FILE

Institute of
Oceanographic Sciences
Deacon Laboratory

INTERNAL DOCUMENT No. 324

An ocean model code for array processor computers

D J Webb

1993

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES
DEACON LABORATORY

INTERNAL DOCUMENT No. 324

An ocean model code for array processor computers

D J Webb

1993

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

SUMMARY

This document describes a version of the standard primitive equation ocean model code (the Bryan-Cox-Semtner model) which has been especially developed for use with array processor computers. It is assumed that the surface of the ocean is split up into small areas and that these are used to define volumes extending from the surface to the bottom of the ocean. Each processor in the array is then responsible for handling all the calculations and storage for one of these small volumes.

For convenience, all the Fortran 'do' loops, in which the workload is distributed between the different processors, have been collected into one high level routine (routine 'step'). For efficiency, the main storage and the loop structure of the lower level routines have been rearranged to enable vectorisation in the vertical. For compatibility, the remaining code and structure of the model is close to that of the GFDL Modular Ocean Model.

The model uses a free surface scheme to solve the depth integrated momentum equation. In most problems tackled using an array processor computer this will be more efficient than the standard stream function scheme.

The code is also suitable for use on other computers in which the main memory is large enough to store all the model data. For such cases a stream function form of the present model is also available. In computers with a scalar architecture and vector computers with a low start up overhead, the code will then often be more efficient than the standard form of the model.

The present code is called the 'moma' code. The name reflects the fact that the code is based on the GFDL Modular Ocean Model ('MOM'), with the extra 'A' indicating that it is an array processor version.

CONTENTS PAGE

INTRODUCTION 7

DEVELOPMENT OF THE MODEL 9

The Cox code 10
The MOM code 10
Replacing the stream function equation by a pressure equation 10
The free surface code 11
The array processor ocean model 11

THE FREE SURFACE MODEL 12

Advection in the baroclinic velocity equation 12
Efficiency of the new code 13
Compatibility with the MOM code 14
Definition and time and timestep in the model 15
Use with non-array processor computers 15
Validation 16
Restarting the model 16

SCHEMES FOR PARTITIONING THE OCEAN 17

FILES USED BY THE MODEL 17

OBTAINING COPIES OF THE MODEL 18

BENCHMARK TESTS USING THE PRESENT CODE 18

The one degree global model 19
The half degree global model 19
The quarter degree global model 20

ACKNOWLEDGEMENTS 21

REFERENCES 21

APPENDIX

(The appendices are not included in this version of the report. See page 18 about obtaining copies of the main program, subroutines and 'include' files.)

INTRODUCTION

To a large extent the dynamic state of the ocean can be described by the velocity, temperature, salinity and pressure at each point within it. In ocean models, temperature is usually represented in the form of potential temperature (relative to a pressure of one atmosphere) because this remains constant under adiabatic changes in pressure. Salinity is a composite variable defined by an international standard which represents the combined effect of the different dissolved salts in the ocean.

The evolution of such a system can be described by a three dimensional momentum equation, an advection-diffusion equation for heat and salt, a continuity equation and an equation of state which defines the density in terms of the temperature, salinity and pressure.

In ocean models, three important approximations are made to reduce the computational load. The first is to assume, in the continuity equation, that the ocean is incompressible. The second is to assume, in the vertical momentum equation, that the vertical velocity is small and that the terms involving it can be neglected. The third is to assume, in the horizontal momentum equations, that small changes in density can be neglected except where they affect the horizontal pressure gradient.

The resulting equations are often called the 'primitive equations' (Bryan 1969). They are:

$$\partial u/\partial t + (u \cdot \nabla)\, u + w\, \partial u/\partial z + f \wedge u = -(1/\rho_0)\, \nabla p + D_u + F_u$$

$$\rho\, g = -\, \partial p/\partial z$$

$$\nabla \cdot u + \partial w/\partial z = 0$$

$$\partial S/\partial t + (u \cdot \nabla)\, S + w\, \partial S/\partial z = D_S + F_S$$

$$\partial T/\partial t + (u \cdot \nabla)\, T + w\, \partial T/\partial z = D_t + F_t$$

$$\rho = \rho\, (T, S, p)$$

The main prognostic variables are u the horizontal velocity, w the vertical velocity, T the potential temperature and S the salinity. The other variables, p the pressure, ρ the density and w the vertical velocity, can be calculated from the prognostic variables.

In these equations t is time and f is the Coriolis term (equal to $2\ \Omega \sin(\theta)$, where Ω is the Earth's rotation rate and θ is the latitude. g represents gravity. The terms D represent diffusion and F the forcing.

The horizontal velocity u is zero on solid sidewall boundaries. The gradients of potential temperature and salinity normal to all solid boundaries (including the bottom) are also zero. The upper surface boundary conditions are used to specify the exchange of heat and fresh water at the air-sea interface and the stress acting on the ocean due to the wind.

Bryan (1969) developed a computer scheme for solving these equations which has been widely followed since. This splits the ocean up into a large number of boxes along lines of constant latitude, longitude and depth. In a coarse resolution ocean model the individual boxes will have a width of typically 2 degrees in the horizontal and a thickness ranging from 20 m for layers near the ocean surface to 500 m at depth in the ocean. Topography is represented by assuming that each box is either a sea box or a land box.

The state of the ocean is represented by values of potential temperature, salinity and horizontal velocity at the centre of each box. The above equations are then integrated over each box to give an equation in which the advection and diffusive terms are replaced by the fluxes through the boundaries of the box and the other terms written in terms of averages over the box.

The equations are then cast in finite difference form with gradients calculated from values at neighbouring boxes. For accuracy the Arakawa-B grid is used, in which the velocities are specified at points offset from those at which tracers (temperature and salinity) are specified. This scheme is better than the other standard Arakawa schemes at representing the propagation of oceanic Rossby waves, when the grid spacing is greater than the Rossby radius of the ocean. The Rossby radius is typically 25 km for the first internal mode of the ocean but is much smaller for higher modes. Even with high resolution ocean models, using grids of 20 km or less, the Arakawa-B grid is still preferred because of its better performance in representing the higher vertical modes.

The equations are used to step the model solution forward in time in a series of discrete timesteps. This is computationally most efficient when the timestep is as large as possible. However for stability reasons the timestep has to be less than the time it takes a wave to move from one grid box to the next.

The fastest wave in the ocean, the external gravity wave, has a speed of about 250 m s^{-1}. The next fastest wave, the first internal gravity wave, has a speed of about one hundredth of this value. If the external gravity wave, which has little effect on the ocean circulation, is filtered out it enables the timestep to be increased by a factor of 100, greatly speeding up the model.

Bryan (1969) did this by placing a rigid lid on the ocean. The lid removes the external gravity waves but he was able to show that it had little effect on the large scale circulation of the

model ocean. The approximation introduces an additional stream function equation which has to be solved but the extra computer time required to do this is relatively small.

Bryan's scheme uses a leapfrog timestep to step the equations forward in time. This is computationally efficient and propagates waves correctly without any change in amplitude. However the scheme is unstable when used with diffusive terms and so for these a simple Euler forward timestep is used. The leapfrog scheme can lead to a splitting of the solution on even and odd timesteps so to prevent this growing too large a forward timestep is introduced every 20 to 100 timesteps for all the terms. This can use the Euler forward scheme but usually a Euler backward scheme is used because this does not increase the amplitude of any waves present, an unphysical process which can readily lead to numerical instability.

Fuller details about the model and the numerical schemes used are given by Bryan (1969), Semtner (1973) and Cox (1984). Good reviews of the properties of the different finite difference schemes are given by Mesinger and Arakawa (1976) and by Roach (1976).

DEVELOPMENT OF THE MODEL

Later developments of the model are based primarily on Semtner's (1974) revision of the initial code. He added Takano's (1974) scheme for handling islands and streamlined the code so that it ran efficiently on vector processing computers.

Ocean models are computationally very expensive to run and so should make very efficient use of the computer time available. For a large ocean model this requires the inner computational loops to be very efficient. Semtner introduced a large number of small vector loops which ran very quickly on the early vector processing machines. He also introduced the idea of moving data between main memory and disk storage in slabs, each containing all the data from one latitude band of the model. This allowed the model to be vectorised in the horizontal, usually giving vectors of 90 elements or more, so that the start up overhead in each vector operation was small. By allowing slabs to be prefetched it also meant that the cpu was only rarely left waiting for input-output operations to complete.

The program is organised with a main program which handles the beginning and end of the run and which includes the main timestepping loop. The details of each timestep are handled by routine 'step'. This calls subroutine 'clinic' to calculate the new baroclinic velocity fields, 'tracer' to calculate the new fields of potential temperature and salinity, and finally calls routine 'relax' to solve the stream function equation.

The Cox code

Semtner's code was widely used by the oceanographic community but it was eventually superseded by the code developed by Cox (1984). This introduced a large number of new options, for example different vertical mixing schemes, and it allowed the horizontal grid spacing to be varied.

This version of the code was also written so that each slab was treated by the processor as a single long vector. This scheme was introduced for use on the Cyber 205 computer, whose vector processing unit was very fast but had a long start up time. Unfortunately, because the scheme carries out many redundant calculations over land points, it is less efficient than the Semtner code on most other computers, be they workstations or Cray supercomputers. The Cox code also replaces some small arrays used in the Semtner code by arrays which are the size of a full slab. These increase the number of main memory to cpu transfers and so will slow down most other computers.

The MOM code

More recently the group at GFDL has revised the Cox code and added a large number of useful options. The new code is called the GFDL Modular Ocean Model or MOM code (Pacanowski et al, 1990). Options and common block specifications are now included using Unix 'C' compiler preprocessing directives. The GFDL group has also removed the very large vectors of the Cox version, so that the code works efficiently on most modern computers.

Replacing the stream function equation by a pressure equation

The treatment of the island boundary conditions in the stream function routine involves an integration around the perimeter of each island. This can be done efficiently on a computer with scalar architecture but this is not so on vector or array processor computers.

To overcome this problem Smith et al (1992) recently proposed replacing the stream function equation by an equation for the surface pressure field. The method for solving the pressure equation is similar to that used for the stream function equation but the island boundary condition no longer involves an integral. As a result the new method works with equal efficiency on scalar, vector and array processor computers. For problems with no islands it requires a similar amount of computer time as the stream function scheme.

The free surface code

Because of the limits in computer power, most early models of the ocean were of modest size, with typically less than 100 points in the horizontal. For such models the stream function (or pressure equation) scheme is the one most efficient for timestepping the barotropic velocity field. In a rectangular ocean a fast fourier transform method can be used to solve the equation, but in practice, because the boundary of the ocean is very complex, a relaxation scheme has to be used. This iterates the solution each timestep, the number of iterations being the same order as the number of grid points in the horizontal.

With large models, an alternative scheme is to timestep the baroclinic velocity field separately, using a vertically integrated velocity equation. This is conventionally called a 'free surface model' because it also removes the rigid lid constraint and includes a variable representing the surface elevation of the ocean. The scheme requires approximately 100 small timesteps for each timestep of the baroclinic velocity field and the tracer fields. Because the number of timesteps does not increase as the size of the model increases, the scheme is eventually more efficient than the stream function or pressure equation methods.

A free surface model for use with the Cox code was developed by Killworth et al (1989).

The array processor ocean model

The reason for developing the present model arose out of a decision by the UK ABRC committee to buy a large array processing computer in 1993/94 for the UK research community. All the ocean model codes described above since Semtner have been designed for efficient use on fast vector processing computers. However the development of vector processors is starting to slow down and in future the main advances in computer power are expected to arise from the development of large arrays of small fast computer units each with its own internal storage. The units will be able to exchange data with each other, but this will be at a rate much slower than the internal transfer rate.

The way that we imagine such an array processor will be used to run an ocean model is that the ocean will be split up into small volumes each handled by a different processor. The processors will carry out timesteps in parallel, each processor requesting boundary data from its 'adjacent' processors when required.

We expect that the most efficient scheme will be the one which chooses the volumes to mimimise the maximum amount of computing carried out by any of the processors and which minimises the maximum amount of data being transferred between one processor and its

neighbours. These constraints will probably produce compact regions containing roughly equal number of oceanic grid points.

An exact minimum is probably not important, so in the code presented here it is assumed that the cells constituting each volume form a compact group in the horizontal and within the region contain all the cells extending from the sea surface to the bottom. Data that needs to be fetched from another processor will always be for cells at the sides of those being processed and never for cells above or below.

This assumption allows the model code to be vectorised in the vertical direction. The way that the ocean region is partitioned between the processors in the horizontal is left to the programmer or compiler to choose, depending on the geometry of the ocean basin being studied. For simplicity, all the do loops involving horizontal indices have been collected in one high level subroutine (subroutine 'step'), so only this routine will need to be changed.

THE FREE SURFACE MODEL

Because the moma code is intended for use with large ocean models it uses a free surface model. The free surface subroutine is based on that of Killworth et al (1989) and uses the Euler backward timestepping option because this is efficient at damping out high frequency surface gravity waves. This prevents aliasing problems associated with the long baroclinic and short barotropic timesteps.

The scheme used differs from that of Killworth et al in that the frictional terms are no longer included within the free surface subroutine. Instead they are calculated once only within each baroclinic timestep. This modification is computationally more efficient and has very little effect on the solution.

A stream function version of the model is also available. It was used to validate the free surface code but contains a stream function solving routine which has not yet been optimised for array processor computers. The stream function version is discussed further in the section on scalar and vector computers.

Advection in the baroclinic velocity equation

In the primitive equation system of equations, the vertical velocity is required at two places, once to advect tracer quantities (ike heat and salt) in the vertical, and once to transport momentum in the vertical. Using the Arakawa B grid these are required at different positions,

so in the standard primitive equation model different schemes are used to calculate the two sets of vertical velocities.

Experience with the model has shown that the vertical velocity field calculated at tracer points is well behaved, but the vertical velocities calculated at velocity points is very noisy. This may be due to the way horizontal advection is defined at velocity points.

When solving the velocity equations, the flux of momentum through the side face of a grid box is calculated from the two velocity points on either side of the face. The divergence of this flow is then used to calculate the fluxes through the top and bottom faces of the box.

When this scheme was checked within the free surface code, it was found that purely horizontal flow fields could result in significant spurious vertical velocities. Such velocities did not occur in the equations for tracers. A check against the standard stream function code showed that the barotropic velocities, calculated from the stream function, produced no spurious vertical velocities but the baroclinic flow field still did so.

The simplest scheme found which overcame these properties defines the advective flux through each side of a velocity box to be the average of the corresponding fluxes for the four surrounding tracer boxes. The new scheme is used in the present free surface version of the code and is recommended for use with stream function version of the code.

The suggestion to use a vertical velocity field at velocity boxes which is the average of that at the surrounding tracer boxes has been made previously. However for consistency, the modified horizontal advection scheme needs to be used as well.

Efficiency of the new code

The present model is designed to make no assumptions about the order in which the columns of ocean boxes are processed. This is done in order to give the programmer or operating system as much freedom as possible to decide how the ocean calculations and storage should be split between the individual array processors. Unfortunately this also means that the model needs more core storage than the 'in core' version of the MOM code. It must also either use very large temporary arrays or repeat parts of the calculation, making it less efficient.

There are three main problem areas. The first is concerned with storing the main data arrays. In their 'disk' versions, the Semtner, Cox and MOM codes store the full model fields from three timesteps on disk. These correspond to the latest timestep and the two previous ones. In their 'in core' versions, they make use of the sequential way in which the calculation is performed to overwrite values from the earliest timestep with the newly calculated values. As a

result the 'in core' versions only need core storage for two full sets of variables* . The present moma code can make no assumptions about the order in which the calculations are carried out and so three full sets of variables must be stored.

Similar problems arise when calculating the advective and diffusive terms in subroutines clinic and tracer and when calculating the pressure gradient terms. For calculating the pressure gradient acting on a velocity grid box, subroutine clinic needs to calculate the density fields in the four surrounding grid boxes. In the standard Semtner scheme this is done once only and the values stored in temporary arrays until required again. Because the model works systematically across the model domain the size of the temporary arrays is quite small. In the present model, no assumptions can be made, and so if a temporary array is used it has to cover the whole model ocean.

In the moma code the choice of whether or not to use a temporary array is included as a precompiler option. If the precompiler name 'presetp' is not set then the model recalculates the pressure field as it is required. If 'presetp' is set then the model uses a temporary array to store the baroclinic pressure field. This reduces the cpu time by about 4% but increases the storage requirement by about 8%.

The Semtner code uses temporary storage in a similar way for the eight horizontal flux fields calculated by clinic and tracer. If used in the present model the temporary arrays would increase the storage requirement by 66%, giving an expected 5% to 10% saving in CPU requirement. The extra storage requirement was thought to be too large and for this reason the option has not been implemented.

Compatibility with the MOM code

The moma code uses the same Unix C preprocessor directives as the MOM code and uses similar naming conventions. To simplify its development the present code contains none of the standard MOM code options but it should be possible to transfer many of these to the present model without much change.

One important difference between the two codes is that the MOM code makes widespread use of statement functions in the timestepping equations. This provides a convenient way of modifying the finite difference scheme when required. However many compilers are inefficient at optimising code including statement functions and so, as they come in the innermost loops of the model, they are not used in the present code.

---

* For the Cox code this is true for Update 5 onwards.

There are also differences in the organisation of some of the common blocks. These are mainly associated with replacing the stream function scheme by the free surface model.

Definition of time and timestep in the model

Most discussions of finite difference schemes, for example Mesinger and Arakawa (1976), consider the problem of stepping the model solution from time t to time $t+dt$, possibly making use of information from time $t-dt$. Users of the present code should be aware that as in Semtner (1973), Cox (1984) and the MOM code (Pacanowski et al, 1980) the model is set up slightly differently.

Let t_0 correspond to the initial time, before any calculations have been carried out, and let dt be the baroclinic timestep. The model is organised so that during timestep 'itt' the solution is advanced from time t_0 to timestep t_0+dt. The variable 'itt' and the time variables ('totsec' and 'totday') are set to their new values at the beginning of each timestep before any other calculations are performed and it is the new values of these variables which control the setting of the time dependent logical switches.

Within the program the logic is structured so that the expected operations occur. Thus if forward timesteps are set to occur every 10 timesteps then these occur during timesteps 1, 11, 21, etc. Similarly the program stops after advancing the solution to the end of the day given by input variable 'days'. (At the end of the run the solution corresponding to the final time is in the arrays pointed at by pointer 'np').

Problems may arise if the forcing functions used by the model contain any rapidly varying time dependent information, like the diurnal variation of sunlight. The subroutines that calculate such terms must then take into account the fact that the central time during a timestep is not that given by the time variable 'totsec', but instead is 'totsec-dt'.

Use with non-array processor computers

Although it is designed primarily for use with array processor computers the present model can be used on scalar and vector computers as long as the main memory is large enough to hold all the data fields. On scalar computers the model may be more efficient than the models discussed earlier because calculations are skipped for land points and points below the bottom of the ocean. However the horizontal advective and diffusion terms are calculated twice in the present model and this introduces an extra overhead.

On a SUN workstation the test model used to validate the present stream function code ran approximately 20% faster than the same model run using the MOM code.

On vector machines the present model should be efficient as long as the vector start up time is not too long. If the computer has a large start up time, it is more efficient to vectorise in the horizontal because the length of the arrays is usually larger in the horizontal direction than in the vertical.

Validation

Validation tests were carried out using the old form of the horizontal advection code at velocity points. Initially the stream function version of the model was validated by checking against an equivalent test model using the MOM code. The only differences found were at the rounding error level.

The free surface version of the code was then validated against the stream function version with the barotropic valocity set to zero. This essentially validated those parts of the code concerned with solving the temperature, salinity and baroclinic velocity equations. Again the only differences were at the rounding error level.

The free surface code itself was checked against the stream function version by running the models with only the free surface and barotropic velocity fields switched on. The agreement between the two solutions was good, the kinetic energy of the free surface solution being just a few per cent higher. It also showed some long term oscillations. Both differences are presumably due to gravity and Kelvin waves present in the free surface solution but filtered out of the stream function solution.

Finally the free surface version was checked against the stream function with all parts of the code switched on. Again there was good agreement between the two solutions.

Restarting the model

The code assumes that the model will always be restarted using a forward timestep. This is done because the forward timestep only needs data from a single timestep whereas a leapfrog timestep needs data from both the restart timestep and the previous timestep. The size of the restart datasets is thus halved. This is significant for the large array processor models envisaged.

SCHEMES FOR PARTITIONING THE OCEAN

As discussed earlier, the present code makes no assumptions about the way the ocean will be partitioned between the individual array processors but for convenience places all the relevant loops over the horizontal indices in subroutine 'step'. In practice a number of schemes could be used.

The simplest, and the one likely to be used by most compilers, is to split the ranges of the latitude and longitude indices 'i' and 'j' equally between the different processors. Each processor is then responsible for storing all the variables and for carrying out all the calculations for its own rectangular region of ocean. When processing points on its own boundary it sends requests to the neighbouring processor for any data it needs for grid points in the neighbouring region.

This scheme has two drawbacks. First some processors will be responsible for regions in which there is little or no computational load. Secondly, when processing the boundary points, each processor may have to wait some considerable time for the data from the neighbouring processor.

The first problem can be overcome by using irregular regions for which the computational load is better balanced. The loops in subroutine 'step' might then not be simple i and j loops over rectangular regions.

The second drawback can be overcome by starting the transfer of boundary information at the start of each timestep. Such a scheme might have to use low level message passing calls between the processors. It might also need additional storage arrays for this data. Calculations would then proceed on interior points, while the transfers are taking place, with calculations on the boundary points themselves being left to the end of the timestep.

FILES USED BY THE MODEL

A copy of the main program and subroutines is given in Appendix I. In standard distributions of the code these are all contained in the file 'moma.F'. The code should be processed before compilation using the Unix C compiler preprocessor 'cpp'. For this it needs the following include files, which contain definitions of the data arrays used:

```
cdiag.h     chmix.h     coord.h     ctmngr.h     cvbc.h      cvmix.h
dncoef.h    frees.h     grdvar.h    index.h      iounit.h    levind.h
param.h     pconst.h    scalar.h    slabs.h      switch.h    thick.h
timelv.h    versno.h
```

The present moma code has three options which are implemented defining the corresponding precompiler name. The three options and corresponding names are:

`'oldadv'` to use the original scheme for the horizontal advective velocity at velocity points.

`'presetp'` to precalculate the full model baroclinic pressure field before use by subroutine 'clinic'.

`'hcomments'` to include comments from the '*.h' include files.

When running the program, the data file 'ocean.in' should be placed in the current directory. Distributed versions of the program also contain the file 'moma.output', the results obtained when running the standard model with the supplied version of 'ocean.in'.

OBTAINING COPIES OF THE MODEL

The latest version of the model can be obtained using anonymous ftp over internet. To obtain the file from a computer attached to internet, make an ftp connection using:

```
ftp unixa.nerc-wormley.ac.uk     (or ftp 192.171.175.1)
```

Give your own name, id and internet address as the password. Once connected change to directory pub/occam using the ftp 'cd' command:

```
cd pub/occam
```

The contents of the directory can be listed using the command 'ls'. Information on the published files is contained in file 'README'. This can be copied to your own machine using the command 'get README'. The latest version of the moma code will be in a file with a name of the form 'moma_v1.11.tar.Z'. Copy this to your own machine using the command:

```
get 'moma_v1.11.tar.Z'
```

The file contains the main program, include files and the test input and output files 'ocean.in' and 'moma.output'. It should be unpacked using the Unix 'compress' command followed by the Unix 'tar' command.

BENCHMARK TESTS USING THE PRESENT CODE

The present code was developed in part to provide a realistic benchmark for use in choosing one of the new generation of array processing computers. The standard code is set up to run the schematic low resolution global model developed by the GFDL group for model tests and distributed with the MOM code. It is set up to cover the ocean from 72°S to 72°N with a

resolution of 4 degrees both east-west and north-south, but may be easily modified for tests at different resolutions. The changes needed for tests at resolutions of one, one-half and one-quarter of a degree are given below.

The one degree global model

1. For the one degree global model the file "param.h" should include before any other declarations, the line:

   parameter(imt=362, jmt=141, km=15, nt=2, imu=imt)

2. The main program should include, immediately after the lines defining the character array 'model', the lines:

```
stlon = -1.0
stlat = -70
dxdeg = 1.0
dydeg = 1.0
```

3. The file 'ocean.in' should include the following namelist statements or their equivalent in the following order:

```
&contrl
      init=.true.,       fnrest = ' ',
      days=10000.0,      restrt='true',
      nmix=48,           eb=.true.,          ncon=1,
      tsi=10.0,          dgnstc=10000.0,     snaps=10.0
&end
&eddy
      am=1.0e8,          ah=2.0e6,
      fpkh=1.0,          fpkm=1.0,           cdbot=0.001
&end
&tsteps
      dtts=900.0,        dttuv=900.0,        dtbt=25.0
&end
```

Note that there should be one space on each line before the ampersand.

The one degree model will require storage for approximately 10 million variables.

The half degree global model

1. For the half degree global model the file "param.h" should include before any other declarations, the line:

   parameter(imt=722, jmt=281, km=15, nt=2, imu=imt)

2. The main program should include, immediately after the lines defining the character array 'model', the lines:

```
stlon = -0.5
stlat = -70
dxdeg = 0.5
dydeg = 0.5
```

3. The file 'ocean.in' should include the following namelist statements or their equivalent in the following order:

```
&contrl
      init=.true.,       fnrest = ' ',
      days=10000.0,      restrt='true',
      nmix=48,           eb=.true.,          ncon=1,
      tsi=10.0,          dgnstc=10000.0,     snaps=10.0
&end
&eddy
      am=1.0e7,          ah=2.0e6,
      fpkh=1.0,          fpkm=1.0,           cdbot=0.001
&end
&tsteps
      dtts=450.0,        dttuv=450.0,        dtbt=12.5
&end
```

Note that there should be one space on each line before the ampersand.

The half degree model will require storage for approximately 40 million variables.

The quarter degree global model

1. For the quarter degree global model the file "param.h" should include before any other declarations, the line:

   parameter(imt=1442, jmt=561, km=15, nt=2, imu=imt)

2. The main program should include, immediately after the lines defining the character array 'model', the lines:

```
stlon = -0.25
stlat = -70
dxdeg = 0.25
dydeg = 0.25
```

3. The file 'ocean.in' should include the following namelist statements or their equivalent in the following order:

```
&contrl
      init=.true.,       fnrest = ' ',
      days=10000.0,      restrt='true',
      nmix=48,           eb=.true.,          ncon=1,
      tsi=10.0,          dgnstc=10000.0,     snaps=10.0
&end
&eddy
      am=2.0e6,          ah=2.0e6,
      fpkh=1.0,          fpkm=1.0,           cdbot=0.001
&end
```

```
&tsteps
      dtts=225.0,      dttuv=225.0,      dtbt=6.25
&end
```

Note that there should be one space on each line before the ampersand.

The quarter degree model will require storage for approximately 160 million variables.

ACKNOWLEDGEMENTS

I wish to acknowledge the important contributions of K. Bryan, A.J. Semtner, M.D. Cox and the GFDL MOM group to the development of this program. Thanks also to M. Ashworth at P.O.L. for discussions on array processor architectures and codes and to B. de Cuevas and A.C. Coward at IOS for their support.

REFERENCES

Bryan, K., 1969: A numerical method for the study of the circulation of the circulation of the world ocean. Journal of Computational Physics, 4, 347.

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

Killworth, P.D., Stainforth, D., Webb, D.J. & Paterson, S.M. 1989: A free-surface Bryan-Cox-Semtner ocean model. IOSDL Report No. 270. Institute of Oceanographic Sciences Deacon Laboratory, Wormley, 184pp.

Killworth, P.D., Stainforth, D., Webb, D.J. and Paterson, S.M. 1991: The development of a free-surface Bryan-Cox-Semtner ocean model. Journal of Physical Oceanography, 21, 1333-1348.

Mesinger, F., and Arakawa, A., 1976: Numerical methods used in atmospheric models. GARP Publication series, No. 17. World Meteorological Organisation, 64pp.

Pacanowski, R.C., Dixon, K. and Rosati, A., 1990: The GFDL Modular Ocean Model users guide, version 1.0. GFDL Group Technical Report No. 2.

Roach, P.J., 1976: Computational Fluid Dynamics. Hermosa Publishers, Albuquerque, N.M., 446pp.

Semtner, A.J., 1974: A general circulation model for the World Ocean. Technical Report No. 9, Department of Meteorology, University of California, Los Angeles, 99pp.

Smith, R.D., Dukowicz, J.K., and Malone, R.C., 1992: Parallel ocean general circulation modelling. Physica D, 60, 38-61.

Takano, K., 1974: A general circulation model of the world ocean. Numerical Simulation of weather and Climate. Technical Report No. 8, Department of Meteorology, University of California, Los Angeles, 47pp.

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