

```

1 | ! =====
2 | ! Validate input symbols
3 | ! =====
4 |
5 | #ifndef _FABM_DIMENSION_COUNT_
6 | # error Preprocessor variable _FABM_DIMENSION_COUNT_ must be defined.
7 | #endif
8 |
9 | #if (_FABM_DIMENSION_COUNT_<0||_FABM_DIMENSION_COUNT_>3)
10 | # error Preprocessor variable _FABM_DIMENSION_COUNT_ takes values between 0 and 3 only.
11 | #endif
12 |
13 | #ifdef _FABM_DEPTH_DIMENSION_INDEX_
14 | # if (_FABM_DEPTH_DIMENSION_INDEX_<1)||(_FABM_DEPTH_DIMENSION_INDEX_>_FABM_DIMENSION_COUNT_)
15 | # error Preprocessor variable _FABM_DEPTH_DIMENSION_INDEX_ takes values between 1 and _FABM_DIMENSION_COUNT_ only.
16 | # endif
17 | #endif
18 |
19 | #ifdef _FABM_VECTORIZED_DIMENSION_INDEX_
20 | # if (_FABM_VECTORIZED_DIMENSION_INDEX_<1)||(_FABM_VECTORIZED_DIMENSION_INDEX_>_FABM_DIMENSION_COUNT_)
21 | # error Preprocessor variable _FABM_VECTORIZED_DIMENSION_INDEX_ takes values between 1 and _FABM_DIMENSION_COUNT_ o
nly.
22 | # endif
23 | #endif
24 |
25 | ! =====
26 | ! End of input symbol validation
27 | ! =====
28 |
29 | #ifndef _NO_DO_CONCURRENT_
30 | # define _DO_CONCURRENT_(iterator,start,stop) do concurrent (iterator=start:stop)
31 | # define _DO_CONCURRENT_WITH_STRIDE_(iterator,start,stop,stride) do concurrent (iterator=start:stop:stride)
32 | #else
33 | # define _DO_CONCURRENT_(iterator,start,stop) do iterator=start,stop
34 | # define _DO_CONCURRENT_WITH_STRIDE_(iterator,start,stop,stride) do iterator=start,stop,stride
35 | #endif
36 |
37 | #if defined(_FABM_VECTORIZED_DIMENSION_INDEX_)||defined(_FABM_DEPTH_DIMENSION_INDEX_)
38 | # define _INTERIOR_IS_VECTORIZED_
39 | #endif
40 |
41 | #if defined(_FABM_VECTORIZED_DIMENSION_INDEX_)&&(_FABM_DEPTH_DIMENSION_INDEX_!=_FABM_VECTORIZED_DIMENSION_INDEX_)
42 | # define _HORIZONTAL_IS_VECTORIZED_
43 | #endif
44 |
45 | #define _I_ l__
46 | #define _J_ m__
47 | #define _N_ cache%n
48 |
49 | #ifdef _INTERIOR_IS_VECTORIZED_
50 | ! Interior fields are 1D
51 | # define _DIMENSION_SLICE_ ,dimension(:)
52 | # define _DIMENSION_SLICE_PLUS_1_ ,dimension(:, :)
53 | # define _DIMENSION_SLICE_PLUS_2_ ,dimension(:, :, :)
54 | # define _INDEX_SLICE_ (_I_)
55 | # define _INDEX_SLICE_PLUS_1_(i) (_I_, i)
56 | # define _INDEX_SLICE_PLUS_2_(i,j) (_I_, i, j)
57 | # define _DIMENSION_SLICE_AUTOMATIC_ ,dimension(_N_)
58 | #else
59 | ! Interior fields are 0D
60 | # define _DIMENSION_SLICE_
61 | # define _DIMENSION_SLICE_PLUS_1_ ,dimension(:)
62 | # define _DIMENSION_SLICE_PLUS_2_ ,dimension(:, :)
63 | # define _INDEX_SLICE_
64 | # define _INDEX_SLICE_PLUS_1_(i) (i)
65 | # define _INDEX_SLICE_PLUS_2_(i,j) (i,j)
66 | # define _DIMENSION_SLICE_AUTOMATIC_
67 | #endif
68 |
69 | #ifdef _HORIZONTAL_IS_VECTORIZED_
70 | ! Horizontal fields are 1D
71 | # define _DIMENSION_HORIZONTAL_SLICE_ ,dimension(:)
72 | # define _DIMENSION_HORIZONTAL_SLICE_PLUS_1_ ,dimension(:, :)
73 | # define _DIMENSION_HORIZONTAL_SLICE_PLUS_2_ ,dimension(:, :, :)
74 | # define _INDEX_HORIZONTAL_SLICE_ (_J_)
75 | # define _INDEX_HORIZONTAL_SLICE_PLUS_1_(i) (_J_, i)
76 | # define _INDEX_HORIZONTAL_SLICE_PLUS_2_(i,j) (_J_, i, j)
77 | # define _DIMENSION_HORIZONTAL_SLICE_AUTOMATIC_ ,dimension(_N_)
78 | #else
79 | ! Horizontal fields are 0D
80 | # define _DIMENSION_HORIZONTAL_SLICE_
81 | # define _DIMENSION_HORIZONTAL_SLICE_PLUS_1_ ,dimension(:)
82 | # define _DIMENSION_HORIZONTAL_SLICE_PLUS_2_ ,dimension(:, :)
83 | # define _INDEX_HORIZONTAL_SLICE_
84 | # define _INDEX_HORIZONTAL_SLICE_PLUS_1_(i) (i)
85 | # define _INDEX_HORIZONTAL_SLICE_PLUS_2_(i,j) (i,j)
86 | # define _DIMENSION_HORIZONTAL_SLICE_AUTOMATIC_
87 | #endif
88 |
89 | ! Preprocessor symbols for procedures operating on an INTERIOR slice
90 | #ifdef _FABM_VECTORIZED_DIMENSION_INDEX_
91 | ! Interior procedures operate in 1D
92 | ! Interior slices MUST be 1D arrays; horizontal slices may be scalars or 1D arrays
93 | # define _LOOP_END_ end do
94 | # ifdef _HORIZONTAL_IS_VECTORIZED_
95 | ! Horizontal slices are 1D arrays
96 | ! For instance, model with i,j,k [MOM,NEMO] or i,j or i,k [FVCOM], or i; vectorized along i or j
97 | # define _DECLARE_INTERIOR_INDICES_ integer :: _I_, _J_

```

```

98 # define _LOOP_BEGIN_EX(cache) do _I_=1,cache%n;_J=_I_
99 # define _CONCURRENT_LOOP_BEGIN_EX(cache) _DO_CONCURRENT(_I_,1,cache%n);_J=_I_
100 # else
101 ! Horizontal slices are scalars
102 ! For instance model with k [GOTM] or i,k, i,j,k, vectorized along k
103 # define _DECLARE_INTERIOR_INDICES_ integer :: _I_
104 # define _LOOP_BEGIN_EX(cache) do _I_=1,cache%n
105 # define _CONCURRENT_LOOP_BEGIN_EX(cache) _DO_CONCURRENT(_I_,1,cache%n)
106 # endif
107 #else
108 ! Interior procedures operate in 0D
109 ! Interior slices may be 0D scalars or 1D arrays [the latter if the model has a vertical dimension]
110 # define _LOOP_BEGIN_EX(cache)
111 # define _CONCURRENT_LOOP_BEGIN_EX(cache)
112 # define _LOOP_END_
113 # ifdef _INTERIOR_IS_VECTORIZED_
114 ! Interior slices are 1D arrays - we will operate on their first element (_I_=1)
115 # define _DECLARE_INTERIOR_INDICES_ integer,parameter :: _I_=1
116 # else
117 ! Interior slices are scalars
118 # define _DECLARE_INTERIOR_INDICES_
119 # endif
120 #endif
121 #define _ARGUMENTS_INTERIOR_ cache
122 #define _DECLARE_ARGUMENTS_INTERIOR_ type (type_interior_cache),intent(inout) :: cache;_DECLARE_INTERIOR_INDICES_
123 #define _LOOP_BEGIN_ _LOOP_BEGIN_EX(cache)
124 #define _CONCURRENT_LOOP_BEGIN_ _CONCURRENT_LOOP_BEGIN_EX(cache)
125
126 ! Preprocessor symbols for procedures operating on a HORIZONTAL slice
127 #ifndef _HORIZONTAL_IS_VECTORIZED_
128 ! Horizontal procedures operate in 1D
129 ! Horizontal and interior slices MUST be 1D. Use same index for horizontal and interior (_I=_J_)
130 # define _DECLARE_HORIZONTAL_INDICES_ integer :: _I_,_J_
131 # define _HORIZONTAL_LOOP_BEGIN_EX(cache) do _J_=1,cache%n;_I=_J_
132 # define _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX(cache) _DO_CONCURRENT(_J_,1,cache%n);_I=_J_
133 # define _HORIZONTAL_LOOP_END_ end do
134 #else
135 ! Horizontal procedures operate in 0D
136 ! Horizontal slices MUST be scalars; interior slices can be scalars or 1D arrays
137 # define _HORIZONTAL_LOOP_BEGIN_EX(cache)
138 # define _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX(cache)
139 # define _HORIZONTAL_LOOP_END_
140 # ifdef _INTERIOR_IS_VECTORIZED_
141 ! Interior slices are 1D arrays - we will operate on their first element (_I_=1)
142 # define _DECLARE_HORIZONTAL_INDICES_ integer,parameter :: _I_=1
143 # else
144 ! Interior slices are scalars
145 # define _DECLARE_HORIZONTAL_INDICES_
146 # endif
147 #endif
148 #define _ARGUMENTS_HORIZONTAL_ cache
149 #define _DECLARE_ARGUMENTS_HORIZONTAL_ type (type_horizontal_cache),intent(inout) :: cache;_DECLARE_HORIZONTAL_INDICES_
150
151 #define _HORIZONTAL_LOOP_BEGIN_ _HORIZONTAL_LOOP_BEGIN_EX(cache)
152 #define _CONCURRENT_HORIZONTAL_LOOP_BEGIN_ _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX(cache)
153 #define _SURFACE_LOOP_BEGIN_ _HORIZONTAL_LOOP_BEGIN_
154 #define _SURFACE_LOOP_END_ _HORIZONTAL_LOOP_END_
155 #define _BOTTOM_LOOP_BEGIN_ _HORIZONTAL_LOOP_BEGIN_
156 #define _BOTTOM_LOOP_END_ _HORIZONTAL_LOOP_END_
157
158 ! Preprocessor symbols for procedures operating on a VERTICAL slice
159 #ifndef _FABM_DEPTH_DIMENSION_INDEX_
160 ! Vertical procedures operate in 1D
161 ! Interior slices MUST be 1D arrays; horizontal slices may be 0D or 1D
162 ! Applies to all models with depth dimension. For instance, model with i,j,k [MOM,NEMO], i,k [FVCOM], or k [GOTM]
163 # define _VERTICAL_LOOP_EXIT_ exit
164 # ifdef _FABM_VERTICAL_BOTTOM_TO_SURFACE_
165 # define _DOWNWARD_LOOP_BEGIN_ do _I=_N_,1,-1
166 # define _UPWARD_LOOP_BEGIN_ do _I_=1,_N_
167 # define _MOVE_TO_BOTTOM_ _I_=1
168 # define _MOVE_TO_SURFACE_ _I=_N_
169 # else
170 # define _DOWNWARD_LOOP_BEGIN_ do _I_=1,_N_
171 # define _UPWARD_LOOP_BEGIN_ do _I=_N_,1,-1
172 # define _MOVE_TO_SURFACE_ _I=1
173 # define _MOVE_TO_BOTTOM_ _I=_N_
174 # endif
175 # define _CONCURRENT_VERTICAL_LOOP_BEGIN_EX(cache) _DO_CONCURRENT(_I_,1,cache%n)
176 # ifdef _HORIZONTAL_IS_VECTORIZED_
177 ! Horizontal slices are 1D arrays - we will operate on their first element (_J_=1)
178 ! For instance, model with i,j,k [MOM,NEMO] or i,k [FVCOM]; vectorized along i or j
179 # define _DECLARE_VERTICAL_INDICES_ integer :: _I_;integer,parameter :: _J_=1
180 # else
181 ! Horizontal slices are scalars
182 ! For instance model with k [GOTM] or i,k, i,j,k, vectorized along k
183 # define _DECLARE_VERTICAL_INDICES_ integer :: _I_
184 # endif
185 #else
186 ! Vertical procedures operate in 0D
187 ! Interior slices may scalars or 1D arrays [the latter if the model is vectorized over a horizontal dimension]
188 ! Applies to all models without depth dimension; for instance, 0D box or model with i,j or i
189 # define _CONCURRENT_VERTICAL_LOOP_BEGIN_EX(cache)
190 # define _VERTICAL_LOOP_END_
191 # define _VERTICAL_LOOP_EXIT_
192 # define _DOWNWARD_LOOP_BEGIN_
193 # define _UPWARD_LOOP_BEGIN_
194 # define _MOVE_TO_SURFACE_

```

```

195 # define _MOVE_TO_BOTTOM_
196 # ifdef _INTERIOR_IS_VECTORIZED_
197 !   Interior slices are 1D arrays. Since the vectorized dimension is not depth, horizontal slices MUST be 1D arrays t
198 !   Operate on their first element (_I=1,_J=1)
199 !   For instance, model with i,j or i; vectorized along i or j
200 #   define _DECLARE_VERTICAL_INDICES_ integer,parameter :: _I=1,_J=1
201 #   else
202 !   Both interior and horizontal slices are scalars
203 !   For instance, 0D box
204 #   define _DECLARE_VERTICAL_INDICES_
205 #   endif
206 #endif
207 #define _VERTICAL_LOOP_BEGIN_ _DOWNWARD_LOOP_BEGIN_
208 #define _DOWNWARD_LOOP_END_ _VERTICAL_LOOP_END_
209 #define _UPWARD_LOOP_END_ _VERTICAL_LOOP_END_
210 #define _ARGUMENTS_VERTICAL_ cache
211 #define _DECLARE_ARGUMENTS_VERTICAL_ type (type_vertical_cache),intent(inout) :: cache;_DECLARE_VERTICAL_INDICES_
212 #define _CONCURRENT_VERTICAL_LOOP_BEGIN_ _CONCURRENT_VERTICAL_LOOP_BEGIN_EX_(cache)
213
214 ! Preprocessor symbols for procedures operating on a single point in space.
215 #ifdef _INTERIOR_IS_VECTORIZED_
216 #   ifdef _HORIZONTAL_IS_VECTORIZED_
217 #       define _ARGUMENTS_LOCAL_ cache,_I,_J
218 #       define _DECLARE_ARGUMENTS_LOCAL_ class (type_cache),intent(in) :: cache;integer,intent(in) :: _I,_J
219 #   else
220 #       define _ARGUMENTS_LOCAL_ cache,_I
221 #       define _DECLARE_ARGUMENTS_LOCAL_ class (type_cache),intent(in) :: cache;integer,intent(in) :: _I
222 #   endif
223 #else
224 #   define _ARGUMENTS_LOCAL_ cache
225 #   define _DECLARE_ARGUMENTS_LOCAL_ class (type_cache),intent(in) :: cache
226 #endif
227
228 ! For BGC models: FABM arguments to routines implemented by biogeochemical models.
229 #define _ARGUMENTS_DO_ _ARGUMENTS_INTERIOR_
230 #define _ARGUMENTS_DO_PPDD_ _ARGUMENTS_INTERIOR_,pp,dd
231 #define _ARGUMENTS_DO_SURFACE_ _ARGUMENTS_HORIZONTAL_
232 #define _ARGUMENTS_DO_BOTTOM_ _ARGUMENTS_HORIZONTAL_
233 #define _ARGUMENTS_DO_BOTTOM_PPDD_ _ARGUMENTS_HORIZONTAL_,pp,dd,benthos_offset
234 #define _ARGUMENTS_DO_COLUMN_ _ARGUMENTS_VERTICAL_
235 #define _ARGUMENTS_GET_VERTICAL_MOVEMENT_ _ARGUMENTS_INTERIOR_
236 #define _ARGUMENTS_GET_EXTINCTION_ _ARGUMENTS_INTERIOR_
237 #define _ARGUMENTS_GET_DRAG_ _ARGUMENTS_HORIZONTAL_
238 #define _ARGUMENTS_GET_ALBEDO_ _ARGUMENTS_HORIZONTAL_
239 #define _ARGUMENTS_CHECK_STATE_ _ARGUMENTS_INTERIOR_
240 #define _ARGUMENTS_CHECK_SURFACE_STATE_ _ARGUMENTS_HORIZONTAL_
241 #define _ARGUMENTS_CHECK_BOTTOM_STATE_ _ARGUMENTS_HORIZONTAL_
242 #define _ARGUMENTS_INITIALIZE_STATE_ _ARGUMENTS_INTERIOR_
243 #define _ARGUMENTS_INITIALIZE_HORIZONTAL_STATE_ _ARGUMENTS_HORIZONTAL_
244
245 ! For BGC models: Declaration of FABM arguments to routines implemented by biogeochemical models.
246 #define _DECLARE_ARGUMENTS_DO_ _DECLARE_ARGUMENTS_INTERIOR_
247 #define _DECLARE_ARGUMENTS_DO_PPDD_ _DECLARE_ARGUMENTS_INTERIOR_;real(rke) _DIMENSION_SLICE_PLUS_2_,intent(inout) :: p
248 #define _DECLARE_ARGUMENTS_DO_BOTTOM_ _DECLARE_ARGUMENTS_HORIZONTAL_
249 #define _DECLARE_ARGUMENTS_DO_BOTTOM_PPDD_ _DECLARE_ARGUMENTS_HORIZONTAL_;real(rke) _DIMENSION_HORIZONTAL_SLICE_PLUS_2_
250 #define _DECLARE_ARGUMENTS_DO_COLUMN_ _DECLARE_ARGUMENTS_VERTICAL_
251 #define _DECLARE_ARGUMENTS_DO_SURFACE_ _DECLARE_ARGUMENTS_HORIZONTAL_
252 #define _DECLARE_ARGUMENTS_GET_VERTICAL_MOVEMENT_ _DECLARE_ARGUMENTS_INTERIOR_
253 #define _DECLARE_ARGUMENTS_GET_EXTINCTION_ _DECLARE_ARGUMENTS_INTERIOR_
254 #define _DECLARE_ARGUMENTS_GET_DRAG_ _DECLARE_ARGUMENTS_HORIZONTAL_
255 #define _DECLARE_ARGUMENTS_GET_ALBEDO_ _DECLARE_ARGUMENTS_HORIZONTAL_
256 #define _DECLARE_ARGUMENTS_CHECK_STATE_ _DECLARE_ARGUMENTS_INTERIOR_
257 #define _DECLARE_ARGUMENTS_CHECK_SURFACE_STATE_ _DECLARE_ARGUMENTS_HORIZONTAL_
258 #define _DECLARE_ARGUMENTS_CHECK_BOTTOM_STATE_ _DECLARE_ARGUMENTS_HORIZONTAL_
259 #define _DECLARE_ARGUMENTS_INITIALIZE_STATE_ _DECLARE_ARGUMENTS_INTERIOR_
260 #define _DECLARE_ARGUMENTS_INITIALIZE_HORIZONTAL_STATE_ _DECLARE_ARGUMENTS_HORIZONTAL_
261
262 #define _ADD_(variable,value) cache%write _INDEX_SLICE_PLUS_1_(variable%sum_index) = cache%write _INDEX_SLICE_PLUS_1_(
263   variable%sum_index) + (value)
264 #define _ADD_HORIZONTAL_(variable,value) cache%write_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%horizontal_sum_index)
265   = cache%write_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%horizontal_sum_index) + (value)
266
267 ! For BGC models: Expressions for setting space-dependent FABM variables defined on the full spatial domain.
268 #define _ADD_SOURCE_(variable,value) _ADD_(variable%sms,(value)*self%rdt__)
269 #define _ADD_BOTTOM_SOURCE_(variable,value) _ADD_HORIZONTAL_(variable%bottom_sms,(value)*self%rdt__)
270 #define _ADD_SURFACE_SOURCE_(variable,value) _ADD_HORIZONTAL_(variable%surface_sms,(value)*self%rdt__)
271 #define _ADD_BOTTOM_FLUX_(variable,value) _ADD_HORIZONTAL_(variable%bottom_flux,(value)*self%rdt__)
272 #define _ADD_SURFACE_FLUX_(variable,value) _ADD_HORIZONTAL_(variable%surface_flux,(value)*self%rdt__)
273 #define _SET_DD_(variable1,variable2,value) dd _INDEX_SLICE_PLUS_2_(variable1%state_index,variable2%state_index) = dd
274   _INDEX_SLICE_PLUS_2_(variable1%state_index,variable2%state_index) + (value)*self%rdt__
275 #define _SET_PP_(variable1,variable2,value) pp _INDEX_SLICE_PLUS_2_(variable1%state_index,variable2%state_index) = pp
276   _INDEX_SLICE_PLUS_2_(variable1%state_index,variable2%state_index) + (value)*self%rdt__
277 #define _ADD_VERTICAL_VELOCITY_(variable,value) _ADD_(variable%movement,(value)*self%rdt__)
278 #define _INVALIDATE_STATE_ cache%valid = .false.
279 #define _REPAIR_STATE_ cache%repair
280
281 #define _GET_WITH_BACKGROUND_(variable,target) target = cache%read _INDEX_SLICE_PLUS_1_(variable%index)+variable%backg
282   round
283 #define _GET_HORIZONTAL_WITH_BACKGROUND_(variable,target) target = cache%read_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(varia
284   ble%horizontal_index)+variable%background
285
286 ! For BGC models: quick expressions for setting a single element in both the destruction and production matrix.
287 #define _SET_DD_SYM_(variable1,variable2,value) _SET_DD_(variable1,variable2,value);_SET_PP_(variable2,variable1,value)
288 #define _SET_PP_SYM_(variable1,variable2,value) _SET_PP_(variable1,variable2,value);_SET_DD_(variable2,variable1,value)

```

```

283 | )
284 | ! For BGC models: macro to determine whether a variable identifier is in use (i.e., has been registered with FABM)
285 | #define _VARIABLE_REGISTERED(variable) associated(variable%link)
286 | #define _AVAILABLE_(variable) variable%index/!=-1
287 | #define _AVAILABLE_HORIZONTAL_(variable) variable%horizontal_index/!=-1
288 |
289 | ! For BGC models: read/write variable access.
290 | #define _GET_(variable,target) target = cache%read _INDEX_SLICE_PLUS_1_(variable%index)
291 | #define _GET_HORIZONTAL_(variable,target) target = cache%read_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%horizontal_i
index)
292 | #define _GET_SURFACE_(variable,target) _GET_HORIZONTAL_(variable,target)
293 | #define _GET_BOTTOM_(variable,target) _GET_HORIZONTAL_(variable,target)
294 | #define _GET_GLOBAL_(variable,target) target = cache%read_scalar(variable%global_index)
295 | #define _SET_(variable,value) cache%set_interior=.true.;cache%read _INDEX_SLICE_PLUS_1_(variable%index) = value
296 | #define _SET_HORIZONTAL_(variable,value) cache%set_horizontal=.true.;cache%read_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(var
iable%horizontal_index) = value
297 | #define _SET_DIAGNOSTIC_(variable,value) cache%write _INDEX_SLICE_PLUS_1_(variable%write_index) = value
298 | #define _SET_HORIZONTAL_DIAGNOSTIC_(variable,value) cache%write_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%horizontal
_write_index) = value
299 | #define _SET_BOTTOM_DIAGNOSTIC_(variable,value) cache%write_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%bottom_write_i
index) = value
300 | #define _SET_SURFACE_DIAGNOSTIC_(variable,value) cache%write_hz _INDEX_HORIZONTAL_SLICE_PLUS_1_(variable%surface_write
_index) = value
301 |
302 | #define _ASSERT_(condition, routine, message) if (.not.(condition)) call driver%fatal_error(routine, message)
303 |
304 | ! Backward compatibility with pre-1.0 FABM (2020-04-22)
305 | #define _SET_ODE_(variable,value) _ADD_SOURCE_(variable,value)
306 | #define _SET_BOTTOM_ODE_(variable,value) _ADD_BOTTOM_SOURCE_(variable,value)
307 | #define _SET_SURFACE_ODE_(variable,value) _ADD_SURFACE_SOURCE_(variable,value)
308 | #define _SET_BOTTOM_EXCHANGE_(variable,value) _ADD_BOTTOM_FLUX_(variable,value)
309 | #define _SET_SURFACE_EXCHANGE_(variable,value) _ADD_SURFACE_FLUX_(variable,value)
310 | #define _SET_VERTICAL_MOVEMENT_(variable,value) _ADD_VERTICAL_VELOCITY_(variable,value)
311 | #define _SET_EXTINCTION_(value) _ADD_(self%extinction_id,value)
312 | #define _SCALE_DRAG_(value) _ADD_HORIZONTAL_(self%surface_drag_id,(value)-1.0_rk)
313 | #define _SET_ALBEDO_(value) _ADD_HORIZONTAL_(self%albedo_id,value)

```