CDFTOOLS: a fortran 90 package of programs and libraries for diagnostic of the DRAKKAR OPA9 output. Part II : Programmer guide

J.M. Molines *

Last update: Rev: 13 Date: 2007-04-16 18:02:53 +0200 (Mon, 16 Apr 2007)

Introduction

This document is a technical description of the different functions and subroutines which belong to cdfio.f90 and eos.f90 fortran 90 modules. They are used basically in the core of the cdftools program either to perform the Netcdf I/O or to compute the equation of state for sea water.

1 cdfio module

TYPE variable

Structure: We defined a derived type for managing the variables attribute. It is defined as follow:

```
TYPE, PUBLIC ::   variable
    character(LEN=80)::  name
    character(LEN=80):: units
    real(kind=4)    :: missing_value
    real(kind=4)    :: valid_min
    real(kind=4)    :: valid_max
    real(kind=4)    :: scale_factor=1.
    real(kind=4)    :: add_offset=0.
    real(kind=4)    :: savelog10=0.
    character(LEN=80):: long_name
    character(LEN=80):: short_name
    character(LEN=80):: online_operation
    character(LEN=80):: axis
    character(LEN=80):: precision='r4'  ! possible values are i2, r4, r8
END TYPE variable
```

Purpose: This is used in the cdftools to avoid the former 'att.txt' file which held the variable attributes. Now, each program needing variables output in a netcdf file, must use a structure (or an array of structure) defining the name and attributes of the variable. This structure or array of structure is passed as argument to the following functions: `createvar, putatt, getvarname`

Example: Self explaining example from cdfpvor.f90:

*Laboratoire des Ecoulements Géophysiques et Industriels, CNRS UMR 5519, Grenoble, France

```
....
  TYPE(variable), DIMENSION(3) :: typvar          !: structure for attribute
....
  ! define variable name and attribute
  typvar(1)%name= 'vorelvor' ; typvar(2)%name= 'vostrvor';  typvar(3)%name= 'vototvor'
  typvar%units='kg.m-4.s-1'   ; typvar%missing_value=0.
  typvar%valid_min= -1000. ;  typvar%valid_max= 1000.
  typvar(1)%long_name='Relative_component_of_Ertel_PV'
  typvar(2)%long_name='Stretching_component_of_Ertel_PV'
  typvar(3)%long_name='Ertel_potential_vorticity'
  typvar(1)%short_name='vorelvor';  typvar(2)%short_name='vostrvor'
  typvar(3)%short_name='vototvor'
  typvar%online_operation='N/A';  typvar%axis='TZYX'
  ncout =create(cfileout, cfilet, npiglo,npjglo,npk)
  ierr= createvar   (ncout ,typvar,3, ipk,id_varout )
  ierr= putheadervar(ncout, cfilet,npiglo,npjglo,npk)
 ....
```

INTERFACE putvar

Generic interface `INTERFACE putvar`

```
   MODULE PROCEDURE putvarr4, putvari2, putvarzo
END INTERFACE
```

Purpose: This generic interface re-direct putvar call to either putvarr4 for real*4 input array, putvari2 for integer*2 input array, or to putvarzo for degenerated 3D-2D arrays corresponding to zonal integration. It also redirect putvar to the reputvarr4 function, which allows to rewrite a variable in an already existing file.

Example: ierr = putvar(ncout, id_varout(jvar) ,i2d, jk, npiglo, npjglo)
...
ierr = putvar(ncout, id_varout(jvar) ,sal, jk, npiglo, npjglo)
Example for reputvarr4
istatus=putvar(cfile,'Bathymetry',jk,npiglo,npjglo, kimin=imin, kjmin=jmin, ptab)

FUNCTION closeout(kout)

Arguments: INTEGER, INTENT(in) :: kout. Netcdf ID of the file to be closed.

Purpose: Close an open netcdf file, specified by its ID

Example: istatus = closeout(ncout)

FUNCTION copyatt(cdvar, kidvar, kcin, kcout)

Arguments:
CHARACTER(LEN=*), INTENT(in) :: cdvar !: Name of the variable
INTEGER,INTENT(in) :: kidvar !: var id of variable cdvar
INTEGER,INTENT(in) :: kcin !: ncid of the file where to read the attributes
INTEGER,INTENT(in) :: kcout !: ncid of the output file. INTEGER :: copyout !: function return value: return an error status.

Purpose: Copy all the attributes for one variable, taking the example from another file, specified by its ncid. Return the status of the function. If $\neq 0$, indicates an error.

Example:

```
istatus = NF90\_DEF\_VAR(icout,'nav\_lon',NF90\_FLOAT,nvdim(1:2),id\_lon)
istatus = copyatt('nav\_lon',id\_lon,ncid,icout)
```

Remark: This function is used internally to cdfio, in the function create.

FUNCTION create(cdfile, cdfilref ,kx,ky,kz, cdep)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile !: name of file to create
CHARACTER(LEN=*), INTENT(in) :: cdfilef !: name of file used as reference for attributes
INTEGER,INTENT(in) :: kx, ky, kz !: value of the dimensions x, y and z (depth)
CHARACTER(LEN=*), OPTIONAL, INTENT(in) :: cdep !: name of depth variable if differs from cdfile
INTEGER :: create !: function return value : the ncid of created variable.

Purpose: Create a netcdf file (IOIPSL type) and copy attributes for nav_lon, nav_lat, depth and time_counter from the reference file given in argument. It is supposed that the reference file is also IOIPSL compliant. For historical reason, there many different names for the depth dimension and variable. If we want to create the new data set with a depth name that differs from the reference file, the cdep optional argument can be used. The return value of the fuction is the ncid of the file just created.

Example:

```
 ! create output fileset
  cfileout='cdfmoy.nc'
  cfileout2='cdfmoy2.nc'
  ! create output file taking the sizes in cfile

  ncout =create(cfileout, cfile,npiglo,npjglo,npk)
  ncout2=create(cfileout2,cfile,npiglo,npjglo,npk)
```

or

```
 ! create output fileset
  cfileout='w.nc'
  ! create output file taking the sizes in cfile

  ncout =create(cfileout, cdfile,npiglo,npjglo,npk,'depthw')
```

FUNCTION createvar (kout,ptyvar,kvar,kpk, kidvo)

Arguments:

```
! * Arguments
INTEGER, INTENT(in) :: kout, kvar
INTEGER, DIMENSION(kvar), INTENT(in) :: kpk
INTEGER, DIMENSION(kvar), INTENT(out) :: kidvo
INTEGER :: createvar
TYPE (variable), DIMENSION(kvar) ,INTENT(in) :: ptyvar
```

Purpose: Creates the kvar variables defined by the ptyvar and kpk arrays. Save the varid's in kidvo.

Example:

```
ncout =create(cfileout, cfile,npiglo,npjglo,npk)
ncout2=create(cfileout2,cfile,npiglo,npjglo,npk)

ierr= createvar(ncout ,typvar,  nvars, ipk, id_varout )
ierr= createvar(ncout2, typvar2, nvars, ipk, id_varout2)
```

FUNCTION getatt(cdfile,cdvar,cdatt)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdatt, & ! attribute name to look for
& cdfile, & ! file to look at
& cdvar
REAL(KIND=4) :: getatt

Purpose: Return a REAL value with the values of the attribute cdatt for all the variable cdvar in cdfile

Example:

```
! get missing_value attribute
spval = getatt( cfile,'votemper','missing_value')
```

FUNCTION getdim (cdfile,cdim_name,cdtrue,kstatus)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile , & ! File name to look at
& cdim_name ! dimension name to look at
CHARACTER(LEN=80),OPTIONAL, INTENT(out) :: cdtrue ! full name of the read dimension
INTEGER, OPTIONAL, INTENT(out) :: kstatus ! status of the nf inquire
INTEGER :: getdim ! the value for dim cdim_name, in file cdfile

Purpose: Return the INTEGER value of the dimension identified with cdim_name in cdfile

Example:

```
 npiglo= getdim (cfile,'x')
 npjglo= getdim (cfile,'y')
 npk   = getdim (cfile,'depth',kstatus=istatus)
....
idum=getdim(cdfilref,'depth',cldep)  ! return in cldep the name of the dim
                                     ! whose 'depth' is used as proxy
```

FUNCTION getvdim (cdfile,cdvar)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! File name to look at
CHARACTER(LEN=*), INTENT(inout) :: cdvar ! variable name to look at.
INTEGER :: getvdim ! number of dim for cdvar

Purpose: Return the number of dimension for variable cdvar in cdfile.

If *cdvar* is not found in *cdfile*, then a list a available variables is displayed and the user is asked to choose the required one. In this case, *cdvar* is updated to the choosen variable name, and is made available to the calling program.

This function is intended to be used with prognostic variables of the model, which are defined in the file either as [TZXY] (3D variable) or as [TXY] (2D variable). The time dimension is not considered. Erroneous results are produced if the variables is [ZXY] or [XY].

Example:

```
...
cvar='variablex'
nvdim  = getvdim(cfilev,cvar)
IF (nvdim == 2 ) nvpk = 1   ! 2D variable ==> 1 level
IF (nvdim == 3 ) nvpk = npk ! 3D variable ==> npk levels
PRINT *, TRIM(cvar),' has ', nvdim,'  dimensions
...
```

FUNCTION getipk (cdfile,knvars,cdep)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! File to look at
INTEGER, INTENT(in) :: knvars ! Number of variables in cdfile
CHARACTER(LEN=*), OPTIONAL, INTENT(in) :: cdep ! optional depth dim name
INTEGER, DIMENSION(knvars) :: getipk ! array (variables) of levels

Purpose: return the number of levels for all the variables in cdfile. Return 0 if the variable in a vector.
returns npk when 4D variables (x,y,z,t)
returns 1 when 3D variables (x,y, t)
returns 0 when other (vectors)
If cdep argument is present, use it as the depth dimension name (instead of default 'dep')

Example:

```
  ! ipk gives the number of level or 0 if not a T[Z]YX  variable
  ipk(:)     = getipk (cfile,nvars)
...
  ipk(:)     = getipk (cisofile, nvars, cdep=sigmalevel)
```

FUNCTION getnvar (cdfile)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! file to look at
INTEGER :: getnvar ! return the number of variables

Purpose: Return the number of variables in cdfile

Example:

```
nvars = getnvar(cfile)
PRINT *,' nvars =', nvars
```

FUNCTION getvar (cdfile,cdvar,klev,kpi,kpj,kimin,kjmin,ktime)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the 2D variable
INTEGER, OPTIONAL, INTENT(in) :: klev ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kimin,kjmin ! Optional : set initial point to get
INTEGER, OPTIONAL, INTENT(in) :: ktime ! Optional variable. If missing 1 is assumed
REAL(KIND=4), DIMENSION(kpi,kpj) :: getvar ! 2D REAL 4 holding variable field at klev

Purpose: Return the 2D REAL variable cdvar, from cdfile at level klev.
kpi,kpj are the horizontal size of the 2D variable

Example:

```
        v2d(:,:)= getvar(cfile, cvarname(jvar), jk ,npiglo, npjglo )
...
        jt=25
        v2d(:,:)= getvar(cfile, cvarname(jvar), jk ,npiglo, npjglo ,ktime=jt)
```

Remark: The optional keyword ktime is **NOT YET** to be used. (working on it).

FUNCTION getvarxz (cdfile,cdvar,kj,kpi,kpz,kimin,kkmin,ktime)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpi,kpz ! size of the 2D variable
INTEGER, INTENT(in) :: kj ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kimin,kkmin ! Optional set initial point to get
INTEGER, OPTIONAL, INTENT(in) :: ktime ! Optional variable. If missing 1 is assumed
REAL(KIND=4), DIMENSION(kpi,kpz) :: getvarxz ! 2D REAL 4 holding variable x-z slab at kj

Purpose: Return the 2D REAL variable x-z slab cvar, from cdfile at j=kj
kpi,kpz are the size of the 2D variable. The time frame can be specified using the optional argument ktime.

Example:

```
      v2d(:,:)= getvarxz(cfile, cvarname(jvar), jj ,npiglo,npk, imin, kmin )
...
      v2d(:,:)= getvarxz(cfile, cvarname(jvar), jj ,npiglo,npk, imin, kmin, ktime=jt)
```

FUNCTION getvaryz (cdfile,cdvar,ki,kpj,kpz,kjmin,kkmin,ktime)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpj,kpz ! size of the 2D variable
INTEGER, INTENT(in) :: ki ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kjmin,kkmin ! Optional set initial point to get
INTEGER, OPTIONAL, INTENT(in) :: ktime ! Optional variable. If missing 1 is assumed
REAL(KIND=4), DIMENSION(kpj,kpz) :: getvaryz ! 2D REAL 4 holding variable x-z slab at kj

Purpose: Return the 2D REAL variable y-z slab cvar, from cdfile at i=ki
kpj,kpz are the size of the 2D variable. The time frame can be specified using the optional argument ktime.

Example:

```
      v2d(:,:)= getvaryz(cfile, cvarname(jvar), ji ,npjglo,npk,jmin,kmin )
 ...
      v2d(:,:)= getvaryz(cfile, cvarname(jvar), ji ,npjglo,npk,jmin,kmin, ktime=jt )
```

SUBROUTINE gettimeseries (cdfile, cdvar, kilook, kjlook,klev)

Arguments:

IMPLICIT NONE
CHARACTER(LEN=*),INTENT(in) :: cdfile, cdvar
INTEGER,INTENT(in) :: kilook,kjlook
INTEGER, OPTIONAL, INTENT(in) :: klev

Purpose: Display a 2 column output (time, variable) for a given variable of a given file at a given point.

Example:

```
   CALL gettimeseries(cfile,cvar,ilook,jlook,klev=ilevel)
...
   CALL gettimeseries(cfile,cvar,ilook,jlook)
```

FUNCTION getvar1d (cdfile,cdvar,kk,kstatus)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kk ! size of 1D vector to be returned
INTEGER, OPTIONAL, INTENT(out) :: kstatus ! return status concerning the variable existence
REAL(KIND=4), DIMENSION(kk) :: getvar1d ! real returned vector

Purpose: Return 1D variable cdvar from cdfile, of size kk

Example:

```
   tim=getvar1d(cfile,'time_counter',1)
....
       z1d=getvar1d(cdfile,'deptht',kpk,idept)
       IF ( idept /= NF90_NOERR ) THEN
         z1d=getvar1d(cdfile,'depthu',kpk,idepu)
         IF ( idepu /= NF90_NOERR ) THEN
           z1d=getvar1d(cdfile,'depthv',kpk,idepv)
           IF ( idepv /= NF90_NOERR ) THEN
             z1d=getvar1d(cdfile,'depthw',kpk,idepv)
             IF ( idepw /= NF90_NOERR ) THEN
               PRINT *,' No depth variable found in ', TRIM(cdfile)
               STOP
             ENDIF
           ENDIF
         ENDIF
       ENDIF
```

This last example shows how to use the optional argument kstatus in order to figure out which is the real name of the depth variable.

FUNCTION getvare3 (cdfile,cdvar,kk)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kk ! size of 1D vector to be returned
REAL(KIND=4), DIMENSION(kk) :: getvare3 ! return e3 variable form the coordinate file

Purpose: Special routine for e3, which in fact is a 1D variable but defined as e3 (1,1,npk,1) in coordinates.nc (!!)

Example:

```
gdepw(:) = getvare3(coordzgr, 'gdepw',npk)
e3t(:)   = getvare3(coordzgr, 'e3t', npk )
```

FUNCTION getvarname (cdfile, knvars,ptypvar)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile ! name of file to work with
INTEGER, INTENT(in) :: knvars ! Number of variables in cdfile
TYPE (variable), DIMENSION (knvars) :: ptypvar ! Retrieve variables attributes CHARACTER(LEN=80), DIMENSION(knvars) :: getvarname ! return an array with the names of the variables

Purpose: Return a character array with the knvars variable names, and the ptypvar structure array filled with the attribute read in cdfile

Example:

```
cvarname(:)=getvarname(cfile,nvars,typvar)
! typvar is output from getvarname
```

FUNCTION putatt (tyvar,kout,kid)

Arguments:

TYPE (variable) ,INTENT(in) :: tyvar INTEGER, INTENT(in) :: kout ! ncid of the output file
INTEGER, INTENT(in) :: kid ! variable id
INTEGER :: putatt ! return variable : error code.

Purpose: Uses the structure tyvar for setting the variable attributes for kid and write them in file id kout.

Example:

```
! add attributes
istatus = putatt(ptyvar(jv), kout,kidvo(jv))
```

Remark: This is almost an internal routine called by createvar.

FUNCTION putheadervar(kout, cdfile, kpi,kpj,kpk,pnavlon, pnavlat,pdep,cdep)

Arguments:

INTEGER, INTENT(in) :: kout ! ncid of the outputfile (already open)
CHARACTER(LEN=*), INTENT(in) :: cdfile ! file from where the headers will be copied
INTEGER, INTENT(in) :: kpi,kpj,kpk ! dimension of nav_lon,nav_lat (kpi,kpj), and depht(kpk)
REAL(KIND=4), OPTIONAL, DIMENSION(kpi,kpj) :: pnavlon, pnavlat ! to get rid of nav_lon , nav_lat of cdfile
REAL(KIND=4), OPTIONAL,DIMENSION(kpk), INTENT(in) :: pdep ! dep array if not on cdfile
CHARACTER(LEN=*), OPTIONAL, INTENT(in) :: cdep ! optional name of vertical variable
INTEGER :: putheadervar ! return status

Purpose: Copy header variables from cdfile to the already open ncfile (ncid=kout)

If the 2 first optional arguments are given, they are taken for nav_lon and nav_lat, instead of those read in file cdfile. This is usefull for f-points results whne no basic "gridF" files exist. If the third optional argument is given, it is taken as the depht(:) array in place of the the depth read in cdfile. If all 3 optional arguments are used, cdfile will not be used and a dummy argument can be passed to the function instead. If optional argument cdep is used, it is then used as the name for the variable associated with the vertical dimension.

Example:

```
ierr= putheadervar(ncout , cfile, npiglo, npjglo, npk)
ierr= putheadervar(ncout2, cfile, npiglo, npjglo, npk)
```

or

```
ierr= putheadervar(ncout , cfile, npiglo, npjglo, npk, glamf, gphif )
```

or

```
ierr= putheadervar(ncout , 'dummy', npiglo, npjglo, npk, glamt, gphit, gdepw )
```

FUNCTION putvar(kout, kid,ptab, klev, kpi, kpj)

Arguments:

INTEGER, INTENT(in) :: kout , & ! ncid of output file
& kid ! varid of output variable
REAL(KIND=4), DIMENSION(kpi,kpj),INTENT(in) :: ptab ! 2D array to write in file
INTEGER, INTENT(in) :: klev ! level at which ptab will be written
INTEGER, INTENT(in) :: kpi,kpj ! dimension of ptab
INTEGER :: putvar ! return status

Purpose: copy a 2D level of ptab in already open file kout, using variable kid

Example:

```
ierr = putvar(ncout, id_varout(jvar) ,rmean, jk, npiglo, npjglo)
```

Remark: Putvar is a generic interface, as explained above. For the interface with reputvar, the syntax is shown below.

FUNCTION reputvarr4 (cdfile,cdvar,klev,kpi,kpj,kimin,kjmin, ktime,ptab)

Arguments:

CHARACTER(LEN=*), INTENT(in) :: cdfile, & ! file name to work with
& cdvar ! variable name to work with
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the 2D variable
INTEGER, OPTIONAL, INTENT(in) :: klev ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: kimin,kjmin ! Optional variable. If missing 1 is assumed
INTEGER, OPTIONAL, INTENT(in) :: ktime ! Optional variable. If missing 1 is assumed
REAL(KIND=4), DIMENSION(kpi,kpj) :: ptab ! 2D REAL 4 holding variable field at klev

Purpose: Change an existing variable in inputfile

Example:

```
ierr = putvar(cfile, 'votemper', 4, npiglo,npjglo, kimin=10, kjmin=200, temperature)
```

Remark: With this function, the input file is modified !

FUNCTION putvar1d(kout,ptab,kk,cdtype)

Arguments:

INTEGER, INTENT(in) :: kout ! ncid of output file
REAL(KIND=4), DIMENSION(kk),INTENT(in) :: ptab ! 1D array to write in file
INTEGER, INTENT(in) :: kk ! number of elements in ptab
CHARACTER(LEN=1), INTENT(in) :: cdtype ! either T or D (for time or depth)
INTEGER :: putvar1d ! return status

Purpose: Copy 1D variable (size kk) hold in ptab, with id kid, into file id kout

Example:

```
                ierr=putvar1d(ncout,timean,1,'T')
                ierr=putvar1d(ncout2,timean,1,'T')
...
                istatus = putvar1d(kout,depw(:),kpk,'D')
```

SUBROUTINE ERR_HDL(kstatus)

Arguments:
INTEGER, INTENT(in) :: kstatus

Purpose: Error handler for NetCDF routine. Stop if kstatus indicates error conditions. Else indicate the error message.

Example:

```
CALL ERR_HDL(istatus)
```

2 eos module

FUNCTION sigma0 (ptem, psal, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: ptem, psal ! Temperature and Salinity arrays
INTEGER,INTENT(in) :: kpi,kpj !: dimension of 2D arrays
REAL(KIND=8), DIMENSION(kpi,kpj) :: sigma0 ! Potential density

Purpose: Compute the potential volumic mass (Kg/m3) from potential temperature and salinity fields

Example:

FUNCTION sigmai(ptem, psal, pref, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: ptem, psal ! Temperature and Salinity arrays
REAL(KIND=4), INTENT(in) :: pref !: reference pressure (dbar)
INTEGER,INTENT(in) :: kpi,kpj !: dimension of 2D arrays
REAL(KIND=8), DIMENSION(kpi,kpj) :: sigmai ! Potential density a level pref

Purpose: Compute the potential volumic mass (Kg/m3) from potential temperature and salinity fields at reference level specified by $pref$.

Example:

FUNCTION eosbn2 (ptem, psal, pdep,pe3w, kpi,kpj,kup,kdown)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj,2), INTENT(in) :: ptem, psal ! temperature and salinity arrays
! (2 levels, only)
REAL(KIND=4) :: pdep ! depthw (W points)
REAL(KIND=4), DIMENSION(kpi,kpj), INTENT(in) :: pe3w ! vertical scale factor at W points
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the grid
INTEGER, INTENT(in) :: kup,kdown ! index cdfmeannd lower layer
! for the actual level
REAL(KIND=4), DIMENSION(kpi,kpj) :: eosbn2 ! result interpolated at T levels

Purpose: Compute the local Brunt-Vaisala frequency

Example:

```
DO jk = npk-1, 2, -1
   PRINT *,'level ',jk
   zmask(:,:)=1.
   ztemp(:,:,iup)= getvar(cfilet, 'votemper',  jk-1 ,npiglo, npjglo)
   WHERE(ztemp(:,:,idown) == 0 ) zmask = 0
   zsal(:,:,iup) = getvar(cfilet, 'vosaline',  jk-1 ,npiglo,npjglo)

   gdepw(:,:) = getvar(coordzgr, 'gdepw', jk, 1, 1)
   e3w(:,:)   = getvar(coordzgr, 'e3w_ps', jk,1, 1 )

   zwk(:,:,iup) = eosbn2 ( ztemp,zsal,gdepw(1,1),e3w, npiglo,npjglo , &
       iup,idown)* zmask(:,:)
   ! now put zn2 at T level (k )
   WHERE ( zwk(:,:,idown) == 0 )
      zn2(:,:) =  zwk(:,:,iup)
   ELSEWHERE
      zn2(:,:) = 0.5 * ( zwk(:,:,iup) + zwk(:,:,idown) ) * zmask(:,:)
   END WHERE

   ierr = putvar(ncout, id_varout(1) ,zn2, jk, npiglo, npjglo )
   itmp = idown ; idown = iup ; iup = itmp

END DO  ! loop to next level
```

FUNCTION albet (ptem, psal, pdep, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj,2), INTENT(in) :: ptem, psal ! temperature and salinity arrays
! (2 levels, only)
REAL(KIND=4) :: pdep ! depthw (W points)
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the grid
! for the actual level
REAL(KIND=4), DIMENSION(kpi,kpj) :: albet ! result interpolated at T levels

Purpose: Compute the ratio alpha/beta

Method: Use the equation of the OPA code (Mc Dougall, 1987)

Remark: This is a function that may be used together with beta for computing the buoyancy flux, from forcing fields.

FUNCTION beta (ptem, psal, pdep, kpi,kpj)

Arguments:

REAL(KIND=4), DIMENSION(kpi,kpj,2), INTENT(in) :: ptem, psal ! temperature and salinity arrays
! (2 levels, only)
REAL(KIND=4) :: pdep ! depthw (W points)
INTEGER, INTENT(in) :: kpi,kpj ! horizontal size of the grid
! for the actual level
REAL(KIND=4), DIMENSION(kpi,kpj) :: beta ! result interpolated at T levels

Purpose: Compute the beta coefficient

Method: Use the equation of the OPA code (Mc Dougall, 1987)

Remark: This is a function that may be used together with albet for computing the buoyancy flux, from forcing fields.

Contents

1 cdfio module **1**
TYPE variable . . . 1
INTERFACE putvar . . . 3
closeout . . . 3
copyatt . . . 3
create . . . 4
createvar . . . 5
getatt . . . 6
getdim . . . 7
getvdim . . . 8
getipk . . . 9
getnvar . . . 10
getvar . . . 11
getvarxz . . . 12
getvaryz . . . 13
gettimeseries . . . 14
getvar1d . . . 15
getvare3 . . . 16
getvarname . . . 17
putatt . . . 18
putheadervar . . . 19
putvar . . . 20
reputvarr4 . . . 20
putvar1d . . . 21
ERR_HDL . . . 22

2 eos module **23**
sigma0 . . . 23
sigmai . . . 23
eosbn2 . . . 24
albet . . . 25
beta . . . 25

Index

albet, 25

beta, 25

closeout, 3
copyatt, 3
create, 4
createvar, 5

eosbn2, 24
ERR_HDL, 22

getatt, 6
getdim, 7
getipk, 9
getnvar, 10
gettimeseries, 14
getvar, 11
getvar1d, 15
getvare3, 16
getvarname, 17
getvarxz, 12
getvaryz, 13
getvdim, 8

INTERFACE putvar, 3

putatt, 18
putheadervar, 19
putvar, 20
putvar1d, 21

reputvarr4, 20

sigma0, 23
sigmai, 23

TYPE variable, 1