In-code documentation for CVMix

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

December 1, 2012

Contents

1 Routine/Function Prologues

1.0.1 vmix_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: vmix_BL_driver-pointers.F90 31 2012-11-14 00:23:35Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/vmix_BL_driver-pointers.F90 $
```

INTERFACE:

```
Program vmix_BL_driver_pointers
```

USES:

```
use vmix_kinds_and_types
use vmix_background
use vmix_convection
use vmix_put_get
use vmix_output
```

1.0.2 vmix_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: vmix_BL_driver-mem_copy.F90 31 2012-11-14 00:23:35Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/vmix_BL_driver-mem_copy.F90 $
```

INTERFACE:

```
Program vmix_BL_driver_mem_copy
```

USES:

```
use vmix_kinds_and_types
use vmix_background
use vmix_convection
use vmix_put_get
use vmix_output
```

1.0.3 vmix_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: vmix_BL_driver-pointers.F90 31 2012-11-14 00:23:35Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/drivers/vmix_BL_driver-pointers.F90 $
```

INTERFACE:

```
Program vmix_KPPshear_driver
```

USES:

```
use vmix_kinds_and_types
use vmix_shear
use vmix_put_get
use vmix_output
```

1.1 Fortran: Module Interface vmix_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: vmix_output.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/vmix_output.F90 $
```

USES:

```
#ifdef _NETCDF
   use netcdf
#endif
   use vmix_kinds_and_types
```

PUBLIC MEMBER FUNCTIONS:

```
  public :: vmix_output_open
  public :: vmix_output_write_viscosity
  public :: vmix_output_write_diffusivity
  public :: vmix_output_close
  public :: print_open_files

  interface vmix_output_write_viscosity
    module procedure vmix_output_write_visc_coeff_single_col
    module procedure vmix_output_write_visc_coeff_multi_col
  end interface

  interface vmix_output_write_diffusivity
    module procedure vmix_output_write_diff_coeff_single_col
    module procedure vmix_output_write_diff_coeff_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 :: vmix_file_entry
    integer :: file_id
    integer :: file_type
    type(vmix_file_entry), pointer :: prev
```

```
    type(vmix_file_entry), pointer :: next
  end type

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

1.1.1 vmix_output_open

INTERFACE:

```
  subroutine vmix_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(vmix_file_entry), pointer :: file_index
```

1.1.2 vmix_output_write_visc_coeff_single_col

INTERFACE:

```
  subroutine vmix_output_write_visc_coeff_single_col(file_id, Vmix_vars)
```

DESCRIPTION:

Routine to write the viscosity coefficients 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_viscosity (see interface in PUBLIC MEMBER FUNCTIONS above).

USES:

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

INPUT PARAMETERS:

```
     integer, intent(in) :: file_id
     type(vmix_data_type), intent(in) :: Vmix_vars
```

LOCAL VARIABLES:

```
     integer :: kw, nw
 #ifdef _NETCDF
     integer :: nw_id, zw_id, visc_id
 #endif
```

1.1.3 vmix_output_write_visc_coeff_multi_col

INTERFACE:

```
   subroutine vmix_output_write_visc_coeff_multi_col(file_id, Vmix_vars)
```

DESCRIPTION:

Routine to write the viscosity coefficients 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_viscosity (see interface in PUBLIC MEMBER FUNCTIONS above).

USES:

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

INPUT PARAMETERS:

```
     integer, intent(in) :: file_id
     type(vmix_data_type), dimension(:), intent(in) :: Vmix_vars
```

LOCAL VARIABLES:

```
     integer :: ncol, nw, icol, kw
     logical :: z_err
     real(kind=vmix_r8), dimension(:,:), allocatable :: lcl_visc
 #ifdef _NETCDF
     integer :: nw_id, ncol_id, zw_id, visc_id
 #endif
```

1.1.4 vmix_output_write_diff_coeff_single_col

INTERFACE:

```
subroutine vmix_output_write_diff_coeff_single_col(file_id, Vmix_vars)
```

DESCRIPTION:

Routine to write the diffusivity coefficients 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(vmix_data_type), intent(in) :: Vmix_vars
```

LOCAL VARIABLES:

```
    integer :: kw, nw
#ifdef _NETCDF
    integer :: nw_id, zw_id, diff_id
#endif
```

1.1.5 vmix_output_write_diff_coeff_multi_col

INTERFACE:

```
subroutine vmix_output_write_diff_coeff_multi_col(file_id, Vmix_vars)
```

DESCRIPTION:

Routine to write the diffusivity coefficients 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(vmix_data_type), dimension(:), intent(in) :: Vmix_vars
```

LOCAL VARIABLES:

```
    integer :: ncol, nw, icol, kw
    logical :: z_err
    real(kind=vmix_r8), dimension(:,:), allocatable :: lcl_diff
#ifdef _NETCDF
    integer :: nw_id, ncol_id, zw_id, diff_id
#endif
```

1.1.6 vmix_output_close

INTERFACE:

```
  subroutine vmix_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(vmix_file_entry), pointer :: ifile, file_to_close
    logical :: file_found
    integer :: file_type
```

1.1.7 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(vmix_file_entry), pointer :: ifile
```

1.2 Fortran: Module Interface vmix_kinds_and_types

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

REVISION HISTORY:

```
SVN:$Id: vmix_kinds_and_types.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/vmix_kinds_and_types.F90 $
```

USES:

```
uses no other modules
```

DEFINED PARAMETERS:

```
! Kind Types:
! The vmix package uses double precision for floating point computations.
integer, parameter, public :: vmix_r8     = selected_real_kind(13), &
                              vmix_strlen = 256

! Global parameters:
! The value for pi is needed for Bryan-Lewis mixing.
real(kind=vmix_r8), parameter, public :: vmix_PI = &
                    2.0_vmix_r8*acos(0.0_vmix_r8)
```

PUBLIC TYPES:

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

    ! Values on interfaces
    ! nlev+1, 2
    real(vmix_r8), dimension(:,:), pointer :: diff_iface => NULL()
    ! nlev+1
    real(vmix_r8), dimension(:),   pointer :: visc_iface => NULL()
    real(vmix_r8), dimension(:),   pointer :: z_iface    => NULL()
    real(vmix_r8), dimension(:),   pointer :: dw_iface   => NULL()
    real(vmix_r8), dimension(:),   pointer :: Ri_t_iface => NULL()
    real(vmix_r8), dimension(:),   pointer :: Ri_u_iface => NULL()
    real(vmix_r8), dimension(:),   pointer :: Ri_g       => NULL()

    ! Values at tracer points
    ! nlev
```

```
    real(vmix_r8), dimension(:),   pointer :: dens     => NULL()
    real(vmix_r8), dimension(:),   pointer :: dens_lwr => NULL()
    real(vmix_r8), dimension(:),   pointer :: z        => NULL()
    real(vmix_r8), dimension(:),   pointer :: dz       => NULL()
end type vmix_data_type

! vmix_global_params_type contains global parameters used by multiple
! mixing methods.
type, public :: vmix_global_params_type
    integer                      :: max_nlev  ! maximum number of levels
    real(vmix_r8)                :: prandtl   ! Prandtl number
end type vmix_global_params_type

! vmix_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 :: vmix_bkgnd_params_type
    real(vmix_r8), allocatable :: static_visc(:,:) ! ncol, nlev+1
    real(vmix_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 vmix_bkgnd_params_type

! vmix_shear_params_type contains the necessary parameters for shear mixing
! (currently just Paconowski-Philander)
type, public :: vmix_shear_params_type
    character(len=vmix_strlen) :: mix_scheme
    real(vmix_r8)              :: alpha
    real(vmix_r8)              :: n
    real(vmix_r8)              :: nu_zero
    real(vmix_r8)              :: Ri_zero
    real(vmix_r8)              :: p_one
end type vmix_shear_params_type

! vmix_conv_params_type contains the necessary parameters for convective
! mixing.
type, public :: vmix_conv_params_type
    real(vmix_r8)              :: convect_diff
    real(vmix_r8)              :: convect_visc
end type vmix_conv_params_type
```

1.3 Fortran: Module Interface vmix_background

This module contains routines to initialize the derived types needed for background mixing (either specifying a scalar, 1D, or 2D field for viscosity and diffusivity coefficients or calculating these coefficients using the Bryan-Lewis method) and to set the viscosity and diffusivity coefficients to this specified value.

REVISION HISTORY:

```
SVN:$Id: vmix_background.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/vmix_background.F90 $
```

USES:

```
use vmix_kinds_and_types, only : vmix_PI,                 &
                                 vmix_r8,                 &
                                 vmix_data_type,          &
                                 vmix_global_params_type, &
                                 vmix_bkgnd_params_type
```

PUBLIC MEMBER FUNCTIONS:

```
public :: vmix_init_bkgnd
public :: vmix_coeffs_bkgnd

interface vmix_init_bkgnd
  module procedure vmix_init_bkgnd_scalar
  module procedure vmix_init_bkgnd_1D
  module procedure vmix_init_bkgnd_2D
  module procedure vmix_init_bkgnd_BryanLewis
end interface vmix_init_bkgnd
```

1.3.1 vmix_init_bkgnd_scalar

INTERFACE:

```
subroutine vmix_init_bkgnd_scalar(Vmix_bkgnd_params, bkgnd_visc, bkgnd_diff)
```

DESCRIPTION:

Initialization routine for background mixing with a static field. 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(vmix_r8), intent(in) :: bkgnd_visc
real(vmix_r8), intent(in) :: bkgnd_diff
```

OUTPUT PARAMETERS:

```
type (vmix_bkgnd_params_type), intent(out) :: Vmix_bkgnd_params
```

1.3.2 vmix_init_bkgnd_1D

INTERFACE:

```
subroutine vmix_init_bkgnd_1D(Vmix_params, Vmix_bkgnd_params, &
                              bkgnd_visc, bkgnd_diff, ncol)
```

DESCRIPTION:

Initialization routine for background mixing with a static field. 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 (vmix_global_params_type), intent(in) :: Vmix_params
real(vmix_r8), dimension(:),    intent(in) :: bkgnd_visc
real(vmix_r8), dimension(:),    intent(in) :: bkgnd_diff
integer, optional,              intent(in) :: ncol
```

OUTPUT PARAMETERS:

```
type (vmix_bkgnd_params_type), intent(out) :: Vmix_bkgnd_params
```

1.3.3 vmix_init_bkgnd_2D

INTERFACE:

```
subroutine vmix_init_bkgnd_2D(Vmix_params, Vmix_bkgnd_params, &
                              bkgnd_visc, bkgnd_diff, ncol)
```

DESCRIPTION:

Initialization routine for background mixing with a static field. 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 (vmix_global_params_type), intent(in) :: Vmix_params
real(vmix_r8), dimension(:,:),  intent(in) :: bkgnd_visc
real(vmix_r8), dimension(:,:),  intent(in) :: bkgnd_diff
integer,                        intent(in) :: ncol
```

OUTPUT PARAMETERS:

```
type (vmix_bkgnd_params_type), intent(out) :: Vmix_bkgnd_params
```

1.3.4 vmix_init_bkgnd_BryanLewis

INTERFACE:

```
subroutine vmix_init_bkgnd_BryanLewis(Vmix_vars, Vmix_params,         &
                                      Vmix_bkgnd_params, colid, ncol, &
                                      bl1, bl2, bl3, bl4)
```

DESCRIPTION:

Initialization routine for background mixing with a Bryan-Lewis mixing. 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 don't mix and match!

The Bryan-Lewis parameterization uses the following:

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

This is all 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,

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

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

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

$\text{bl4} = 2500\ \text{m}$

USES:

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

INPUT PARAMETERS:

```
type (vmix_data_type),          intent(in) :: Vmix_vars   ! Depth, nlev
type (vmix_global_params_type), intent(in) :: Vmix_params ! Prandtl
integer,                        intent(in) :: colid, ncol

! Units are first column if Vmix_data%depth is m, second if cm
real(vmix_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 (vmix_bkgnd_params_type), intent(out) :: Vmix_bkgnd_params
```

1.3.5 vmix_coeffs_bkgnd

INTERFACE:

```
subroutine vmix_coeffs_bkgnd(Vmix_vars, Vmix_bkgnd_params, colid)
```

DESCRIPTION:

Computes vertical diffusion coefficients for static mixing. This routine simply copies viscosity / diffusivity values from Vmix_bkgnd_params to Vmix_vars.

USES:

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

INPUT PARAMETERS:

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

INPUT/OUTPUT PARAMETERS:

```
type (vmix_data_type), intent(inout) :: Vmix_vars
```

1.4 Fortran: Module Interface vmix_shear

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

REVISION HISTORY:

```
SVN:$Id: vmix_shear.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/vmix_shear.F90 $
```

USES:

```
use vmix_kinds_and_types, only : vmix_r8,                 &
                                 vmix_strlen,             &
                                 vmix_data_type,          &
                                 vmix_global_params_type, &
                                 vmix_bkgnd_params_type,  &
                                 vmix_shear_params_type
```

PUBLIC MEMBER FUNCTIONS:

```
public :: vmix_init_shear
public :: vmix_coeffs_shear
```

1.4.1 vmix_init_shear

INTERFACE:

```
subroutine vmix_init_shear(Vmix_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 Paconowski-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` (ν_0), `alpha` (α), 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 ν_b and κ_b are set in `vmix_init_bkgnd()`, which needs to be called separately from this routine.

KPP requires setting `nu_zero` (ν^0), `p_one` (p_1), and `Ri_zero` (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(vmix_r8), optional, intent(in) :: alpha, n, nu_zero, ri_zero, p_one
```

OUTPUT PARAMETERS:

```
type(vmix_shear_params_type), intent(inout) :: Vmix_shear_params
```

1.4.2 vmix_coeffs_shear

INTERFACE:

```
subroutine vmix_coeffs_shear(Vmix_vars, Vmix_shear_params, &
                             Vmix_bkgnd_params, colid)
```

DESCRIPTION:

Computes vertical diffusion 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(vmix_shear_params_type),           intent(in) :: Vmix_shear_params
! PP mixing requires Vmix_bkgnd_params
type(vmix_bkgnd_params_type), optional, intent(in) :: Vmix_bkgnd_params
! colid is only needed if Vmix_bkgnd_params%lvary_horizontal is true
integer,                      optional, intent(in) :: colid
```

INPUT/OUTPUT PARAMETERS:

```
type(vmix_data_type), intent(inout) :: Vmix_vars
```

1.5 Fortran: Module Interface vmix_convection

This module contains routines to initialize the derived types needed for convective mixing and to set the viscosity and diffusivity in unstable columns.

REVISION HISTORY:

```
SVN:$Id: vmix_convection.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/vmix_convection.F90 $
```

USES:

```
use vmix_kinds_and_types, only : vmix_r8,               &
                                 vmix_data_type,        &
                                 vmix_conv_params_type
```

PUBLIC MEMBER FUNCTIONS:

```
public :: vmix_init_conv
public :: vmix_coeffs_conv
```

1.5.1 vmix_init_conv

INTERFACE:

```
subroutine vmix_init_conv(Vmix_conv_params, convect_diff, convect_visc)
```

DESCRIPTION:

Initialization routine for convective mixing.

USES:

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

OUTPUT PARAMETERS:

```
type (vmix_conv_params_type), intent(out) :: Vmix_conv_params
```

INPUT PARAMETERS:

```
real(vmix_r8), intent(in) :: &
   convect_diff,      &! diffusivity to mimic convection
   convect_visc        ! viscosity to mimic convection
```

1.5.2 vmix_coeffs_conv

INTERFACE:

```
subroutine vmix_coeffs_conv(Vmix_vars, Vmix_conv_params)
```

DESCRIPTION:

Computes vertical diffusion coefficients for convective mixing.

USES:

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

INPUT PARAMETERS:

```
type (vmix_conv_params_type), intent(in)  :: Vmix_conv_params
```

INPUT/OUTPUT PARAMETERS:

```
type (vmix_data_type), intent(inout) :: Vmix_vars
```

1.6 Fortran: Module Interface vmix_put_get

This module contains routines to pack data into the vmix 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: vmix_put_get.F90 32 2012-12-01 22:14:25Z mike.levy.work@gmail.com $
SVN:$URL: https://cvmix.googlecode.com/svn/trunk/src/shared/vmix_put_get.F90 $
```

USES:

```
use vmix_kinds_and_types, only : vmix_r8,                 &
                                 vmix_strlen,             &
                                 vmix_data_type,          &
                                 vmix_global_params_type, &
                                 vmix_bkgnd_params_type
```

PUBLIC MEMBER FUNCTIONS:

```
public :: vmix_put

interface vmix_put
  module procedure vmix_put_int
  module procedure vmix_put_real
  module procedure vmix_put_real_1D
  module procedure vmix_put_bkgnd_real_1D
  module procedure vmix_put_global_params_int
  module procedure vmix_put_global_params_real
end interface vmix_put
```

1.6.1 vmix_put_int

INTERFACE:

```
subroutine vmix_put_int(Vmix_vars, varname, val, opts)
```

DESCRIPTION:

Write an integer value into a vmix_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 (vmix_data_type), intent(inout) :: Vmix_vars
```

1.6.2 vmix_put_real

INTERFACE:

```
subroutine vmix_put_real(Vmix_vars, varname, val, opts)
```

DESCRIPTION:

Write a real value into a vmix_data_type variable.

USES:

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

INPUT PARAMETERS:

```
character(len=*),           intent(in) :: varname
real(vmix_r8),              intent(in) :: val
character(len=*), optional, intent(in) :: opts
```

OUTPUT PARAMETERS:

```
type (vmix_data_type), intent(inout) :: Vmix_vars
```

1.6.3 vmix_put_real_1D

INTERFACE:

```
subroutine vmix_put_real_1D(Vmix_vars, varname, val, opts)
```

DESCRIPTION:

Write an array of real values into a vmix_data_type variable.

USES:

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

INPUT PARAMETERS:

```
character(len=*),            intent(in) :: varname
real(vmix_r8), dimension(:), intent(in) :: val
character(len=*), optional,  intent(in) :: opts
```

OUTPUT PARAMETERS:

```
type (vmix_data_type), intent(inout) :: Vmix_vars
```

1.6.4 vmix_put_bkgnd_real_1D

INTERFACE:

```
subroutine vmix_put_bkgnd_real_1D(varname, Vmix_bkgnd_params, val, &
                                  opts)
```

DESCRIPTION:

Write an array of real values into a vmix_bkgnd_params_type variable.

USES:

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

INPUT PARAMETERS:

```
character(len=vmix_strlen),  intent(in)          :: varname
real(vmix_r8), dimension(:), intent(in)          :: val
character(len=vmix_strlen), optional, intent(in) :: opts
```

OUTPUT PARAMETERS:

```
type (vmix_bkgnd_params_type), intent(out) :: Vmix_bkgnd_params
```

1.6.5 vmix_put_global_params_int

INTERFACE:

```
subroutine vmix_put_global_params_int(Vmix_params, varname, val, opts)
```

DESCRIPTION:

Write an integer value into a vmix_global_params_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 (vmix_global_params_type), intent(inout) :: Vmix_params
```

1.6.6 vmix_put_global_params_real

INTERFACE:

```
subroutine vmix_put_global_params_real(Vmix_params, varname, val, opts)
```

DESCRIPTION:

Write a real value into a vmix_global_params_type variable.

USES:

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

INPUT PARAMETERS:

```
character(len=*),           intent(in) :: varname
real(vmix_r8),              intent(in) :: val
character(len=*), optional, intent(in) :: opts
```

OUTPUT PARAMETERS:

```
type (vmix_global_params_type), intent(inout) :: Vmix_params
```