the calling process with the above identifiers, which then can be passed to MPI_SENDRECV. The user specifies the coordinate direction and the size of the step (positive or negative). The function is local.

MPI_CART_SHIFT(comm, direction, disp, rank_source, rank_dest)

IN	comm	communicator with Cartesian structure (handle)
IN	direction	coordinate dimension of shift (integer)
IN	disp	displacement (> 0: upwards shift, < 0: downwards shift) (integer)
OUT	rank_source	rank of source process (integer)
OUT	rank_dest	rank of destination process (integer)

```
int MPI_Cart_shift(MPI_Comm comm, int direction, int disp,
              int *rank_source, int *rank_dest)
```

```
MPI_CART_SHIFT(COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR)
    INTEGER COMM, DIRECTION, DISP, RANK_SOURCE, RANK_DEST, IERROR
```

ticket150.

```
{void MPI::Cartcomm::Shift(int direction, int disp, int& rank_source,
              int& rank_dest) const (binding deprecated, see Section 15.2) }
```

ticket150.

ticket41.

[The direction argument indicates the dimension of the shift, i.e., the coordinate which value is modified by the shift. The coordinates are numbered from 0 to ndims-1, when ndims is the number of dimensions.] The direction argument indicates the coordinate dimension to be traversed by the shift. The dimensions are numbered from 0 to ndims-1, where ndims is the number of dimensions.

ticket41.

Depending on the periodicity of the Cartesian group in the specified coordinate direction, MPI_CART_SHIFT provides the identifiers for a circular or an end-off shift. In the case of an end-off shift, the value MPI_PROC_NULL may be returned in rank_source or rank_dest, indicating that the source or the destination for the shift is out of range.

It is erroneous to call MPI_CART_SHIFT with a direction that is either negative or greater than or equal to the number of dimensions in the Cartesian communicator. This implies that it is erroneous to call MPI_CART_SHIFT with a comm that is associated with a zero-dimensional Cartesian topology.

Example 7.5 The communicator, comm, has a two-dimensional, periodic, Cartesian topology associated with it. A two-dimensional array of REALs is stored one element per process, in variable A. One wishes to skew this array, by shifting column i (vertically, i.e., along the column) by i steps. [

ticket122.

```
%....
%C find process rank
%      CALL MPI_COMM_RANK(comm, rank, ierr))
%C find Cartesian coordinates
%      CALL MPI_CART_COORDS(comm, rank, maxdims, coords, ierr)
%C compute shift source and destination
%      CALL MPI_CART_SHIFT(comm, 0, coords(2), source, dest, ierr)
```

```
%C skew array
%      CALL MPI_SENDRECV_REPLACE(A, 1, MPI_REAL, dest, 0, source, 0, comm,
%     +                          status, ierr)
%
]                                                                    ticket122.
....
C find process rank
      CALL MPI_COMM_RANK(comm, rank, ierr)
C find Cartesian coordinates
      CALL MPI_CART_COORDS(comm, rank, maxdims, coords, ierr)
C compute shift source and destination
      CALL MPI_CART_SHIFT(comm, 0, coords(2), source, dest, ierr)
C skew array
      CALL MPI_SENDRECV_REPLACE(A, 1, MPI_REAL, dest, 0, source, 0, comm,
     +                          status, ierr)
```

Advice to users. In Fortran, the dimension indicated by DIRECTION = i has DIMS(i+1) nodes, where DIMS is the array that was used to create the grid. In C, the dimension indicated by direction = i is the dimension specified by dims[i]. (*End of advice to users.*)

7.5.6 Partitioning of Cartesian structures

MPI_CART_SUB(comm, remain_dims, newcomm)

IN	comm	communicator with Cartesian structure (handle)
IN	remain_dims	the i-th entry of remain_dims specifies whether the i-th dimension is kept in the subgrid (`true`) or is dropped (`false`) (logical vector)
OUT	newcomm	communicator containing the subgrid that includes the calling process (handle)

```
int MPI_Cart_sub(MPI_Comm comm, int *remain_dims, MPI_Comm *newcomm)
```

```
MPI_CART_SUB(COMM, REMAIN_DIMS, NEWCOMM, IERROR)
    INTEGER COMM, NEWCOMM, IERROR
    LOGICAL REMAIN_DIMS(*)
```

ticket150.

{`MPI::Cartcomm MPI::Cartcomm::Sub(const bool remain_dims[]) const` *(binding deprecated, see Section 15.2)* } ticket150.

If a Cartesian topology has been created with MPI_CART_CREATE, the function MPI_CART_SUB can be used to partition the communicator group into subgroups that form lower-dimensional Cartesian subgrids, and to build for each subgroup a communicator with the associated subgrid Cartesian topology. If all entries in remain_dims are false or comm is already associated with a zero-dimensional Cartesian topology then newcomm is associated with a zero-dimensional Cartesian topology. (This function is closely related to MPI_COMM_SPLIT.)