Appendix E

Glossary

active
: Of elements and positions: Participating in parallel operations. Parallel operations within a **`where`** statement are carried out only on parallel variable elements left active by the **`where`** statement.

axis
: A dimension of a shape. Axes are numbered starting with 0 and are read from left to right in a left index. For example, if a shape is declared as "**`[256][512]ShapeA`**", shape **`ShapeA`** has 256 positions along axis 0 and 512 positions along axis 1.

`bool`
: An unsigned single-bit integer data type.

collision
: An attempt by more than one parallel variable element to send values to or get a value from the same element at the same time. C* provides mechanisms for avoiding collisions.

combiner type
: In communication functions: The type of operation to be carried out by the function—for example, add values, multiply them, or perform a bitwise logical AND.

context
: The active positions of a shape as set by a **`where`** statement.

coordinate A number that identifies a position or an element along an axis. For example, the coordinates of parallel variable element **`[6][14]p1`** are 6 for axis 0 and 14 for axis 1.

corresponding elements
Elements of different parallel variables that are at the same position. Corresponding elements have the same coordinates and the same shape.

current shape The shape on whose parallel variables parallel operations can be performed. The **`with`** statement selects the current shape.

`current` *predeclared shape name*
A shape name that C* equates to the current shape. Variables declared to be of shape **`current`** (for example, in a function) are of the shape that is current when the declaration is made.

direction In communication functions: The direction along an axis in which a function is to perform its operation. An upward direction is from lower-numbered coordinates to higher; a downward direction is from higher-numbered coordinates to lower.

element An individual data point of a parallel variable. A parallel variable has one element at each position in its shape.

exclusive operation
In communication functions: An operation that excludes the first position of a segment-bit scan set, and that includes the first position of a start-bit scan set in the operation for the preceding scan set. Compare *inclusive operation*.

general communication
Communication in which any parallel variable element can send a value to or get a value from any other element, whether or not their positions are in the same shape. Compare *grid communication*.

get operation An operation in which a parallel variable gets values from another parallel variable. For example: "**`dest = [index]source;`**".

grid communication
Communication in which a parallel variable sends values to or gets values from another parallel variable in the same shape, using the coordinates of the parallel variable's elements. Compare *general communication*.

hyperplane In communication functions: A set of positions whose coordinates are allowed to differ along more than one axis. Compare *scan class*.

inactive Of elements and positions: Not participating in parallel operations.

inclusive operation
In communication functions: An operation that includes the first position of the scan set. Compare *exclusive operation*.

intrinsic function A function that is defined as part of the language.

left indexing A method of specifying an element or elements of a parallel variable, or the dimension(s) of a shape, using values in brackets to the left of the variable or shape's name.

multicoordinate A value obtained by the **`make_multi_coordinate`** function that specifies which element of a parallel variable is to be spread through each hyperplane for the **`copy_multispread`** function.

notify bit In the **`send`** function: a **`bool`**-sized parallel variable, each element of which can be set when the corresponding element of the destination parallel variable receives a value.

parallel operation An operation carried out on more than one element of a parallel variable at the same time.

parallel variable A variable consisting of multiple data points, called *elements*, arranged in a specified shape. The declaration "**`int:ShapeA p1;`**" declares **`p1`** to be an **`int`**-length parallel variable of shape **`ShapeA`**. Compare *scalar variable*.

`pcoord` *function* An intrinsic function that returns a parallel variable whose elements are initialized to their coordinates along a specified axis.

`physical` *shape* A shape predeclared by C*. It is 1-dimensional, with the number of positions equal to the number of physical processors allocated to the program at run time.

position An area of a shape that can contain parallel variable elements. A shape declared as **`[8192]ShapeB`** contains 8192 positions, arranged along one dimension. A parallel variable of a given shape has an element in each position of that shape.

predeclared shape name
A shape name provided as part of the language. The three predeclared shape names are **`current`**, **`physical`**, and **`void`**.

promotion Changing a scalar variable into a parallel variable by replicating the value of the scalar variable in each position of the shape.

rank The number of dimensions of a shape. A shape declared as **`[512][256]ShapeA`** has rank 2. A shape can have up to 31 dimensions.

reduction operator
An operator that reduces a parallel variable to a single scalar value by performing a combining operation. For example, the reduction operator **`+=`** adds the values of active elements of a parallel variable.

sbit — In communication functions: A **bool**-sized parallel variable. An element of an sbit, when set to 1, marks the beginning of a scan set at the element's position. An sbit can be interpreted as a *segment bit* or as a *start bit*, depending on the value of the **smode** argument to the function.

scalar variable — A Standard C variable, having only one value. Compare *parallel variable*.

scan class — In communication functions: A set of positions whose coordinates differ only along a specified axis. Compare *hyperplane*, *scan set*.

scan set — In communication functions: A subset of a scan class, the beginning of which is marked by an sbit.

segment bit — In communication functions: The interpretation of an sbit when the value of the **smode** argument is **CM_segment_bit**. When an sbit is a segment bit: 1) the sbit starts a scan set when the value of its element is 1, whether or not it is in an active position; 2) scan sets are not affected by the direction of the operation; and 3) operations in one scan set never affect values of elements in another scan set. Compare *start bit*.

send address — An address that, along with a position's coordinates, uniquely identifies that position among all positions in all shapes.

send operation — An operation in which a parallel variable element sends a value to another element. For example: "**[index]dest = source;**".

shape — A template for parallel data. A shape is declared in a **shape** statement and consists of a number of positions organized in up to 31 dimensions. All parallel variables must have a shape, and no parallel operations can be carried out unless a shape is made current by a **with** statement.

shape-valued expression
An expression that can be resolved to a shape name, and can be used anywhere a shape name is used. For example, "**`shapeof(p1)`**" returns the name of the parallel variable **`p1`**'s shape and can be used in place of that shape's name.

start bit In communication functions: The interpretation of an sbit when the value of the **`smode`** argument is **`CM_start_bit`**. When an sbit is a start bit: 1) an sbit starts a scan set only when the value of its element is 1 and the element's position is active; 2) when the direction is downward, scan sets are created from the higher coordinate to the lower coordinate; and 3) in an exclusive operation, the position whose sbit element is 1 receives a value from the preceding scan set, if there is one. Compare *segment bit*.

torus A doughnut-shaped surface. C* "torus" communication functions use a grid as if it were wrapped into a torus, with the opposite borders of the grid connected. An element that requires a value from beyond the border gets it from the other side of the grid.

`void` *predeclared shape name*
An extension of the ANSI keyword **`void`**. It specifies a shape without indicating what the shape's name is. The **`void`** predeclared shape name can be used only as the target shape of a scalar-to-parallel pointer.

`where` *statement* A statement that sets the context for parallel operations within its body. For example, "**`where (p1 = 4)`**" causes parallel operations to be carried out only on elements in positions where the parallel variable **`p1`** is equal to 4.

`with` *statement* A statement that chooses the current shape. Parallel operations within the body of a **`with`** statement must (with some exceptions) be carried out on parallel variables of the current shape.

wrapping In communication functions: Obtaining values from the other side of the grid.