Appendix D

CM-5 C* Table Lookup Utility

CM-5 C* provides an efficient mechanism for parallel lookups into a single table. If you use this mechanism, C* replicates the table once per node or vector unit, rather than in each position of a shape.

To use the table lookup utility, include the file **<cstable.h>**.

The utility consists of four functions:

- Call **CMC_allocate_shared_table** to allocate the table on the nodes. It takes as its argument the size of the table (the total number of elements in the table times their size in bytes), and it returns a pointer to a parallel variable that indicates the table's location on the nodes. Its definition is:

  ```
  void:void *CMC_allocate_shared_table(size_t table_size);
  ```

  It is legal to use the pointer returned by this function only with the other table lookup functions.

- Call **CMC_initialize_shared_table** to put values into the table. Its definition is:

  ```
  void CMC_initialize_shared_table(void:void *table,
                           const void *values,
                           size_t table_size);
  ```

  where:

  table is the pointer to the table, returned by **CMC_allocate_shared_table**.

  values is a pointer to the scalar table values.

table_size is the size of the table in bytes. This is the same size specified to the **CMC_allocate_shared_table** function.

- Call **CMC_lookup_shared_table** to do a lookup in the table on the nodes. Its definition is:

```
void CMC_lookup_shared_table(void:current *result,
                             void:void *table,
                             int:current index,
                             size_t element_size);
```

where:

result is a pointer to a parallel variable of the current shape that holds the results of the lookup.

table is the parallel pointer to the table.

index is a parallel **int** of the current shape; its values are the indices into the table.

element_size is the size of each element in the table, in bytes.

- Call **CMC_free_shared_table** to deallocate the memory allocated on the nodes to the table. Its definition is:

```
void CMC_free_shared_table(void:void *table);
```

where **table** is the pointer returned by **CMC_allocate_shared_table**.

D.1 An Example

In this example, a table of 24 **int**s is allocated and initialized in a 16384-position shape on the nodes. Random numbers are used as the index into the table, and the results of some lookups are printed. Finally, the memory for the table is freed.

```
#include <stdio.h>
#include <stdlib.h>
#include <cstable.h>

int table_data[24] = {
    14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10,
    23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2
};
```

```
main()
{
    shape [16384]s;
    int:s index, result;
    void:void *table;
    int i;

    table = CMC_allocate_shared_table(sizeof(table_data));
    CMC_initialize_shared_table(table, table_data,
                sizeof(table_data));

    with(s)
    {
         index = prand() % 24;
         CMC_lookup_shared_table(&result, table, index,
                sizeof(result));
    }

    for(i = 0; i < 20; ++i)
    {
         printf("%d\n", [i]result);
    }

    CMC_free_shared_table(table);
}
```