```
#if _FABM_DIMENSION_COUNT_==0

! ---------------------
! 0D spatial context
! ---------------------

#define _LOCATION_
#define _LOCATION_DIMENSIONS_
#define _LOCATION_RANGE_
#define _BEGIN_GLOBAL_LOOP_
#define _END_GLOBAL_LOOP_

#elif _FABM_DIMENSION_COUNT_==1

! ---------------------
! 1D spatial context
! ---------------------

#define _LOCATION_ i__
#define _LOCATION_DIMENSIONS_ :
#define _LOCATION_RANGE_ istart__,istop__
#define _BEGIN_GLOBAL_LOOP_ do i__=istart__,istop__
#define _END_GLOBAL_LOOP_ end do

#ifdef _FABM_DEPTH_DIMENSION_INDEX_
#  define _BEGIN_OUTER_VERTICAL_LOOP_
#  define _END_OUTER_VERTICAL_LOOP_
#  define _GLOBAL_VERTICAL_(it) it
#endif

#ifdef _FABM_VECTORIZED_DIMENSION_INDEX_
#  define _INTERIOR_FIXED_LOCATION_
#  define _GLOBAL_INTERIOR_(it) it
#  define _BEGIN_OUTER_INTERIOR_LOOP_
#  define _END_OUTER_INTERIOR_LOOP_
#endif

#elif _FABM_DIMENSION_COUNT_==2

! ---------------------
! 2D spatial context
! ---------------------

#define _LOCATION_ i__,j__
#define _LOCATION_DIMENSIONS_ :,:
#define _LOCATION_RANGE_ istart__,istop__,jstart__,jstop__
#define _BEGIN_GLOBAL_LOOP_ do j__=jstart__,jstop__;do i__=istart__,istop__
#define _END_GLOBAL_LOOP_ end do;end do

#ifdef _FABM_DEPTH_DIMENSION_INDEX_
#  if _FABM_DEPTH_DIMENSION_INDEX_==1
#    define _HORIZONTAL_LOCATION_ j__
#    define _HORIZONTAL_LOCATION_RANGE_ jstart__,jstop__
#    define _BEGIN_OUTER_VERTICAL_LOOP_ do j__=jstart__,jstop__
#    define _END_OUTER_VERTICAL_LOOP_ end do
#    define _GLOBAL_VERTICAL_(it) it,j__
#  elif _FABM_DEPTH_DIMENSION_INDEX_==2
#    define _HORIZONTAL_LOCATION_ i__
#    define _HORIZONTAL_LOCATION_RANGE_ istart__,istop__
#    define _BEGIN_OUTER_VERTICAL_LOOP_ do i__=istart__,istop__
#    define _END_OUTER_VERTICAL_LOOP_ end do
#    define _GLOBAL_VERTICAL_(it) i__,it
#  endif
#  define _HORIZONTAL_LOCATION_DIMENSIONS_ :
#endif

#if _FABM_VECTORIZED_DIMENSION_INDEX_==1
#  define _INTERIOR_FIXED_LOCATION_ j__
#  define _GLOBAL_INTERIOR_(it) it,j__
#  if _FABM_DEPTH_DIMENSION_INDEX_==2
#    define _GLOBAL_HORIZONTAL_(it) it
#  endif
#  define _BEGIN_OUTER_INTERIOR_LOOP_ do j__=jstart__,jstop__
#  define _END_OUTER_INTERIOR_LOOP_ end do
#elif _FABM_VECTORIZED_DIMENSION_INDEX_==2
#  define _INTERIOR_FIXED_LOCATION_ i__
#  define _GLOBAL_INTERIOR_(it) i__,it
#  if _FABM_DEPTH_DIMENSION_INDEX_==1
#    define _GLOBAL_HORIZONTAL_(it) it
#  endif
#  define _BEGIN_OUTER_INTERIOR_LOOP_ do i__=istart__,istop__
#  define _END_OUTER_INTERIOR_LOOP_ end do
#endif

#elif _FABM_DIMENSION_COUNT_==3

! ---------------------
! 3D spatial context
! ---------------------

#define _LOCATION_ i__,j__,k__
#define _LOCATION_DIMENSIONS_ :,:,:
#define _LOCATION_RANGE_ istart__,istop__,jstart__,jstop__,kstart__,kstop__
#define _BEGIN_GLOBAL_LOOP_ do k__=kstart__,kstop__;do j__=jstart__,jstop__;do i__=istart__,istop__
#define _END_GLOBAL_LOOP_ end do;end do;end do

#ifdef _FABM_DEPTH_DIMENSION_INDEX_
#  if _FABM_DEPTH_DIMENSION_INDEX_==1
```

```
#    define _HORIZONTAL_LOCATION_ j__,k__
#    define _HORIZONTAL_LOCATION_RANGE_ jstart__,jstop__,kstart__,kstop__
#    define _BEGIN_OUTER_VERTICAL_LOOP_ do k__=kstart__,kstop__;do j__=jstart__,jstop__
#    define _END_OUTER_VERTICAL_LOOP_ end do;end do
#    define _GLOBAL_VERTICAL_(it) it,j__,k__
#  elif _FABM_DEPTH_DIMENSION_INDEX_==2
#    define _HORIZONTAL_LOCATION_ i__,k__
#    define _HORIZONTAL_LOCATION_RANGE_ istart__,istop__,kstart__,kstop__
#    define _BEGIN_OUTER_VERTICAL_LOOP_ do k__=kstart__,kstop__;do i__=istart__,istop__
#    define _END_OUTER_VERTICAL_LOOP_ end do;end do
#    define _GLOBAL_VERTICAL_(it) i__,it,k__
#  elif _FABM_DEPTH_DIMENSION_INDEX_==3
#    define _HORIZONTAL_LOCATION_ i__,j__
#    define _HORIZONTAL_LOCATION_RANGE_ istart__,istop__,jstart__,jstop__
#    define _BEGIN_OUTER_VERTICAL_LOOP_ do j__=jstart__,jstop__;do i__=istart__,istop__
#    define _END_OUTER_VERTICAL_LOOP_ end do;end do
#    define _GLOBAL_VERTICAL_(it) i__,j__,it
#  endif
#  define _HORIZONTAL_LOCATION_DIMENSIONS_ :,:
#endif

#if _FABM_VECTORIZED_DIMENSION_INDEX_==1
#  define _INTERIOR_FIXED_LOCATION_ j__,k__
#  define _GLOBAL_INTERIOR_(it) it,j__,k__
#  if _FABM_DEPTH_DIMENSION_INDEX_==2
#    define _HORIZONTAL_FIXED_LOCATION_ k__
#    define _GLOBAL_HORIZONTAL_(it) it,k__
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do k__=kstart__,kstop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  elif _FABM_DEPTH_DIMENSION_INDEX_==3
#    define _HORIZONTAL_FIXED_LOCATION_ j__
#    define _GLOBAL_HORIZONTAL_(it) it,j__
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do j__=jstart__,jstop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  endif
#  define _BEGIN_OUTER_INTERIOR_LOOP_ do k__=kstart__,kstop__;do j__=jstart__,jstop__
#  define _END_OUTER_INTERIOR_LOOP_ end do;end do
#elif _FABM_VECTORIZED_DIMENSION_INDEX_==2
#  define _INTERIOR_FIXED_LOCATION_ i__,k__
#  define _GLOBAL_INTERIOR_(it) i__,it,k__
#  if _FABM_DEPTH_DIMENSION_INDEX_==1
#    define _HORIZONTAL_FIXED_LOCATION_ k__
#    define _GLOBAL_HORIZONTAL_(it) it,k__
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do k__=kstart__,kstop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  elif _FABM_DEPTH_DIMENSION_INDEX_==3
#    define _HORIZONTAL_FIXED_LOCATION_ i__
#    define _GLOBAL_HORIZONTAL_(it) i__,it
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do i__=istart__,istop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  endif
#  define _BEGIN_OUTER_INTERIOR_LOOP_ do k__=kstart__,kstop__;do i__=istart__,istop__
#  define _END_OUTER_INTERIOR_LOOP_ end do;end do
#elif _FABM_VECTORIZED_DIMENSION_INDEX_==3
#  define _INTERIOR_FIXED_LOCATION_ i__,j__
#  define _GLOBAL_INTERIOR_(it) i__,j__,it
#  if _FABM_DEPTH_DIMENSION_INDEX_==1
#    define _HORIZONTAL_FIXED_LOCATION_ j__
#    define _GLOBAL_HORIZONTAL_(it) j__,it
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do j__=jstart__,jstop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  elif _FABM_DEPTH_DIMENSION_INDEX_==2
#    define _HORIZONTAL_FIXED_LOCATION_ i__
#    define _GLOBAL_HORIZONTAL_(it) i__,it
#    define _BEGIN_OUTER_HORIZONTAL_LOOP_ do i__=istart__,istop__
#    define _END_OUTER_HORIZONTAL_LOOP_ end do
#  endif
#  define _BEGIN_OUTER_INTERIOR_LOOP_ do j__=jstart__,jstop__;do i__=istart__,istop__
#  define _END_OUTER_INTERIOR_LOOP_ end do;end do
#endif

#endif

#if _FABM_VECTORIZED_DIMENSION_INDEX_==1
#  define _ITERATOR_ i__
#  define _START_ istart__
#  define _STOP_ istop__
#elif _FABM_VECTORIZED_DIMENSION_INDEX_==2
#  define _ITERATOR_ j__
#  define _START_ jstart__
#  define _STOP_ jstop__
#elif _FABM_VECTORIZED_DIMENSION_INDEX_==3
#  define _ITERATOR_ k__
#  define _START_ kstart__
#  define _STOP_ kstop__
#else
#  define _BEGIN_OUTER_INTERIOR_LOOP_ _BEGIN_GLOBAL_LOOP_
#  define _END_OUTER_INTERIOR_LOOP_ _END_GLOBAL_LOOP_
#endif

! If there is no depth dimension, horizontal dimensions match full dimensions.
#ifndef _FABM_DEPTH_DIMENSION_INDEX_
#  define _HORIZONTAL_FIXED_LOCATION_ _INTERIOR_FIXED_LOCATION_
#  define _HORIZONTAL_LOCATION_ _LOCATION_
#  define _HORIZONTAL_LOCATION_RANGE_ _LOCATION_RANGE_
#  define _HORIZONTAL_LOCATION_DIMENSIONS_ _LOCATION_DIMENSIONS_
#  define _HORIZONTAL_DIMENSION_COUNT_ _FABM_DIMENSION_COUNT_
#  define _BEGIN_OUTER_VERTICAL_LOOP_ _BEGIN_GLOBAL_LOOP_
```

```
197 #  define _END_OUTER_VERTICAL_LOOP_ _END_GLOBAL_LOOP_
198 #else
199 #  define _HORIZONTAL_DIMENSION_COUNT_ _FABM_DIMENSION_COUNT_-1
200 #endif
201
202 #if (!defined(_FABM_DEPTH_DIMENSION_INDEX_)||_FABM_DEPTH_DIMENSION_INDEX_==_FABM_VECTORIZED_DIMENSION_INDEX_)
203 #  define _BEGIN_OUTER_HORIZONTAL_LOOP_ _BEGIN_OUTER_INTERIOR_LOOP_
204 #  define _END_OUTER_HORIZONTAL_LOOP_ _END_OUTER_INTERIOR_LOOP_
205 #endif
206
207 #if defined(_FABM_VECTORIZED_DIMENSION_INDEX_)&&!defined(_FABM_DEPTH_DIMENSION_INDEX_)
208 #  define _GLOBAL_HORIZONTAL_(it) _GLOBAL_INTERIOR_(it)
209 #endif
210
211 ! Check for additional required preprocessor variables.
212 #ifndef _LOCATION_
213 #  error BUG: Preprocessor variable _LOCATION_ must be defined.
214 #endif
215 #ifndef _LOCATION_DIMENSIONS_
216 #  error BUG: Preprocessor variable _LOCATION_DIMENSIONS_ must be defined.
217 #endif
218 #if _HORIZONTAL_DIMENSION_COUNT_>0
219 #  ifndef _HORIZONTAL_LOCATION_
220 #    error BUG: Preprocessor variable _HORIZONTAL_LOCATION_ must be defined.
221 #  endif
222 #  ifndef _LOCATION_DIMENSIONS_
223 #    error BUG: Preprocessor variable _HORIZONTAL_LOCATION_DIMENSIONS_ must be defined.
224 #  endif
225 #endif
226 #if defined(_FABM_VECTORIZED_DIMENSION_INDEX_)&&!defined(_GLOBAL_INTERIOR_)
227 #  error BUG: Preprocessor variable _GLOBAL_INTERIOR_ must be defined since _FABM_VECTORIZED_DIMENSION_INDEX_ is set.
228 #endif
229 #if defined(_FABM_DEPTH_DIMENSION_INDEX_)&&!defined(_GLOBAL_VERTICAL_)
230 #  error BUG: Preprocessor variable _GLOBAL_VERTICAL_ must be defined since _FABM_DEPTH_DIMENSION_INDEX_ is set.
231 #endif
232 #if defined(_FABM_VECTORIZED_DIMENSION_INDEX_)&&_FABM_VECTORIZED_DIMENSION_INDEX_!=_FABM_DEPTH_DIMENSION_INDEX_&&!defi
    ned(_GLOBAL_HORIZONTAL_)
233 #  error BUG: Preprocessor variable _GLOBAL_HORIZONTAL_ must be defined since _FABM_VECTORIZED_DIMENSION_INDEX_ is set
     and not equal to _FABM_DEPTH_DIMENSION_INDEX_.
234 #endif
235
236 ! ================================================================================================================
237 ! Process spatial mask, based on the following variables provided by the driver:
238 !   _FABM_MASK_TYPE_ (data type of mask elements, e.g., logical, integer or real)
239 !   _FABM_MASKED_VALUE_ or _FABM_UNMASKED_VALUE_ (mask value for masked and unmasked cells, respectively)
240 ! ================================================================================================================
241
242 #ifdef _FABM_MASK_TYPE_
243 #  define _HAS_MASK_
244 #endif
245
246 #ifdef _HAS_MASK_
247 #  ifndef _INTERIOR_IS_VECTORIZED_
248 #    error _FABM_MASK_TYPE_/_FABM_MASKED_VALUE_/_FABM_UNMASKED_VALUE_ are not used if no dimension is vectorized.
249 #  endif
250 #  ifdef _FABM_IS_UNMASKED_
251 #    define _IS_UNMASKED_(maskvalue) _FABM_IS_UNMASKED_(maskvalue)
252 #  elif defined(_FABM_MASKED_VALUE_)
253 #    define _IS_UNMASKED_(maskvalue) maskvalue/=_FABM_MASKED_VALUE_
254 #  elif defined(_FABM_UNMASKED_VALUE_)
255 #    define _IS_UNMASKED_(maskvalue) maskvalue==_FABM_UNMASKED_VALUE_
256 #  else
257 #    error If _FABM_MASK_TYPE_ is set, _FABM_MASKED_VALUE_, _FABM_UNMASKED_VALUE_ or _FABM_IS_UNMASKED_ must be set as
     well.
258 #  endif
259 #else
260 #  ifdef _FABM_IS_UNMASKED_
261 #    error To use _FABM_IS_UNMASKED_, _FABM_MASK_TYPE_ must be set as well.
262 #  endif
263 #  ifdef _FABM_MASKED_VALUE_
264 #    error To use _FABM_MASKED_VALUE_, _FABM_MASK_TYPE_ must be set as well.
265 #  endif
266 #  ifdef _FABM_UNMASKED_VALUE_
267 #    error To use _FABM_UNMASKED_VALUE_, _FABM_MASK_TYPE_ must be set as well.
268 #  endif
269 #  define _IS_UNMASKED_(maskvalue) .true.
270 #endif
271
272 #if defined(_FABM_CONTIGUOUS_)&&!defined(_NO_CONTIGUOUS_)
273 #  define _CONTIGUOUS_ ,contiguous
274 #else
275 #  define _CONTIGUOUS_
276 #endif
277
278 #ifndef _BEGIN_OUTER_HORIZONTAL_LOOP_
279 #  define _BEGIN_OUTER_HORIZONTAL_LOOP_
280 #  define _END_OUTER_HORIZONTAL_LOOP_
281 #endif
282
283 ! ============================================================================
284 ! Further preprocessor macros for specifying spatial dimensionality and position
285 ! ============================================================================
286
287
288 #ifdef _FABM_VECTORIZED_DIMENSION_INDEX_
289 #  define _DIMENSION_EXT_SLICE_ ,dimension(_START_:)
290 #  define _DIMENSION_EXT_SLICE_PLUS_1_ ,dimension(_START_:,:)
291 #  define _DIMENSION_EXT_SLICE_PLUS_2_ ,dimension(_START_:,:,:)
```

```
292 #  define _INDEX_EXT_SLICE_ (_START_+_I_-1)
293 #  define _INDEX_EXT_SLICE_PLUS_1_(i) (_START_+_I_-1,i)
294 #  define _INDEX_EXT_SLICE_PLUS_2_(i,j) (_START_+_I_-1,i,j)
295 #else
296 #  define _DIMENSION_EXT_SLICE_
297 #  define _DIMENSION_EXT_SLICE_PLUS_1_ ,dimension(:)
298 #  define _DIMENSION_EXT_SLICE_PLUS_2_ ,dimension(:,:)
299 #  define _INDEX_EXT_SLICE_
300 #  define _INDEX_EXT_SLICE_PLUS_1_(i) (i)
301 #  define _INDEX_EXT_SLICE_PLUS_2_(i,j) (i,j)
302 #endif
303
304 #ifdef _HORIZONTAL_IS_VECTORIZED_
305 ! Horizontal fields are 1D
306 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_ _DIMENSION_EXT_SLICE_
307 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_PLUS_1_ _DIMENSION_EXT_SLICE_PLUS_1_
308 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_PLUS_2_ _DIMENSION_EXT_SLICE_PLUS_2_
309 #  define _INDEX_EXT_HORIZONTAL_SLICE_ _INDEX_EXT_SLICE_
310 #  define _INDEX_EXT_HORIZONTAL_SLICE_PLUS_1_(i) _INDEX_EXT_SLICE_PLUS_1_(i)
311 #  define _INDEX_EXT_HORIZONTAL_SLICE_PLUS_2_(i,j) _INDEX_EXT_SLICE_PLUS_2_(i,j)
312 #else
313 ! Horizontal fields are 0D
314 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_
315 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_PLUS_1_ ,dimension(:)
316 #  define _DIMENSION_EXT_HORIZONTAL_SLICE_PLUS_2_ ,dimension(:,:)
317 #  define _INDEX_EXT_HORIZONTAL_SLICE_
318 #  define _INDEX_EXT_HORIZONTAL_SLICE_PLUS_1_(i) (i)
319 #  define _INDEX_EXT_HORIZONTAL_SLICE_PLUS_2_(i,j) (i,j)
320 #endif
321
322 ! ------------------------------------------------------------------------------
323 ! Dimension attribute and index specifyer for horizontal (2D) fields.
324 ! ------------------------------------------------------------------------------
325
326 #if _HORIZONTAL_DIMENSION_COUNT_>0
327 #  define _INDEX_HORIZONTAL_LOCATION_ (_HORIZONTAL_LOCATION_)
328 #  define _DIMENSION_GLOBAL_HORIZONTAL_ ,dimension(_HORIZONTAL_LOCATION_DIMENSIONS_)
329 #  define _ATTRIBUTES_GLOBAL_HORIZONTAL_ _DIMENSION_GLOBAL_HORIZONTAL_ _CONTIGUOUS_
330 #  define _ARG_HORIZONTAL_LOCATION_ _HORIZONTAL_LOCATION_
331 #  define _POSTARG_HORIZONTAL_LOCATION_ ,_ARG_HORIZONTAL_LOCATION_
332 #  define _POSTARG_HORIZONTAL_LOCATION_RANGE_ ,_HORIZONTAL_LOCATION_RANGE_
333 #  define _PREARG_HORIZONTAL_LOCATION_ _ARG_HORIZONTAL_LOCATION_,
334 #  define _PREARG_HORIZONTAL_LOCATION_DIMENSIONS_ _HORIZONTAL_LOCATION_DIMENSIONS_,
335 #  define _DECLARE_ARGUMENTS_HORIZONTAL_LOCATION_ integer,intent(in) :: _HORIZONTAL_LOCATION_
336 #  define _DECLARE_ARGUMENTS_HORIZONTAL_LOCATION_RANGE_ integer,intent(in) :: _HORIZONTAL_LOCATION_RANGE_
337 #  define _DECLARE_HORIZONTAL_LOCATION_ integer :: _HORIZONTAL_LOCATION_
338 #else
339 #  define _INDEX_HORIZONTAL_LOCATION_
340 #  define _DIMENSION_GLOBAL_HORIZONTAL_
341 #  define _ATTRIBUTES_GLOBAL_HORIZONTAL_
342 #  define _ARG_HORIZONTAL_LOCATION_
343 #  define _POSTARG_HORIZONTAL_LOCATION_
344 #  define _POSTARG_HORIZONTAL_LOCATION_RANGE_
345 #  define _PREARG_HORIZONTAL_LOCATION_
346 #  define _PREARG_HORIZONTAL_LOCATION_DIMENSIONS_
347 #  define _DECLARE_ARGUMENTS_HORIZONTAL_LOCATION_
348 #  define _DECLARE_ARGUMENTS_HORIZONTAL_LOCATION_RANGE_
349 #  define _DECLARE_HORIZONTAL_LOCATION_
350 #endif
351
352 ! ------------------------------------------------------------------------------
353 ! Dimension attribute and index specifyer for full 3D fields.
354 ! ------------------------------------------------------------------------------
355
356 #if _FABM_DIMENSION_COUNT_>0
357 #  define _INDEX_LOCATION_ (_LOCATION_)
358 #  define _DIMENSION_GLOBAL_ ,dimension(_LOCATION_DIMENSIONS_)
359 #  define _ATTRIBUTES_GLOBAL_ _DIMENSION_GLOBAL_ _CONTIGUOUS_
360 #  define _POSTARG_LOCATION_ ,_LOCATION_
361 #  define _POSTARG_LOCATION_RANGE_ ,_LOCATION_RANGE_
362 #  define _PREARG_LOCATION_ _LOCATION_,
363 #  define _PREARG_LOCATION_DIMENSIONS_ _LOCATION_DIMENSIONS_,
364 #  define _DECLARE_ARGUMENTS_LOCATION_ integer,intent(in) :: _LOCATION_
365 #  define _DECLARE_ARGUMENTS_LOCATION_RANGE_ integer,intent(in) :: _LOCATION_RANGE_
366 #  define _DECLARE_LOCATION_ integer :: _LOCATION_
367 #else
368 #  define _INDEX_LOCATION_
369 #  define _DIMENSION_GLOBAL_
370 #  define _ATTRIBUTES_GLOBAL_
371 #  define _POSTARG_LOCATION_
372 #  define _POSTARG_LOCATION_RANGE_
373 #  define _PREARG_LOCATION_
374 #  define _PREARG_LOCATION_DIMENSIONS_
375 #  define _DECLARE_ARGUMENTS_LOCATION_
376 #  define _DECLARE_ARGUMENTS_LOCATION_RANGE_
377 #  define _DECLARE_LOCATION_
378 #endif
379
380 #define _DIMENSION_GLOBAL_PLUS_1_ ,dimension(_PREARG_LOCATION_DIMENSIONS_ :)
381 #define _DIMENSION_GLOBAL_HORIZONTAL_PLUS_1_ ,dimension(_PREARG_HORIZONTAL_LOCATION_DIMENSIONS_ :)
382
383 #ifdef _GLOBAL_INTERIOR_
384 !  Interior is vectorized; forward provided iterator
385 #  define _INDEX_GLOBAL_INTERIOR_(it) (_GLOBAL_INTERIOR_(it))
386 #  define _INDEX_GLOBAL_INTERIOR_PLUS_1_(it,j) (_GLOBAL_INTERIOR_(it),j)
387 #else
388 !  Interior is not vectorized; just index to local point in space.
389 #  define _INDEX_GLOBAL_INTERIOR_(it) _INDEX_LOCATION_
```

```
390 #  define _INDEX_GLOBAL_INTERIOR_PLUS_1_(it,j) (_PREARG_LOCATION_ j)
391 #endif
392
393 #ifdef _GLOBAL_HORIZONTAL_
394 !  Interior is vectorized; forward provided iterator
395 #  define _INDEX_GLOBAL_HORIZONTAL_(it) (_GLOBAL_HORIZONTAL_(it))
396 #  define _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(it,j) (_GLOBAL_HORIZONTAL_(it),j)
397 #else
398 !  Interior is not vectorized; just index to local point in space.
399 #  define _INDEX_GLOBAL_HORIZONTAL_(it) _INDEX_HORIZONTAL_LOCATION_
400 #  define _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(it,j) (_PREARG_HORIZONTAL_LOCATION_ j)
401 #endif
402
403 #ifdef _GLOBAL_VERTICAL_
404 !  Interior is vectorized; forward provided iterator
405 #  define _INDEX_GLOBAL_VERTICAL_(it) (_GLOBAL_VERTICAL_(it))
406 #  define _INDEX_GLOBAL_VERTICAL_PLUS_1_(it,j) (_GLOBAL_VERTICAL_(it),j)
407 #else
408 !  Interior is not vectorized; just index to local point in space.
409 #  define _INDEX_GLOBAL_VERTICAL_(it) _INDEX_LOCATION_
410 #  define _INDEX_GLOBAL_VERTICAL_PLUS_1_(it,j) (_PREARG_LOCATION_ j)
411 #endif
412
413 #ifdef _FABM_VECTORIZED_DIMENSION_INDEX_
414 !  -----------------------------------------------------------------------------
415 !  INTERIOR procedures operate on a data slice over one spatial dimension.
416 !  -----------------------------------------------------------------------------
417 #  if _FABM_DIMENSION_COUNT_>1
418 #    define _ARG_INTERIOR_FIXED_LOCATION_ ,_INTERIOR_FIXED_LOCATION_
419 #  else
420 #    define _ARG_INTERIOR_FIXED_LOCATION_
421 #  endif
422 #  define _ARG_INTERIOR_IN_ _START_,_STOP_ _ARG_INTERIOR_FIXED_LOCATION_
423 #  define _POSTARG_INTERIOR_IN_ ,_ARG_INTERIOR_IN_
424 #  define _PREARG_INTERIOR_IN_ _ARG_INTERIOR_IN_,
425 #  define _DECLARE_ARGUMENTS_INTERIOR_IN_ integer,intent(in) :: _START_,_STOP_ _ARG_INTERIOR_FIXED_LOCATION_
426 #else
427 !  -----------------------------------------------------------------------------
428 !  INTERIOR procedures operate on one point at a time.
429 !  -----------------------------------------------------------------------------
430 #  define _ARG_INTERIOR_IN_ _LOCATION_
431 #  define _POSTARG_INTERIOR_IN_ _POSTARG_LOCATION_
432 #  define _PREARG_INTERIOR_IN_ _PREARG_LOCATION_
433 #  define _DECLARE_ARGUMENTS_INTERIOR_IN_ _DECLARE_ARGUMENTS_LOCATION_
434 #endif
435
436 #ifdef _HORIZONTAL_IS_VECTORIZED_
437 !  -----------------------------------------------------------------------------
438 !  HORIZONTAL procedures operate on a data slice over one spatial dimension.
439 !  This will be the same dimension that INTERIOR procedures operate upon.
440 !  -----------------------------------------------------------------------------
441 #  if (_FABM_DIMENSION_COUNT_>2¦¦(_FABM_DIMENSION_COUNT_==2&&!defined(_FABM_DEPTH_DIMENSION_INDEX_)))
442 #    define _ARG_HORIZONTAL_FIXED_LOCATION_ ,_HORIZONTAL_FIXED_LOCATION_
443 #  else
444 #    define _ARG_HORIZONTAL_FIXED_LOCATION_
445 #  endif
446 #  define _ARG_HORIZONTAL_IN_ _START_,_STOP_ _ARG_HORIZONTAL_FIXED_LOCATION_
447 #  define _POSTARG_HORIZONTAL_IN_ ,_ARG_HORIZONTAL_IN_
448 #  define _PREARG_HORIZONTAL_IN_ _ARG_HORIZONTAL_IN_,
449 #  define _DECLARE_ARGUMENTS_HORIZONTAL_IN_ integer,intent(in) :: _START_,_STOP_ _ARG_HORIZONTAL_FIXED_LOCATION_
450 #else
451 !  -----------------------------------------------------------------------------
452 !  HORIZONTAL procedures operate on one point at a time.
453 !  -----------------------------------------------------------------------------
454 #  define _ARG_HORIZONTAL_IN_ _ARG_HORIZONTAL_LOCATION_
455 #  define _POSTARG_HORIZONTAL_IN_ _POSTARG_HORIZONTAL_LOCATION_
456 #  define _PREARG_HORIZONTAL_IN_ _PREARG_HORIZONTAL_LOCATION_
457 #  define _DECLARE_ARGUMENTS_HORIZONTAL_IN_ _DECLARE_ARGUMENTS_HORIZONTAL_LOCATION_
458 #endif
459
460 #ifdef _FABM_DEPTH_DIMENSION_INDEX_
461 !  -----------------------------------------------------------------------------
462 !  VERTICAL procedures operate on a data slice over one spatial dimension.
463 !  -----------------------------------------------------------------------------
464 #  if _FABM_DEPTH_DIMENSION_INDEX_==1
465 #    define _VERTICAL_ITERATOR_ i__
466 #    define _VERTICAL_START_ istart__
467 #    define _VERTICAL_STOP_ istop__
468 #  elif _FABM_DEPTH_DIMENSION_INDEX_==2
469 #    define _VERTICAL_ITERATOR_ j__
470 #    define _VERTICAL_START_ jstart__
471 #    define _VERTICAL_STOP_ jstop__
472 #  else
473 #    define _VERTICAL_ITERATOR_ k__
474 #    define _VERTICAL_START_ kstart__
475 #    define _VERTICAL_STOP_ kstop__
476 #  endif
477 #  if _FABM_DIMENSION_COUNT_==1
478 #    define _ARG_VERTICAL_FIXED_LOCATION_
479 #  else
480 #    define _ARG_VERTICAL_FIXED_LOCATION_ ,_HORIZONTAL_LOCATION_
481 #  endif
482 #  define _ARG_VERTICAL_IN_ _VERTICAL_START_,_VERTICAL_STOP_ _ARG_VERTICAL_FIXED_LOCATION_
483 #  define _POSTARG_VERTICAL_IN_ ,_ARG_VERTICAL_IN_
484 #  define _PREARG_VERTICAL_IN_ _ARG_VERTICAL_IN_,
485 #  define _DECLARE_ARGUMENTS_VERTICAL_IN_ integer,intent(in) :: _VERTICAL_START_,_VERTICAL_STOP_ _ARG_VERTICAL_FIXED_L
    OCATION_
486 #else
```

```
!  ------------------------------------------------------------------------------
!  VERTICAL procedures operate on one point at a time.
!  ------------------------------------------------------------------------------
#  define _ARG_VERTICAL_IN_ _LOCATION_
#  define _POSTARG_VERTICAL_IN_ _POSTARG_LOCATION_
#  define _PREARG_VERTICAL_IN_ _PREARG_LOCATION_
#  define _DECLARE_ARGUMENTS_VERTICAL_IN_ _DECLARE_ARGUMENTS_LOCATION_
#endif

#ifdef _HAS_MASK_
#  define _PACK_GLOBAL_(in,out,i,cache) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_SLICE_PLUS_1_(i) = in _INDEX_GLOBAL_INTERIOR_(cache%ipack(_I_));_LOOP_END_
#  define _PACK_GLOBAL_PLUS_1_(in,i,out,j,cache) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_INTERIOR_PLUS_1_(cache%ipack(_I_),i);_LOOP_END_
#  define _UNPACK_(in,i,out,cache,missing) _DO_CONCURRENT_(_I_,_START_,_STOP_);out(_I_) = in(cache%iunpack(_I_),i);end do
#  define _UNPACK_TO_PLUS_1_(in,i,out,j,cache,missing) _DO_CONCURRENT_(_I_,_START_,_STOP_);out(_I_,j) = in(cache%iunpack(_I_),i);end do
#  define _UNPACK_AND_ADD_TO_PLUS_1_(in,i,out,j,cache) _DO_CONCURRENT_(_I_,_START_,_STOP_);out(_I_,j) = out(_I_,j) + in(cache%iunpack(_I_),i);end do
#  define _UNPACK_TO_GLOBAL_(in,i,out,cache,missing) _DO_CONCURRENT_(_I_,_START_,_STOP_);out _INDEX_GLOBAL_INTERIOR_(_I_) = in(cache%iunpack(_I_),i);end do
#  define _UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _DO_CONCURRENT_(_I_,_START_,_STOP_);out _INDEX_GLOBAL_INTERIOR_PLUS_1_(_I_,j) = in(cache%iunpack(_I_),i);end do
#else
#  define _PACK_GLOBAL_(in,out,i,cache) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_SLICE_PLUS_1_(i) = in _INDEX_GLOBAL_INTERIOR_(_START_+_I_-1);_LOOP_END_
#  define _PACK_GLOBAL_PLUS_1_(in,i,out,j,cache) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_INTERIOR_PLUS_1_(_START_+_I_-1,i);_LOOP_END_
#  define _UNPACK_(in,i,out,cache,missing) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_SLICE_ = in _INDEX_SLICE_PLUS_1_(i);_LOOP_END_
#  define _UNPACK_TO_PLUS_1_(in,i,out,j,cache,missing) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_SLICE_PLUS_1_(j) = in _INDEX_SLICE_PLUS_1_(i);_LOOP_END_
#  define _UNPACK_AND_ADD_TO_PLUS_1_(in,i,out,j,cache) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_SLICE_PLUS_1_(j) = out _INDEX_EXT_SLICE_PLUS_1_(j) + in _INDEX_SLICE_PLUS_1_(i);_LOOP_END_
#  define _UNPACK_TO_GLOBAL_(in,i,out,cache,missing) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_GLOBAL_INTERIOR_(_START_+_I_-1) = in _INDEX_SLICE_PLUS_1_(i);_LOOP_END_
#  define _UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _CONCURRENT_LOOP_BEGIN_EX_(cache);out _INDEX_GLOBAL_INTERIOR_PLUS_1_(_START_+_I_-1,j) = in _INDEX_SLICE_PLUS_1_(i);_LOOP_END_
#endif

#if defined(_HORIZONTAL_IS_VECTORIZED_)&&defined(_HAS_MASK_)
#  define _HORIZONTAL_PACK_GLOBAL_(in,out,j,cache) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_HORIZONTAL_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_HORIZONTAL_(cache%ipack(_J_));_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_PACK_GLOBAL_PLUS_1_(in,i,out,j,cache) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_HORIZONTAL_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(cache%ipack(_J_),i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_(in,i,out,cache,missing) _DO_CONCURRENT_(_J_,_START_,_STOP_);out(_J_) = in(cache%iunpack(_J_),i);end do
#  define _HORIZONTAL_UNPACK_TO_PLUS_1_(in,i,out,j,cache,missing) _DO_CONCURRENT_(_J_,_START_,_STOP_);out(_J_,j) = in(cache%iunpack(_J_),i);end do
#  define _HORIZONTAL_UNPACK_AND_ADD_TO_PLUS_1_(in,i,out,j,cache) _DO_CONCURRENT_(_J_,_START_,_STOP_);out(_J_,j) = out(_J_,j) + in(cache%iunpack(_J_),i);end do
#  define _HORIZONTAL_UNPACK_TO_GLOBAL_(in,i,out,cache,missing) _DO_CONCURRENT_(_J_,_START_,_STOP_);out _INDEX_GLOBAL_HORIZONTAL_(_J_) = in(cache%iunpack(_J_),i);end do
#  define _HORIZONTAL_UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _DO_CONCURRENT_(_J_,_START_,_STOP_);out _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(_J_,j) = in(cache%iunpack(_J_),i);end do
#else
#  define _HORIZONTAL_PACK_GLOBAL_(in,out,j,cache) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_HORIZONTAL_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_HORIZONTAL_(_START_+_J_-1);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_PACK_GLOBAL_PLUS_1_(in,i,out,j,cache) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_HORIZONTAL_SLICE_PLUS_1_(j) = in _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(_START_+_J_-1,i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_(in,i,out,cache,missing) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_HORIZONTAL_SLICE_ = in _INDEX_HORIZONTAL_SLICE_PLUS_1_(i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_TO_PLUS_1_(in,i,out,j,cache,missing) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_HORIZONTAL_SLICE_PLUS_1_(j) = in _INDEX_HORIZONTAL_SLICE_PLUS_1_(i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_AND_ADD_TO_PLUS_1_(in,i,out,j,cache) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_EXT_HORIZONTAL_SLICE_PLUS_1_(j) = out _INDEX_EXT_HORIZONTAL_SLICE_PLUS_1_(j) + in _INDEX_HORIZONTAL_SLICE_PLUS_1_(i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_TO_GLOBAL_(in,i,out,cache,missing) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_GLOBAL_HORIZONTAL_(_START_+_J_-1) = in _INDEX_HORIZONTAL_SLICE_PLUS_1_(i);_HORIZONTAL_LOOP_END_
#  define _HORIZONTAL_UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _CONCURRENT_HORIZONTAL_LOOP_BEGIN_EX_(cache);out _INDEX_GLOBAL_HORIZONTAL_PLUS_1_(_START_+_J_-1,j) = in _INDEX_HORIZONTAL_SLICE_PLUS_1_(i);_HORIZONTAL_LOOP_END_
#endif

#if defined(_FABM_DEPTH_DIMENSION_INDEX_)&&defined(_HAS_MASK_)
#  define _VERTICAL_UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _DO_CONCURRENT_(_I_,_VERTICAL_START_,_VERTICAL_STOP_);out _INDEX_GLOBAL_VERTICAL_PLUS_1_(_I_,j) = in(cache%iunpack(_I_),i);end do
#else
#  define _VERTICAL_UNPACK_TO_GLOBAL_PLUS_1_(in,i,out,j,cache,missing) _CONCURRENT_VERTICAL_LOOP_BEGIN_EX_(cache);out _INDEX_GLOBAL_VERTICAL_PLUS_1_(_VERTICAL_START_+_I_-1,j) = in _INDEX_SLICE_PLUS_1_(i);_VERTICAL_LOOP_END_
#endif
```