# In-code documentation for CVMix

Many contributors from GFDL, LANL, and NCAR
*GFDL, LANL, and NCAR*

January 16, 2013

## Contents

# 1 Routine/Function Prologues

### 1.0.1 cvmix_BL_driver_pointers

A stand-alone driver for the CVMix package. This particular driver generates the Bryan-Lewis coefficients in two columns, one that represents tropical latitudes and one that represents subtropical latitudes. All memory is declared in the driver, and the CVMix data type points to the local variables.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_BL_driver-pointers.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/cvmix_BL_driver-pointers.F90 $
```

**INTERFACE:**

```
Program cvmix_BL_driver_pointers
```

**USES:**

```
  use cvmix_kinds_and_types, only : cvmix_r8,                &
                                    cvmix_data_type,         &
                                    cvmix_global_params_type, &
                                    cvmix_bkgnd_params_type
  use cvmix_background,      only : cvmix_init_bkgnd,        &
                                    cvmix_coeffs_bkgnd
  use cvmix_put_get,         only : cvmix_put
  use cvmix_output,          only : cvmix_output_open,       &
                                    cvmix_output_write,      &
                                    cvmix_output_close
```

---

### 1.0.2 cvmix_BL_driver_mem_copy

A stand-alone driver for the CVMix package. This particular driver generates the Bryan-Lewis coefficients in two columns, one that represents tropical latitudes and one that represents subtropical latitudes. All memory is declared in the driver and then copied into the CVMix data structures.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_BL_driver-mem_copy.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/cvmix_BL_driver-mem_copy.F90 $
```

**INTERFACE:**

```
Program cvmix_BL_driver_mem_copy
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                &
                                  cvmix_data_type,         &
                                  cvmix_global_params_type, &
                                  cvmix_bkgnd_params_type
use cvmix_background,      only : cvmix_init_bkgnd,        &
                                  cvmix_coeffs_bkgnd
use cvmix_put_get,         only : cvmix_put
use cvmix_output,          only : cvmix_output_open,       &
                                  cvmix_output_write,      &
                                  cvmix_output_close
```

---

### 1.0.3 cvmix_KPPshear_driver

A stand-alone driver for the CVMix package. This particular driver generates the shear-mixing coefficient defined in Equation (28) of Large, et al., in a single column and then outputs the column to allow recreation of Figure 3 from the same paper. Note that here each "level" of the column denotes a different local gradient Richardson number rather than a physical ocean level. All memory is declared in the driver, and the CVMix data type points to the local variables.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_KPP-shear_driver.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/cvmix_KPP-shear_driver.F90 $
```

**INTERFACE:**

```
Program cvmix_KPPshear_driver
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                &
                                  cvmix_data_type,         &
                                  cvmix_global_params_type, &
                                  cvmix_shear_params_type
use cvmix_shear,           only : cvmix_init_shear,        &
                                  cvmix_coeffs_shear
use cvmix_put_get,         only : cvmix_put
use cvmix_output,          only : cvmix_output_open,       &
                                  cvmix_output_write,      &
                                  cvmix_output_close
```

---

### 1.0.4 cvmix_Simmonstidal_driver

A stand-alone driver for the CVMix package. This particular driver generates the tidal-mixing coefficient defined in Equation (??) of Simmons, et al., in a single column and then outputs the column to allow recreation of Figure ? from the same paper.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_Simmons-tidal_driver.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/cvmix_Simmons-tidal_driver.F9
```

**INTERFACE:**

```
Program cvmix_Simmonstidal_driver
```

**USES:**

```
  use cvmix_kinds_and_types, only : cvmix_r8,                &
                                    cvmix_data_type,         &
                                    cvmix_global_params_type, &
                                    cvmix_tidal_params_type
  use cvmix_tidal,           only : cvmix_init_tidal,        &
                                    cvmix_coeffs_tidal
  use cvmix_put_get,         only : cvmix_put
  use cvmix_output,          only : cvmix_output_open,       &
                                    cvmix_output_write,      &
                                    cvmix_output_close
```

---

### 1.0.5 cvmix_ddiff_driver

A stand-alone driver for the CVMix package. This particular driver generates the double diffusion mixing coefficients.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_ddiff_driver.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/cvmix_ddiff_driver.F90 $
```

**INTERFACE:**

```
Program cvmix_ddiff_driver
```

**USES:**

```
  use cvmix_kinds_and_types, only : cvmix_r8,                &
                                    cvmix_data_type,         &
                                    cvmix_global_params_type, &
```

```
                                    cvmix_ddiff_params_type
   use cvmix_ddiff,           only : cvmix_init_ddiff,         &
                                     cvmix_coeffs_ddiff
   use cvmix_put_get,         only : cvmix_put
   use cvmix_output,          only : cvmix_output_open,        &
                                     cvmix_output_write,      &
                                     cvmix_output_close
```

## 1.1 Fortran: Module Interface cvmix_output

This module contains routines to output CVmix variables to data files. Currently only ascii output is supported, but the plan is to also include plain binary and netCDF output as well.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_output.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/cvmix_output.F90 $
```

**USES:**

```
   use cvmix_kinds_and_types, only : cvmix_data_type
#ifdef _NETCDF
   use cvmix_kinds_and_types, only : cvmix_r8
   use netcdf
#endif
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
  public :: cvmix_output_open
  public :: cvmix_output_write
  public :: cvmix_output_close
  public :: print_open_files

  interface cvmix_output_write
    module procedure cvmix_output_write_single_col
    module procedure cvmix_output_write_multi_col
  end interface
```

**DEFINED PARAMETERS:**

```
  integer, parameter :: ASCII_FILE_TYPE  = 1
  integer, parameter :: BIN_FILE_TYPE    = 2
  integer, parameter :: NETCDF_FILE_TYPE = 3
  integer, parameter :: FILE_NOT_FOUND   = 404

  ! Probably not the best technique, but going to use a linked list to keep
  ! track of what files are open / what format they are (ascii, bin, or nc)
  type :: cvmix_file_entry
    integer :: file_id
    integer :: file_type
    type(cvmix_file_entry), pointer :: prev
    type(cvmix_file_entry), pointer :: next
  end type

  type(cvmix_file_entry), allocatable, target :: file_database(:)
```

---

### 1.1.1 cvmix_output_open

**INTERFACE:**

```
subroutine cvmix_output_open(file_id, file_name, file_format)
```

**DESCRIPTION:**

Routine to open a file for writing. Goal is to support writing files in plain text (currently working), netCDF, and plain binary. Besides opening the file, this routine also adds an entry to file_database, a linked list that keeps track of what files are open and what type of file each identifier refers to. So it will be possible to output the same data in ascii and netCDF, for example.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*), intent(in) :: file_name, file_format
```

**OUTPUT PARAMETERS:**

```
integer, intent(out) :: file_id
```

**LOCAL VARIABLES:**

```
type(cvmix_file_entry), pointer :: file_index
```

---

### 1.1.2 cvmix_output_write_single_col

**INTERFACE:**

```
subroutine cvmix_output_write_single_col(file_id, CVmix_vars, var_names)
```

**DESCRIPTION:**

Routine to write the requested variables from a single column to a file (file must be opened using vmix_output_open to ensure it is written correctly). Called with vmix_output_write (see interface in PUBLIC MEMBER FUNCTIONS above).

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
    integer,                          intent(in) :: file_id
    type(cvmix_data_type),             intent(in) :: CVmix_vars
    character(len=*), dimension(:), intent(in) :: var_names
```

**LOCAL VARIABLES:**

```
    integer :: kw, var
#ifdef _NETCDF
    integer                               :: nw, nw_id
    integer, dimension(:), allocatable :: var_id
#endif
```

---

### 1.1.3 cvmix_output_write_multi_col

**INTERFACE:**

```
  subroutine cvmix_output_write_multi_col(file_id, CVmix_vars, var_names)
```

**DESCRIPTION:**

Routine to write the requested variables from multiple columns to a file (file must be opened using vmix_output_open to ensure it is written correctly). Called with vmix_output_write (see interface in PUBLIC MEMBER FUNCTIONS above).

**USES:**

```
   Only those used by entire module.
```

**INPUT PARAMETERS:**

```
    integer,                              intent(in) :: file_id
    type(cvmix_data_type), dimension(:), intent(in) :: CVmix_vars
    character(len=*),      dimension(:), intent(in) :: var_names
```

**LOCAL VARIABLES:**

```
    integer :: ncol, nw, icol, kw, var
    logical :: z_err
#ifdef _NETCDF
    integer                                              :: nw_id, ncol_id
    integer,             dimension(:),   allocatable :: var_id
    real(kind=cvmix_r8), dimension(:,:), allocatable :: lcl_visc, lcl_diff
#endif
```

---

### 1.1.4 cvmix_output_close

**INTERFACE:**

```
subroutine cvmix_output_close(file_id)
```

**DESCRIPTION:**

Routine to close a file once all writing has been completed. In addition to closing the file, this routine also deletes its entry in file_database to avoid trying to write to the file in the future.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
integer, intent(in) :: file_id
```

**LOCAL VARIABLES:**

```
type(cvmix_file_entry), pointer :: ifile, file_to_close
logical                         :: file_found
integer                         :: file_type
```

---

### 1.1.5 get_file_type

**INTERFACE:**

```
function get_file_type(file_id)
```

**DESCRIPTION:**

Returns the file format (enumerated in DEFINED PARAMETERS section) of a given file. If the file is not in the database, returns FILE_NOT_FOUND.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
integer, intent(in) :: file_id
```

**OUTPUT PARAMETERS:**

```
integer             :: get_file_type
```

**LOCAL VARIABLES:**

```
type(cvmix_file_entry), pointer :: ifile
```

---

## 1.2 Fortran: Module Interface cvmix_kinds_and_types

This module contains the declarations for all required vertical mixing data types. It also contains several global parameters used by the cvmix package, such as kind numbers and string lengths.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_kinds_and_types.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_kinds_and_types.F90 $
```

**USES:**

```
uses no other modules
```

---

**DEFINED PARAMETERS:**

```
! Kind Types:
! The cvmix package uses double precision for floating point computations.
integer, parameter, public :: cvmix_r8     = selected_real_kind(13), &
                              cvmix_strlen = 256

! Global parameters:
! The value for pi is needed for Bryan-Lewis mixing.
real(cvmix_r8), parameter, public :: cvmix_PI = &
                    2.0_cvmix_r8*acos(0.0_cvmix_r8)
```

**PUBLIC TYPES:**

```
! cvmix_input_type contains every possible necessary input field for all
! supported types of mixing.
type, public :: cvmix_data_type
    integer :: nlev = -1 ! Number of levels in column
                         ! Setting default to -1 might be F95...

    ! Values on interfaces
    ! nlev+1, 2
    real(cvmix_r8), dimension(:,:), pointer :: diff_iface => NULL()
    ! nlev+1
    real(cvmix_r8), dimension(:),   pointer :: visc_iface => NULL()
    real(cvmix_r8), dimension(:),   pointer :: z_iface    => NULL()
    real(cvmix_r8), dimension(:),   pointer :: dw_iface   => NULL()
    real(cvmix_r8), dimension(:),   pointer :: Ri_iface => NULL()

    ! Values at tracer points
    ! nlev
    real(cvmix_r8), dimension(:),   pointer :: dens     => NULL()
    real(cvmix_r8), dimension(:),   pointer :: dens_lwr => NULL()
```

```
    real(cvmix_r8), dimension(:),   pointer :: z        => NULL()
    real(cvmix_r8), dimension(:),   pointer :: dz       => NULL()
end type cvmix_data_type

! cvmix_global_params_type contains global parameters used by multiple
! mixing methods.
type, public :: cvmix_global_params_type
    integer                         :: max_nlev  ! maximum number of levels
    real(cvmix_r8)                  :: prandtl   ! Prandtl number
end type cvmix_global_params_type

! cvmix_bkgnd_params_type contains the necessary parameters for background
! mixing. Background mixing fields can vary from level to level as well as
! over latitude and longitude.
type, public :: cvmix_bkgnd_params_type
    real(cvmix_r8), allocatable :: static_visc(:,:) ! ncol, nlev+1
    real(cvmix_r8), allocatable :: static_diff(:,:) ! ncol, nlev+1

    ! Note: need to include some logic to avoid excessive memory use
    !       when static_visc and static_diff are constant or 1-D
    logical                     :: lvary_vertical   ! True => second dim not 1
    logical                     :: lvary_horizontal ! True => first dim not 1
end type cvmix_bkgnd_params_type

! cvmix_shear_params_type contains the necessary parameters for shear mixing
! (currently Pacanowski-Philander or Large et al)
type, public :: cvmix_shear_params_type
    character(len=cvmix_strlen) :: mix_scheme
    real(cvmix_r8)              :: alpha
    real(cvmix_r8)              :: n
    real(cvmix_r8)              :: nu_zero
    real(cvmix_r8)              :: Ri_zero
    real(cvmix_r8)              :: p_one
end type cvmix_shear_params_type

! cvmix_tidal_params_type contains the necessary parameters for shear mixing
! (currently just Simmons)
type, public :: cvmix_tidal_params_type
    character(len=cvmix_strlen) :: mix_scheme
end type cvmix_tidal_params_type

! cvmix_ddiff_params_type contains the necessary parameters for double
! diffusion mixing (currently just a place-holder variable)
type, public :: cvmix_ddiff_params_type
    real(cvmix_r8) :: deleteme
end type cvmix_ddiff_params_type

! cvmix_conv_params_type contains the necessary parameters for convective
```

```
! mixing.
type, public :: cvmix_conv_params_type
    real(cvmix_r8)              :: convect_diff
    real(cvmix_r8)              :: convect_visc
end type cvmix_conv_params_type
```

## 1.3 Fortran: Module Interface cvmix_background

This module contains routines to initialize the derived types needed for time independent static background mixing coefficients. It specifies either a scalar, 1D, or 2D field for viscosity and diffusivity. It also calculates the background diffusivity using the Bryan-Lewis method. It then sets the viscosity and diffusivity to the specified value.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_background.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_background.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_PI,                  &
                                  cvmix_r8,                  &
                                  cvmix_data_type,           &
                                  cvmix_global_params_type,  &
                                  cvmix_bkgnd_params_type
use cvmix_put_get, only         : cvmix_put
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_init_bkgnd
public :: cvmix_coeffs_bkgnd

interface cvmix_init_bkgnd
  module procedure cvmix_init_bkgnd_scalar
  module procedure cvmix_init_bkgnd_1D
  module procedure cvmix_init_bkgnd_2D
  module procedure cvmix_init_bkgnd_BryanLewis
end interface cvmix_init_bkgnd
```

---

### 1.3.1 cvmix_init_bkgnd_scalar

**INTERFACE:**

```
subroutine cvmix_init_bkgnd_scalar(CVmix_bkgnd_params, bkgnd_visc, bkgnd_diff)
```

**DESCRIPTION:**

Initialization routine for static background mixing coefficients. For each column, this routine sets the static viscosity / diffusivity to the given scalar constants.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
real(cvmix_r8), intent(in) :: bkgnd_visc
real(cvmix_r8), intent(in) :: bkgnd_diff
```

**OUTPUT PARAMETERS:**

```
type (cvmix_bkgnd_params_type), intent(out) :: CVmix_bkgnd_params
```

---

### 1.3.2 cvmix_init_bkgnd_1D

**INTERFACE:**

```
subroutine cvmix_init_bkgnd_1D(CVmix_params, CVmix_bkgnd_params, &
                               bkgnd_visc, bkgnd_diff, ncol)
```

**DESCRIPTION:**

Initialization routine for static background mixing coefficients. For each column, this routine sets the static viscosity / diffusivity to the given 1D field. If field varies horizontally, need to include ncol!

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_global_params_type), intent(in) :: CVmix_params
real(cvmix_r8), dimension(:),   intent(in) :: bkgnd_visc
real(cvmix_r8), dimension(:),   intent(in) :: bkgnd_diff
integer, optional,              intent(in) :: ncol
```

**OUTPUT PARAMETERS:**

```
type(cvmix_bkgnd_params_type), intent(out) :: CVmix_bkgnd_params
```

---

### 1.3.3 cvmix_init_bkgnd_2D

**INTERFACE:**

```
subroutine cvmix_init_bkgnd_2D(CVmix_params, CVmix_bkgnd_params, &
                               bkgnd_visc, bkgnd_diff, ncol)
```

**DESCRIPTION:**

Initialization routine for static background mixing coefficients. For each column, this routine sets the static viscosity / diffusivity to the given 2D field.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_global_params_type), intent(in) :: CVmix_params
real(cvmix_r8), dimension(:,:), intent(in) :: bkgnd_visc
real(cvmix_r8), dimension(:,:), intent(in) :: bkgnd_diff
integer,                        intent(in) :: ncol
```

**OUTPUT PARAMETERS:**

```
type(cvmix_bkgnd_params_type), intent(out) :: CVmix_bkgnd_params
```

---

### 1.3.4 cvmix_init_bkgnd_BryanLewis

**INTERFACE:**

```
subroutine cvmix_init_bkgnd_BryanLewis(CVmix_vars, CVmix_params,             &
                                       CVmix_bkgnd_params, bl1, bl2, bl3, bl4)
```

**DESCRIPTION:**

Initialization routine for Bryan-Lewis diffusivity/viscosity calculation. For each column, this routine sets the static viscosity & diffusivity based on the specified parameters. Note that the units of these parameters must be consistent with the units of viscosity and diffusivity – either cgs or mks, but do not mix and match!

The Bryan-Lewis parameterization is based on the following:

$$\begin{aligned} \kappa_{BL} &= \text{bl1} + \frac{\text{bl2}}{\pi} \tan^{-1}\left(\text{bl3}(z - \text{bl4})\right) \\ \nu_{BL} &= \text{Pr} \cdot \kappa_{BL} \end{aligned}$$

This method is based on the following paper:

> *A Water Mass Model of the World Ocean*
> K. Bryan and L. J. Lewis
> Journal of Geophysical Research, vol 84 (1979), pages 2503-2517.

In that paper, they recommend the parameters

$\text{bl1} = 8 \cdot 10^{-5}$ m$^2$/s

$\text{bl2} = 1.05 \cdot 10^{-4}$ m$^2$/s

$\text{bl3} = 4.5 \cdot 10^{-3}$ m$^{-1}$

$\text{bl4} = 2500$ m

However, more recent usage of their scheme may warrant different settings. **USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_data_type),          intent(in) :: CVmix_vars   ! Depth, nlev
type(cvmix_global_params_type), intent(in) :: CVmix_params ! Prandtl

! Units are first column if CVmix_data%depth is m, second if cm
real(cvmix_r8), intent(in) :: bl1,     &! m^2/s or cm^2/s
                              bl2,     &! m^2/s or cm^2/s
                              bl3,     &! 1/m   or 1/cm
                              bl4       ! m     or cm
```

**OUTPUT PARAMETERS:**

```
type(cvmix_bkgnd_params_type), intent(out) :: CVmix_bkgnd_params
```

---

### 1.3.5 cvmix_coeffs_bkgnd

**INTERFACE:**

```
subroutine cvmix_coeffs_bkgnd(CVmix_vars, CVmix_bkgnd_params, colid)
```

**DESCRIPTION:**

Computes vertical tracer and velocity mixing coefficients for static background mixing. This routine simply copies viscosity / diffusivity values from CVmix_bkgnd_params to CVmix_vars.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_bkgnd_params_type), intent(in) :: CVmix_bkgnd_params
! Need to know column for pulling data from static_visc and _diff
integer,                       intent(in) :: colid
```

**INPUT/OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

## 1.4 Fortran: Module Interface cvmix_shear

This module contains routines to initialize the derived types needed for shear mixing, and to set the viscosity and diffusivity coefficients. Presently this scheme has implemented the shear mixing parameterizations from Pacanowski & Philander (1981) and Large, McWilliams, & Doney (1994).

**REVISION HISTORY:**

```
SVN:$Id: cvmix_shear.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_shear.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                &
                                  cvmix_data_type,         &
                                  cvmix_bkgnd_params_type, &
                                  cvmix_shear_params_type
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_init_shear
public :: cvmix_coeffs_shear
```

---

### 1.4.1 cvmix_init_shear

**INTERFACE:**

```
subroutine cvmix_init_shear(CVmix_shear_params, mix_scheme,  &
                            alpha, n, nu_zero, Ri_zero, p_one)
```

**DESCRIPTION:**

Initialization routine for shear (Richardson number-based) mixing. There are currently two supported schemes - set `mix_scheme = 'PP'` to use the Pacanowski-Philander mixing scheme or set `mix_scheme = 'KPP'` to use the interior mixing scheme laid out in Large et al.

PP requires setting `nu_zero` ($\nu_0$), `alpha` ($\alpha$), and `n` ($n$), and returns

$$\begin{aligned} \nu_{PP} &= \frac{\nu_0}{(1+\alpha \mathrm{Ri})^n} + \nu_b \\ \kappa_{PP} &= \frac{\nu}{1+\alpha \mathrm{Ri}} + \kappa_b \end{aligned}$$

Note that $\nu_b$ and $\kappa_b$ are set in `cvmix_init_bkgnd()`, which needs to be called separately from this routine.

KPP requires setting `nu_zero` ($\nu^0$), `p_one` ($p_1$), and `Ri_zero` ($\mathrm{Ri}_0$), and returns

$$\nu_{KPP} = \begin{cases} \nu^0 & \mathrm{Ri} < 0 \\ \nu^0 \left[1 - \frac{\mathrm{Ri}}{\mathrm{Ri}_0}^2\right]^{p_1} & 0 < \mathrm{Ri} < \mathrm{Ri}_0 \\ 0 & \mathrm{Ri}_0 < \mathrm{Ri} \end{cases}$$

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*),         intent(in) :: mix_scheme
real(cvmix_r8), optional, intent(in) :: alpha, n, nu_zero, ri_zero, p_one
```

**OUTPUT PARAMETERS:**

```
type(cvmix_shear_params_type), intent(inout) :: CVmix_shear_params
```

---

### 1.4.2 cvmix_coeffs_shear

**INTERFACE:**

```
subroutine cvmix_coeffs_shear(CVmix_vars, CVmix_shear_params,    &
                              CVmix_bkgnd_params, colid, no_diff)
```

**DESCRIPTION:**

Computes vertical tracer and velocity mixing coefficients for shear-type mixing parameterizatiions. Note that Richardson number is needed at both T-points and U-points.

**USES:**

```
only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_shear_params_type),           intent(in) :: CVmix_shear_params
! PP mixing requires CVmix_bkgnd_params
type(cvmix_bkgnd_params_type), optional, intent(in) :: CVmix_bkgnd_params
! colid is only needed if CVmix_bkgnd_params%lvary_horizontal is true
integer,                       optional, intent(in) :: colid
logical,                       optional, intent(in) :: no_diff
```

**INPUT/OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

## 1.5 Fortran: Module Interface cvmix_tidal

This module contains routines to initialize the derived types needed for tidal mixing (currently just the Simmons scheme) and to set the viscosity and diffusivity coefficients accordingly.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_tidal.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_tidal.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                &
                                  cvmix_data_type,         &
                                  cvmix_tidal_params_type
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_init_tidal
public :: cvmix_coeffs_tidal
```

---

### 1.5.1 cvmix_init_tidal

**INTERFACE:**

```
subroutine cvmix_init_tidal(CVmix_tidal_params, mix_scheme)
```

**DESCRIPTION:**

Initialization routine for tidal mixing. There is currently just one supported schemes - set `mix_scheme = 'simmons'` to use the Simmons mixing scheme. **USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*), intent(in) :: mix_scheme
```

**OUTPUT PARAMETERS:**

```
type(cvmix_tidal_params_type), intent(inout) :: CVmix_tidal_params
```

---

### 1.5.2 cvmix_coeffs_tidal

**INTERFACE:**

```
subroutine cvmix_coeffs_tidal(CVmix_vars, CVmix_tidal_params)
```

**DESCRIPTION:**

Computes vertical diffusion coefficients for tidal mixing parameterizatiions.

**USES:**

```
only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_tidal_params_type), intent(in) :: CVmix_tidal_params
```

**INPUT/OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

## 1.6 Fortran: Module Interface cvmix_ddiff

This module contains routines to initialize the derived types needed for double diffusion mixing and to set the viscosity and diffusivity coefficients accordingly.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_ddiff.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_ddiff.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                &
                                  cvmix_data_type,         &
                                  cvmix_ddiff_params_type
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_init_ddiff
public :: cvmix_coeffs_ddiff
```

---

### 1.6.1 cvmix_init_ddiff

**INTERFACE:**

```
subroutine cvmix_init_ddiff(CVmix_ddiff_params)
```

**DESCRIPTION:**

Initialization routine for double diffusion mixing. **USES:**

```
Only those used by entire module.
```

**OUTPUT PARAMETERS:**

```
type(cvmix_ddiff_params_type), intent(inout) :: CVmix_ddiff_params
```

---

### 1.6.2 cvmix_coeffs_ddiff

**INTERFACE:**

```
subroutine cvmix_coeffs_ddiff(CVmix_vars, CVmix_ddiff_params)
```

**DESCRIPTION:**

Computes vertical diffusion coefficients for the double diffusion mixing parameterizatiion.

**USES:**

```
only those used by entire module.
```

**INPUT PARAMETERS:**

```
type(cvmix_ddiff_params_type), intent(in) :: CVmix_ddiff_params
```

**INPUT/OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

## 1.7 Fortran: Module Interface cvmix_convection

This module contains routines to initialize the derived types needed for specifying mixing coefficients to parameterize vertical convective mixing, and to set the viscosity and diffusivity in gravitationally unstable portions of the water column.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_convection.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_convection.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,               &
                                  cvmix_data_type,        &
                                  cvmix_conv_params_type
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_init_conv
public :: cvmix_coeffs_conv
```

---

### 1.7.1 cvmix_init_conv

**INTERFACE:**

```
subroutine cvmix_init_conv(CVmix_conv_params, convect_diff, convect_visc)
```

**DESCRIPTION:**

Initialization routine for specifying convective mixing coefficients.

**USES:**

```
Only those used by entire module.
```

**OUTPUT PARAMETERS:**

```
type (cvmix_conv_params_type), intent(out) :: CVmix_conv_params
```

**INPUT PARAMETERS:**

```
real(cvmix_r8), intent(in) :: &
   convect_diff,      &! diffusivity to parameterize convection
   convect_visc        ! viscosity to parameterize convection
```

---

### 1.7.2 cvmix_coeffs_conv

**INTERFACE:**

```
subroutine cvmix_coeffs_conv(CVmix_vars, CVmix_conv_params)
```

**DESCRIPTION:**

Computes vertical diffusion coefficients for convective mixing.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
type (cvmix_conv_params_type), intent(in)  :: CVmix_conv_params
```

**INPUT/OUTPUT PARAMETERS:**

```
type (cvmix_data_type), intent(inout) :: CVmix_vars
```

---

## 1.8 Fortran: Module Interface cvmix_put_get

This module contains routines to pack data into the cvmix datatypes (allocating memory as necessary) and then unpack the data out. If we switch to pointers, the pack will just point at the right target and the unpack will be un-necessary.

**REVISION HISTORY:**

```
SVN:$Id: cvmix_put_get.F90 -1   $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/cvmix_put_get.F90 $
```

**USES:**

```
use cvmix_kinds_and_types, only : cvmix_r8,                 &
                                  cvmix_strlen,             &
                                  cvmix_data_type,          &
                                  cvmix_global_params_type, &
                                  cvmix_bkgnd_params_type
```

---

**PUBLIC MEMBER FUNCTIONS:**

```
public :: cvmix_put

interface cvmix_put
  module procedure cvmix_put_int
  module procedure cvmix_put_real
  module procedure cvmix_put_real_1D
  module procedure cvmix_put_bkgnd_real    ! untested
  module procedure cvmix_put_bkgnd_real_1D
  module procedure cvmix_put_bkgnd_real_2D ! untested
  module procedure cvmix_put_global_params_int
  module procedure cvmix_put_global_params_real
end interface cvmix_put
```

---

### 1.8.1 cvmix_put_int

**INTERFACE:**

```
subroutine cvmix_put_int(CVmix_vars, varname, val, opts)
```

**DESCRIPTION:**

Write an integer value into a cvmix_data_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*),           intent(in) :: varname
integer,                    intent(in) :: val
character(len=*), optional, intent(in) :: opts
```

**OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

### 1.8.2 cvmix_put_real

**INTERFACE:**

```
subroutine cvmix_put_real(CVmix_vars, varname, val, opts)
```

**DESCRIPTION:**

Write a real value into a cvmix_data_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*),           intent(in) :: varname
real(cvmix_r8),              intent(in) :: val
character(len=*), optional, intent(in) :: opts
```

**OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

### 1.8.3 cvmix_put_real_1D

**INTERFACE:**

```
subroutine cvmix_put_real_1D(CVmix_vars, varname, val, opts)
```

**DESCRIPTION:**

Write an array of real values into a cvmix_data_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*),             intent(in) :: varname
real(cvmix_r8), dimension(:), intent(in) :: val
character(len=*), optional,   intent(in) :: opts
```

**OUTPUT PARAMETERS:**

```
type(cvmix_data_type), intent(inout) :: CVmix_vars
```

---

### 1.8.4 cvmix_put_bkgnd_real

**INTERFACE:**

```
subroutine cvmix_put_bkgnd_real(CVmix_bkgnd_params, varname, val)
```

**DESCRIPTION:**

Write a real value into a cvmix_bkgnd_params_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=cvmix_strlen), intent(in) :: varname
real(cvmix_r8),              intent(in) :: val
```

**OUTPUT PARAMETERS:**

```
type(cvmix_bkgnd_params_type), intent(inout) :: CVmix_bkgnd_params
```

---

### 1.8.5 cvmix_put_bkgnd_real_1D

**INTERFACE:**

```
subroutine cvmix_put_bkgnd_real_1D(CVmix_bkgnd_params, varname, val, &
                                   ncol, nlev)
```

**DESCRIPTION:**

Write an array of real values into a cvmix_bkgnd_params_type variable. You must use `opt='horiz'` to specify that the field varies in the horizontal direction, otherwise it is assumed to vary in the vertical.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*),             intent(in) :: varname
real(cvmix_r8), dimension(:), intent(in) :: val
integer, optional,            intent(in) :: ncol, nlev
```

**OUTPUT PARAMETERS:**

```
type (cvmix_bkgnd_params_type), intent(inout) :: CVmix_bkgnd_params
```

---

### 1.8.6 cvmix_put_bkgnd_real_2D

**INTERFACE:**

```
subroutine cvmix_put_bkgnd_real_2D(CVmix_bkgnd_params, varname, val, &
                                   ncol, nlev)
```

**DESCRIPTION:**

Write a 2D array of real values into a cvmix_bkgnd_params_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=cvmix_strlen),    intent(in) :: varname
real(cvmix_r8), dimension(:,:), intent(in) :: val
integer,                        intent(in) :: ncol, nlev
```

**OUTPUT PARAMETERS:**

```
type (cvmix_bkgnd_params_type), intent(out) :: CVmix_bkgnd_params
```

---

### 1.8.7 cvmix_put_global_params_int

**INTERFACE:**

```
subroutine cvmix_put_global_params_int(CVmix_params, varname, val)
```

**DESCRIPTION:**

Write an integer value into a cvmix_global_params_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*), intent(in) :: varname
integer,          intent(in) :: val
```

**OUTPUT PARAMETERS:**

```
type (cvmix_global_params_type), intent(inout) :: CVmix_params
```

---

### 1.8.8 cvmix_put_global_params_real

**INTERFACE:**

```
subroutine cvmix_put_global_params_real(CVmix_params, varname, val)
```

**DESCRIPTION:**

Write a real value into a cvmix_global_params_type variable.

**USES:**

```
Only those used by entire module.
```

**INPUT PARAMETERS:**

```
character(len=*), intent(in) :: varname
real(cvmix_r8),   intent(in) :: val
```

**OUTPUT PARAMETERS:**

```
type(cvmix_global_params_type), intent(inout) :: CVmix_params
```