```
! ================================================================================================
! This module declares objects ("identities") for standard physical-biogeochemical variables
! that have a well-defined interpretation and unit.
!
! To use these identities, "use" the fabm_types module, which provides a single "standard_variables"
! variable (derived type) that has all standard identities as its members. This can then be used as
! standard_variables%temperature, standard_variables%wind_speed, etc.
! For a list of all supported variables, please see:
! https://github.com/fabm-model/fabm/wiki/List-of-standard-variables
!
! Biogeochemical models can use these "identity" objects in two ways. First, they can access the value
! of the corresponding variable by registering them as dependency. To do so, call register_dependency
! with the "identity" object (e.g., standard_variables%temperature) as argument.
! Additionally, biogeochemical models can assign a standard identities to their own variables during
! registration. FABM will couple all variables that have been assigned the same identity. Thus, using
! standard identities enables implicit variable coupling.
! ================================================================================================
! The names of standard variables are based on the Standard Name Table from the NetCDF Climate and
! Forecast (CF) Metadata Convention. See http://cfconventions.org/.
! In deriving names from the CF convention, the following exceptions are made to account for the fact
! that FABM handles both marine and limnic systems and has the water column as default domain:
! - "sea_water_" prefix is suppressed.
! - "_in_sea_water" suffix is suppressed.
! - instead of the "_at_sea_floor" suffix a "bottom_" prefix is used, analogous to the "surface_"
!   prefix used in CF.
! - the "sea_floor_" prefix is replaced by a "bottom_" prefix.
! ================================================================================================

module fabm_standard_variables

   implicit none

   private

   public type_base_standard_variable
   public type_universal_standard_variable, type_domain_specific_standard_variable, type_interior_standard_variable, type_horizontal_standard_variable, type_surface_standard_variable, type_bottom_standard_variable, type_global_standard_variable
   public type_standard_variable_node, type_standard_variable_set
   public standard_variables

   ! ================================================================================================
   ! Data types that contain all metadata needed to describe standard variables.
   ! ================================================================================================

   type type_base_standard_variable
      character(len=256) :: name  = ''    ! Name
      character(len=64)  :: units = ''    ! Units
      character(len=512) :: cf_names = '' ! Comma-separated list of standard names defined in the NetCDF CF convention
                                          ! (http://cfconventions.org/standard-names.html)
      logical            :: aggregate_variable = .false. ! Whether biogeochemical models can contribute (add to) this variable.
                                                         ! If .true., this variable is always available with a default value of 0.
      logical, private   :: resolved = .false.
   contains
      procedure :: resolve         => base_standard_variable_resolve
      procedure :: assert_resolved => base_standard_variable_assert_resolved
   end type

   type, extends(type_base_standard_variable) :: type_domain_specific_standard_variable
      type (type_universal_standard_variable), pointer :: universal => null()
   contains
      procedure :: typed_resolve => domain_specific_standard_variable_typed_resolve
   end type

   type, extends(type_domain_specific_standard_variable) :: type_interior_standard_variable
   end type

   type, extends(type_domain_specific_standard_variable) :: type_horizontal_standard_variable
   end type

   type, extends(type_horizontal_standard_variable) :: type_surface_standard_variable
   end type

   type, extends(type_horizontal_standard_variable) :: type_bottom_standard_variable
   end type

   type, extends(type_domain_specific_standard_variable) :: type_global_standard_variable
   end type

   type, extends(type_base_standard_variable) :: type_universal_standard_variable
      logical                                                  :: conserved = .false.          ! Whether this variable should be included in lists of conserved quantities.
      type (type_interior_standard_variable),   pointer, private :: pin_interior   => null()
      type (type_horizontal_standard_variable), pointer, private :: pat_interfaces => null()
      type (type_surface_standard_variable),    pointer, private :: pat_surface    => null()
      type (type_bottom_standard_variable),     pointer, private :: pat_bottom     => null()
   contains
      procedure :: typed_resolve => universal_standard_variable_typed_resolve
      procedure :: in_interior   => universal_standard_variable_in_interior
      procedure :: at_interfaces => universal_standard_variable_at_interfaces
      procedure :: at_surface    => universal_standard_variable_at_surface
      procedure :: at_bottom     => universal_standard_variable_at_bottom
   end type

   type type_standard_variable_node
```

```
 93      class (type_base_standard_variable), pointer :: p    => null()
 94      logical, private                             :: own = .false.
 95      type (type_standard_variable_node),  pointer :: next => null()
 96   end type
 97
 98   type type_standard_variable_set
 99      type (type_standard_variable_node), pointer :: first => null()
100   contains
101      procedure :: contains_variable => standard_variable_set_contains_variable
102      procedure :: contains_name     => standard_variable_set_contains_name
103      generic   :: contains => contains_variable, contains_name
104      procedure :: add      => standard_variable_set_add
105      procedure :: update   => standard_variable_set_update
106      procedure :: finalize => standard_variable_set_finalize
107   end type
108
109   type type_standard_variable_collection
110      type (type_standard_variable_node), pointer :: first => null()
111#include "standard_variables.h"
112   contains
113      procedure :: initialize => standard_variable_collection_initialize
114      procedure :: finalize   => standard_variable_collection_finalize
115   end type
116
117   ! Single instance of the collection that contains all standard variables.
118   type (type_standard_variable_collection), save :: standard_variables
119
120contains
121
122   recursive function base_standard_variable_resolve(self) result(p)
123      class (type_base_standard_variable), intent(in), target :: self
124      class (type_base_standard_variable), pointer            :: p
125
126      type (type_standard_variable_node), pointer :: node
127
128      if (self%resolved) then
129         p => self
130         return
131      end if
132
133      node => standard_variables%first
134      do while (associated(node))
135         if (compare(self, node%p)) then
136            p => node%p
137            return
138         end if
139         node => node%next
140      end do
141
142      allocate(p, source=self)
143      call add(p, own=.true.)
144
145   contains
146
147      logical function compare(variable1, variable2)
148         class (type_base_standard_variable), intent(in) :: variable1, variable2
149
150         ! Compare the type of the standard variables.
151         compare = same_type_as(variable1, variable2) .or. extends_type_of(variable1, variable2) .or. extends_type_of(
    variable2, variable1)
152
153         ! Compare the metadata of the standard variables.
154         if (compare) compare = (variable1%name  == '' .or. variable2%name  == '' .or. variable1%name  == variable2%na
    me ) &
155                           .and. (variable1%units == '' .or. variable2%units == '' .or. variable1%units == variable2%un
    its)
156      end function
157
158   end function base_standard_variable_resolve
159
160   recursive subroutine add(standard_variable, own)
161      class (type_base_standard_variable), target, intent(inout) :: standard_variable
162      logical, optional,                           intent(in)    :: own
163
164      type (type_standard_variable_node), pointer :: node
165      type (type_interior_standard_variable)   :: in_interior
166      type (type_horizontal_standard_variable) :: at_interfaces
167      type (type_surface_standard_variable)    :: at_surface
168      type (type_bottom_standard_variable)     :: at_bottom
169
170      select type (standard_variable)
171      class is (type_universal_standard_variable)
172         call add_child(in_interior,   trim(standard_variable%name),                      standard_variable%units, stan
    dard_variable)
173         call add_child(at_surface,    trim(standard_variable%name) // '_at_surface',    trim(standard_variable%units)
     // '*m', standard_variable)
174         call add_child(at_bottom,     trim(standard_variable%name) // '_at_bottom',     trim(standard_variable%units)
     // '*m', standard_variable)
175         call add_child(at_interfaces, trim(standard_variable%name) // '_at_interfaces', trim(standard_variable%units)
     // '*m', standard_variable)
176      end select
177
178      allocate(node)
179      node%p => standard_variable
180      if (present(own)) node%own = own
181      node%next => standard_variables%first
182      standard_variables%first => node
183      standard_variable%resolved = .true.
```

```
184
185    contains
186
187       subroutine add_child(standard_variable, name, units, universal_standard_variable)
188          class (type_domain_specific_standard_variable), target :: standard_variable
189          character(len=*), intent(in)                           :: name, units
190          type (type_universal_standard_variable),        target :: universal_standard_variable
191          class (type_domain_specific_standard_variable), pointer :: p
192
193          standard_variable%name = name
194          standard_variable%units = units
195          p => standard_variable%typed_resolve()
196          p%universal => universal_standard_variable
197          p%aggregate_variable = universal_standard_variable%aggregate_variable
198          select type (p)
199          type is (type_interior_standard_variable);   universal_standard_variable%pin_interior => p
200          type is (type_surface_standard_variable);    universal_standard_variable%pat_surface => p
201          type is (type_bottom_standard_variable);     universal_standard_variable%pat_bottom => p
202          type is (type_horizontal_standard_variable); universal_standard_variable%pat_interfaces => p
203 #ifndef NDEBUG
204          class default
205             write (*,*) 'fabm_standard_variables::add::add_child: BUG wrong type returned'
206             stop 1
207 #endif
208          end select
209       end subroutine
210
211    end subroutine
212
213    function universal_standard_variable_typed_resolve(self) result(p)
214       class (type_universal_standard_variable), intent(in), target  :: self
215       class (type_universal_standard_variable), pointer             :: p
216
217       class (type_base_standard_variable), pointer :: presolved
218
219       presolved => self%resolve()
220       select type (presolved)
221       class is (type_universal_standard_variable)
222          p => presolved
223 #ifndef NDEBUG
224       class default
225          write (*,*) 'universal_standard_variable_typed_resolve: BUG wrong type returned'
226          stop 1
227 #endif
228       end select
229    end function
230
231    function universal_standard_variable_in_interior(self) result(p)
232       class (type_universal_standard_variable), intent(in), target  :: self
233       class (type_interior_standard_variable), pointer              :: p
234
235       class (type_universal_standard_variable), pointer  :: presolved
236
237       presolved => self%typed_resolve()
238       p => presolved%pin_interior
239    end function
240
241    function universal_standard_variable_at_surface(self) result(p)
242       class (type_universal_standard_variable), intent(in), target  :: self
243       class (type_surface_standard_variable), pointer               :: p
244
245       class (type_universal_standard_variable), pointer  :: presolved
246
247       presolved => self%typed_resolve()
248       p => presolved%pat_surface
249    end function
250
251    function universal_standard_variable_at_bottom(self) result(p)
252       class (type_universal_standard_variable), intent(in), target  :: self
253       class (type_bottom_standard_variable), pointer                :: p
254
255       class (type_universal_standard_variable), pointer  :: presolved
256
257       presolved => self%typed_resolve()
258       p => presolved%pat_bottom
259    end function
260
261    function universal_standard_variable_at_interfaces(self) result(p)
262       class (type_universal_standard_variable), intent(in), target  :: self
263       class (type_horizontal_standard_variable), pointer            :: p
264
265       class (type_universal_standard_variable), pointer  :: presolved
266
267       presolved => self%typed_resolve()
268       p => presolved%pat_interfaces
269    end function
270
271    function domain_specific_standard_variable_typed_resolve(self) result(p)
272       class (type_domain_specific_standard_variable), intent(in), target  :: self
273       class (type_domain_specific_standard_variable), pointer             :: p
274
275       class (type_base_standard_variable), pointer :: pbase
276
277       pbase => self%resolve()
278       select type (pbase)
279       class is (type_domain_specific_standard_variable)
280          p => pbase
281 #ifndef NDEBUG
```

```
282       class default
283          write (*,*) 'domain_specific_standard_variable_typed_resolve: BUG wrong type returned'
284          stop 1
285 #endif
286       end select
287    end function
288
289    subroutine base_standard_variable_assert_resolved(self)
290       class (type_base_standard_variable), intent(in) :: self
291
292       if (self%resolved) return
293       write (*,*) 'FATAL ERROR: standard_variable_collection_assert_contains: "' // trim(self%name) // '" not in stand
    ard variable collection."'
294       stop 1
295    end subroutine
296
297    subroutine standard_variable_collection_initialize(self)
298       class (type_standard_variable_collection), intent(inout) :: self
299 #include "standard_variable_assignments.h"
300    end subroutine
301
302    subroutine standard_variable_collection_finalize(self)
303       class (type_standard_variable_collection), intent(inout) :: self
304
305       type (type_standard_variable_node), pointer :: node, next
306
307       node => self%first
308       do while (associated(node))
309          next => node%next
310          if (node%own) deallocate(node%p)
311          deallocate(node)
312          node => next
313       end do
314       self%first => null()
315    end subroutine standard_variable_collection_finalize
316
317    logical function standard_variable_set_contains_variable(self, standard_variable)
318       class (type_standard_variable_set),  intent(in) :: self
319       class (type_base_standard_variable), target     :: standard_variable
320
321       type (type_standard_variable_node), pointer :: node
322       logical,                            pointer :: pmember
323
324 #ifndef NDEBUG
325       call standard_variable%assert_resolved()
326 #endif
327       standard_variable_set_contains_variable = .true.
328       node => self%first
329       pmember => standard_variable%aggregate_variable
330       do while (associated(node))
331 #ifndef NDEBUG
332          call node%p%assert_resolved()
333 #endif
334          ! Note: for Cray 10.0.4, the comparison below fails for class pointers! Therefore we compare type member refe
    rences.
335          if (associated(pmember, node%p%aggregate_variable)) return
336          node => node%next
337       end do
338       standard_variable_set_contains_variable = .false.
339    end function standard_variable_set_contains_variable
340
341    logical function standard_variable_set_contains_name(self, name)
342       class (type_standard_variable_set), intent(in) :: self
343       character(len=*),                   intent(in) :: name
344
345       type (type_standard_variable_node), pointer :: node
346
347       standard_variable_set_contains_name = .true.
348       node => self%first
349       do while (associated(node))
350          if (node%p%name == name) return
351          node => node%next
352       end do
353       standard_variable_set_contains_name = .false.
354    end function standard_variable_set_contains_name
355
356    subroutine standard_variable_set_add(self, standard_variable)
357       class (type_standard_variable_set),  intent(inout) :: self
358       class (type_base_standard_variable), target        :: standard_variable
359
360       type (type_standard_variable_node), pointer :: node
361
362 #ifndef NDEBUG
363       call standard_variable%assert_resolved()
364 #endif
365
366       if (self%contains(standard_variable)) return
367
368       if (.not. associated(self%first)) then
369          allocate(self%first)
370          node => self%first
371       else
372          node => self%first
373          do while (associated(node%next))
374             node => node%next
375          end do
376          allocate(node%next)
377          node => node%next
```

```
      end if
      node%p => standard_variable
   end subroutine standard_variable_set_add

   subroutine standard_variable_set_update(self, other)
      class (type_standard_variable_set), intent(inout) :: self
      class (type_standard_variable_set), intent(in)    :: other

      type (type_standard_variable_node), pointer :: node

      node => other%first
      do while (associated(node))
         call self%add(node%p)
         node => node%next
      end do
   end subroutine standard_variable_set_update

   subroutine standard_variable_set_finalize(self)
      class (type_standard_variable_set), intent(inout) :: self

      type (type_standard_variable_node), pointer :: node, next_node

      node => self%first
      do while (associated(node))
         next_node => node%next
         deallocate(node)
         node => next_node
      end do
      self%first => null()
   end subroutine standard_variable_set_finalize

end module fabm_standard_variables
```