calculate the nine coefficients of the third order approximation to the

equation of state for each level. Fortunately, one of the modules supplied with the MOM code is a stand-alone program designed to perform the required calculation and create the include file, dncoef.h, which holds the resulting data statements. The process is as follows:

1. Set the new level thicknesses in the include file: thick.h
2. Edit the module: denscoef.F, comment out the subroutine declaration and reinstate the program statement.
3. Compile and run denscoef.F
4. A new version of dncoef.h will be created and will overwrite the existing version. By default denscoef.F (actually named eqstat by the program statement) uses the UNESCO equation of state. Older equations of state can be selected by using the appropriate preprocessor directives.

Finally, run control parameters can be set in the control file: ocean.in (the name of which is actually set in ocean.F). These parameters include:

Variable	Description
init	*Logical flag set true if run should start from initial conditions*
days	Number of days for integration (can include fractions of days)
dgnstc	Number of days between diagnostic dumps
tsi	Number of days between output of standard run information (i.e. total k.e., dtemp, dsalt etc.)
nmix	Number of timesteps between mixing steps
eb	Logical flag, true if Euler backwards step is used for mixing
restrt	Logical flag, true if a restart dataset is produced at the end of run.

The eddy, tsteps and params namelist entries in the control file conform to the standard Cox setup. *The namelist entry, &iland, contains co-ordinate information about the islands. This has* changed from the original Cox setup in as much as it is no longer necessary to describe a surrounding box for each island. The MOM module iperim.F calculates island perimeters from a 'seed' point. It is, therefore, only necessary to supply a single co-ordinate pair (lat,lon) which points to an arbitrary point within each island (nisle in total).

3. CONSTRUCTING THE MODEL TOPOGRAPHY

Having set the model domain, one of the first tasks in improving the model is to apply a more realistic bottom topography. The best description of ocean topography currently available to

us is the digital bathymetric 5 minute by 5 minute data (dbdb5) supplied by the Naval Ocean Research and Development Office (NORDA) and the US Naval Oceanographic Office (USNOO). A fair representation of bottom topography can be obtained at coarser resolutions by calculating the median of the dbdb5 data in each grid cell. The dbdb5 data is supplied on two 9-track tapes. Each tape covers one hemisphere and each hemisphere is itself divided into sixteen 45° x 45° areas.

The areas are contained within a separate file and each file is subdivided into eighty-one 5° x 5° blocks. Each block has its own header and contains 61 x 61 values. The extra row and column (i.e. 61 instead of the expected 60) is an overlap with the blocks' eastern and northern neighbours.

Reading the dbdb5 data and obtaining median data on a 1/4° x1/4° grid was the subject of an internal IOS report and computer program produced by Nick Plummer (1991). Obtaining data f o r a 2° x 1° grid required a major reworking of the original program because every fifth 2° x 1° grid cell straddles two of the original dbdb5 files — an occurrence not allowed for in the original program. The solution is presented in appendix D (db2med.f). This program was constructed specifically to obtain data for the current application. It may, however, serve as a guide for anyone else intending to make use of the dbdb5 data.

Db2med.f will create a file of median data covering the entire hemisphere. It was discovered that the easiest method of working was to run db2med on the IBM (where there is easy access to the tape drives and large temporary disks) and then to transfer the output file (median data a) to a workstation for post-processing. Post-processing consists of:

1. Reading the 'median data' file, selecting the area covered by the intended model and writing out the data as a single two-dimensional field.
2. Taking the two-dimensional topography field and constructing the model topography array (i.e. 'snapping' the topography to the nearest model level).

The first of these steps is performed by the program med2cram.f (Appendix D (ii)). This program produces the 'real' topography array in two forms:

(i) A full accuracy unformatted dump (rawcram.dbdb5) which is used for stage 2.
(ii) An ascout cards file (topog.dbdb5) which can be viewed using the FRAM graphics programs.

(Note: Preprocessor directives are used to include the ascout0 and header subroutines. Therefore, med2cram.f (in common with all subsequent programs) will need to be preprocessed.

The commands:

```
cc -P med2cram.F
mv med2cram.i med2cram.f
f77 -o med2cram med2cram.f
```

should work in all UNIX environments).

The second stage is performed by the program makekm.F (Appendix D(iii)). This code reads the 'raw' median data and optionally applies zero, one or two smoothing passes before converting the depths to the nearest model level.

Isolated bays which will be unaffected by advection are then removed and there is also the option of removing any isolated land points. The model depths are calculated from the level thicknesses held in thick.h, so a change in the vertical distribution will be automatically picked up by makekm.F at compilation (n.b. makekm.F requires preprocessing as above). The three main model parameters, imt,jmt and km, are set in a parameter statement at the top of makekm.F. A change in these parameters could also be accounted for automatically by using the include file param.h. However, in one respect the current program is specific to the model domain and resolution of the Southern Ocean model. Namely the north and south islands of New Zealand are joined by a section of fixed code. The inclusion of the fixed parameters is intended as a reminder of this fact.

Output is again to two files: one unformatted file (sardepths21) and one ascout cards file for viewing (topsar21.cards). The MOM code contains checks for possible causes of Killworth-type topographic instabilities and any such occurrences will be flagged at runtime (Killworth, 1987). Therefore the run journal should be checked for any warnings after changing the topography.

Having created a new model topography it is relatively easy to adapt the MOM code to use it. The simplest method is to exchange the call to topog in ocn1st.F with code to read in the kmt field from the depths file. ocn1st.F is the routine which is called if the logical flag, init, is true (i.e. the routine which sets up all initial values). topog.F is the routine which will set up an idealised topography and is therefore no longer required. Makekm.F automatically applies cyclic boundary conditions on the kmt field. Code will have to be included to override this if it is not part of the intended configuration. The code changes necessary to read in the kmt field created by makekm.F have been incorporated into ocn1st.F (SCCS version number 1.2) and can be activated by preprocessing the routine with the -Dmytopog commandline option (see appendix A).

The topography used for the Southern Ocean model was produced using makekm with one smoothing pass and retaining isolated land points. The effect of makekm can be seen by comparing figures 1 and 2. Figure 1 is the 'raw' median data as produced by med2cram.F. Figure 2 is the model topography created by makekm.F

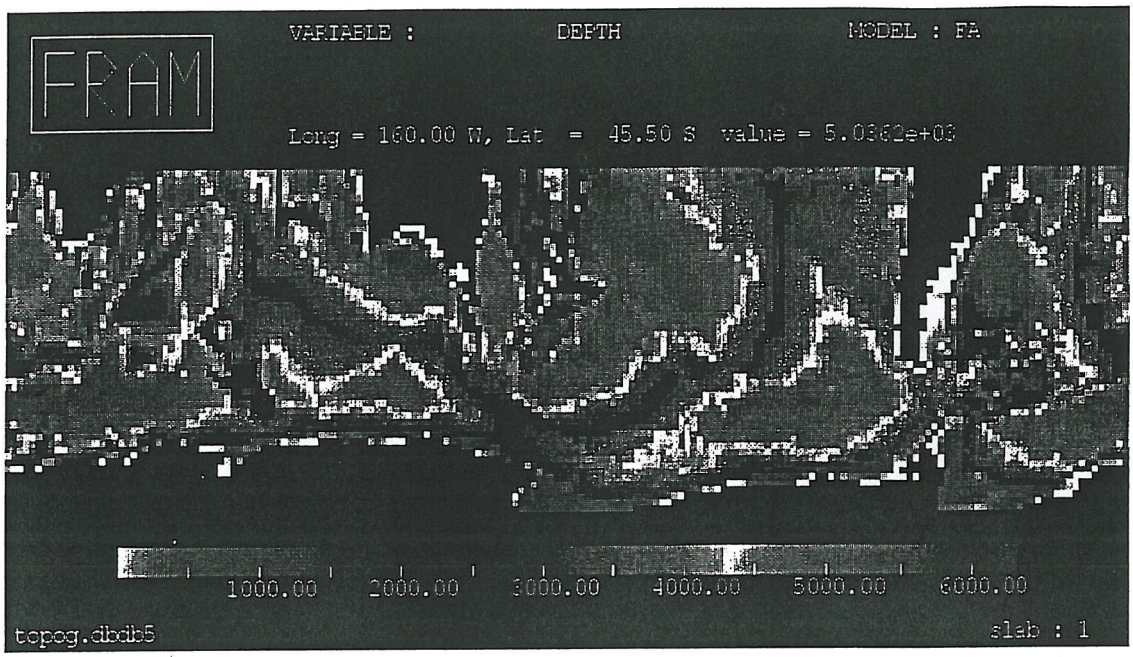

Figure 1 : The 'raw' median data as produced by med2cram.F.

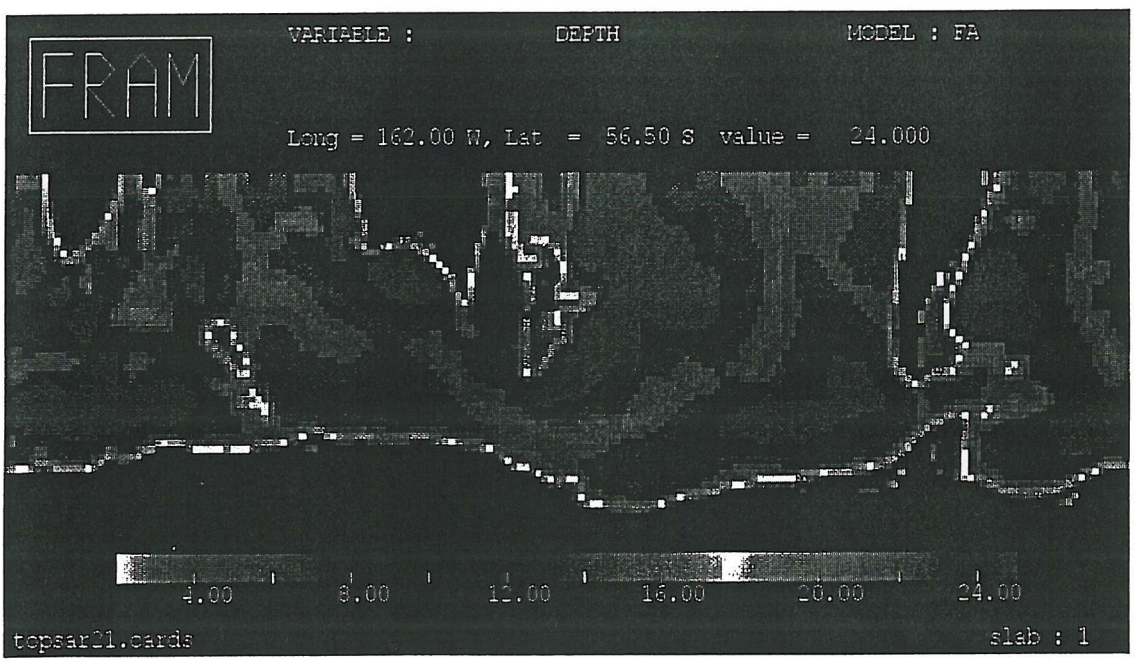

Figure 2 : The model topography created by makekm.F.

4. THE OPEN BOUNDARY CONDITION

One of the major features of the FRAM model (not available in the released version of the MOM code) are the open boundary conditions on the northern boundary of the FRAM region. Stevens (1990, 1991) gives details of the theory behind the open boundary condition. In practise it was possible to lift the relevant code from the Cox-based FRAM code and adapt it to the MOM code. Most changes are reasonably apparent. For example, the arrays T,TA,TB,TP,TM etc. are absorbed into the elements of the 5-D array t(i,k,jptr,time_ptr,n_tracer). Some care is required, however, because there are some subtle re-definitions of familiar variables. For example, the reciprocal of twice the vertical separation between centres of levels is defined in the Cox code as:

```
DZZ2R(K) = 1.0/ (DZ(K-1) + DZ(K))
```

whereas in the MOM code what, at first, appears to be the same variable is defined as:

```
dzw2r(k) = 1.0/ (dzt(k) + dzt(k+1))
```

So, unless extreme care is exercised, the task of converting Cox updates for use in the MOM code can introduce some well-hidden bugs.

The changes carried out to include the open boundary condition in the MOM code are a good illustration of how such enhancements should be implemented. All the additional code is placed within `#ifdef openbc` and `#endif` delimiters, and any changes to existing code do not overwrite but are added as `#ifdef` *`openbc new_code`* `#else` *`old_code`* `#endif` constructions.

For example, the reference array for the tracers on the northern boundary (which is used when the direction of flow is into the domain) is stored in the kontrl file. The kontrl file, therefore, has to be increased in size if the open boundary condition is in use. The following code changes in ocean.F achieve this:

```
call ostart (kontrl, 2, 2, 1)
```

becomes:

```
#ifdef openbc
 nkntrl=2+imt*km*nt
 call ostart (kontrl, nkntrl, nkntrl, 1)
#else
 call ostart (kontrl, 2, 2, 1)
#endif
```

Using this methodology, the open boundary condition was inserted into the MOM code. The original model, with a closed northern boundary, can still be created at any time by preprocessing the code without the -Dopenbc flag (see appendix A).

There are several points to make about the open boundary code as currently implemented. Firstly, the code works only with the 'rigidlid' model formulation and is currently coded only into the 'hypergrid' external mode solver. This solver is the same checker-board relaxation method that was used successfully for the FRAM model. Adding the open boundary code to the conjugate-gradient techniques (congr5pt.F or congr9pt.F) should be possible but this has not yet been investigated. Secondly, the relaxation timescale used to restore values towards Levitus values on the boundary, when flow is into the domain (rtscale), is set within step.F. Currently it is set to the order of 10 model days. The final (and most unsatisfactory) point concerns the imposing of a western boundary current in each basin. In the FRAM model a 'ball-park' width of 230km was assumed for each western boundary current. Unfortunately, this width is less than 2 grid points wide with a resolution of 2° x 1°. The present code will therefore allow a minimum of 5 grid points in which to 'ramp-down' the stream function values at each western boundary. This fix works well numerically, but obviously imposes an unrealistically wide western boundary current at the northern extent of each basin.

Code changes and additions required to implement the open boundary condition can be found in the following modules:

File	SCCS version number
emode.h	1.2
param.h	1.5
temp.h	1.2
blkdta.F	1.3
checks.F	1.3
clinic.F	1.2
hyper.F	1.2
ocean.F	1.7
ocn1st.F	1.3
odam.F	1.2
setvbc.F	1.2
step.F	1.2
windwt.F	1.3

and all subsequent versions of these modules. As previously stated, in order to activate the open boundary condition use the -Dopenbc preprocessor commandline option.

5. APPLYING ANNUAL MEAN WINDS

Boundary conditions at the surface and bottom of the ocean are calculated by the routine setvbc.F. In the basic model this routine calls the interpolation routine, bcest (see section 2). The zonal average values produced by this routine are applied across the entire latitude band. The resulting descriptions of SST, surface salinity and wind stress are clearly inadequate.

One of the first tasks to improve the model is to apply a more realistic wind stress. There are several choices of climatological wind datasets. The set employed by FRAM and hence the most readily available is that compiled by Hellerman and Rosenstein (1983). This dataset gives annual mean values for the horizontal wind stress components over the world ocean at a resolution of 2° x 2°. Data are also available for monthly averages at the same resolution. Allowing seasonal variation in the wind stress will be a necessary enhancement for the biological model. However, as a first step, the code changes required to read and apply an annual mean wind have been implemented.

Because the resolution of the Hellerman data nearly matches the current model resolution, interpolation is only required in the meridional direction. For this, standard linear interpolation has been used. A simple-minded approach has been applied at this stage and no generality to different model resolutions or domains should be assumed. The data is read from two files (data/windx and data/windy) which each hold the appropriate wind stress component data over the model region (i.e. 180 x 28 values). These values are read in by the routine anlwind.F and stored in a common block defined in anlwind.h. Code changes to setvbc.F ensure that these values are interpolated (if necessary) and assigned to the surface momentum flux array (smf).

Code changes and additions required to implement the annual mean winds can be found in the following modules:

File	SCCS version number
anlwind.h	1.1
hyper.F	1.3
ocean.F	1.8
setvbc.F	1.3
step.F	1.3
windwt.F	1.4

and all subsequent versions of these modules. In order to activate the annual mean winds use the -Dannwind preprocessor commandline option. It will also be necessary to ensure that the routine anlwind.F is included with the main modules.

6. ACQUIRING LEVITUS DATA ON THE MODEL GRID

Levitus climatological data are available from the National Oceanographic Data Centre, Washington D.C.. The data represent the result of objective analyses performed on a one-degree latitude-longitude grid at a number of surfaces of constant depth within the world ocean. As there is a lack of synoptic data, the mean values are based on a composite of all available data regardless of year of observation. Data available include: annual summaries of temperature; salinity; dissolved oxygen; percent oxygen saturation and seasonal summaries of temperature and salinity. The 33 analysis levels, 1° latitude-longitude grid and data format are common to all datasets.

These data are commonly used for surface forcing, reference fields for robust diagnostic relaxation and initial conditions. The task of interpolating the data onto the model grid has to be approached with some care. Problems can arise, for example, where a model sea-point overlies a Levitus land-point (values in the Levitus datasets are not interpolated over land). A Fortran77 program, levi2mom.F, has been developed which will produce datasets of potential temperature and salinity on the current MOM-grid from the original Levitus datasets. The program uses the MOM modules to define the model grid, so a change in model grid or domain will be automatically adjusted by recompiling levi2mom.F with the same preprocessor directives that are used for the main model.

The procedure followed by the program is as follows:

(1) Define model grid using setgrid.F. Read in kmt field from file produced by makekm.F

(2) Open original Levitus temperature and salinity files (or previously created potential temperature file).

(3) Set all array values in the 'Levitus' arrays to the land mask value (this is necessary because land points are excluded from the dataset).

(4) Read through datasets and perform steps (5) to (7) for each station.

(5) Unpack data, convert temperature to potential temperature if necessary.

(6) Vertically interpolate from the 33 NODC levels onto the model levels. Note: if the lower point is a Levitus land-point then the value at the model level is left undefined.

(7) Store values at as many model levels as memory limitations permit for each Levitus station. That is, the storage requirement is at least 360 x 180 Levitus stations x 'km' model levels x 2 tracers.

(8) Perform horizontal interpolation. The steps taken to assign values to each model point are as follows:

   (a) Find the four stations which surround the model point

(b) If all four are Levitus sea-points then perform standard 4-pt interpolation.

(c) Else if only three are Levitus sea-points then take the average of the three values.

(d) Else leave point undefined.

(e) When as many points as possible on the model grid have been filled in using steps 8(a)

to 8(d), the remaining undefined model sea-points are set iteratively:

(i) Working on the model grid attempt to set each undefined sea-point as an average of the nine surrounding values.

(ii) If any model sea-points remain unset after a full pass (i.e. those points which were previously completely surrounded by undefined points) then perform a second pass.

(iii) Repeat step 8(e) as many times as necessary until all model sea-points have been set.

(9) Store model level as ascout slices and in unformatted 'j-slabs' suitable for use in the main model.

(10) Perform steps 8 and 9 for each model level in store.

(11) If all model levels have been set then close files and exit. Else rewind Levitus datasets and return to step 3

There are two options for converting temperature to potential temperature coded into the current version of levi2mom.F. By default, the routine ptmp83a is used. This routine uses a 4th-order Runge Kutta integration of the Bryden (1973) equation for adiabatic lapse rate. The alternative is to use pottem routine developed by Webb (1992). This routine accurately solves the adiabatic lapse rate equation by direct integration with a pressure increment. This method is the most accurate to date but is computationally very expensive. The pottem routine can be used in preference to ptmp83a by preprocessing levi2mom.F with the -Dpottem commandline option.

The current version of levi2mom.F (SCCS version number 1.4) assumes a uniform model grid spacing but alternative distributions could be allowed for by re-defining the functional forms of the variables 'xsm' and 'ysm'.

The two unformatted, direct-access files created by levi2mom.F (dalevs21 and dalevt21) are used by the main model if either of two new options are active:

Levitus (note the capital letter) : This option causes the model temperature and salinity fields to be initialised from the Levitus data (if init is .true.) and, if **restorst** is also active, will use the Levitus values when calculating the surface restoring force.

robustd : This option will run the model in robust diagnostic mode with values relaxed towards Levitus at all depths with a timescale of 'rests' (set in blkdta.F)

On reflection, there is an option missing here. Namely the ability to start from a cold, saline ocean (or even the zonal values) and relax towards Levitus in a similar manner to the first six years of the FRAM integration. With the FRAM integration this approach was necessary because the system was unstable when started from Levitus. It is obviously advantageous to start from Levitus data (or its equivalent) whenever possible, but should it be necessary to initialise with other values then other options can be easily incorporated into ocn1st.F.

Code changes and additions required to implement the Levitus and robustd options are located in the following modules:

Option number	File	SCCS version
Levitus	ocean.F	1.8
	ocn1st.F	1.4
	setvbc.F	1.4
robustd	ocean.F	1.8
	setvbc.F	1.4
	step.F	1.4
	tracer.F	1.4

and all subsequent versions of these modules.

7. PRELIMINARY RUNS AND RESULTS

The model described in this report has been successfully integrated for periods up to 200 days. The Levitus values produced from levi2mom.F were used as initial conditions for temperature and salinity. Despite the obvious inertial shock resulting from using an initially incompatible velocity field (i.e. stationary) the solution process remained stable. The initial stages were carried out with a range of timesteps due to concern for the stability. This concern seems to have been unnecessary although tests with timesteps of 2 hours or more were unstable.

A run of 260 days was accomplished using the following timesteps:

days 0 to 14 : timestep = 20 minutes
days 14 to 28 : timestep = 1 hour
days 28 to 260 : timestep = 1.5 hours

The stream-function field was stored at regular intervals and an animation sequence suitable for viewing with imagetool has been produced. The sequence shows the first 28 days of

the model integration with one frame every 4 model hours. Total kinetic energy and global rate of change of temperature and salinity can be seen in Figure 3.

The model was run on a Sun Sparcstation IPC with 24 Mbytes of memory. There was sufficient memory to run the model in-core (-D diskless) and in this mode the model progressed at an average rate of 150 seconds per timestep. (Note this is single precision arithmetic only.)

As an experiment the model was restarted from Levitus but this time with the velocity field from the end of the first run. The graph in Figure 4 shows fewer inertial oscillations and a marked decrease in the initial rate of warming. A program, reset2levi.F has been supplied to take a restart dataset and replace the temperature and salinity fields with Levitus values.

Plots of stream function and surface velocity field have also been included (Figures 5 and 6).

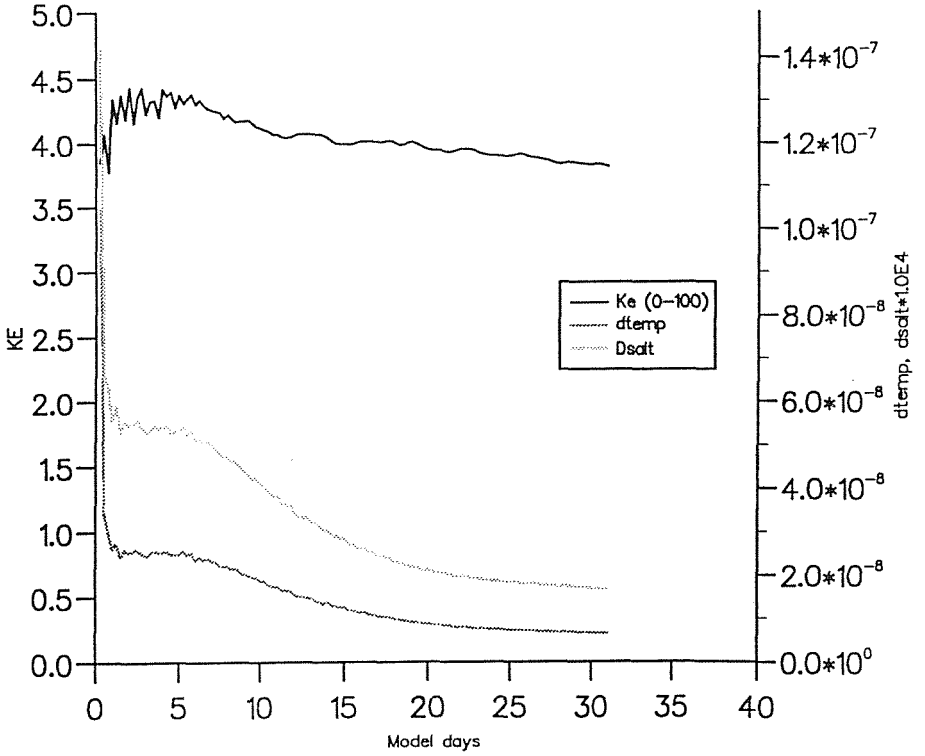

Figure 3 : The total kinetic energy and global rate of chage of temperature and salinity. Run 1 : Start from Levitus climatology, with stationary state.

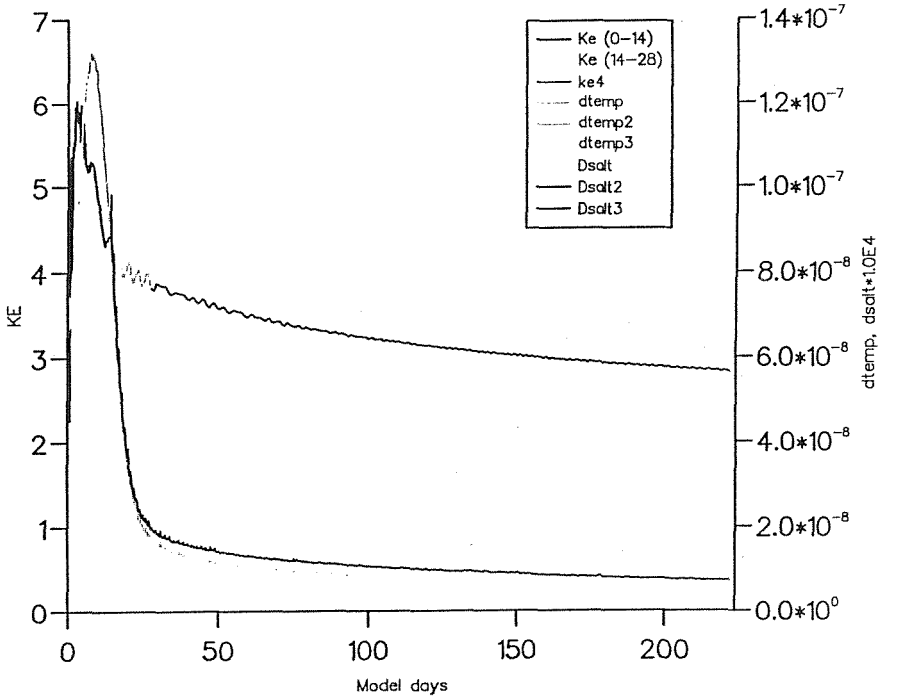

Figure 4: The total kinetic energy and global rate of change of temperature and salinity. Run 2 : Reset initial tracers to Levitus climatology, retaining developed velocity field.

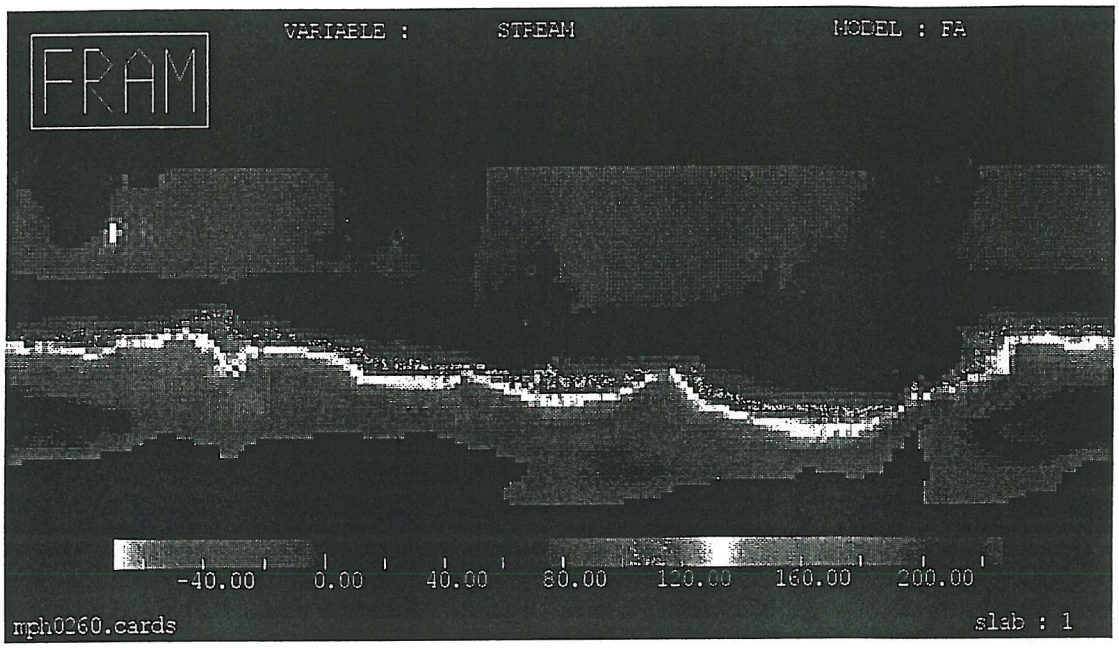

Figure 5 : Stream-function field after approximately 9 months of model integration.

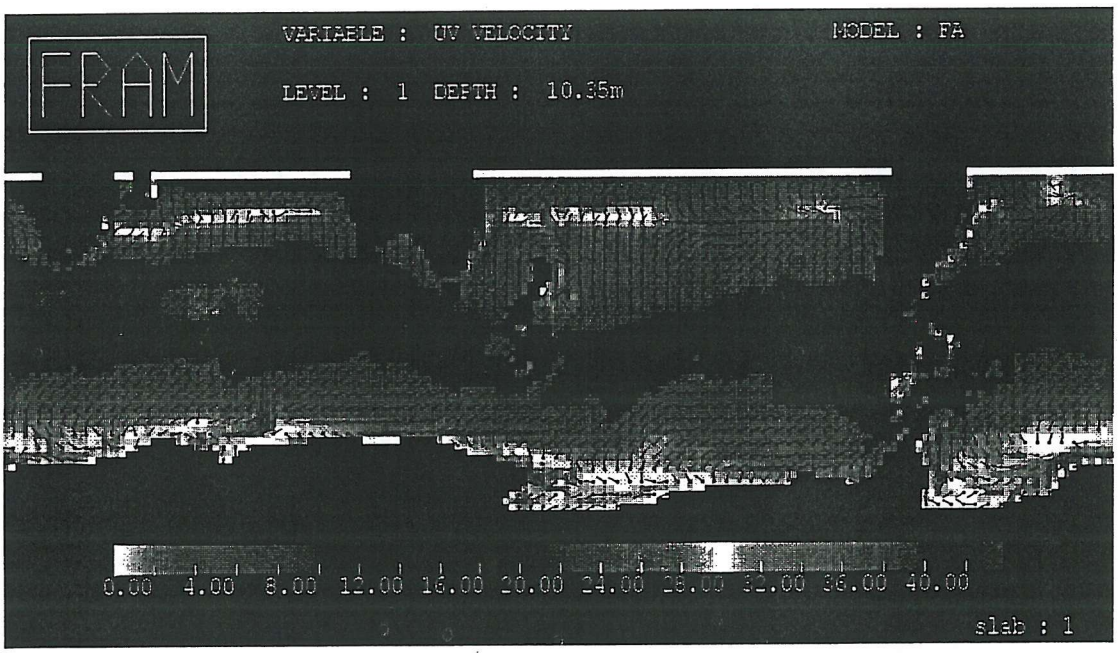

Figure 6 : Surface velocity field after approximately 9 months of model integration.

REFERENCES

Bryden, H. 1973
Deep-Sea Research, 20, 401-408.

Cox, M.D. 1984 A primitive equation, 3-dimensional model of the ocean.
GFDL Ocean Group Technical Report No. 1, Geophysical Fluid Dynamics Laboratory/NOAA, Princeton University, Princeton, N.J. 08542, U.S.A.

de Cuevas, B. 1992 The main runs and datasets of the Fine Resolution Antarctic Model Project (FRAM). Part 1: The coarse resolution runs.
Institute of Oceanographic Sciences Deacon Laboratory, Internal Document No. 315.

de Cuevas, B. 1993 The main runs and datasets of the Fine Resolution Antarctic Model Project (FRAM). Part II: The fine resolution runs.
Institute of Oceanographic Sciences Deacon Laboratory, Internal Document No. 318.

Hellerman, S & Rosenstein, M 1983 Normal monthly wind stress over the world ocean with error estimates.
Journal of Physical Oceanography, 13, 1093-1104.

Killworth, P.D. 1987 Topographic Instabilities in Level Model OGCMs.
'Ocean Modelling', Issue 75, November 1987.

Levitus, S. 1982 Climatological Atlas of the World Ocean.
NOAA Professional Paper 13, US Government Printing Office, Washington D.C.

Pacanowski, R.C., Dixon, K & Rosati, A. 1990 The GFDL Modular Ocean Model 1.0.
Geophysical fluid Dynamics Laboratory / NOAA, Princeton University, Princeton, N.J. U.S.A. (Unpublished manuscript)

Sarmiento, J.L. & Toggweiler, J.R. 1986 A preliminary model of the role of upper-ocean chemical dynamics in determining oceanic oxygen and atmospheric carbon dioxide levels.
Dynamic processes in the chemistry of the upper ocean. (Ed. J.D. Burton et al.)
New York, Plenum Press, 246 pp.

Fasham, M.J.R., Sarmiento, J.L., Slater, R.D., Ducklow, H.W. & Williams, R. 1993 Ecosystem behaviour at Bermuda Station 'S' and Ocean Weather Station 'India': a general circulation model and observational analysis.
Global Biogeochemical Cycles, 7(2), 379-415.

Plummer, N.P. 1991 DBDB5 data set of global gridded bathymetry.
Institute of Oceanographic Sciences Deacon Laboratory, Internal Document No. 300.

Stevens, D.P 1990 On open boundary conditions for three dimensional primitive equation ocean general circulation models.
Geophysical and Astrophysical Fluid Dynamics, 51, pp 103-133.

Stevens, D.P. 1991 The open boundary condition in the United Kingdom Fine Resolution Antarctic Model.
Journal of Physical Oceanography, 21(9), pp 1494-99.

Webb, D.J. 1992 The equation of state algorithms used by the FRAM model.
Institute of Oceanographic Sciences Deacon Laboratory, Internal Document No. 313.

APPENDIX A: MANAGING THE MOM CODE IN A UNIX ENVIRONMENT

This appendix outlines the method of working with the GFDL Modular Ocean Model in a UNIX environment, which has evolved in the light of the experience gained in developing the 2° x 1° resolution model. Obviously every modeller has a preferred method of working with and developing code, but the MOM code presents a few challenges through sheer size and distribution (87 files for the base code alone). The techniques required to efficiently handle the model are no more than the standard software development practises which are commonly used in the commercial sector. Fundamental rules are:

(1) All files should have a development history and previous versions should be recoverable at any time.

(2) Only one version of each file should be available for editing at any one time. (This prevents different and possibly incompatible changes being made simultaneously to different copies of the current version).

The Source Code Control System (SCCS) available on the Sun workstations provides precisely the required control system. SCCS or software offering similar functionality is available in most UNIX environments. The current Southern Ocean model has been developed under SCCS control and is being distributed to interested parties in this form. That is to say, the current model is being supplied together with its development history. Recipients will be able to retrieve the source code in its most up-to-date form, in its original form or in any of the intermediate states. Appendix B contains details of the development history of all the files that constitute the current model. The information presented in Appendix B is that given by the command:

```
sccs prt *.h *.F
```

With the benefit of hindsight, it should be pointed that the way in which the current model has been developed is not ideal. The problem lies in the fact that some modules required changes at several development stages while others were altered at fewer stages (if at all). As a result the current model uses, for example, version 1.9 of ocean.F but version 1.4 of step.F. This is less of a problem than might be expected because the current model uses the latest versions of all routines and, by default, the command:

```
sccs get *
```

retrieves the latest version of all files. One way around this apparent inconsistency is to 'check out' all mom-files prior to a development stage and to create change 'deltas' for all files once the stage

has been completed (regardless of whether any changes have been made or not). The procedural sequence:

```
sccs edit SCCS
.....make changes....
.....debug and verify changes....
sccs delta SCCS
```

should achieve this and will only prompt for a single comment which it will apply to all files. If individual files require a more specific comment their delta commentary can be changed using the `sccs fix` command. This command retrieves a version for editing but when checked back in the version number is retained. Thus to change a delta commentary use the following procedure:

```
sccs fix -rvid filename
sccs delta filename
.... new comments in response to prompt....
```

where *vid* is the latest version number. The `fix` subcommand is also useful for correcting minor bugs discovered after a delta has been made but which do not justify the creation of a new delta.

For colleagues wishing to develop their own models from the mom base code, I have supplied a script: *get_debugged_mom*, which will retrieve the mom base code plus the bug fixes suggested in MOM_NEWS1.0. Colleagues constructing models from this corrected base code would be advised to copy the retrieved files to a fresh directory and create another sccs base version set using the command:

```
sccs create *
```

Changes to this set could then be progressed in a consistent manner, as suggested above. Note the `sccs create` command will do the following for all files matched by the wildcard:

(1) Create a file called `s.`*filename* in the SCCS subdirectory

(2) Rename *filename* by placing a comma in front of the name

(3) Retrieve a read-only version of each file using the `sccs get` command.

Once you are satisfied that the retrieved versions are identical to the renamed files it is best to delete all the *,filename* files.

Of course, keeping track of your edits is only one of the problems associated with having the code separated into a large number of files. Equally important is understanding the full impact of the changes that are being implemented. Simple facts like knowing if a particular variable is defined in a subroutine can involve a considerable amount of tracing. The UNIX `grep` command

can be extremely useful in this context. First of all, the common block in which a variable is held can be located by searching the .h include files. For example, searching for the grid spacing array, dxt:

```
grep dxt *.h
```

would yield:

```
fdift.h:     UTx(i,k) = (aux1(i,k) - aux1(i+1,k))*dxt4r(i)
fdift.h:     Txx(i,k) = bbt(j)*dxt4r(i)*
fdift.h:     Txx(i,k) = bbtj(i,k)*dxt4r(i)*
fdift.h:     Txx(i,k) = bbt(j)*dxt4r(i)*
fdift.h:     Tisox(i,k) =  (e(i,k,1) - e(i-1,k,1))*dxtr(k)*cstr(j)
grdvar.h:c    dxt      = longitudinal width of "t" grid box (in cm)
grdvar.h:c    dxtr     = reciprocal of "dxt"
grdvar.h:c    dxt2r    = reciprocal of "2*dxt"
grdvar.h:c    dxt4r    = reciprocal of "4*dxt"
grdvar.h:     common /grdvar/ dxt(imt),  dxtr(imt),  dxt2r(imt),
dxu(imt)
grdvar.h:    $,               dxur(imt), dxu2r(imt), dxu4r(imt),
dxt4r(imt)
```

Note also that this is an efficient way of finding out the definition of a variable because all the include files are extremely well commented.

The subroutines can then be checked to discover which of these include grdvar.h, i.e.:

```
grep grdvar.h *.F
```

Since initially compiling the model , two new features have been introduced. Firstly, a script: *setarg* which simply sets a shell variable, `arglist`, to the complete set of preprocessor commandline options that are used for the current model. These are:

```
-Drestrt10d -Dascdump -DLevitus -Dannwind -Dopenbc -Dcyclic
-Ddiskless -Drestorst -Dconstvmix -Dconsthmix -Drigidlid
-Dhypergrid -Dislands -Dnohilats
```

(Appendix C contains a complete list of the new options that have been introduced to the base code.) Secondly, I have defined an 'ordered list' of the files that constitute the model. This list is held in the file *catmom*. The list is used at compilation to construct the complete source code prior to preprocessing. I have found it convenient to order the list into some semblance of the original Cox code and to exclude files that are not used by the current set of options. I have included a script: *list_missing*, which will compare the contents of *catmom* to the complete list of .F and .f files in the current directory and list those currently omitted from *catmom*. The complete list of original MOM files is kept for reference in the file *momfiles*.

The script: *makeup*, uses *setarg* and *catmom* to preprocess and compile the model. The complete source code after preprocessing is held in the file: *momver2.f*. Listings of all the scripts are included at the end of this appendix. The script '*makeup*' is also listed here so that its structure can be referenced in the proceding paragraphs:

```
#
# Version 1 of the makeup script. The code is preprocessed
# and compiled as suggested in the "initial report on the
# GFDL Modular ocean code". The file catmom holds an ordered
# list of the component files. Using this method both the
# unprocessed code (momver1.f) and the preprocessed code
# (momver2.f) contain modules in the order determined by the
# catmom file.
#
set oceanobj = ocean2
rm momver1.f
rm momver2.f
cat `cat catmom` >momver1.f
source setarg
set echo
cc -P $arglist momver1.f
mv momver1.i momver2.f
 f77 -o $oceanobj momver2.f
 rm *.o
unset echo
echo "executable file:" $oceanobj "has been created
exit
```

The file *momver2.f* can be useful when it comes to debugging. This file contains all the code in-line and it is, therefore, possible to see the results of the particular set of options chosen at compilation. At this stage, however, the thorough commenting of the include files is detrimental. Producing a source listing at this stage would be extremely wasteful due to the number of times each set of comments are included. There are two ways around this problem and each solution has its own merits. The first method is to simply remove all the comments lines. The editing sequence:

```
ed momver2.f
g/^c/d
w
q
```

will achieve this result. This reduces the code from over 20,000 lines to under 8,000. However, unless you are extremely familiar with the code the listing is difficult to follow. The main benefit of

this form is for use in an interactive debugging tool (such as dbxtool) where problem lines can be quickly identified without having to wade through pages of comments. A second '*makeup*' script (*makeup2*) has been supplied which will perform these edits on the *momver2.f* file and compile with the debug flag set (i.e. f77 -g ...).

The second method of producing a reduced listing is to comment out the `#include` statements before preprocessing. The resulting *momver2.f* file will not compile but is useful for checking the effect of preprocessing on the conditional constructions. This, together with an up-to-date listing of the include files, provides the most useful reference listings. A script, *makeup_listing*, has been supplied which will create two files: *code_listing* and *include_listing*. Together these files have less than 10,000 lines.

A final requirement for efficient working is the ability to view the results of a model run quickly and easily. For this, a utility similar to the FRAM extract program has been developed. The utility, `getslice.F`, can be used to extract standard direction slices (i.e. constant depth, constant latitude or constant longitude) from a MOM restart dataset. Again the MOM modules have been used extensively, which should mean that a change in model setup will be automatically accounted for by recompiling `getslice.F` with the same preprocessor commandline options used for the main model. This is where the script *setarg* can save some typing, e.g.:

```
source setarg
cc -P $arglist getslice.F
mv getslice.i getslice.f
f77 -o getslice getslice.f
```

The program will prompt for an input filename, choice of variable etc. Output is in the form of single-slice ascout 'cards' files. The names of the files are seven-character configurations (plus a .cards extension), formulated as follows:

```
m (for MOM)
one of {t,s,u,v,p or d} (i.e. temperature, salinity, u-velocity,
                                 v-velocity, stream-fn or depth)
one of {h,n,e} (i.e. horizontal, north-south or east-west)
a four-digit daynumber
```

These files have correct headers and can viewed using the FRAM graphics programs.

SCRIPT LISTINGS:

makeup:

```
#
# Version 1 of the makeup script.  The file catmom holds an ordered
# list of the component files.  Using this method both the
# unprocessed code (momver1.f) and the preprocessed code (momver2.f)
# contain modules in the order determined by the catmom file.
#
set oceanobj = ocean2
rm momver1.f
rm momver2.f
cat `cat catmom` >momver1.f
source setarg
set echo
cc -P $arglist momver1.f
mv momver1.i momver2.f
 f77 -o $oceanobj momver2.f
 rm *.o
unset echo
echo "executable file:" $oceanobj "has been created
exit
```

makeup2:

```
#
# Version 2 of the makeup script.  This version compiles the code
with
# the debug (-g) option set on f77.  The script also edits the pre-
# processed file, momver2.f, and removes all comment lines.  The
# resulting file is more compact and easier to manipulate using the
# source code browser in dbxtool.
#
set oceanobj = ocean2
rm momver1.f
rm momver2.f
cat `cat catmom` >momver1.f
source setarg
set echo
cc -P $arglist momver1.f
mv momver1.i momver2.f
ed - momver2.f << EOF
g/^c/d
w
```

```
q
EOF
 f77 -g -o $oceanobj momver2.f
 rm *.o
unset echo
echo "executable file: " $oceanobj " has been created"
exit
```

makeup_listing:

```
#
# Script to makeup listings of the current model code.  The resulting
# files code_listing and include_listing are not compilable but are
# useful for reference.
#
rm momver1.f
rm code_listing
rm include_listing
cat `cat catmom` >momver1.f
cat *.h >inclusions_tmp.f
#
ed - momver1.f << EOFA
1,\$s/\#include/C_INCLUDE/
w
q
EOFA
#
ed - inclusions_tmp.f << EOFB
1,\$s/\#include/C_INCLUDE/
w
q
EOFB
#
source setarg
set echo
cc -P $arglist momver1.f
cc -P $arglist inclusions_tmp.f
mv momver1.i code_listing
mv inclusions_tmp.i include_listing
unset echo
rm inclusions_tmp.f
exit
```

setarg:

```
#
set arglist = " -Drestrtl0d -Dascdump -DLevitus -Dannwind -Dopenbc
-Dcyclic -Ddiskless -Drestorst -Dconstvmix -Dconsthmix -Drigidlid
-Dhypergrid -Dislands -Dnohilats"
echo arglist set to $arglist
```

list_missing:

```
#
set vars = `cat catmom`
set vars2 = `ls *.F *.f`
#
echo "The following source files are located in the current
      directory"
echo "but are NOT included in the catmom file:"
echo "---------"
foreach var ($vars2)
set yes = no
foreach varr ($vars)
 if ($varr == $var ) set yes = yes
end
 if ($yes == 'no') echo $var
end
exit
```

get_debugged_mom:

```
#
# Script to retrieve a debugged version of the MOM base code.
# (i.e. a version with all fixes suggested in MOM_NEWS1.0
# implemented.)
#
sccs get *.h *.F *.f
sccs get -r1.2 slabs.h
sccs get -r1.2 checks.F
sccs get -r1.2 denscoef.F
sccs get -r1.2 docmnt.F
sccs get -r1.5 ocean.F
sccs get -r1.2 reg1st.F
sccs get -r1.2 restio.F
sccs get -r1.2 tracer.F
exit
```

APPENDIX B

SCCS development history of the .h include files

(Files which have been unaltered since creation will have a SCCS version number of 1.1)

```
accel.h
anlwind.h

cbihar.h
ccfl.h
cdiag.h
chmix.h
cisop.h
cnlmix.h
coord.h
cpolar.h
cppmix.h
cprnts.h
cregin.h
crelax.h
cshrbf.h
ctask.h
ctcmix.h
ctmngr.h
cvbc.h
cvmix.h

------------>
dncoef.h
D 1.3     91/11/19 11:33:51 acc   3 2   00113/00159/00072
density coefficients for 25 vertical levels (Sarmiento)

D 1.2     91/06/19 09:22:17 acc   2 1   00184/00072/00047
Reworked coefficients (using denscoef.F (eqstat)) for Cram
thicknesses

D 1.1     91/05/08 09:33:01 acc   1 0   00119/00000/00000
date and time created 91/05/08 09:33:01 by acc
<------------
docnam.h

------------>
emode.h
D 1.2     92/01/17 14:28:32 acc   2 1   00003/00000/00045
First working version with open boundary

D 1.1     91/05/08 09:33:03 acc   1 0   00045/00000/00000
date and time created 91/05/08 09:33:03 by acc
<------------
```

fdifm.h
fdift.h

grdvar.h

index.h
iounit.h

levind.h

ndcon.h

---------->
param.h
D 1.5 92/01/17 14:27:50 acc 5 4 00001/00001/00086
First working version with open boundary

D 1.4 91/06/28 15:18:29 acc 4 3 00001/00001/00086
Set number of islands=4 for new 2x1 topography

D 1.3 91/06/20 17:24:35 acc 3 2 00001/00001/00086
Corrected number of islands (nisle) for Fram 2 by 1

D 1.2 91/06/18 17:02:34 acc 2 1 00001/00001/00086
Changed resolution to Fram 2 by 1.

D 1.1 91/05/08 09:33:10 acc 1 0 00087/00000/00000
date and time created 91/05/08 09:33:10 by acc
<----------
pconst.h
pfil.h

scalar.h
---------->
slabs.h
D 1.2 91/05/08 09:36:03 acc 2 1 00001/00001/00173
Fixed bug reported in MOM_NEWS no.1 (improper dimensions)

D 1.1 91/05/08 09:33:13 acc 1 0 00174/00000/00000
date and time created 91/05/08 09:33:13 by acc
<----------
---------->
switch.h
D 1.2 92/02/19 12:07:58 acc 2 1 00006/00000/00076
Added a new flag: t10day used by the restrt10d option to detect
the end of a 10-day period.

D 1.1 91/05/08 09:33:14 acc 1 0 00076/00000/00000
date and time created 91/05/08 09:33:14 by acc
<----------

tcslab.h
---------->

```
temp.h
D 1.2     92/01/17 14:29:06 acc   2 1   00004/00000/00013
First working version with open boundary

D 1.1     91/05/08 09:33:16 acc   1 0   00013/00000/00000
date and time created 91/05/08 09:33:16 by acc
<--------------
-------------->
thick.h
D 1.3     91/12/16 14:02:11 acc   3 2   00005/00006/00013
Changed to the Sarmiento model thicknesses (25 levels)

D 1.2     91/06/19 09:04:47 acc   2 1   00006/00004/00013
Changed thickness to the CRAM dimensions (32 levels).

D 1.1     91/05/08 09:33:17 acc   1 0   00017/00000/00000
date and time created 91/05/08 09:33:17 by acc
<--------------
timelv.h

versno.h
```

SCCS development history of .F files

```
-------------->
anlwind.F
D 1.2     92/01/27 16:54:11 acc   2 1   00050/00050/00007
Working version with annual Hellerman winds

D 1.1     92/01/17 17:10:58 acc   1 0   00057/00000/00000
date and time created 92/01/17 17:10:58 by acc
<--------------
annwind.F

bcest.F
-------------->
blkdta.F
D 1.3     92/01/17 14:28:53 acc   3 2   00002/00002/00247
First working version with open boundary

D 1.2     91/06/18 17:15:47 acc   2 1   00008/00008/00241
Changed grid parameters to Fram 2 by 1 resolution.

D 1.1     91/05/08 09:20:00 acc   1 0   00249/00000/00000
date and time created 91/05/08 09:20:00 by acc
<--------------

cfl.F
-------------->
checks.F
D 1.3     92/01/17 14:26:15 acc   3 2   00021/00000/00505
First working version with open boundary
```

```
D 1.2     91/05/08 09:32:10 acc   2 1   00002/00002/00503
Fixed bug reported in MOM-NEWS no.1 (missing commas)

D 1.1     91/05/08 09:20:02 acc   1 0   00505/00000/00000
date and time created 91/05/08 09:20:02 by acc
<-------------
------------->
clinic.F
D 1.2     92/01/17 14:23:05 acc   2 1   00085/00000/00984
First working version with open boundary

D 1.1     91/05/08 09:20:03 acc   1 0   00984/00000/00000
date and time created 91/05/08 09:20:03 by acc
<-------------
cnvmix.F
congr5.F
congr9.F


delsq.F
------------->
denscoef.F
D 1.3     91/06/20 18:26:50 acc   3 2   00002/00002/01226
changed program to Fram 2 by 1 set-up

D 1.2     91/05/08 09:28:36 acc   2 1   00004/00004/01224
Fixed bugs reported in MOM_NEWS no.1 (incorrect variable types)

D 1.1     91/05/08 09:20:08 acc   1 0   01228/00000/00000
date and time created 91/05/08 09:20:08 by acc
<-------------
diag.F
diag2.F
------------->
docmnt.F
D 1.2     91/05/08 10:18:50 acc   2 1   00001/00001/00473
Fixed bug reported in MOM_NEWS no.1 (incorrect logical variable)

D 1.1     91/05/08 09:20:12 acc   1 0   00474/00000/00000
date and time created 91/05/08 09:20:12 by acc
<-------------

filfir.F
filt.F
filtr.F
filuv.F
filz.F
findex.F
```

```
------------>
getslice.F
D 1.2     92/02/11 14:39:56 acc   3 1   00071/00988/00746
Revision of first working version to improve modularity and
documentation

D 1.1     92/02/06 17:20:38 acc   1 0   01734/00000/00000
date and time created 92/02/06 17:20:38 by acc
<------------

header.F
------------>
hyper.F
D 1.3     92/02/19 11:31:35 acc   3 2   00003/00000/00469
Included common blocks for annual wind data if annwind option is
selected.

D 1.2     92/01/17 14:23:43 acc   2 1   00099/00001/00370
First working version with open boundary

D 1.1     91/05/08 09:20:18 acc   1 0   00371/00000/00000
date and time created 91/05/08 09:20:18 by acc
<------------

implq.F
invtri.F
iperim.F
isopyc.F

matrix.F

nlmix.F
------------>
ocean.F
D 1.9     92/02/19 12:02:08 acc   10 9  00039/00008/01238
Inserted several options: units 85 and 86 are connected to the
Levitus data files created by levi2mom.F in either Levitus or robustd
options are active.  restrt10d will dump a full restart dataset every
10 days (filename=Mrxxxx.data where xxxx is the day number).  ascdump
will dump a cards image of the stream-function to data/image
/mpzzzz.cards every 16 timesteps (here zzzz is the timestep).

D 1.8     92/01/27 16:53:52 acc   9 8   00008/00000/01238
Working version with annual Hellerman winds

D 1.7     92/01/17 14:28:01 acc   8 7   00029/00000/01209
First working version with open boundary

D 1.6     91/06/20 18:26:02 acc   7 6   00020/00000/01189
inserted ascdump option to dump sf field at end of run.

D 1.5     91/05/09 16:55:48 acc   6 5   00000/00000/01189
Excluded delta 1.2 (discovered that the -Dtiming option only applies
to the CRAY)
```

```
D 1.4     91/05/08 10:08:10 acc   5 4   00010/00001/01184
Fixed bugs reported in MOM_NEWS no.1 (div zero & term balances)

D 1.3     91/05/08 09:48:42 acc   4 3   00001/00001/01184
Fixed bug reported in MOM_NEWS no.1 (vertical region masks)

D 1.2     91/05/03 12:44:12 acc   3 1   00009/00004/01176
Commented out references to second() and timef() which don"t appear
to exist on the SUN.

D 1.1     91/05/03 12:32:25 acc   1 0   01180/00000/00000
date and time created 91/05/03 12:32:25 by acc
<--------------
-------------->
ocn1st.F
ocn1st.F:
D 1.4     92/02/20 12:48:28 acc   4 3   00031/00000/00202
Added option: Levitus. Temperature and salinity (including the tn
array if openbc defined) are now initialised from datasets created by
levi2mom.F

D 1.3     92/02/20 12:46:25 acc   3 2   00033/00000/00169
Added option:openbc. All arrays are now initialised correctly for
open boundary calculations.

D 1.2     92/02/20 12:44:59 acc   2 1   00018/00000/00151
Added option: mytopog to read kmt field from file created by makekm.F

D 1.1     92/02/20 12:08:32 acc   1 0   00151/00000/00000
date and time created 92/02/20 12:08:32 by acc

-------------->
odam.F
D 1.2     92/01/17 14:24:22 acc   2 1   00004/00000/00227
First working version with open boundary

D 1.1     91/05/08 09:20:25 acc   1 0   00227/00000/00000
date and time created 91/05/08 09:20:25 by acc
<--------------

ppmix.F

-------------->
reg1st.F
D 1.2     91/05/08 11:49:01 acc   3 1   00022/00000/00156
Fixed bug reported in MOM_NEWS no.1 (vertical region masks)

D 1.1     91/05/08 09:20:27 acc   1 0   00156/00000/00000
date and time created 91/05/08 09:20:27 by acc
<--------------
region.F
relax.F
reset2levi.F
```

```
------------>
restio.F
D 1.2     91/05/08 10:15:42 acc   2 1   00004/00000/00126
Fixed bug reported in MOM_NEWS no. 1 (disk restart io error)

D 1.1     91/05/08 09:20:29 acc   1 0   00126/00000/00000
date and time created 91/05/08 09:20:29 by acc
<------------

------------>
setgrid.F
D 1.2     92/02/07 15:00:42 acc   4 1   00010/00000/00367
Incorporated simple "noreport" option to suppress printing of arrays
on stdout

D 1.1     91/05/08 09:20:30 acc   1 0   00367/00000/00000
date and time created 91/05/08 09:20:30 by acc
<------------
setkmp.F
------------>
setvbc.F
D 1.4     92/02/19 11:48:08 acc   4 3   00018/00000/00117
Inserted option: Levitus to read in and restore surface values to
Levitus values (requires data files as created by levi2mom.F)

D 1.3     92/01/27 16:53:10 acc   3 2   00032/00000/00085
Working version with annual Hellerman winds

D 1.2     92/01/17 14:27:36 acc   2 1   00004/00000/00081
First working version with open boundary

D 1.1     91/05/08 09:20:32 acc   1 0   00081/00000/00000
date and time created 91/05/08 09:20:32 by acc
<------------

------------>
size.F
D 1.2     91/06/18 16:51:00 acc   2 1   00004/00004/00545
Changed resolution to Fram 2 by 1 (made main prog)

D 1.1     91/05/08 09:20:33 acc   1 0   00549/00000/00000
date and time created 91/05/08 09:20:33 by acc
<------------
state.F
------------>
step.F
D 1.4     92/02/19 11:52:12 acc   4 3   00018/00002/00842
Inserted option: robustd to restore t and s to Levitus values
throughout (i.e. robust diagnostic mode). Requires data files as
created by levi2mom.F. Also reset the open boundary relaxation
timescale to a consistent value (rtscale).

D 1.3     92/01/27 16:53:35 acc   3 2   00003/00000/00841
Working version with annual Hellerman winds
```

```
D 1.2     92/01/17 14:25:12 acc   2 1   00433/00000/00408
First working version with open boundary

D 1.1     91/05/08 09:20:35 acc   1 0   00408/00000/00000
date and time created 91/05/08 09:20:35 by acc
<-------------
```

tcmix.F

```
------------->
```

tmngr.F

```
D 1.2     92/02/19 11:55:20 acc   2 1   00008/00000/00528
Inserted switch (t10day) used by the option:restrt10d which will dump
a restart dataset every 10 model days

D 1.1     91/05/08 09:20:38 acc   1 0   00528/00000/00000
date and time created 91/05/08 09:20:38 by acc
<-------------
```

topog.F

```
------------->
```

tracer.F

```
D 1.4     92/02/19 12:12:37 acc   4 3   00001/00001/00641
Added the robustd option which in the case of tracer.F means simply
stopping the restorst option from duplicating the setup of the sourct
array already performed in step.F

D 1.3     92/01/17 14:25:28 acc   3 2   00005/00001/00637
First working version with open boundary

D 1.2     91/05/08 10:10:27 acc   2 1   00001/00001/00637
Fixed bug reprted in MOM_NEWS no.1 (incorrect term balances)

D 1.1     91/05/08 09:20:40 acc   1 0   00638/00000/00000
date and time created 91/05/08 09:20:40 by acc
<-------------
```

vort.F

```
------------->
```

windwt.F

```
D 1.4     92/01/27 16:54:28 acc   4 3   00007/00004/00016
Working version with annual Hellerman winds
D 1.3     92/01/17 14:29:20 acc   3 2   00000/00006/00020
First working version with open boundary
D 1.2     91/12/09 14:38:59 acc   2 1   00014/00008/00012
First version
D 1.1     91/11/14 12:21:20 acc   1 0   00020/00000/00000
date and time created 91/11/14 12:21:20 by acc
<-------------
```

APPENDIX C : SUMMARY OF NEW OPTIONS

This appendix contains a summary of the new options which have been introduced during the construction of the 2° x 1° resolution model. Most of these options have been described in sections 1 to 5.

Option	description
mytopog	Causes the kmt field to be read in from an unformatted file on fortran unit 53. This option is only effective at initialisation (i.e. init = .true.). The unformatted kmt data can be created using makekm.F.
openbc	Causes the model to be constructed with an open northern boundary. At present the reference array (tn) holding values of the tracers along the northern boundary is set to the initial values at the jmt row. This is correct when starting the model from Levitus data.
annwind	Causes the wind stress components to be read from the files containing Hellerman and Rosenstein mean annual wind data. This option is quite specific to the 2° x 1° Southern ocean model but could be generalised with moderate effort.
Levitus	(Note the capital letter.) This option causes the model temperature and salinity fields to be initialised from the Levitus data (if init is .true.) and, if restorst is also active, it will use the Levitus values when calculating the surface restoring force.
robustd	This option will run the model in robust diagnostic mode with values relaxed towards Levitus at all depths with a timescale of "rests" (set in blkdta.F)
noreport	A simple option added to setgrid.F to suppress the printing of the grid-spacing arrays at start-up.
ascdump	An option which will cause an ascout dump of the stream-function at a preset interval (currently set to 16 timesteps). This is specific to routine requirements, but could be adapted easily. All related code is in ocean.F. The output is intended for animation via imagetool (after post-processing) and is placed in data/image/mpxxxx.cards, where xxxx is the timestep.
restrt10d	Causes a full restart dataset to be written every 10 model days. Output is written to a file: Mrxxxx.data where xxxx is the daynumber.

In addition to these options for the main model, some of the utility programs have their own options:

option	file	description
nomodel header files. and MOM The MOM	header.F	header.F is a re-working of the old FRAM routine for creating headers for ascout This version uses some of the MOM modules to determine starting latitudes and longitudes grid spacings. For applications outside of these values can be set within the routine. nomodel option suppresses inclusion of the "hard-wired" values.
vtsteps variation timesteps	getslice.F	vtsteps activates code to calculate the day number from itt according to a preset in the timestep. Such variations in are common in the early stages of a run.
pottem to	levi2mom.F	Causes the potential temperature calculation be performed by direct integration of the adiabatic lapse rate equation. This is the most accurate method but is computationally expensive. By default the equation is solved using a Runge Kutta numerical integration.

APPENDIX D (i)

```
      program  db2med
c     ****************
c     This program is a re-working of Nick plummer's program to retrieve
c     dbdb5 data on a 1/4 degree resolution grid using the median value
c     of all the 5 min data within each grid cell (Plummer, 1991).  This
c     version uses the same approach to extract data on a 2° x 1° grid.
c     This causes additional difficulties because each file in the
c     orignal datasets holds a 5 degree square of data.  Every third
c     2° x 1° grid cell therefore straddles two files. Hence this
c     extensive re-working of N. Plummer's original program.
c
c     This version is designed to be run on the IBM (using temporary
c     disks).  The output files can be transferred to the SUN work-
c     stations and used by the utility 'makekm' to construct the
c     topography array for a MOM code application.
c     ********************************************************************
c     * first section of the program is concerned with selecting the     *
c     * files you wish to work with.  Each file contains depth values    *
c     * from the northern/southern hemispheres.                          *
c     ********************************************************************
      character * 80 record,header
      character * 20 fname, filename(2), testdata(2)
      integer file1,file2,file3,file4,nrec,nfiles,position
      integer a(4000),b(122,61),median,n,loop,f,z,med(25),ideg(288)
      data filename /'stor1 data t', 'stor2 data t'/
      data testdata /'test1 data t', 'test2 data t'/
      nrec=18954
c
c     Open the original dbdb5 data file. Direct access file already
      copied to on-line storage
c
      fname='dbdb5 data a'
c
c IBM ndopen routine: call ndopen(unit,file,direct-access,read-only,
c                                 status,recl,nrec,return-status)
c
      call ndopen(3,fname,3,1,'old',80,303264,istat)
      if (istat.ne.0) then
        write(6,*)'error in opening file ', fname
        write(6,*)'istat= ',istat
        stop
      endif
c
      fname='median data a'
c
c IBM ndopen routine: call ndopen(unit,file,sequential,read/write
c                                 status,recl,nrec,return-status)
c
      call ndopen (4,fname,1,3,'unknown',110,0,istat)
```

```
       if (istat.ne.0) then
         write(6,*)'error in opening file',fname
         write(6,*)'istat = ',istat
       endif
c
c
         do 999 file1=1,15,2
         file2=file1
         nblock=0
   1     continue
         write(6,*) 'working on file ',file2
         position=(18954*(file2-1)+1)
         file2=file1+1
         nblock=nblock+1
c
c
c      Open one of the pair of temporary files to receive data
c
c IBM ndopen routine: call ndopen(unit,file,sequential,read/write
c                                 status,recl,nrec,return-status)
c
       call ndopen(15,filename(nblock),1,3,'unknown',80,18954,istat)
       if (istat.ne.0) then
            write(6,*)'error in opening file',filename(nblock)
            write(6,*)'istat = ',istat
       endif

         do j=1,nrec
           read (3,'(a80)',rec=position)record
           write (15,'(a80)')record
           position=position+1
       enddo
c
c
c
c Open "test data" files to receive temporary re-formatted data
       call nclose(15,istat)
       call ndopen(16,testdata(nblock),1,3,'unknown',305,0,istat)
       if (istat.ne.0) then
            write(6,*)'error in opening file',testdata(nblock)
            write(6,*)'istat = ',istat
       endif
c
c reopen file as read-only
       call ndopen(15,filename(nblock),1,1,'unknown',80,18954,istat)
       if (istat.ne.0) then
            write(6,*)'error in opening file',filename(nblock)
            write(6,*)'istat = ',istat
       endif

c      ****************************************************************
c      * loop 81 times for all blocks in a file                       *
c      * sort data into 61i5 format and output to 'testdata' file     *
c      ****************************************************************
```

```
c      Each file contains data covering a 45° x 45° area. The data are
c      arranged in 81, 5x5 degree blocks each with its own header.  Each
c      block has an additional eastern column and northern row which
c      overlaps the neighbouring square.

       do k=1,81
         read (15,'(a80)') header
         write(16,'(a80)') header

c Rearrange the 61x61 (i.e.(5*12+1)**2) values originally written
c as 80 character records into a 61x61i5 array
c
         do i=1,3728,16
           read (15,'(16i5)') (a(i+j-1),j=1,16)
         enddo
         do i=1,3668,61
           write (16,'(61i5)') (a(i+j-1),j=1,61)
         enddo
       enddo
c
c      close 'test data' file
c
       call nclose(15,istat)
       call nclose(16,istat)
       if(nblock.eq.1) goto 1
c      write (6,*) 'after do i=1,3668'

c      ****************************************************************
c      * read information from a 5° square; breakdown that information  *
c      * into 2° x 1° squares and obtain median pts.  Then store        *
c      * results gathered into various files.                           *
c      ****************************************************************
c
       call ndopen (16,testdata(1),1,1,'old',305,0,istat)
       if (istat.ne.0) then
         write(6,*)'error in opening file',testdata(1)
         write(6,*)'istat = ',istat
       endif
       call ndopen (17,testdata(2),1,1,'old',305,0,istat)
       if (istat.ne.0) then
         write(6,*)'error in opening file',testdata(2)
         write(6,*)'istat = ',istat
       endif
c
c      *read all numbers in file into the b array*
c      Note there are three cases to consider depending upon whether the
c      2° x 1° area lies wholly in the first file, straddles both files
c      or lies wholly in the second file.
c
       nbound=0
       do 666 z=1,81
         nbound=nbound+1
         if(nbound.eq.10) nbound=1
c
```

```
c nbound < 4 implies area lies wholly in the first file
c
      if(nbound.le.4) then
      nb=1
        read(16,'(a80)')header
        do j=1,61
        read(16,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
      nb=2
        read(16,'(a80)')header
        do j=1,61
        read(16,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
c
c nbound = 5 implies area straddles the two files
c
      elseif(nbound.eq.5) then
      nb=1
        read(16,'(a80)')header
        do j=1,61
        read(16,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
      nb=2
        read(17,'(a80)')header
        do j=1,61
        read(17,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
c
c nbound > 5 implies area lies wholly in the second file
c
      else
      nb=1
        read(17,'(a80)')header
        do j=1,61
        read(17,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
      nb=2
        read(17,'(a80)')header
        do j=1,61
        read(17,'(61i5)') (b(i,j),i=1+(nb-1)*61,61+(nb-1)*61)
        enddo
      endif
        icounter=0
c
c write header to median data file
        write(4,*)
        write(4,*)header
c
c     ****************************************************************
c     * the 7442 data points are arranged in a (122,61) array.  The  *
c     * calculations below locate the data points required by taking *
c     * the median of the 288 values in each 2° x 1° square sub-     *
c     * division of the original 10° square.                         *
c     ****************************************************************
```

```
        loop=0
        do l=1,5
          do k=1,5
            do i=((l-1)*12)+1,((l-1)*12)+12
              do j=((k-1)*24)+1,((k-1)*24)+24
                icounter=icounter+1
                ideg(icounter)=b(j,i)
              enddo
            enddo
            n=icounter
c
c Use IBM library routine to sort the 288 values into ascending order
c
            call rsort(ideg,288,istat)

c     ***************************************************************
c     * the numbers within the 288 array are now sorted and the median *
c     * value is retrieved.  (The 144th value is selected rather than  *
c     * the true median which could introduce half metres.)            *
c     ***************************************************************
            loop = loop + 1
            med(loop)=ideg(144)
            icounter=0
          enddo
        enddo
c
c     *output med array to median data a (stream 4)*

      do i=1,5
        write (4,'(5i5)') (med((i-1)*5 + j),j=1,5)
      enddo
c
 666  continue
c
c median data a should be transferred to Workstation for post-processing
c (see program med2cram.f)
c
c     *close all streams*
c
      call nclose(16,istat)
      if (istat.ne.0) then
        write (6,*)'error in closing file',testdata
        write (6,*)'istat =',istat
      endif
      call nclose(17,istat)
      if (istat.ne.0) then
        write (6,*)'error in closing file',testdata
        write (6,*)'istat =',istat
      endif
  999 continue
c
      call nclose(4,istat)
```

```
if (istat.ne.0) then
  write (6,*)'error in closing file',fname
  write (6,*)'istat =',istat
endif
stop
end
```

APPENDIX D (ii)

```
      program  med2cram
      ******************
c     Program to convert 'raw' median data produced on the IBM by
c     db2med.f to a topography file for a MOM CRAM run.  This topography
c     will need to be smoothed and checked by the program makekm.f
      (version 1.2 or higher).
C
c     The data file produced by db2med.f gives median depths every two
c     degrees longitudinally and every degree latitudinally, starting
c     at 1.0E, -89.5S.
c
c     The MOM CRAM run requires data at the same resolution starting at
c     1.0E, -78.5S with imt=180,jmt=56.
c
        parameter(imt=180,jmt=56)
        real depths(180,90),vmask(4),cramd(imt,jmt)
        integer median(5,5)
        character*80 line
        data vmask/-10.,3*0.0/
c
c Set the southernmost t-latitude
c
        crams=-78.5
c
c Open the following units:
c 53 - input median data, formatted file produced by db2med.f
c 54 - output file to receive an ascout cards file for viewing
c 55 - output file to receive a full precision unformatted form of the
c      topography array.
c
        open(unit=53,file='smedian.data')
        open(unit=54,file='topog.dbdb5')
        open(unit=55,file='rawcram.dbdb5',form='unformatted')
c
c Create header for cards file:
        call header(54,'depth','stream',1,imt,1,jmt,1,0,'CD','FAA')
c
c Read in median values and re-arrange into a continuous 2-D array:
        do 2 nhalf=1,2
        do 5 nfile=1,4
        do 10 nb2=1,9
        do 20 nb=1,9
          read(53,*)
          read(53,'(a)') line
          do 30 j=1,5
            read(53,'(5i5)') (median(i,j),i=1,5)
 30       continue
c
          id=(nfile-1)*45+(nb-1)*5
          jd=(nhalf-1)*45+(nb2-1)*5
c
```

```
          do 40 i=1,5
            do 50 j=1,5
              depths(id+i,jd+j)=median(i,j)
 50         continue
 40       continue
c
 20    continue
 10    continue
  5    continue
  2    continue
c
c Now select the reduced area (latitude reduction only):
       do 100 j=1,90
       degs=(j-1)*1.-89.5
       if(degs.eq.crams) then
        do 110 i=1,imt
         do 120 jj=j,j+jmt-1
          cramd(i,jj-j+1)=depths(i,jj)
120      continue
110     continue
       goto 99
       endif
100    continue
            write(6,*) 'crams not found, crams= ',crams
c
c Output array and stop:
  99   call ascout0(cramd,imt,imt,jmt,vmask,2,54)
       write(55) cramd
       stop
       end
#include "../ascout0.f"
c include a version of header which doesn't take its parameters from
c the MOM common blocks:
#define nomodel
#include "../header.F"
```

APPENDIX D (iii)

```
      program  makekm
      ****************
c     Program to read in a cards file of median depths produced from
c     DBDB5 data and apply smoothing operations and interpolations to
C     produce a full depths file for MOM
         parameter(imtold=180,jmtold=56,imt=182,jmt=56,km=25,
     +             stlondb=1.0,stlatdb=-78.5,dxdb=2.0,dydb=1.0,
     +             stlon  =1.0,stlat  =-78.5,dxdeg=2.0,dydeg=1.0)
c     parameters:
c     imtold = horiziontal size of DBDB5 cards file data
c     jmtold = latitudinal size of DBDB5 cards file data
c     imt    = horiziontal size of MOM t-grid (including cyclic overlap)
c     jmt    = latitudinal size of MOM t-grid (inc. northern boundary)
c     km     = No. of vertical levels in MOM grid
C
c     The remaining parameters are not used but are included for
      reference:
C     stlondb= starting longitude of first DBDB5 value
c     stlatdb= starting latitude of first DBDB5 value
c     dxdb   = longitudinal resolution (degrees) of DBDB5 cards data
c     dydb   = latitudinal resolution (degrees) of DBDB5 cards data
c     stlon  = starting longitude of first MOM t-point
c     stlat  = starting latitude of first MOM t-point
c     dxdeg  = longitudinal resolution (degrees) of MOM grid
c     dydeg  = latitudinal resolution (degrees) of MOM grid
c
          parameter(imtom2=imtold-2,jmtom2=jmtold-2,
     +              imtm2=imt-2,
     +              jmtm2=jmt-2)
          real fkmold(imtold,jmtold) ,fkmnew(imtm2,jmtm2)
          real tmp(imtold,jmtold)
          real fkmt(imt,jmt),dzt(km),zt(0:km)
          integer kmt(imt,jmt),kmu(imt,jmt)
          real vmask(4)
          character*80 line,ans*1
c
#include "../thick.h"
          data vmask/0.,3*0.0/
          rmax=-1.e7
          rmin=1.e7
          isea=0
          iland=0
          write(6,*) 'Enter number of smoothing passes (0,1 or 2)'
          read(5,*) npass
          write(6,*) 'Remove isolated land points? (y/n)'
          read(5,'(a)') ans
c
c Calculate depths of t-points in MOM grid
c
          zt(0) = 0.0
          zt(1) = dzt(1)*0.5*1.E-2
```

```
          do 700 k=1,km-1
            zt(k+1)  = zt(k) + 0.5*1.E-2*(dzt(k)+dzt(k+1))
            write(6,*) 'Model depth of t-point for k= ',k,' = ',zt(k)
700       continue

          open(unit=20,file='rawcram.dbdb5',form='unformatted')
          open(unit=21,file='topsar21.cards')
c
c
c Create header for cards file:
          call header(21,'depth','stream',1,imt,1,jmt,1,0,'CD','FAA')
c
c Read in "raw" topography as created by med2cram.f
          read(20) fkmold
c
c Smooth topography by one or two passes of this filter:
c         +++++1+2+1+++++
c         +++++2+4+2+++++
c         +++++1+2+1+++++
c
          if(npass.ge.1) then
c First pass:
          do 110 i=1,imtold
          ilt=i-1
          irt=i+1
          if(ilt.eq.0) ilt=imtold
          if(irt.eq.imtold+1) irt=1
           do 100 j=2,jmtold-1
           if(fkmold(i,j).le.0.0) then
            tmp(i,j)=fkmold(i,j)
           else
           tmp(i,j)=(1./16.)*(
     +              fkmold(ilt,j+1)+2.*fkmold(i,j+1)+fkmold(irt,j+1)
     +       + 2.*(fkmold(ilt,j)  +2.*fkmold(i,j)  +fkmold(irt,j)  )
     +       +     fkmold(ilt,j-1)+2.*fkmold(i,j-1)+fkmold(irt,j-1))
           endif
100        continue
           if(fkmold(i,1).ge.1.e-5) then
            write(6,*) 'Bottom boundary set to land at i= ',i,
     +                 ' value was: ',fkmold(i,1)
           endif
c Apply N and S boundary conditions:
           tmp(i,1)=0.0
           tmp(i,jmtold)=tmp(i,jmtold-1)
110        continue
c
           if(npass.eq.1) then
            do 800 i=1,imtold
            do 800 j=1,jmtold
             fkmold(i,j)=tmp(i,j)
800         continue
           else

c
```

```
c Second pass:
            do 210 i=1,imtold
            ilt=i-1
            irt=i+1
            if(ilt.eq.0) ilt=imtold
            if(irt.eq.imtold+1) irt=1
             do 200 j=2,jmtold-1
             if(tmp(i,j).le.0.0) then
              fkmold(i,j)=tmp(i,j)
             else
             fkmold(i,j)=(1./16.)*(
     +               tmp(ilt,j+1)+2.*tmp(i,j+1)+tmp(irt,j+1)
     +        +  2.*(tmp(ilt,j)  +2.*tmp(i,j)  +tmp(irt,j)  )
     +        +      tmp(ilt,j-1)+2.*tmp(i,j-1)+tmp(irt,j-1))
            endif
200         continue
c Apply N and S boundary conditions:
            fkmold(i,1)=0.0
            fkmold(i,jmtold)=fkmold(i,jmtold-1)
210         continue
             endif
c
            endif
c
c
c Adjust depths to nearest MOM vertical level
c
           do 20  j=1,jmtold
            do 30 i=1,imtold
            if(abs(fkmold(i,j)).ge.1.e-4) then
             isea=isea+1
             do 35 k=1,km
              if(zt(k).gt.fkmold(i,j)) then
               kup=k-1
                deltaz1=zt(k)-fkmold(i,j)
                deltaz2=fkmold(i,j)-zt(kup)
c
                if(deltaz1.lt.deltaz2) then
                 fkmnew(i,j)=k
                else
                 fkmnew(i,j)=kup
                endif
               goto 31
              endif
c
  35          continue
              fkmnew(i,j)=km
            else
             iland=iland+1
             fkmnew(i,j)=fkmold(i,j)
            endif
  31        if(fkmnew(i,j).ge.rmax) rmax=fkmnew(i,j)
            if(fkmnew(i,j).le.rmin) rmin=fkmnew(i,j)
  30        continue
```

```
   20         continue
              write(6,*) isea,' sea points ',iland,' land points '
              do 40 j=1,jmt-1
              do 50 i=1,imtm2
               kmt(i,j)= fkmnew(i,j)
   50         continue
   40         continue
c
c Apply cyclic conditions
c
              do 60 j=1,jmt-1
               kmt(imt-1,j)=kmt(1,j)
               kmt(imt,j)  =kmt(2,j)
   60         continue
c
c Apply open northern boundary condition
c (The MOM code will automatically override this if the northern
c boundary is closed within the model)
c
              do 70 i=1,imt
               kmt(i,1)=0
               kmt(i,jmt)=kmt(i,jmt-1)
   70         continue
c
      if(ans.eq.'y'.or.ans.eq.'Y') then
c-----------------------------------------
c remove isolated island points
c-----------------------------------------
      do 250 j=2,jmt-1
      do 260 i=2,imt-1
       if(kmt(i,j).eq.0) then
        if(kmt(i-1,j).ne.0
     +    .and.kmt(i-1,j-1).ne.0
     +    .and.kmt(i-1,j).ne.0
     +    .and.kmt(i+1,j-1).ne.0
     +    .and.kmt(i+1,j).ne.0
     +    .and.kmt(i+1,j+1).ne.0
     +    .and.kmt(i,j-1).ne.0
     +    .and.kmt(i,j+1).ne.0) then
           kmt(i,j) = min(kmt(i-1,j-1),
     +    kmt(i-1,j),kmt(i-1,j),kmt(i+1,j-1),kmt(i+1,j),
     +    kmt(i+1,j+1),kmt(i,j-1),kmt(i,j+1))
           write(6,*) 'Isolated land point removed at: ',i,',',j
         endif
       endif
260   continue
250   continue
      endif
c
c Calculate kmu field
c
      do 310 j=1,jmt
        kmu(imt,j) = 0
310   continue
```

```
c
c
      do 340 j=1,jmt-1
        do 330 i=1,imt-1
         kmu(i,j) = min (kmt(i,j), kmt(i+1,j), kmt(i,j+1), kmt(i+1,j+1))
330     continue
340   continue
      do 350 j=1,jmt
        kmu(imt,j) = kmu(2,j)
        kmu(imt-1,j)=kmu(1,j)
350   continue
      do 320 i=1,imt
        kmu(i,jmt) = kmu(i,jmt-1)
320   continue
c
c-----------------------------------------------------------------------
c     search for isolated bays... "t" grid boxes at the surface which
c     cannot be influenced by advection
c-----------------------------------------------------------------------
c
      do 400 j=2,jmt-1
        do 390 i=2,imt-1
          if (kmt(i,j) .ne. 0) then
            if (kmu(i,j) .eq. 0 .and. kmu(i-1,j) .eq. 0 .and.
     $         kmu(i,j-1) .eq. 0 .and. kmu(i-1,j-1) .eq. 0) then
               write (6,'(10x,a42,i4,a1,i4,a9,i3,a20)')
     $         '==>   Warning: isolated "kmt" at (i,j) = (',i,',',j
     $,        '), kmt = ', kmt(i,j),' is being reset to 0'
              kmt(i,j) = 0
            endif
          endif
390     continue
400   continue
c
c Do likewise for all depths:
c
      do 900 j=2,jmt-1
        do 990 i=2,imt-1
          if (kmt(i,j) .ne. 0) then
            m=kmt(i,j)
            if (kmu(i,j) .lt. m .and. kmu(i-1,j) .lt. m .and.
     $        kmu(i,j-1) .lt. m .and. kmu(i-1,j-1) .lt. m) then
              write (6,'(10x,a42,i4,a1,i4,a9,i3,a20)')
     $        '==>   Warning: isolated "kmt" at (i,j) = (',i,',',j
     $,       '), kmt = ', kmt(i,j),' is being reset to max. kmu'
              kmt(i,j)=max(kmu(i,j),kmu(i-1,j),kmu(i,j-1),kmu(i-1,j-1))
            endif
          endif
990     continue
900   continue
c
      do 500 i=1,imt
      do 510 j=1,jmt
       fkmt(i,j)=kmt(i,j)
```

```
510   continue
500   continue
c
c
c Join N and S islands of New Zealand at 176.E 40.5S
c
           inz=(176. -stlon )/dxdeg +1
           jnz=(-40.5 -stlat)/dydeg +1
           fkmt(inz,jnz)=0.0
         write(6,*)
       + 'Warning ==> N and S islands of New Zealand have been joined'
         write(6,*) '             by fixed code. Check for validity'
         write(6,*) 'Point set was 176.E,40.5S, array coords: ', inz, jnz
c
           write(6,*) 'data min= ',rmin, 'data max= ',rmax
           call ascout0(fkmt,imt,imt-2,jmt,vmask,2,21)
c
           open(unit=24,file='sardepths21',form='unformatted')
            write(24) fkmt
           close(unit=24)
           end
#include "../ascout0.f"
#include "../asciin.f"
#define nomodel 1
#include "../header.F"
```

APPENDIX D (iv)

```
      program  levi2mom
      ******************
c
c Program to read in levitus temp & salinity data from formatted packed
c files and extract a subset of the data.  The data are stored in integer
c format of length 5 with a 10000 offset & multiplied by 1000.
c This format is identical to the original format supplied by GFDL.
c Data represent values at the centre of 1 degree squares.
c
#include "../param.h"
#include "../scalar.h"
#include "../coord.h"
#include "../grdvar.h"
#include "../levind.h"
c
      parameter(mxl=25,npsla=4*imt*km)
c
c mxl is the maximum number of 'levitus levels' that can be retained in
c     memory.  Ideally mxl should equal km but if less than (360*180*km*2)
c     values can be accommodated then several passes can be made through
c     the Levitus datasets in order to achieve the same result.
c     e.g mxl=km or (km/2 + mod(km,2)) or ...  etc.
c npsla is the record length for each direct-access j-slab in the final
c     output file
c
      character *165 trecrd
      character *165 srecrd
      character*1 ans
      real rtcol(33),rscol(33),rpcol
      real vmask(4),levtemp,levsali
      real rkmt(imt,jmt),dmom(km)
      real thic(km),frac1(km),frac2(km)
      real*8 z2pb,pottem,ptmp83a
      common/tracr/ levtemp(360,180,mxl),levsali(360,180,mxl),
     +              tmom(imt,jmt),smom(imt,jmt),tsla(imt,km),
     +              ssla(imt,km)
      integer tcol(33),scol(33),kmL(360,180,mxl),imoml2(imt,jmt)
      integer levdepth(33),lup(km),llo(km),imoml(imt,jmt),
     +        imoml3(imt,jmt)
      logical ioerror,reiter,around,potential,create
      data potential,create/.false.,.false./
      data vmask/1.e7,2.e7,3.e7,4.e7/
c
c These are the depths (m) of the standard 33 levels from the original
  dataset:
      data levdepth/0,10,20,30,50,75,100,125,150,200,250,
     + 300,400,500,600,700,800,900,1000,1100,1200,1300,
     + 1400,1500,1750,2000,2500,3000,3500,4000,4500,5000,5500/
c
```

```
c
      pi     = c4*atan(c1)
      radian = c360/(c2*pi)
      slonLev=0.5
      slatLev=-89.5
      dxLev=1.0
      dyLev=1.0
c
      write(6,*) 'Program for interpolating'
      write(6,*) 'Levitus temperature and salinity onto'
      write(6,*) 'the current MOM grid from the original'
      write(6,*) '1x1 grid at the following standard levels:'
      write(6,*) 'index----depth(m)      index----depth(m)'
      do 111 lk=1,16
        write(6,'(i5,4x,i5,9x,i5,4x,i5)') lk,levdepth(lk),
     +                                    lk+16,levdepth(lk+16)
111   continue
      write(6,'(23x,i5,4x,i5)') 33,levdepth(33)
c
c set offset and mult. factor for unpacking of data
c
      roff=10000.0
      factor=1e-3
c
c Define the current MOM grid
      call grids
      slonmom=xt(1)
      slatmom=yt(1)
      dxmom=xmin(1)
      dymom=ymin(1)
c Define depths of mom-t-points in metres:
c
      dmom(1)=p5*dzt(1)*1.E-2
      do 17 k=2,km
       dmom(k)=dmom(k-1)+p5*(dzt(k-1)+dzt(k))*1.E-2
17    continue
c
c Read in the current depths array (kmt)
c (as produced by the program makekm.f)
c
      inquire(file='sardepths21',exist=around)
      if(around) then
       open(unit=53,file='sardepths21',status='old',form='UNFORMATTED')
      else
       write(6,*) 'MOM depths file: sardepths21, not found'
       write(6,*) '...stopping'
       stop
      endif
c
       read(53) rkmt
      close(unit=53)
         do 13 j=1,jmt
         do 13 i=1,imt
```

```
#ifndef openbc
          if(j.eq.jmt) rkmt(i,j)=0.0
          if(i.eq.1.and.j.eq.jmt) write(stdout,*)
     +                      'Northern boundary closed'
#else
          if(j.eq.jmt) rkmt(i,j)=rkmt(i,j-1)
#endif
          kmt(i,j)=rkmt(i,j)+0.5
  1      continue
c
c open original (or potential temperature) levitus files
c
      inquire(file='potemp.levi',exist=potential)
      if(.not.potential) then
c
       open(unit=8,file='temp.levi',status='old',iostat=istat)
       if (istat.ne.0) then
         write (6,14) 'Levitus temperature',istat
         stop
       endif
c
       write(6,*) 'Working from original Levitus temperature file'
       write(6,*)
     + 'Do you wish to create a potential temperature version? (y/n)'
       read(5,'(a)') ans
       if(ans.eq.'y'.or.ans.eq.'Y') create=.true.
c
      else
c
       open(unit=8,file='potemp.levi',status='old',iostat=istat)
       if (istat.ne.0) then
         write (6,14) 'Levitus potential temperature',istat
         stop
       endif
c
      endif
c
      open(unit=9,file='salin.levi',status='old',iostat=istat)
      if (istat.ne.0) then
        write (6,14) 'Levitus salinity',istat
        stop
      endif
c
c open new levitus files
c first the files for ascout slices:
      open(unit=10,file='l1temp')
      open(unit=11,file='l1salin')
      call header(10,'temperature','depth',1,imtm2,1,jmt,1,
     +               levk,'CD','FA Levitus temperature annual mean')
      call header(11,'salinity','depth',1,imtm2,1,jmt,1,
     +                  levk,'CD','FA Levitus salinity annual mean')
c
c And then direct access files for 'j-slab' output
```

```
c
      open(unit=12,file='dalevt21',access='direct',recl=npsla)
      open(unit=13,file='dalevs21',access='direct',recl=npsla)
      if(create) open(unit=14,file='potemp.levi')
c
14    format(1x,A,' file open error number ',i5)
c
c For each mom level decide which two Levitus levels bracket the level
c and calculate vertical interpolation factors
       do 555 levk=1,km
       do 665 kLev=1,33
        if(levdepth(kLev).le.dmom(levk).and.
     +     levdepth(kLev+1).ge.dmom(levk)) then
        lup(levk)=kLev
        llo(levk)=kLev+1
        thic(levk)=levdepth(llo(levk))-levdepth(lup(levk))
        frac1(levk)=(dmom(levk)-levdepth(lup(levk)))/thic(levk)
        frac2(levk)=(levdepth(llo(levk))-dmom(levk))/thic(levk)
        goto 555
       endif
665   continue
555   continue
c
c Decide on the number of passes through the Levitus data; dependent
c upon memory restrictions it may be necessary to carry out the
c interpolation process in several passes through the Levitus dataset.
      if(mod(km,mxl).eq.0) then
       nsub=km/mxl
      else
       nsub=km/mxl + 1
      endif
c
      do 669 nmem=1,nsub
c
19    ioerror=.false.
      rewind 8
      rewind 9
c
c
c
c Initialise data on current horizontal slab (since land points are
c excluded from the Levitus data sets then set all points initially to
c the land mask value).  Use the kmL array to hold a land/sea indicator.
c
      do 4 m=1,mxl
      do 4 j=1,180
      do 4 i=1,360
       levtemp(i,j,m)=vmask(1)
       levsali(i,j,m)=vmask(1)
       kmL(i,j,m)=0
4     continue
c
c read in text lines
c
```

```
       do 5 i=1,3
         read (8,*)
         read (9,*)
5      continue
c
       if(create) then
       write(14,*)
      +'ANNUAL MEAN POTENTIAL TEMPERATURE ANALYSES FROM LEVITUS'
       write(14,*)
      +'UNITS OF TEMPERATURE ARE DEGREES CENTIGRADE AFTER UNPACKING.'
       write(14,*)
      +'10.1000. Temperatures converted using routines z2pb and ptmp83a'
       endif
c
c
c read through all longitudes, latitudes
c
c
c read in first line of next set of records from original temp file
c
333       read (8,15,err=10,end=10) trecrd(1:55),n,lon,lat
          read (9,15,err=10,end=10) srecrd(1:55),n1,lon1,lat1
c
c first check to see if final row of 9's has been reached
          if(n.eq.99999.and.n1.eq.99999) goto 10
c
c else check validity and common location of data
c
          if(n.ne.1.or.n1.ne.1) then
           write(6,*) 'Input data misplaced '
           ioerror=.true.
          endif
          if(lon.ne.lon1) then
           write(6,*) 'Temperature and salinity data mismatched, '
           ioerror=.true.
          endif
          if(lat.ne.lat1) then
           write(6,*) 'Temperature and salinity data mismatched, '
           ioerror=.true.
          endif
          if(ioerror) then
           write(6,*) 'Current temperature station: ',n,lon,lat
           write(6,*) 'Current salinity station:    ',n1,lon1,lat1
           stop
          endif
c
c
c read in remaining temp/salinity data for this lon,lat
c
          read (8,15,err=10,end=10) trecrd(56:110),n,lon,lat
          read (8,15,err=10,end=10) trecrd(111:165),n,lon,lat
          read (9,15,err=10,end=10) srecrd(56:110),n,lon,lat
          read (9,15,err=10,end=10) srecrd(111:165),n,lon,lat
c
```

```
c Unpack data and scale back to true figures
c
        read(trecrd,'(33i5)') (tcol(kcol),kcol=1,33)
        read(srecrd,'(33i5)') (scol(kcol),kcol=1,33)
        do 6 kcol=1,33
c
         if(scol(kcol).eq.0) then
          rscol(kcol)=vmask(2)
         else
          rscol(kcol)=(scol(kcol)-roff)*factor
         endif
c
         if(tcol(kcol).eq.0) then
          rtcol(kcol)=vmask(2)
         else
          rtcol(kcol)=(tcol(kcol)-roff)*factor
c
         if(.not.potential) then
c Convert Levitus depths (m) to pressure (dB)
c
          xlat=lat-90
          rpcol=real(z2pb(dble(levdepth(kcol)),dble(xlat)))
c
c Convert "in-situ" temperatures to potential temperatures:
c (by whichever method was selected at compilation)
#ifdef pottem
          rtcol(kcol)=real(pottem(dble(rtcol(kcol)),
     +                dble(rscol(kcol)),dble(rpcol),0d0,1d0))
#else
          rtcol(kcol)=real(ptmp83a(dble(rpcol),dble(rtcol(kcol)),
     +                        dble(rscol(kcol)),0d0))
#endif
c
          if(create) tcol(kcol)=rtcol(kcol)/factor + roff
         endif
         endif
6       continue
c
       if(create) then
        write(trecrd,'(33i5)') (tcol(kcol),kcol=1,33)
        write (14,15) trecrd(1:55),1,lon,lat
        write (14,15) trecrd(56:110),2,lon,lat
        write (14,15) trecrd(111:165),3,lon,lat
       endif
c
       do 222 m=1,mxl
        levk=(nmem-1)*mxl + m
        if(levk.gt.km) goto 222
c
c Perform vertical interpolation and store required slab (first check if
c lower point is Levitus land ; if it is then leave point undefined)
         if(abs(rtcol(llo(levk))-vmask(2)).le.1.e-4) then
          kmL(lon,lat,m)=0
```

```
          else
           levtemp(lon,lat,m)=rtcol(lup(levk))*frac2(levk)
     +                      + rtcol(llo(levk))*frac1(levk)
           levsali(lon,lat,m)=rscol(lup(levk))*frac2(levk)
     +                      + rscol(llo(levk))*frac1(levk)
           kmL(lon,lat,m)=1
          endif
          if(abs(rtcol(lup(levk))-vmask(2)).le.1.e-4) kmL(lon,lat,m)=0
          if(abs(rtcol(lup(levk))-vmask(1)).le.1.e-4) kmL(lon,lat,m)=0
c
222    continue
        goto 333
10     continue
c
c Perform horizontal interpolation
c
       israd=5
       do 223 m=1,mxl
        levk=(nmem-1)*mxl + m
        if(levk.gt.km) goto 223
        write(6,*) '----------------------------------------'
        write(6,*) 'Level: ',levk
        write(6,*) '----------------------------------------'
c
c Intialise mom grid land mask for this depth:
       nland=0
       do 866 i=1,imt
       do 866 j=1,jmt
        imoml(i,j)=1
        imoml2(i,j)=1
        if(kmt(i,j).lt.levk) then
         imoml(i,j)=0
         imoml2(i,j)=0
         nland=nland+1
        endif
866    continue
c
       do 900 im=1,imtm2
       xsm=slonmom+(im-1)*dxmom
c
c Find surrounding Levitus points
c
       do 910 il=1,359
        xsl=slonLev+(il-1)*dxLev
        xsr=xsl+dxLev
        if(xsl.le.xsm.and.xsr.ge.xsm) goto 800
910    continue
       write(6,*) 'Bracketing Levitus column not found'
       write(6,*) 'im= ',im
       stop
c
800    do 900 jm=1,jmt
c
```

```
c if land at this mom-pt then look no further
      if(kmt(im,jm).lt.levk) then
       tmom(im,jm)=0.0
       smom(im,jm)=0.0
       goto 899
      endif
c
      ysm=slatmom+(jm-1)*dymom
c
c Find surrounding Levitus points
c
      do 920 jl=1,179
       ysl=slatLev+(jl-1)*dyLev
       ysr=ysl+dyLev
       if(ysl.le.ysm.and.ysr.ge.ysm) goto 810
920   continue
      write(6,*) 'Bracketing Levitus row not found'
      write(6,*) 'jm= ',jm
      stop
c
810   continue
c
c check if land at any of the corner nodes
c
       ilp=il+1
       jlp=jl+1
       nsea=kmL(il,jlp,m)+kmL(ilp,jlp,m)+kmL(il,jl,m)+kmL(ilp,jl,m)
c
c If two or fewer corner nodes are land then leave equal to the previous
c layer but set land flag in order to attempt horizontal interpolation
c later.
      if(nsea.le.2) then
           imoml2(im,jm)=0
c
c
c else if all 4 corner nodes are sea perform standard linear
c interpolation
c
         elseif(nsea.eq.4) then
          a=(xsm-xsl)/dxLev
          b=(ysm-ysl)/dyLev
          tmom(im,jm)=(1.-a)*(1.-b)*levtemp(il,jl,m)+
     +           a*(1.-b)*levtemp(ilp,jl,m)+
     +           a*b*levtemp(ilp,jlp,m)+
     +           (1.-a)*b*levtemp(il,jlp,m)
c
          smom(im,jm)=(1.-a)*(1.-b)*levsali(il,jl,m)+
     +           a*(1.-b)*levsali(ilp,jl,m)+
     +           a*b*levsali(ilp,jlp,m)+
     +           (1.-a)*b*levsali(il,jlp,m)
c
c else if 3 corner nodes are sea take an average value
c
```

```
            elseif(nsea.eq.3) then
             tmom(im,jm) = (kmL(il,jl,m)*levtemp(il,jl,m)
     +                    + kmL(ilp,jl,m)*levtemp(ilp,jl,m)+
     +                      kmL(il,jlp,m)*levtemp(il,jlp,m)+
     +                      kmL(ilp,jlp,m)*levtemp(ilp,jlp,m))/nsea
c
             smom(im,jm) = (kmL(il,jl,m)*levsali(il,jl,m)
     +                    + kmL(ilp,jl,m)*levsali(ilp,jl,m)+
     +                      kmL(il,jlp,m)*levsali(il,jlp,m)+
     +                      kmL(ilp,jlp,m)*levsali(ilp,jlp,m))/nsea
            endif
899         if(kmt(im,jm).eq.0) then
             tmom(im,jm)=vmask(1)
             smom(im,jm)=vmask(1)
            elseif(kmt(im,jm).lt.levk) then
             tmom(im,jm)=vmask(2)
             smom(im,jm)=vmask(2)
            endif
c
900         continue
c
#ifdef cyclic
      do 905 j=1,jmt
        tmom(imtm1,j) = tmom(1,j)
        smom(imtm1,j) = smom(1,j)
        imoml2(imtm1,j)=imoml2(1,j)
c
        tmom(imt,j) = tmom(2,j)
        smom(imt,j) = smom(2,j)
        imoml2(imt,j)=imoml2(2,j)
905   continue
#endif
c
c Now iterate to fill in any sea points which fell in Levitus land areas
c
       niter=1
       reiter=.false.
       nland2=0
       nland3=-1
c
746    do 747 j=1,jmt
       do 747 i=1,imt
        if(niter.eq.1) imoml3(i,j)=imoml2(i,j)
        if(imoml2(i,j).eq.0) nland2=nland2+1
747    continue
       if(nland2.ne.nland) then
c
       do 748 j=1,jmt
       do 748 i=2,imtm1
        if(imoml2(i,j).eq.0.and.imoml(i,j).ne.0) then
c
```

```
c Take average of all surrounding sea-points
c
        ip1=i+1
        im1=i-1
        jp1=min(j+1,jmt)
        jm1=max(j-1,1)
c
        nsurr=imoml2(im1,jm1)+imoml2(i,jm1)+imoml2(ip1,jm1)
      +       +imoml2(im1,j)                +imoml2(ip1,j)
      +       +imoml2(im1,jp1)+imoml2(i,jp1)+imoml2(ip1,jp1)
c
        if(nsurr.gt.0) then
        tmom(i,j)=(imoml2(im1,jm1)*tmom(im1,jm1)
      +           + imoml2(i,jm1)*tmom(i,jm1)
      +           + imoml2(ip1,jm1)*tmom(ip1,jm1)
      +           + imoml2(im1,j)*tmom(im1,j)
      +           + imoml2(ip1,j)*tmom(ip1,j)
      +           + imoml2(im1,jp1)*tmom(im1,jp1)
      +           + imoml2(i,jp1)*tmom(i,jp1)
      +           + imoml2(ip1,jp1)*tmom(ip1,jp1))/nsurr
c
        smom(i,j)=(imoml2(im1,jm1)*smom(im1,jm1)
      +           + imoml2(i,jm1)*smom(i,jm1)
      +           + imoml2(ip1,jm1)*smom(ip1,jm1)
      +           + imoml2(im1,j)*smom(im1,j)
      +           + imoml2(ip1,j)*smom(ip1,j)
      +           + imoml2(im1,jp1)*smom(im1,jp1)
      +           + imoml2(i,jp1)*smom(i,jp1)
      +           + imoml2(ip1,jp1)*smom(ip1,jp1))/nsurr
         imoml3(i,j)=1
        else
         reiter=.true.
        endif
c
        endif
748     continue
#ifdef cyclic
       do 750 j=1,jmt
        tmom(1,j) = tmom(imtm1,j)
        tmom(imt,j) = tmom(2,j)
        smom(1,j) = smom(imtm1,j)
        smom(imt,j) = smom(2,j)
        imoml3(1,j) = imoml3(imtm1,j)
        imoml3(imt,j)= imoml3(2,j)
750    continue
#endif
c
        endif
c
        if(reiter) then
        niter=niter+1
        nland3=0
        reiter=.false.
        write(6,*) 'Iteration number= ',niter
```

```
        do 749 j=1,jmt
        do 749 i=1,imt
         imoml2(i,j)=imoml3(i,j)
         if(imoml2(i,j).eq.0) nland3=nland3+1
749     continue
        if(nland3.ne.nland2) then
        nland2=0
        if(niter.lt.50) goto 746
        else
c
        write(6,*) 'Filling in isolated basins'
        endif
        endif
c
        call ascout0(tmom,imt,imtm2,jmt,vmask,2,10)
        call ascout0(smom,imt,imtm2,jmt,vmask,2,11)
c
c Write out slab in such a way that MOM type latitude slabs can be
c retrieved later.
c Note Salinity is scaled ready for use in the MOM model
c
      do 123 j=1,jmt
       if(levk.ne.1) then
        read(12,rec=j) tsla
        read(13,rec=j) ssla
       endif
       do 124 i=1,imt
        tsla(i,levk)=tmom(i,j)
        ssla(i,levk)=(smom(i,j)-35.)*1.e-3
124    continue
       write(12,rec=j) tsla
       write(13,rec=j) ssla
123    continue

223   continue
669   continue
      15    format(a55,3i5)
      close (8)
      close (9)
      close (10)
      close (11)
      close (12)
      close (13)
      if(create) close (14)
      stop
      end
#ifdef pottem
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      real*8 function pottem(tt,ss,p0,p1,dpp)
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c
c Subroutine to calculate the final temperature of water moved
```

```
c adiabatically from an initial temperature tt, salinity ss and pressure
c p0, to a final pressure p1.
c
c The integral equation is solved by direct integration with a pressure
c increment dpp - using the bryden equation for the adiabatic lapse rate
c (subroutine atg).
c
c  t      = surface (potential) temperature in degrees centigrade
c  s      = salinity in nsu
c  p0     = initial pressure in decibars
c  p1     = final pressure in decibars
c  dpp    = pressure step to use
c  pottem = result in degrees centigrade
c
c tests with dpp values ranging from 1 to 128 decibars showed the most
c accurate results were obtained with dpp equal to 1.
c
      implicit real*8 (a-h,o-z)
c
      if(p0.lt.0d0.or.p0.gt.20000.0
     &   .or.p1.lt.0d0.or.p1.gt.20000.0)then
        print *,' subroutine pottem stopping - pressures out of range'
        print *,' pressures p0 and p1 = ',p0,p1
        print *,' allowed range has min of 0.0, max of 20,000'
        stop
      endif
c
      dp = sign(dpp,p1-p0)
      p  = p0
      t  = tt
      tb = t  - atg(p0,t,ss)*dp
c
 10   ta = tb + 2d0*atg(p,t,ss)*dp
      p  = p  + dp
      tb = t
      t  = ta
      test=(p-p1)*(p-dp-p1)
      if(test.gt.0d0)goto 10
      pottem = ((p1-p+dp)*t + (p-p1)*tb)/dp
      return
      end
c--------------------------------------------------------------------------
c--------------------------------------------------------------------------
      double precision function atg(p,t,s)
c--------------------------------------------------------------------------
c--------------------------------------------------------------------------
c
c  adiabatic temperature gradient deg c per decibar
c  ref: bryden,h.,1973,deep-sea res.,20,401-408
c  units:
c        pressure       p        decibars
c        temperature    t        deg celcius (ipts-68)
c        salinity       s        (pss-78)
c        adiabatic      atg      degrees celcius per decibar
```

```
c  check value: atg=3.255976e-4 deg c/dbar
c  for s=40 (pss-78), t=40 deg c, p=10000 decibars
c
      implicit real*8(a-h,o-z)
c
      ds=s-35d0
      atg=(((-2.1687d-16*t+1.8676d-14)*t-4.6206d-13)*p
     &+((2.7759d-12*t-1.1351d-10)*ds+((-5.4481d-14*t
     &+8.733d-12)*t-6.7795d-10)*t+1.8741d-8))*p
     &+(-4.2393d-8*t+1.8932d-6)*ds
     &+((6.6228d-10*t-6.836d-8)*t+8.5258d-6)*t+3.5803d-5
      return
      end
#endif
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      function dpth80 (pin,xlat)
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      implicit double precision(a-h,o-z)
c......
c......*************************************
c......*** d s collins  ios(w)  7-may-81 ***
c......*************************************
c......
c      depth in meters from pressure in decibars using Saunder's and
c      Fofonoff's method.  Deep Sea Res., 1976,23,109-111.
c      formula refitted for eos80
c      check value: 9712.654 m for pin=10000 decibars,latitude = 30 deg.
c
c..... convert pressure to bars
      p=pin*0.1
      x=sin(xlat/57.29578d0)
      x=x*x
      gr=9.780318d0*(1.0+(5.2788d-3+2.36d-5*x)*x)+1.092d-5*p
      dpth80=(((-1.82d-11*p+2.279d-7)*p-2.2512d-3)*p+97.2659)*p
      dpth80=dpth80/gr
      return
      end
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      real*8 function z2pb(z,xlat)
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      implicit real*8(a-h,o-z)
c
c  Function to calculate pressure in decibars from depth in metres using
c  an'exact' iterative inverse of saunders and fofonoff's algorithm
c  (routine dpth80).  Iterates until convergence or 30 iterations reach
c  convergence criteria are zero error or a two point limit cycle.
c  Error exit if final error > eps.
c
      data eps/1d-6/
c
```

```
      p=z
      zz=-999.0
      z1=-999.0
      do 20 i=1,30
      z2=z1
      z1=zz
      zz=dpth80(p,xlat)
      if(z.eq.zz.or.(abs(z-zz).lt.eps.and.zz.eq.z2))goto 50
      p =p+z-zz
   20 continue
      if(abs(z-zz).lt.eps)goto 50
c
      print *,'subroutine z2pb.  iteration has not converged after',
     &        ' 30 iterations'
      print *,'object depth =',z,' last three estimates are:'
      print *,'iteration      depth                depth error'
      print *,28,z2,z-z2
      print *,29,z1,z-z1
      print *,30,zz,z-zz
      stop
c
   50 z2pb=p
      return
      end
#ifndef pottem
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      double precision function ptmp83a(p0,t0,s,pr)
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      implicit double precision (a-h,o-z)
c
c..... to compute local potential temperature at pr.  Using Bryden 1973
c..... polynomial for adiabatic lapse rate and Runge-Kutta 4-th order
c..... integration algorithm.  Ref: Bryden,H.,1973, Deep-Sea Res., 20,
c..... 401-408.  Fofonoff,N.,1977, Deep-Sea Res., 24, 489-491.
c..... Check value: ptmp83 =36.89072 for s=40 nsu,t=40 deg c, p0
c..... (measured pressure) = 10000 decibars, pr = 0 bars.
c
c  This has been modified so that the constants are calculated on first
c  entry to full precision.  D.J.Webb, Jan 1992.
c
      data in/0/
      if(in.eq.0)then
        c1=0.5d0
        c2=dsqrt(0.5d0)
        c3=dsqrt(2d0)
        c4=1d0-c2
        c5=1d0+c2
        c6=1d0/6d0
        c7=2d0-c3
        c8=2d0+c3
        c9 =-2d0+3d0*c2
        c10=-2d0-3d0*c2
```

```
          in=1
        endif
        save c1,c4,c5,c6,c7,c8,c9,c10,in
        p=p0
        t=t0
        h=pr-p
        xk=h*atgr83(p,t,s)
        t=t+c1*xk
        q=xk
        p=p+c1*h
        xk=h*atgr83(p,t,s)
        t=t+c4*(xk-q)
        q=c7*xk+c9*q
        xk=h*atgr83(p,t,s)
        t=t+c5*(xk-q)
        q=c8*xk+c10*q
        p=p+c1*h
        xk=h*atgr83(p,t,s)
        ptmp83a=t+(xk-2d0*q)*c6
        return
        end
        real*8 function atgr83(pin,t,s)
        implicit real*8(a-h,o-z)
c.....
c
c......Adiabatic temperature gradient deg c/bar
c......Ref: Bryden,H.,1973, Deep-Sea Res., 20, 401-408
c......Check value: atgr80=3.255976e-3 for s=40 nsu,t=40 deg c,
c......pin=10000 decibars
c
c..... convert pressure to bars
       p=pin*0.1
       ds=s-35d0
       atgr83=(((-2.1687d-13*t+1.8676d-11)*t-4.6206d-10)*p
      &+((2.7759d-10*t-1.1351d-8)*ds+((-5.4481d-12*t
      &+8.733d-10)*t-6.7795d-8)*t+1.8741d-6))*p
      &+(-4.2393d-7*t+1.8932d-5)*ds
      &+((6.6228d-9*t-6.836d-7)*t+8.5258d-5)*t+3.5803d-4
c
c......as from 19 july 1983, gradient is per decibar, not per bar
       atgr83=atgr83*0.1
       return
       end
#endif
#define noreport 1
#include "../setgrid.F"
#include "../header.F"
#include "../blkdta.F"
#include "../ascout0.f"
```

APPENDIX D (v)

```
      program  getslice
      ******************
c
c Program to retrieve standard direction slices from MOM restart data-
c sets.  Output is in the form of ascout cards files suitable for
c viewing via the FRAM plotting programs.  Many of the modules used in
c the construction of this program are taken directly from the MOM
c source code.  This should mean that any reconfiguring of the ocean
c model will be automatically accounted for by simply re-compiling this
c extraction program with the same preprocessor directives.
c
c e.g.: cc -P -Dopenbc -Ddiskless etc. getslice.F
c       mv getslice.i getslice.f
c       f77 -o getslice getslice.f
c
c will produce a code capable of correctly extracting slices from a
c restart dataset produced by running the model in core with an open
c northern boundary.
c
c Additional preprocessor directives unique to this program are:
c vtsteps   : used to calculate the model day from the timestep
c             according to a set variation in the length of the timestep
c             (e.g. if the timestep has been changed part-way through a
c             run).
c
#include "param.h"
#include "ctmngr.h"
#include "emode.h"
#include "iounit.h"
#include "levind.h"
#include "grdvar.h"
#include "coord.h"
#include "scalar.h"
#ifdef multitasking
#include "cshrbf.h"
#else
#include "slabs.h"
#endif
      real islice(jmt,km),jslice(imt,km),kslice(imt,jmt)
      real vmask(4)
      integer tpts
      logical around
      character*50 namrun,rstrtfn
      character*9 vars(6),dims(4)
      character*2 fnames(6),opform
      character*1 orien(3),dayno*4,fname*7,ans
      external blkdta
c
```

```
c set up character strings for the header subroutine
      data vars/'temperature','salinity','uvelocity','vvelocity',
     +          'stream','depths'/dims/'latitude','longitude','depth',
     +          'stream'/
      data fnames/'mt','ms','mu','mv','mp','md'/orien/'e','n','h'/
c
c Load the values used for masking in ascout
      data vmask/1.e7,2.e7,3.e7,4.e7/
c
c Load some strings used by the header routine
      data namrun/'Fasham model 2x1 resolution'/opform/'CD'/
c
      pi     = c4*atan(c1)
      radian = c360/(c2*pi)
      omega  = pi/43082.0
c
      itry=1
      tpts=1
c
#ifdef vtsteps
c if variable length timesteps have been used the flag the present
c settings:
      write(6,*) 'Extraction program assuming varying timesteps'
      write(6,*) 'Current settings are: '
      write(6,*) 'days 0 to 14   : 20 minutes'
      write(6,*) 'days 14+ to 28: 60 minutes'
      write(6,*) 'days 28+       : 90 minutes'
#else
c Set the number of time steps per day
      tperday=24
      write(6,*) 'Extraction program assuming 24 timesteps per day'
#endif
      write(6,*) '-------------------------------------------------'
 5    write(6,*) 'Enter filename of restart dataset: '
      read(5,'(A)') rstrtfn
c
        inquire(file=rstrtfn,exist=around)
        if(.not.around) then
         itry=itry+1
         write(6,*) 'File not found, Please try again or enter quit'
         if(rstrtfn(1:4).eq.'quit') then
           write(6,*) 'User requested exit'
           stop
         endif
         if(itry.gt.5) then
          write(6,*) 'Repeated filename failure...program halted'
          stop
         endif
         goto 5
        endif
c
      write(6,*) 'Enter choice of variable: '
      write(6,*) '(1) Temperature '
      write(6,*) '(2) Salinity '
```

```
      write(6,*) '(3) U-velocity'
      write(6,*) '(4) V-velocity'
      write(6,*) '(5) Stream-function'
      write(6,*) '(6) Topography '
c
      ilim=6
#ifdef openbc
      write(6,*) '(7) Reference temperature along open boundary'
      write(6,*) '(8) Reference salinity along open boundary'
      ilim=8
#endif
c
      read(5,*) itype
      if(itype.lt.1.or.itype.gt.ilim) then
       write(6,*) 'Illegal variable choice ...defaulting to type 1'
       itype=1
      endif
      if(itype.eq.3.or.itype.eq.4) tpts=0
c
      IF (itype.lt.5) then
       write(6,*) 'Enter choice of slice direction: '
       write(6,*) '(1) E-W slice'
       write(6,*) '(2) N-S slice'
       write(6,*) '(3) Horizontal slice'
       read(5,*) ichoice
       if(ichoice.lt.1.or.ichoice.gt.3) then
        write(6,*) 'Illegal direction choice ...defaulting to choice 3'
        ichoice=3
       endif
       if(ichoice.eq.1) then
        write(6,*) 'Enter jrow of slice: '
        read(5,*) jrow
        if(jrow.lt.1.or.jrow.gt.jmt) then
          write(6,*) 'Illegal row choice ...defaulting to jmt/2'
          jrow=jmt/2
        endif
       elseif(ichoice.eq.2) then
        write(6,*) 'Enter irow of slice: '
        read(5,*) irow
        if(irow.lt.1.or.irow.gt.imt) then
          write(6,*) 'Illegal column choice ...defaulting to imt/2'
          irow=imt/2
        endif
        jrow=jmt
       else
        write(6,*) 'Enter k-level: '
        read(5,*) klevel
        if(klevel.lt.1.or.klevel.gt.km) then
          write(6,*) 'Illegal level choice ...defaulting to k=1'
          klevel=1
        endif
        jrow=jmt
      endif
c
```

```
      ELSEIF(itype.eq.5.or.itype.eq.6) then
c
       ichoice=3
       klevel=1
c
      ELSE
       ichoice=1
       jrow=jmt
c
      ENDIF
c
c Call the MOM routine grids to set up the grid.  The version actually
c included with this program is identical to the original grids routine
c (MOM_1.0) but does not detail the grid arrays on stdout.
c If setgrid.F has been altered since MOM_1.0 then update the version
c included below.
      call grids
c
#ifdef openbc
      nkntrl=2+imt*km*nt
      call ostart (kontrl, nkntrl, nkntrl, 1)
#else
      call ostart (kontrl, 2, 2, 1)
#endif
      call ostart (kflds, nkflds*nwds, nwds, 1)
      call ostart (labs(1), jmt*nslab, nslab, nbuf)
      call ostart (labs(2), jmt*nslab, nslab, nbuf)
      call ostart (labs(3), jmt*nslab, nslab, nbuf)
c
      open(iorest,file=rstrtfn,access='SEQUENTIAL',form='UNFORMATTED')
c
c Read the restart dataset using rdrest from the standard MOM module:
restio.F
      call rdrest
      call oget(kontrl,2,1,itt)
c
c-----------------------------------------------------------------------
c     compute permuting disc indicators and read in 2 levels of
c     stream function.
c-----------------------------------------------------------------------
c
#ifdef diskless
#  ifdef multitasking
        ndiskb = mod(itt+2,ntlev) + 1
        ndisk  = mod(itt  ,ntlev) + 1
        ndiska = mod(itt+1,ntlev) + 1
#  else
        ndiskb = mod(itt+1,ntlev) + 1
        ndisk  = mod(itt  ,ntlev) + 1
        ndiska = ndiskb
#  endif
#else
        ndiskb = mod(itt+2,ntlev) + 1
        ndisk  = mod(itt  ,ntlev) + 1
```

```
        ndiska = mod(itt+1,ntlev) + 1
#endif
c
c
c Retrieve stream-fn and topography arrays
      call oget (kflds, nwds, (ndisk-1)*nwds+1, p(1,1,2))
      call oget (kflds, nwds, (ndiska-1)*nwds+1, p(1,1,1))
c
      call oget(kflds,nwds,5*nwds+1,kmt)
c
c-----------------------------------------------------------------------
c     compute number of vertical levels on the "u" grid
c-----------------------------------------------------------------------
c
      do 800 j=1,jmt
        kmu(imt,j) = 0
800   continue
c
      do 900 i=1,imt
        kmu(i,jmt) = 0
900   continue
c
      do 1000 j=1,jmtm1
        do 990 i=1,imtm1
         kmu(i,j) = min (kmt(i,j), kmt(i+1,j), kmt(i,j+1), kmt(i+1,j+1))
990     continue
1000  continue
#ifdef openbc
c
c     set open boundary topographic conditions
c
      do 737 i=1,imtm1
       kmu(i,jmt)=kmu(i,jmtm1)
 737  continue
#endif
#ifdef cyclic
c
c     set cyclic conditions
c
      do 1100 j=1,jmt
        kmu(imt,j) = kmu(2,j)
1100  continue
#endif
#ifdef symmetry
c
c     set symmetry conditions
c
      do 1200 i=1,imt
        kmu(i,jmt) = kmu(i,jmtm2)
1200  continue
#endif
c
```

```
c-----------------------------------------------------------------------
c     compute depths and reciprocal depths
c-----------------------------------------------------------------------
c
      do 1400 j=1,jmt
        do 1390 i=1,imt
          hr(i,j) = c0
          h(i,j)  = c0
          if (kmu(i,j) .ne. 0) then
            hr(i,j) = c1/zw(kmu(i,j))
            h (i,j) = zw(kmu(i,j))
          endif
1390    continue
1400  continue

c
c
c Makeup filename:
#ifdef vtsteps
      if(itt.le.1008) then
       NSLA=itt/72
      elseif(itt.gt.1008.and.itt.le.1334) then
       NSLA=14 + (itt - 1008)/24
      else
       NSLA=28 + (itt - 1334)/16
      endif
#else
      NSLA=itt/tperday
#endif
      write(6,*) 'Data extracted from timestep: ', itt, ' day: ',NSLA
      write(dayno,'(i4.4)') NSLA
#ifdef openbc
      if(itype.eq.7.or.itype.eq.8) then
      itrace=itype-6
      fname=fnames(itrace)//orien(ichoice)//dayno
      else
#endif
      fname=fnames(itype)//orien(ichoice)//dayno
#ifdef openbc
      endif
#endif
c
      nc=mod(itt,ntau)+1
      nm=mod(itt-1,ntau)+1
      jc=mod(jrow,nslabs)+1
c
c Check validity of output filename and request permission to
c overwrite if necessary:
      inquire(file=fname//'.cards',exist=around)
      if(around) then
       write(6,*) 'Output file: '//fname//'.cards already exists'
       write(6,*) 'Ok to overwrite? (y/n)'
       read(5,'(A)') ans
```

```
        if(ans.eq.'y'.or.ans.eq.'Y') then
         open(unit=35,file=fname//'.cards')
c
        else
c
c Search for an unused filename by appending single characters to the
c existing filename:
         write(6,*) 'Searching for alternative name...'
         do 333 ich=ichar('a'),ichar('z')
          inquire(file=fname//char(ich)//'.cards',exist=around)
          if(.not.around) goto 444
          write(6,*) fname//char(ich)//'.cards exists'
 333     continue
         write(6,*) 'Alternative name not found...stopping'
         stop
 444     write(6,*) 'Using file: '//fname//char(ich)//'.cards'
         open(unit=35,file=fname//char(ich)//'.cards')
        endif
       else
        open(unit=35,file=fname//'.cards')
       endif
c
c
c-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
c If Stream-function then mask,scale and output here:
c
      if(itype.eq.5) then
      do 100 j=1,jmt
        do 90 i=1,imt
          kslice(i,j)=p(i,j,1)*1.E-12
          if (kmt(i,j) .le. 1) then
            kslice(i,j) = vmask(1)
          endif
90      continue
100   continue
      call header(35,vars(itype),dims(4),1,imtm2,1,jmt,tpts,
     +                     0,opform,namrun)
      call ascout0(kslice,imt,imtm2,jmt,vmask,2,35)
      stop
      endif
c-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
c-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
c If Topography then mask,scale and output here:
c
      if(itype.eq.6) then
      do 155 j=1,jmt
        do 955 i=1,imt
          kslice(i,j)=kmt(i,j)
          if (kmt(i,j) .le. 1) then
            kslice(i,j) = vmask(1)
          endif
955      continue
155   continue
```

```
      call header(35,vars(itype),dims(4),1,imtm2,1,jmt,tpts,
     +                  0,opform,namrun)
      call ascout0(kslice,imt,imtm2,jmt,vmask,2,35)
      stop
      endif
c-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
#ifdef openbc
c==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==
c If tracer on open northern boundary then mask and output here:
c
      call oget(kontrl,imt*km*nt,3,tn(1,1,1))
      if(itype.eq.7.or.itype.eq.8) then
       do 555 k=1,km
       do 555 i=1,imt
        jslice(i,k)=tn(i,k,itrace)
       if (kmt(i,jrow).le.1) then
        jslice(i,k)=vmask(1)
       elseif (kmt(i,jrow).lt.k) then
        jslice(i,k)=vmask(2)
       endif
 555   continue
       call header(35,vars(itrace),dims(ichoice),1,imtm2,1,km,tpts,
     +                  jrow,opform,namrun)
       call ascout0(jslice,imt,imtm2,km,vmask,2,35)
       stop
       endif
c==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==
c
#endif

c For Horizontal and N-S slices loop through whole dataset:
c-------------------------------------------------------------
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      if(ichoice.ge.2) then
       do 221 j=1,jrow
       jc=mod(j,nslabs)+1
c
c Retrieve slab information using routine getrow.  This routine is
c similar to the MOM module getvar (included as part of step.F) but
c includes additions to mask the topography ready for ascout.
c
       call getrow (j,jc,itype, vmask)
c
       if(ichoice.eq.3) then
        do 121 i=1,imt
          if    (itype.le.nt) then
           kslice(i,j)=t(i,klevel,jc,nc,itype)
          elseif(itype.eq.nt+1) then
           kslice(i,j)=u(i,klevel,jc,nc)
          elseif(itype.eq.nt+2) then
           kslice(i,j)=v(i,klevel,jc,nc)
          endif
 121    continue
c
```

```
         elseif (ichoice.eq.2) then
          do 321 k=1,km
          if    (itype.le.nt) then
           islice(j,k)=t(irow,k,jc,nc,itype)
           elseif(itype.eq.nt+1) then
            islice(j,k)=u(irow,k,jc,nc)
           elseif(itype.eq.nt+2) then
            islice(j,k)=v(irow,k,jc,nc)
           endif
 321      continue
         endif
c
 221   continue
c
       if(ichoice.eq.3) then
        call header(35,vars(itype),dims(ichoice),1,imtm2,1,jmt,tpts,
     +                  klevel,opform,namrun)
        call ascout0(kslice,imt,imtm2,jmt,vmask,2,35)
       elseif(ichoice.eq.2) then
        call header(35,vars(itype),dims(ichoice),1,jmt,1,km,tpts,
     +                  irow,opform,namrun)
        call ascout0(islice,jmt,jmt,km,vmask,2,35)
       endif
c
c-----------------------------------------
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
c Else select only the required E-W slice:
c-----------------------------------------
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      elseif(ichoice.eq.1) then
        call getrow (jrow,jc,itype,vmask)
c
        do 222 k=1,km
        do 222 i=1,imt
         if(itype.le.nt) then
          jslice(i,k)=t(i,k,jc,nc,itype)
         elseif (itype.eq.nt+1) then
          jslice(i,k)=u(i,k,jc,nc)
         elseif (itype.eq.nt+2) then
          jslice(i,k)=v(i,k,jc,nc)
         endif
 222    continue
c
        call header(35,vars(itype),dims(ichoice),1,imtm2,1,km,tpts,
     +                  jrow,opform,namrun)
        call ascout0(jslice,imt,imtm2,km,vmask,2,35)
      endif
c-----------------------------------------
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      close (unit=35)
c
      end
      subroutine getrow (jrowt, jptr ,itype, vmask)
```

```
#ifdef multitasking
cfpp$ noconcur r
#endif
c
c=======================================================================
c
c     Read prognostic variables from row "jrowt" on disk into the memory
c     slab window at position "jptr". Set the masks for "jrowt" and save
c     a copy of the internal modes before constructing the full velocity.
c
c=======================================================================
c
#include "param.h"
#include "ctask.h"
#include "emode.h"
#include "grdvar.h"
#include "coord.h"
#include "scalar.h"
#include "iounit.h"
#include "levind.h"
#include "slabs.h"
#include "switch.h"
c
      dimension ubar(imt), vbar(imt)
      real vmask(4)
c
c     limit size of jrow
c
      jrow = min(jrowt,jmt)
c
c-----------------------------------------------------------------------
c     read slabs from row "jrow" on disk into the memory slab window at
c     row "jptr"
c-----------------------------------------------------------------------
c
      if (jrow .le. jmt) then
        if (mixts) then
          call oget (labs(ndisk), nslab, (jrow-1)*nslab+1, bufsl)
        else
          call oget (labs(ndiskb), nslab, (jrow-1)*nslab+1, bufsl)
        endif
        call xfer (bufsl, t(1,1,jptr,nm,1))
        call oget (labs(ndisk ), nslab, (jrow-1)*nslab+1, bufsl)
        call xfer (bufsl, t(1,1,jptr,nc,1))
      endif
c
c-----------------------------------------------------------------------
c     set masks for row "jrow"
c-----------------------------------------------------------------------
c
      do 100 k=1,km
        do 90 i=1,imt
c
```

```
c If Salinity then rescale
c
          if(itype.eq.2)
     +      t(i,k,jptr,nc,itype)=(t(i,k,jptr,nc,itype)+0.035)*1000.
c
          if(itype.le.2) then
             fm(i,k,jptr)=1.0
           if (kmt(i,jrow) .le. 1) then
             fm(i,k,jptr)=0.0
             t(i,k,jptr,nc,itype) = vmask(1)
           elseif (kmt(i,jrow).lt.k) then
             fm(i,k,jptr)=0.0
             t(i,k,jptr,nc,itype) = vmask(2)
           endif
          else
             gm(i,k,jptr)=1.0
           if (kmu(i,jrow) .le. 1) then
             gm(i,k,jptr)=0.0
             u(i,k,jptr,nc) = vmask(1)
             v(i,k,jptr,nc) = vmask(1)
           elseif (kmu(i,jrow).lt.k) then
             gm(i,k,jptr)=0.0
             u(i,k,jptr,nc) = vmask(2)
             v(i,k,jptr,nc) = vmask(2)
           endif
          endif
90       continue
100    continue
c
c-----------------------------------------------------------------------
c     set pointers (indices) to cycle internal modes
c     (also used to cycle del**2 quantities for biharmonic option)
c-----------------------------------------------------------------------
c
      jpt1 = mod (jrowt+1,numjpt) + 1
      jpt2 = mod (jrowt+2,numjpt) + 1
      jpt  = jpt2
#ifdef biharmonic
      jpt3 = mod (jrowt+3,numjpt) + 1
      jpt  = jpt3
#endif
c
      if(itype.lt.3) return
c
c-----------------------------------------------------------------------
c     save a copy of the internal mode velocity from row "jrow"
c     (row "jrow + 1" if the biharmonic option is enabled) for use in
c     constructing 'fvsu' in clinic and diagnostic caluclations
c-----------------------------------------------------------------------
c
      do 200 k=1,km
        do 190 i=1,imt
          uclin(i,k,jpt) = u(i,k,jptr,nc)
          vclin(i,k,jpt) = v(i,k,jptr,nc)
```

```
190     continue
200   continue
c
c-----------------------------------------------------------------------
c     add external mode velocity to internal mode velocity for row
c     row 'jsrow' (tau & tau-1)
c-----------------------------------------------------------------------
c
      if (jrow .le. jmtm1) then
        do 300 m=1,2
          if (m .eq. 1) then
            n = nc
          else
            n = nm
          endif
          do 270 i=1,imtm1
            diag1   = p(i+1,jrow+1,m) - p(i  ,jrow,m)
            diag0   = p(i  ,jrow+1,m) - p(i+1,jrow,m)
            ubar(i) = -(diag1+diag0)*dyu2r(jrow)*hr(i,jrow)
            vbar(i) =  (diag1-diag0)*dxu2r(i)*hr(i,jrow)*csur(jrow)
270       continue
#ifdef cyclic
c
c         set cyclic boundary conditions
c
          ubar(imt) = ubar(2)
          vbar(imt) = vbar(2)
#endif
          do 290 k=1,km
            do 280 i=1,imu
              u(i,k,jptr,n) = u(i,k,jptr,n) + ubar(i)*gm(i,k,jptr)
              v(i,k,jptr,n) = v(i,k,jptr,n) + vbar(i)*gm(i,k,jptr)
280         continue
290       continue
300     continue
      endif
c
      return
      end
      subroutine header(op,trac,depvar,ibase1,itop1,ibase2,itop2,tpts,
     +                  intlv,opform,namrun)
c
c Subroutine to produce a header for ascout files.
c
c In the current version the following assumptions are made:
c 1. The grid spacing is uniform in lat/long directions
c 2. The first longitudinal "t-point" is on the Greenwich meridian
c 3. stlat is the latitude of the southern-most "u-point"
c
c Input arguments are:
c
c name      type         description
c-----------------------------------
c op        integer      Output fortran unit
```

```
c trac        character    Variable type of output field
c                                     (e.g. Temperature)
c depvar      character    dimension held constant for this slice. Options
c                              are : latitude, longitude, depth or stream
c ibase1      integer      start column of output field
c itop1       integer      end column of output field
c ibase2      integer      start row of output field
c itop2       integer      end row of output field
c tpts        integer      flag to indicate whether output field is on
c                              t-points (1) or u-points (0)
c intlv       integer      Row, column or depth indicator (used for
                               annotation)
c opform      character    output form no longer used but included for
c                              backward compatibility
c namrun      character    text for inclusion in the header's comments
c                              field
c
#ifndef nomodel
#include "param.h"
#include "grdvar.h"
#include "scalar.h"
#include "coord.h"
#endif
      character trac*(*),depvar*9,opform*(*),namrun*(*)
      character*9 quan(3),from(3),incr(3),to(3)
      integer nop(3),op,tpts
      common /tstep/ ndfir,ndlas,ndinc
#ifdef nomodel
c -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
c If using header outside a MOM-code application it may be necessary
c to fill in the following definitions:
       real dxt(1),dyt(1)
       dtts=3600.
       ttsec=0000.
       itt=288
       xincr =2.0
       yincr =1.0
       stlat=-79.
c -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
#else
c
      xincr=dxt(1)*radian/radius
      yincr=dyt(1)*radian/radius
#endif
c convert input strings to upper-case
      call conv2up(trac)
      call conv2up(depvar)
      call conv2up(opform)
      call conv2up(namrun)
      rintlv=float(intlv)
c
      if(depvar(1:6).eq.'STREAM') then
        quan(1) = 'LONGITUDE'
        quan(2) = ' LATITUDE'
```

```
      write(from(1),'(f9.3)') (ibase1-1)*xincr +(1-tpts)*0.5*xincr
      write(from(2),'(f9.3)')
     +                    (ibase2-1)*yincr + tpts*0.5*yincr +stlat
      write(incr(1),'(f9.3)') xincr
      write(incr(2),'(f9.3)') yincr
      write(to(1),'(f9.3)')   (itop1-1)*xincr + (1-tpts)*0.5*xincr
      write(to(2),'(f9.3)')
     +                       (itop2-1)*yincr +tpts*0.5*yincr +stlat
      nop(1) = itop1-ibase1+1
      nop(2) = itop2-ibase2+1
C
C
      elseif(depvar(1:3).eq.'DEP') then
        quan(1) = 'LONGITUDE'
        quan(2) = ' LATITUDE'
        write(from(1),'(f9.3)')
     +                     (ibase1-1)*xincr  + (1-tpts)*0.5*xincr
        write(from(2),'(f9.3)')
     +                  (ibase2-1)*yincr  + tpts*0.5*yincr +stlat
        write(incr(1),'(f9.3)')   xincr
        write(incr(2),'(f9.3)')   yincr
        write(to(1),'(f9.3)')
     +                      (itop1-1)*xincr  + (1-tpts)*0.5*xincr
        write(to(2),'(f9.3)')
     +                   (itop2-1)*yincr  + tpts*0.5*yincr +stlat
        nop(1) = itop1-ibase1+1
        nop(2) = itop2-ibase2+1
C
      elseif(depvar(1:3).eq.'LAT') then
        quan(1) = 'LONGITUDE'
        quan(2) = '    DEPTH'
        write(from(1),'(f9.3)')
     +                       (ibase1-1)*xincr  + (1-tpts)*0.5*xincr
        write(from(2),'(f9.3)')   float(ibase2)
        write(incr(1),'(f9.3)')   xincr
        write(incr(2),'(f9.3)')   1.0
        write(to(1),'(f9.3)')
     +                        (itop1-1)*xincr  + (1-tpts)*0.5*xincr
        write(to(2),'(f9.3)')     float(itop2)
        nop(1) = itop1-ibase1+1
        nop(2) = itop2-ibase2+1
        rintlv = (intlv-1)*yincr  + tpts*0.5*yincr +stlat
C
      elseif(depvar(1:3).eq.'LON') then
        quan(1) = ' LATITUDE'
        quan(2) = '    DEPTH'
        write(from(1),'(f9.3)')
     +                    (ibase1-1)*yincr  + tpts*0.5*yincr +stlat
        write(from(2),'(f9.3)')   float(ibase2)
        write(incr(1),'(f9.3)')   yincr
        write(incr(2),'(f9.3)')   1.0
        write(to(1),'(f9.3)')
     +                     (itop1-1)*yincr  + tpts*0.5*yincr +stlat
        write(to(2),'(f9.3)')     float(itop2)
```

```
          nop(1) = itop1-ibase1+1
          nop(2) = itop2-ibase2+1
          rintlv = (intlv-1)*xincr  + (1-tpts)*0.5*xincr
        endif
c
      quan(3)=' TIMESTEP'
      ndinc=0
      ndfir1 = itt
      ndlas = itt
      write(from(3),'(i9)')ndfir1
      write(incr(3),'(i9)')ndinc
      write(to(3),'(i9)')ndlas
      nop(3) = 1
c
      if (depvar(1:3) .eq. 'STR') then
        write(op,5101)trac,opform
       else
        write(op,5100) trac,depvar,opform
      endif
c
      write(op,5102) namrun(1:2),namrun
      write(op,5103) (i,i=1,3)
      write(op,5104) (quan(i),i=1,3)
      write(op,5105) (from(i),i=1,3)
      write(op,5106) (incr(i),i=1,3)
      write(op,5107) (to(i),i=1,3)
      write(op,5108) (nop(i),i=1,3)
      if (depvar(1:6) .eq. 'STREAM') then
        write(op,5109) ' '
      else
        write(op,5110) rintlv
      endif
      write(op,5111) ttsec ,dtts
 5100 format('VARIABLE :',a15,2x,a9,t41,'FORMAT   :',a2)
 5101 format('VARIABLE :',a15,t41,'FORMAT   :',a2)
 5102 format('MODEL : ',a2,'      COMMENTS:',a55)
 5103 format('INDEX ',9x,':',3('   ',i1,'     :'))
 5104 format('QUANTITY ',6x,':',a9,':',a9,':',a9,':')
 5105 format('FROM     ',6x,':',a9,':',a9,':',a9,':')
 5106 format('INCREMENT',6x,':',a9,':',a9,':',a9,':')
 5107 format('TO       ',6x,':',a9,':',a9,':',a9,':')
 5108 format('NO.OF POINTS ',2x,':',i9,':',i9,':',i9,':')
 5109 format(a50)
 5110 format(f7.3)
 5111 format('FIRST TTSEC ',f12.0,' DTTS ',f5.0)
      return
      end
      subroutine conv2up(mixcase)
      character mixcase*(*)
      integer upa
c
      nchar=lnblnk(mixcase)
c
```

```
      lowa=ichar('a')
      lowz=ichar('z')
      upa =ichar('A')
c
      do 10 n=1,nchar
      nc=ichar(mixcase(n:n))
      if(nc.ge.lowa.and.nc.le.lowz) then
      nc=upa+(nc-lowa)
      mixcase(n:n) = char(nc)
      endif
 10   continue
c
      end
c-----------------------------------------------------------------------
c Include a version of setgrid.F which differs only in the exclusion of
c some report output to stdout:
c-----------------------------------------------------------------------
#define noreport 1
#include "setgrid.F"
c-----------------------------------------------------------------------
c End of adapted setgrid.F
c-----------------------------------------------------------------------
c
#include "odam.F"
#include "restio.F"
#include "blkdta.F"
#include "ascout0.f"
```

Brook Road, Wormley, Godalming
Surrey, GU8 5UB,
United Kingdom
Telephone +44 (0) 428-684141
Facsimile +44 (0) 428-683066
Telex 858833 OCEANS G

Natural
Environment
Research
Council