<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

READ_ME file for GFDL MOM 1.0

R.C. Pacanowski   K. Dixon   A. Rosati

Dec 1990

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

Table of Contents:

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(i) General Overview:

The GFDL Modular Ocean Model (MOM) is a primitive equation general ocean circulation model intended to be a flexible tool for exploring ocean and coupled air-sea applications over a wide range of space and time scales. Its modular design is specifically intended to aid in flexibility, ease of use, and continued development without sacrificing clarity.

MOM has been written as a collaborative effort by Ron Pacanowski, Keith Dixon, and Anthony Rosati at the National Oceanographic and Atmospheric Administration's Geophysical Fluid Dynamics Laboratory in Princeton, New Jersey. It is the successor to the code written by Michael Cox, documented in the GFDL Ocean Tech Report #1, (1984). As was the case for the Cox model and the Semtner model (UCLA Dept. of Meteorology Tech. Report No. 9, 1974) that preceded it, MOM is a fortran implementation of equations described by Kirk Bryan (1969, J. Computat. Phys., 4, 347-376).

In the future, a User's Guide to MOM will become available. For now, this "READ_ME" file is intended to serve as a guide. Users unfamiliar with the Cox model are referred to GFDL Ocean Tech Report #1 and the J. Computat Phys article for a discusion of the numerics.

Comments and suggestions concerning MOM are welcome. Anyone wishing to develop refinements and/or additions will be acknowledged for their effort in subsequent releases of MOM. Please contact one of us if you have something you feel might be worthwhile to add. We suggest that when developing code, the style and modularity of MOM be followed. As time permits, new features will be incorporated into MOM.

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(ii) Files & Access:

```
The following files are included:

     a)   splitlib.c & mergelib  ... two utilities. mergelib
          concatenates all files with suffixes F,f, or h (abbreviated
          as *.[Ffh] files) into one big file.
          splitlib does the inverse... breaking the mergelibed file
          back into its component files ( *.[Ffh] files)
          (Note: Functionally, the UNIX "tar" command does the same but
           there may be machine dependencies. If it turns out that there
           are no problems, we will go to "tar" instead.)

     b)   mom_x      ... the MOM code in mergelibed form. (the
                         the "x" is the version number. ie: mom_1.0)

     c)   printout   ... output from a 3x4 degree world ocean sample
                         run on a CRAY YMP

     d)   ocean.in   ... the input file containing "namelist" input
                         needed to run the sample case


     e)   READ_ME    ... the file you're looking at

     f)   upgrade    ... a sample script for source management on
                          SUN workstations (for upgrading your changes
                          to future versions of MOM)

     g)   cray.run   ... a sample run deck for running the test case
                         on a CRAY YMP

     h)   PREP_DATA  ... is a directory which contains a collection of
                         files (routines & rundecks) for interfacing
                         Scripps 1 deg topography, Hellerman & Rosenstein
                         wind stress, Oort air temperature, & Levitus
                         climatologies directly to MOM. Check the
                         included READ_ME file for details.

     i)   MOM_NEWS   ... if this file exists, it will contain an
                         up to date list of bugs & problems with MOM
                         along with other relevant info.


How To Access GFDL:

For now, a user in netland will have to use server1's IP address to gain
access.  In the future, we will have domain name service and users will
be able to use a domain address to find us. So for now, in order to
connect to GFDL, a user will have to key in the following command:

ftp 140.208.1.9

Where 140.208.1.9 is the IP address of server1.  The user can then log
in as either "ftp" or "anonymous".

Example:

    First, make a directory & a subdirectory and change to it:

mkdir my_mom          ( for the MOM files)
cd my_mom
mkdir prep_data       ( for the DATA files)
cd prep_data
```

    Then, do the ftp & transfer the files:

```
ftp -i 140.208.1.9
Name (140.208.1.9:xxx): ftp
Password: anything              (your last name would do nicely)
cd pub/GFDL_MOM/PREP_DATA       (this is the DATA directory)
mget *                          ( transfer the DATA files )
lcd ../                         ( change local directory)
cd ../                          ( change remote directory)
mget *                          ( transfer the MOM files )
quit
```

    Then, change directories & do a directory list:

```
cd ../
ls -alF
```

To work with MOM, first create a directory & copy the original files into it. (Always save your originals. They are needed when upgrading your changes to newer versions of MOM)
Now, break mom_1.x into its component files by doing:

```
       cc -o splitlib splitlib.c  (to compile the "c" program)
       splitlib mom_1.x            (to run it)
```

Now there should be lots of files. They are:

     a)   *.F and *.f files:
....these are the Fortran source files (subroutines, main programs, and functions)

     b)   *.h files:
....these are the "include" files (groups of parameter statements, common blocks, etc., that are inserted into a *.F file via "#include" directives)

To merge them back into mergelibed form, do the following:

```
        mergelib > onebigfile
```

Note that all the *.[Ffh] files are still left in the directory and there is a new one : "onebigfile"
Caution: be careful that there are no other .F, .f or .h files in the directory or they will be incorporated into the "onebigfile".

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(iii) Recommendations & Assumptions:

     * We recommend that you first scan the ".h" files after reading this file. A convenient way to do this (under UNIX) is to use:

```
   cat *.h > allh
```

to append all the ".h" files together into file "allh" and then print it out. All variables should be described in comments (if we missed some, let us know). Then look at the sample run in file "printout". Note the section on model options which is produced by subroutine "docmnt.F". MOM may be "customized" for a particular application by selecting various combinations of these options.

     * It is strongly recommended that source management be done

```
on a UNIX based computer (workstation or mainframe). At the very
least, the C-language preprocessor ("cpp") is required. Any computer
that runs "C" should have it. The "cpp" is used stand alone or by
compilers to allow sections of the Fortran code to be selectively
turned on or off during compilation via "ifdef" options. Many
compilers incorporate "cpp" into their processing, even if not
explicitly stated. We use this feature to control which options and/or
modules are used in MOM. While we do source code management and
version control using the UNIX "SCCS" system, the method of source
management is left to the individual. We’ve included "upgrade", based
on SCCS, as an example of one way to do source management on a SUN
workstation:


 ..... Suppose you’ve been working with MOM.old and have added local
       changes to make "MOM.yours". When MOM.new becomes available, you
       would like to take advantage of its new features. How do
       you get MOM.yours upgraded from MOM.old to MOM.new? If your
       modifications are mostly sections of code grouped together (as
       opposed to being spewen everywhere) you might simply "cut &
       paste" them into MOM.new. More extensive changes can be handled
       more or less in an automated way.
       An example using the "upgrade" script follows:

       Create a working directory and copy four files into it: the
       mergelibed forms of MOM.old, MOM.yours, & MOM.new along with
       the "upgrade" script.

       Now run "upgrade" and at the prompts input MOM.old, MOM.yours
       and MOM.new. The script will produce "new_source"  which is
       MOM.yours upgraded to  MOM.new. It will also most likely point
       out places where possible problems occurred by showing the line
       numbers in  "new_source".

       After investigating & fixing all problems (start from the
       highest line numbers (bottom) & work backwords in your favorite
       editor), you should then compare "new_source" with MOM.new
       as in:

       diff MOM.new new_source > my.chgs

       and inspect "my.chgs" to make sure they got in properly.
       Note: If code is altered in MOM.yours and the same code is
       altered in MOM.new but NOT in MOM.old, both the altered code
       from MOM.yours and MOM.new will get into "new_source"

       When satisified, generate the component *.[Ffh] files by
       using:
       splitlib new_source

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(iv) Coding Design and Features:

     * Users of previous versions of GFDL’s ocean model codes will
notice that there are many more subroutines in the MOM code. The
modular design is intended to aid in the logical organization of the
code with "hooks" readily available for adding new options to the
model.  Deciding which modules to use is done at the pre-processing
level. Modules do NOT interact with each other, but interact with the
main code in only a few places through a short argument list and/or
the include files. This approach to interfacing tends to localize code
modifications, thereby keeping the code structure simple, under-
```

standable, and easily supplemented. Again, we strongly suggest that
this approach be followed.

* Variables have been organized and grouped in terms of physical
(and/or logical) significance. This allows for better modularity.
Look at the *.h files. Note that all variables are commented. To find
out where variable yyy is used, use the UNIX command:

```
     grep -i -n yyy *.[Ffh]
```

which searches all files with .F, .f, or .h suffixes for the yyy.
(The use of "grep" is invaluable in tracing variables & options and
its use is strongly encouraged)
Note also that the include files may be nested (see the bottom of file
"param.h" for example). When modifying the code, one should
consider whether it is appropriate to add new variables to existing
common blocks and ".h" files, or if a new ".h" file and common block
are called for to enhance modularity. When in doubt, try to minimize
the clutter factor.

* There are no out-of-bounds references in the MOM code. This
can simplify debugging new code by turning on a compiler's bounds
checking option. If, on execution, an array subscript goes out of
bounds, the compiler's bounds checker will let you know where & when.

* The "slabs" have been defined as four dimensional variables,
utilizing the concept of a memory slab window. Look at file "slabs.h"
for details. This increases the generality and flexibility of the code
and simplifies management of the data flow from disk through memory.

* Statement functions are used to define physical operators.
This allows one to write code that more closely resembles the
equations. Another advantage is that complicated code can be made more
understandable. At the same time, more operations per loop often allows a
compiler to generate more efficient code. On computers like the CRAY YMP,
this can lead to significantly more parallelism. Look in "tracer.F" and
"clinic.F" to see how we have used them. We've also included an option
that allows the use of arrays (instead of statement functions) to
retain terms in momentum and tracer equations. This can be useful when
using the terms often in more than one place. The price for this is
increased memory (but less computation).

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(v) General Considerations:

* MOM is written in standard Fortran-77, except for the use of
the common extension of "namelist".

* Naming conventions have also been adopted, and appear as
comments in the model. Conventions exist for naming numerical
constants, grid variables, reciprocals, etc. (see "pconst.h"). By
adhering to these rules, one can more easily analyze unfamiliar
sections of the code.

* The code has been written with the underlying assumption that
memory size has been increasing for the types of computers that the
code is typically run on. So, there are places where we trade-off extra
memory usage in order to achieve clarity, modularity, and to minimize
the probabilty of introducing errors when modifying the code in the
future.

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(vi) Setting Up A Model & Stand-Alone Programs:

There are 4 main programs: "ocean", "eqstat", "size",and "depths". The UNIX loader can only handle one main program in a directory at a time (makefiles can be utilized to get around this constraint). The program statements for "eqstat" in "denscoef.F" and "depths" in "depths.f" have been commented out and replaced by subroutine statements to prevent problems. The same has been done in "size.f"

The test case is a 4 x 3 degree ("param.h" & "blkdta.F") by 15 level ("depths.f") world ocean model with idealized geometry ("topog.F") forced by zonally averaged annual mean data ("bcest.F") with initial conditions specified in "onc1st.F". The domain & resolution can be easily changed (described below) and the geometry, forcing & initial conditions adapt themselves accordingly.

* When setting up a model, first decide on the model dimensional parameters in "param.h" (Set "imt" through "maxipp" which are on the first two lines of the parameter statement).

* Set miscellaneous items to the desired values in "blkdta.F". As a minimum, these include all quantites that define the grid (stlon, stlat, xmin, xmax, xwid, idir, ymin, ymax, ywid, & jdir. see "coord.h" for an explanation of what they are). Ideally, regions of interest should have constant, though not necessarily the same, grid spacing in longitude, latitude & depth. Why? Because the numerics are second order accurate for constant grid spacing. There can be severe truncation in areas of abrupt changes in resolution. An example of a reasonable setup might be 1/3 deg spacing in latitude equatorward of 10 deg and slowly increasing to 1.0 deg at 30 deg. See "coord.h" for how to do it.

* Set the number & distribution of vertical levels. the program statement should be placed back into "depths.f", and subroutine statements put into "eqstat.F" and "ocean.F". You can then compile and run "depths.f". It is an interactive program that will allow you to produce a vertical distribution of model levels. It will then produce a "thick.h" file , which will be used by the model.
(Note: the version included here is a prototype, and is being refined).

* Next, prepare "eqstat" to be the main program and compile and run it. It’s located in the module "denscoef.F". It will produce density coefficients (using the "thick.h" file) in file "dncoef.h", which are used by the model in subroutine "state.F".

```
    f77 -o eqstat denscoef.F           ...uses the UNESCO equation of
state, while to use the Knudsen-Ekman equation set the "ifdef" option...
    f77 -Dknudsen -o eqstat denscoef.F
```

* Prepare the file "ocean.in", which will be read as namelist input by MOM. Variables that control MOM’s flow (ie: length of integration, when diagnostics are to done, etc) are documented in "switch.h", "scalar.h" and initialized in "blkdta.F". Many of them may be reset by altering their values in the namelist.
Also decide which model "ifdef" options are to be enabled. When using the "islands" option, the "iland" namelist variables "alonis" & "alatis" (in ocean.in) must be set. See "index.h" for their definitions ... since there are no defaults.

* Now make "ocean.F" the main program, compile and run. In a UNIX environment, use this form to match the sample printout provided:

```
   f77 -Dcyclic -Ddiskless -Drestorst -Dconstvmix -Dconsthmix -Dtiming
        -Drigidlid -Dcongrad9pt -Dislands -Dfourfil -o ocean *.[Ff]
   ocean > printout
```

"cray.run" is a sample run deck for a CRAY YMP. The "-D" options turn on
the "ifdef" options in the model code. The ones shown here are those
that were used to produce the sample printout file.
The previous example showing the use of "grep" to find variables is
also useful for finding out where code related to certain "ifdef"
options is located. (ie: grep -i -n "skipland" *.[Ffh] ... will
show you where code involving the skipland option is.)

The fourth program "size.F" is a stand alone utility. Replace the
subroutine statement with a program statement, compile, and run it.
It will allow you to quickly estimate the memory & disk resources for
various configurations of MOM.

ie: to see the requirements when biharmonic mixing and multitasking are
turned on use:

```
f77 -D biharmonic -D multitasking -o size size.F
size
```

(Note: change the parameters within size.F to alter dimensional info)

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(vii) Some New Features & Notes for Previous Users of the Cox Code

As noted before, the MOM code is intended to be much more modular
in design than the Cox model and other earlier codes. Increased
modularity and an increased number of code options result in there
being many more subroutines and include files in MOM.
The simplified flow chart in this "READ_ME" file should help users
familiar with the Cox code to follow the flow of MOM.

Refer to the sample "printout" and section (viii) to see the complete
list of currently allowable "ifdef" options. Note that there are options
for doing calculations only over ocean points (as opposed to
everyhwere), choosing various lateral and vertical mixing
parameterizations (explicit or implicity solved) plus a selection of
Poisson solvers including a very accurate 9 pt conjugate gradient scheme
which allows for direct reconstruction of the rigid lid surface
pressure. Various MOM options are controlled by "ifdef" options, rather
than Cox's "UPDOC" program. It is recommended that you put "ifdef"
options around your code modules and try to keep modules from
interacting as in MOM. Some UNIX tools for source management include:
diff, sdiff, diff3, & SCCS.

Note also that the slabs are defined in a way such that no boundary
conditions have been added onto the slabs. The vertical boundary
conditions can be found in their own common block in "cvbc.h" (common
of vertical boundary conditions). By increasing the number of dimensions
on the slabs to include time level and j-row location "pointers", the
number of slab variable names and do loops that switch time and j-row
positions have been reduced.

New variables have been added and some old ones redefined
(see "coord.h" for examples). There are also "source terms" for the
momentum & tracer equations which allow easy inclusion of sources
and sinks (ie: biology, imposing baroclinic flows, etc)

Consistancy checks are carried out for "ifdef" options along with
various other conditions by subroutine "checks.F". Notice the "warning"

messages. There are printed by "checks.F".

Island calculations are done by line (rather than area) integrals and
the island definition has been simplified by the automatic calculation
of island perimeters ("iperim.F") requiring only that one point within
the island be specified. This allows for easy use of islands within
complex geometries.

A time manager (see "tmngr.F" and "ctmngr.h") controls all time
dependent decisions within the model. Time dependent information such
as length of integration, time between energy diagnostics, etc., are
entered through namelist in units of days (only nmix, the number of
timesteps between mixing timesteps, is not in units of days). This
information is used by "tmngr.F" to decide how logical variables in
"switch.h" are to be set each time step. These logical variables in
"switch.h" control time dependent data flow.

Included in "tmngr.F" are utilities for aiding in interpolating time
dependent boundary conditions (ie: monthly wind stress, etc) to the
time step. However, these are not used in the test case.

Another utility is function "indp", found in "setgrid.F". This function is
used in many places and gives an easy way to map real world coordinates
(latitudes, longitudes, & depths) to the nearest model grid point. This
allows all spatial information to be independent of changes in model
resolution.

In addition to outlawing out-of bounds references in order to improve
the code’s clarity, an attempt has been made to minimize equivalence
statements. Equivalence statements should be added with caution.

The ability to compute volume weighted tracer averages, surface fluxes,
and term balances for the momentum & tracer equations has been added
to facilitate analysis & debugging. These computations can be done
over arbitrary volumes in arbitrary locations. The volume sizes and
locations can be specified using horizontal & vertical masks. (ocean.F)

There is a "snapshot" feature to write instantaneous data for offline
analysis. (See "switch.h" & "iounit.h" for details)

There is also a bottom drag in MOM. It is controlled by the linear
drag coefficient "cdbot". Ther is no bottom drag when "cdbot" = 0.
(See "scalar.h" & "blkdta.F") As in the COX model, lateral boundaries
are no slip for momentum & no flux across boundaries for tracers.

MOM uses "docmnt.F" to arrange data to be written out as a
form of documentation summarizing the model characteristics that
uniquely define a given model run. It also lists the requested "ifdef"
options. The user will need to review this routine when configuring
a model run or adding new options to the code. This feature can be
beneficial to those setting up new model runs, comparing different model
runs, and for analysis purposes.

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(viii) Options:

A very powerful feature of MOM is the ability to define ifdef options
and configure the model in a manner suitable for the problem being
investgated. The options are quite extensive and range from various
subgrid mixing parameterizations to computing performance enhancements.
The current implementation is such that if there are more than two
options within a catagory there is no default value, so care should be

```
taken to define the desired option. A subroutine "checks" ,wihin MOM,
searches the selcted options to see if any of them conflict. For example,
if a user in selecting an I/O scheme turned on both "diskless" (fully
contained within memory) and "fio" (Fortran dirct access), an obvious
inconsistency exists and a message to this effect will be printed and
MOM will halt.

Options are enabled by using the "-D option" form on the compiling
statement (as shown above in (vi)). This form may also be used on a
"cpp" statement (see "cray.run") to preprocess the code before sending
separate it to the compiler. Here is a terse summary of what’s currently
available:



External mode:

 Basically, there are four Poisson solvers. All of them require the
 "rigidlid" option to also be enabled:

 "oldrelax" is the one from Mike Cox’s ocean model. If you use it,
 you should twiddle with "sor" (the overrelaxation constant) to
 minimize the number of scans for your geometry. "crit" controls the
 accuracy. See "relax.F" for more details.

 "congrad5pt" is a conjugate gradient technique. "sor" is not needed
 here but "crit" still controls the accuracy. This scheme is faster
 than "oldrelax" but may be a bit less stable in the presents of steep
 topographic gradients. The surface pressure is calculated only as
 a basis for comparison  with "congrad9pt". See "congr5.F" for more
 details.

 "congrad9pt" is another conjugate gradient solver which uses a 9 point
 laplacian instead of the 5 point laplacians used by the other schemes.
 It’s advantage is that it is the most accurate & allows the surface
 pressure to be calculated directly. Reducing "crit" will give better
 results (as shown by the closed line surface pressure integrals in the
 test case) than "congrad5pt" because, as in all other schemes, there
 is significant truncation due to the 5 point numerics. In fact, the
 accuracy of the streram funtion solution is limited only by "crit"
 and computer precision.
 Its down side is that it may have stability problems with for
 topographies with sharp gradients and will take more time. How much
 you ask? It depends on your configuration. Try it and see. See
 "congr9.F" for more details.

 "hypergrid" is a version of "oldrelax" solved using a checkerboard
 technique: first on the black squares, then on the red ones:
 Numerically it’s very much like "oldrelax" ("sor" & "crit" comments
 apply)


Lateral mixing schemes:

Lateral mixing applies to both momentum and tracers. One (and only one)
scheme must be enabled.

 "consthmix" is a linear second order mixing with constant  eddy
 coefficients: "am" for momentum & "ah" for heat. The idea is
 to choose them just large enough to suppress small scale numerical
 noise. Typical vaules would be "am" = 1.e7 cm**2/sec for a one degree
 resolution. Usually "ah" is chosen smaller so as not to diffuse
 away frontal features.
```

```
  "biharmonic" is a linear fourth order mixing with constant
  coefficients as in "consthmix": "am" for momentum & "ah" for heat.
  This is more scale selective than "consthmix" (ie: more severly
  damps the small scale features).  "am" should be roughly 10**19
  to 10**20 cm**4/sec (the minus sign is in the eqns) for a 1/3 degree
  resolution.

  "nlhmix" is the non-linear horizontal subgrid-scale mixing after
  Smagorinsky. In this formulation, the horizontal eddy diffusion
  coefficient is proportional to the horizontal grid length and to the
  local deformation field. The coefficients are sensitive to the spatial
  scales of motion and therefore are relatively small in the open ocean,
  where the scales of motion are comparatively large, and are relatively
  large in regions where the scales are comparatively small.


 Vertical mixing schemes:

 Vertical mixing applies to both momentum and tracers. One (and only
 one) scheme must be enabled. The default is for explicit solution.
 Option "implicitvmix" will solve them implicitly.


  "constvmix" uses constant eddy coeficients "fkpm"  & "fkph" for linear
  second order mixing of momentum & heat in the vertical. In the
  explicit case (no "implicitvmix"), convective
  adjustment handles regions of gravitational instability. If
  "implicitvmix" is selected, then the convective section is bypassed
  and the large limits "vdclim" & "vvclim" are used to set the mixing
  coeffiecints which handles the instability.


  "ppvmix" is a vertical mixing scheme based on the Pacanowski &
  Philander richardson mixing scheme (jpo vol 11, #11, 1981).
  This parameterization was designed primarily for equatorial models
  with vertical resolution of about 10 meters between levels in the
  upper ocean (we were most interested in the structure of the
  undercurrent). Previous versions of this code assumed the "t" grid was
  coincident with the "u" grid and gave good results. In the present
  case, we relaxed this assumption. We tried this a few ways and got
  lots of numerical noise, (particularly on the equator off Brazil).
  The present configuration minimizes noise. If noise develops off
  steep shelf breaks it can sometimes be suppressed by turning on a
  bottom drag ("cdbot"). The background mixing "bvdc"  should be
  kept << 1.0 to avoid diffusing the thermocline away on long time
  scales.
  In regions of gravitational instability, vertical mixing limits
  "vdclim" & "vvclim" are used. In the explicit case (no "implicitvmix")
  they must satisfy the "cfl" criterion and may be too small to remove
  the instability. However, the convective adjustment is operative and
  tries to remove the instability. Whether it does or not depends on
  "ncon" which controlls the number of passes through the convective
  section. If "implicitvmix" is used, "vdclim" & "vvclim" are set large
  and the convective adjustment section is bypassed.
  This scheme also assumes the use of equal time steps on the density
  and momentum equations. If a longer time step is desired, try
  using the "implicitvmix" option.

  "tcvmix" invokes the Mellor-Yamada level 2.5 turbulence closure scheme.
  Here the mixing coefficients are a function of the turbulence length scale
  (l), the turbulent kinetic energy (q**2), the analytically derived
  stability factors (Sm & Sh), and the boundary conditions. There are two
  options to compute the length scale:

     "leq" solves an additional equation for q**2l from which
      the length scale may be derived at each level.

      "lalg" obtains the length scale from an algebraic relationship.
      This option may be sufficient for the boundary layer but not
      in cases where there would be multiple turbulent regimes.

  The horizontal diffusion coefficient of turbulent kinetic energy is set
  using the variable "aq" ("ctcmix.h", namelist, & "blkdta.f"). For this
  scheme to be effective "implicitvmix" should be enabled and also the
  vertical resolution should be sufficient.


  "implicitvmix" solves the vertical diffusion term implicitly for all
  vertical mixing schemes. This allows for large mixing coefficients without
   the need to reduce the timestep. With this option enabled, convective
   adjustment is not done and the unstable density profile is mixed using
   large mixing coefficient limits (vvclim & vdclim). For "constvmix" and
   "ppvmix" these limits are set in "blkdta.F" and for "tcvmix" the
    coefficients are computed.
  "implicitvmix" also requires that "aidif" - the implicit factor - be set
  (see "cvmix.h", namelist, and "blkdta.F").

          0 < aidif < .5 , stable if kappa*dt/dz**2 < 1/(2-4*aidif) ;
         .5 < aidif < 1 , always stable
         where kappa is the max vertical mixing coefficient



 Hybrid mixing schemes:

   These are schemes which apply to either momentum or tracers but not
   both.

   For all cases in which "implicitvmix" is not enabled, convective
   adjustment is the default. This mixes tracers vertically in regions
   of gravitational instability but NOT momentum. The effectiveness of
   convective adjustment is controlled by "ncon" which is the number of
   passes through the convective section (use "grep -i -n ncon *.[Ffh]"
   to see its usage). If "impllicitvmix" is enabled, the convective
   adjustment section is bypassed. The assumption is that the vertical
   mixing schemes will handle it.

  "isopycmix" is a scheme in which a mixing tensor is computed from
  the local isopycnal slope and the diffusion of tracers is then conducted
  along that direction.  It therefore influences both the lateral and
  vertical mixing of tracers in the model.  Its use is intended to partially
  mitigate a perceived shortcoming of z-coordinate primitive equation ocean
  models in parameterizing oceanic mixing due to mesoscale motions.  Since
  such mixing is believed to take place along isopycnal surfaces, the
  "isopycmix" option seeks to orient the mixing of tracers along isopycnal
  surfaces rather than purely horizontal surfaces.  As described by Cox
  (Ocean Modelling, issue 74, pg 1-5, June 1987), in addition to specifying
  the mixing coefficient for along isopycnal diffusion "ahisop", a non-zero
  background horizontal mixing coefficient "ah" is often needed to supress
  gridpoint noise.  Additionally, a constraint is placed on the steepest
  isopycnal slope "slmxr" that the scheme will consider when computing the
  components of the mixing tensor.  A typical value of "slmxr" is 100.0,
  which translates into a slope of 1:100.  Steeper slopes would then be
  considered to be 1:100 for the purpose of computing the mixing tensor.
  A vertical mixing scheme must be specified for use with "isopycmix".
```

The zz component of the isopycnal mixing is added to the vertical diffusion coefficient produced by the vertical mixing scheme. Since "isopycmix" only affects tracers, one must also specify a lateral mixing scheme from the above list to be used for momentum. This additional lateral mixing scheme has no effect on tracer diffusion.

Grid options:

"cyclic" is for setting cyclic boundary conditons in a zonal direction. If enabled, anything exiting the eastern side of the grid comes back in through the western side. If not enabled, then solid walls are assumed on the eastern and western boundaries of the model.

"symmetry" is for setting a symmetric condition across the northern boundary of the model. This assumes the line of symmetry is at the equator which should be row jmt-1 on the velocity grid.
The condition is :
Tracers at row jmt on the "t" grid = tracers at row jmt-1 on the "t" grid.
meridional velocity at row jmt-1 on the "u" grid is zero.
meridional velocity at row jmt on the "u" grid is the negative of the meridional velocity at row jmt-2 on the "u" grid.
stream funcion at row jmt = negative of the stream function at row jmt-1 on the "t" grid.
If not enabled, solid walls exist at the northern and southern boundaries.

Optimizations:

"skipland" allows calculations to be done over blocks of ocean while skipping land points. The trade off here is the time saved by not doing the caluclation over land versus the time used in starting & stoping a lot of vectors. It’s really a function of land mass geometry. A global ocean would be a good candidate for this option, but a idealized basin woulod not.

"multitasking" allows the slabs to be divided into tasks. Each task contains approximately the same number of slabs and there should ideally be one task per processor. This is all handled by simply spedcifying "ntasks" to be the number of processors (we’ve defaulted it to 8 in "blkdta.F"). Since all tasks are independent, all tasks can be worked on simultaneously. This reduces the total wall clock time for the job but NOT the total cpu time.
Note: "ntasks" may be set > 1 even when unitasking (one processor) but this is not recommended since there is extra work being done at the interfaces between tasks. Also, on diagnostic, tracer averaging, and data saving time steps, MOM reverts to one task of jmt-2 slabs so effectively multitasking is shut down for these time steps.
"multitasking" currently doesn’t work with the "diskless" (see below) option because only two ("ntlev"=2) time levels are used for slabs on simulated disk to save memory and this is incompatible with "ntasks" > 1. This condition may, in principle, be removed by setting "ntlev" = 3 when enabeling the "diskless" option. We haven’t explored it yet.
So far we have gotten to the point where all combinations of unitasked, multitasked, & autotasked runs give the same answers down to the last hex digit.
To really utilize this feature on the CRAY YMP, the SSD must be used for the slabs (unless large memory & "diskless" is used). We have not

yet added the best i/o scheme for the SSD, so the wall clock times are higher than they should be.

"timing" gives cp & wall clock times for running on a CRAY YMP. Additionally, the time per grid point per time step is calculated.

Filtering:

Filtering is used to combat the effect of the convergence of meridians (shrinking longitudinal grid spacing) near the poles. The problem is that the grid spacing severly limits the time step (especially when using large density time steps) due to the "CFL" criterian. Filtering relaxes this constraint by truncating high wave numbers.

"fourfil" is a fourier filter which acts on longitudinal strips of ocean points. The number of wave numbers to be allowed at each latitude is decided upon & wavenumbers above are truncated (without using any reasonable window). The best that can be said for it is that it is time consuming, messes the solution up & should only be done as a last resort. It is provided as a backward compatability feature for the COX code. Note: Both in MOM and in the COX code, the amount of filtering depends on the number of ocean points within the longitudinal stip. This actually leads to possibly different truncations at the same latitude!

"firfil" is a simple finite impluse response filter based on the familiar 1/4, 1/2, 1/4 weights (the response function is a cosine). It comes in two flavors: One to with symmetric boundary conditions (ie: for tracers) & one for asymmetric boundary conditons (ie: momentum). It also works on longitudinal strips of ocean points & may be applied an arbitrary number of times for an arbitrary number of latitudes. The best that can be said for it is that the reason for using it is the same as for "fourfil", it is quite fast, & we really haven’t done any long running comparisons to comapre it to "fourfil".

Miscellaneous:

"keepterms" allows the use of arrays instead of statement functions for component terms in the equations. This may be desirable when using component terms over & over for some purpose. The trade off is the extra memory needed for the arrays versus the extra time needed to recalculate the statement functions. See "size.F" to look at resource requirements.

"nohilats" stands for no high latitudes. If the latitudinal domain of the model is limited to equatorial redions, for instance, then the metric nonlinear terms in the momentum equations may be dropped. It will save some time.

"restorst" stands for restore surface tracers to some prescribed values by a newtonian damping term. In the test case, the restoration is globally back to the annual mean conditions on a time scale of 50 days. Without much work, this option could easily be extended to damp certain regions while leaving others alone by simple making the damping time scale a function of latitude (for instance).

"testcfl" tests whether any velocity exceeds the local "cfl" criterion by a factor of "cflcrt" (see "blkdta.F"). It will print out the places where the "cfl" criterion is most nearly met on diagnostic time steps. If the "cflcrt" factor is exceeded on any time step, MOM

```
  will stop & print out the surronding variables. It’s not meant to be
  left on all the time. If MOM blows up, it can quickly point to where
  it’s  happening.


 I/O options:

   "diskless" simulates disk usage using an array in memory. Since
   there is no "wall clock" time lost during disk transfers (because
   they are memory to memory transfers), this method gives the best
   efficiency (cp time / total time). The down side is that the model
   may not be able to fit into available memory. As a memory saving
   feature, "diskless" keeps only two time levels of prognostic
   variables instead of the usual three. See "size.F"  to look at
   resource requirements. Also see "multitasking"


   "crayio" for now uses the "getwa/putwa" routines. If the disk units
   are set up on "non SSD" type disk, then the efficiency (cp time /
   total time) will be bad. This shows the mismatch in speed between
   computation & disk use. SSD, however, is really the way to go. It
   can be looked at as an extended memory & the efficiency is much
   better.

   "fio" is for fortran direct access i/o. The comments from "crayio"
   apply here also.

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(ix) Cray Y-MP Considerations & Multi-tasking:

 This is not the place to weigh advantages and disadvantages of
multi-tasking on the CRAY Y-MP, however a few things should be
stated. When MOM (with a 1 degree longitudinal resolution with
options "firfil" and "congrad") was autotasked on a 4 processor Y-MP,
the wall clock time was 2.5 times less than the wall clock on a
single processor. That translates to 80% parallelism. However,
CP time increased by 50%. That’s the overhead of synchronization
and idle processor CP time charged to the user. Higher resolutions
will appear more parallel, because the autotasking overhead accounts
for a smaller fraction of the total work done.

The MOM code’s "multitasking" option is still being tested. We expect
very high parallelism with this approach. However, it will have
drawbacks. One is memory explosion (a very large increase in
memory requirements) due to copies of task common needed for
each processor. Another is increased possibility for introducing bugs,
because one has to be much more aware of where variables can be
changed (i.e., setting private variables outside of parallel structures).
In the future, we will improve model performance using both
techniques.

It should be noted that unitasked, autotasked & multitasked versions
of the model give identical anwsers down to the last hex digit! As
time permits, we will continue to look into the performance issues for
each approach.

On one processor of a CRAY Y-MP, MOM runs about 2.6 times faster
than its counterpart on the CYBER 205 (full precision). To give an
overall speed improvement is misleading, since it depends on how
the model is configured. For instance, if options "firfil" and "congrad"
are used it can be 5 times faster than the CYBER 205. Compared to
the Cox model on the CRAY Y-MP on one processor, MOM runs at the
```

same speed. However, if the "skipland" option is used, MOM can be roughly
25% faster for high resolution (1 degree) world ocean models. On multiple
processors the "autotasked" Cox code is only 37% parallel as compared to
65% for the comparable "autotasked" version of MOM (as measured by one of
our early 6 degree global test cases).

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(x) MOM’s Simplified Flow Chart:

```
OCEAN (main program)
      BLKDTA (initializes variables)
      GRIDS (sets up the grids)
            MESH (utility for setting up grids)
      CMESH (utility for setting up grids)
      INDP (utility for finding nearest grid point)
      DOCMNT (documents the run)
      CNVMIX (sets coefficients for constant vertical mixing)
      OSTART (i/o routine)
      OCN1ST (sets up initial conditions)
            TOPOG (sets up idealized world topography)
            OPUT (i/o routine)
      OPUT (i/o routine)
      IPERIM  (calculate island perimeters)
      REG1ST (set up for regional averages of tracers)
      FINDEX (calculates indices for filtering)
      OGET (i/o routine)
      CHECKS (test for consistancy of configuration)
      TMNGR (time manager called once per time step)
            TMNSET (set time dependent logical switches)
            INDP (utility for finding nearest grid point)
      STEP (cycles latitude rows between disk & memory slab window)
            OPUT (i/o routine)
            OFIND (i/o routine)
            OGET (i/o routine)
            GETVAR (gets disk data into memory slab window and adds
                    external mode to internal mode velocities)
            DELSQ (calculates quantites for biharmonic mixing)
            SETVBC (set vertical boundary conditions)
            PPMIX  (Pacanowski/Philander richardson mixing)
            TCMIX  (Mellor Yamada turbulence closure mixing)
            ISOP0  (Isopycnal mixing)
            CFL    (for cfl monitoring)
            CLINIC (calculates internal mode velocities)
                  STATE (density calculation)
                  NLMIX (nonlinear Smagorinsky mixing)
                  INVTRI (for implicit vertical diffusion)
                  Filtering (FIRFIL or FOURFIL filter u & v at high
                  latitudes)
            TRACER (calculates tracers)
                  REGION (does basin averages of tracers)
                  STATEC (density calculation for vertical stability)
                  INVTRI (for implicit vertical diffusion)
                  Filtering (FIRFIL or FOURFIL filter tracers at high
                  latitudes)
            DIAG (energy diagnostic and term balance calculations)
                  INDP (utility for finding nearest grid point)
                  MATRIX (printouts)
      VORT (calculates vorticity)
            Filtering (FIRFIL or FOURFIL filter at high latitudes)
      Poisson Solver (RELAX, HYPER, CONGR5, or CONGR9)
            OPUT (i/o routine)
            OGET (i/o routine)
            OFIND (i/o routine)
      DIAG2 (energy diagnostic & term balance printouts)
           OCLOSE (i/o routine)



EQSTAT (main program)
      Density Routine (KNUEKM or UNESCO)
      POTEM (returns potential temperature)
      Density Routine (KNUEKM or UNESCO)
      LSQSL2 (iteratively computes a least squares fit)


DEPTHS (main program)


SIZE   (main program)


<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
```

(xi) Preview of what’s next:

There is no time schedule for the next release ... but here are some of the things we’re working on:

1) An airsea interface for coupling MOM to atmospheric models.

2) The astronomical relationship between earth & sun for defining the solar radiation at the top of the atmosphere at any point on the earth and at any time (within a few million years(+/-)) along with a surface heat budget (longwave, shortwave, sensible, latent).

3) Short wave penetration (beyond the first vertical level) for solar radiation

4) Free surface option

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>

(xii) Addresses For Correspondence:

Please report errors, bugs, comments, and suggestions to us.

	e-mail address	phone number
Keith Dixon	kd@gfdl.princeton.edu	(609) 452-6574
Ron Pacanowski	rcp@gfdl.princeton.edu	(609) 452-6527
Tony Rosati	ar@gfdl.princeton.edu	(609) 452-6584

Omnet/Sciencenet mailbox: K.DIXON   fax: (609) 987-5063

Geophysical Fluid Dynamics Lab
US Dept. of Commerce / N.O.A.A.
Princeton Univ. - P.O. Box 308
Princeton, NJ 08542

If you are considering using MOM, please fill-out this form and send it to one of the above addresses. If you don’t, we will not be able to let you know of new releases.

```
-----------------------------------------------------------------

GFDL MOM
NAME:         ________________________________________________
ADDRESS:      ________________________________________________
              ________________________________________________
              ________________________________________________
              ________________________________________________
       PHONE: _____________________ FAX: _____________________
E-MAIL ADDRESS: ________________________________________________
On what type of computer(s) would you expect to run this code?
__________________________________________________________
On what type of computer(s) would you expect to manage this
source code? ________________________________________________

-----------------------------------------------------------------

<> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
```