Chapter 1

Introduction

The utility library provides convenient access from CM Fortran to the capabilities of lower-level CM software. The purpose is typically to achieve functionality or performance beyond what the compiler currently provides.

As the compiler continues to develop, some of the utility procedures become redundant with CM Fortran language features. Each chapter of Part I compares utility procedures with the corresponding language features in the current release and notes any differences in behavior or performance.

1.1 Why a Utility Library?

CM Fortran programmers can use utility library procedures in situations where one is tempted to make explicit calls to lower-level software. There are several advantages to using the utility library instead in these situations:

- *Convenience*. The utility procedures take CM Fortran array names and other CM Fortran data objects as arguments. There is no need to convert CM Fortran objects into the data types used by lower-level software.

- *CM Portability*. With the few exceptions noted, the utility procedures support all CM hardware configurations and execution models, regardless of the particular lower-level software involved. There is no need to recode these calls to port a program from one CM system to another, even though the underlying system software may be quite different.

Copyright © 1994 Thinking Machines Corporation

- *Support*. The utility library is a supported part of the CM Fortran product. Unlike the underlying system software, the library remains stable over time so that programs using it do not require maintenance.

1.2 Contents of the Utility Library

The chapters that follow describe the utility library procedures under these categories:

- inquiries
  - system inquiry
  - array inquiry
- random number generation
- dynamic array allocation
- array aliasing
  - alias construction
  - alias operations
  - alias descriptor inquiry
  - execution environment
- data motion (interprocessor)
  - array transfers
  - scatters with combining (plus array address construction)
  - parallel prefix operations
  - ranking and sorting
- data motion (local)
  - table look-ups
  - gathers/scatters on serial axes
- parallel I/O
  - CM file operations
  - CM I/O via devices or sockets
- initialization

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

1.3 The Utility Library Header Files

Each program unit that uses procedures from the utility library must include its header file:

```
INCLUDE 'CMF_defs.h'
```

CM **INCLUDE** files are also accessible by absolute pathname or by using the **-I** switch. Check with your site's system administrator for the files' location.

The compiler command **cmf** links with the utility library automatically; no explicit linking is required.

The include file **CMF_defs.h** contains external declarations for certain functions. Their purpose is to prevent users from inadvertently referencing these functions as variables — for example, assigning the function name, such as **n = CMF_NUMBER_OF_PROCESSORS**, instead of assigning its return value, **n = CMF_NUMBER_OF_PROCESSORS()**.

However, external declarations are not permitted in block data program units. Therefore, an alternative include file, **CMF_params.h**, exists, containing only the parameter definitions from **CMF_defs.h**. Use **CMF_params.h** instead of **CMF_defs.h** in block data program units.

```
INCLUDE 'CMF_params.h'
```

1.4 Restrictions on Utility Procedures

- *Aligned arrays*. The utility procedures do not operate on arrays that are aligned with other arrays of higher rank or aligned with nonzero dimension offset(s) with any other array.
- *Lower bounds*. The procedure arguments that index into arrays assume that the array dimensions have a lower bound of 1. Any other lower bound value is ignored in these cases. This restriction is noted under the procedures to which it applies, such as **CMF_DEPOSIT_GRID_COORDINATE** and **CMF_AREF_1D**.
- *The 64-bit integer type.* On CM-5 systems with vector units, you can use the **INTEGER*8** type for any CM array argument to the utility procedures. You cannot, however, use an **INTEGER*8** value for scalar

Copyright © 1994 Thinking Machines Corporation

arguments. For front-end array arguments, the **INTEGER*8** type is accepted but the extra length is not used.

A few additional restrictions apply only to particular procedures. These are noted both in the text discussing the functional categories and in the man pages for those procedures.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation