Chapter 3

Random Numbers

Three subroutines serve to fill a CM array with pseudo random numbers:

```
CALL CMF_RANDOMIZE( SEED )

CALL CMF_RANDOM( DEST, LIMIT )

CALL CMF_RANDOM_LONG_S_INTEGER( DEST, LIMIT )
```

CMF_RANDOMIZE sets a seed for the random number generator used by the other two subroutines. **CMF_RANDOM** and **CMF_RANDOM_LONG_S_INTEGER** use the initialized random number generator to store a pseudo random number in each element of the **DEST** array. For **CMF_RANDOM**, the **DEST** array can be of type **INTEGER*4**, **INTEGER*8**, **REAL**, **DOUBLE PRECISION**, **COMPLEX**, or **DOUBLE COMPLEX**. For **CMF_RANDOM_LONG_INTEGER**, the **DEST** array must be of type **INTEGER*8**.

LIMIT is of type **INTEGER*4** for **CMF_RANDOM** and **INTEGER*8** for **CMF_RANDOM_LONG_S_INTEGER**. The **LIMIT** argument should always be specified as 1.0 for floating-point values. For integers, the argument serves as the exclusive upper bound of the values generated. If you do not want to set a limit for integer values, specify the **LIMIT** argument as 0.

The random number generator algorithm used by these routines is Wolfram's Rule 30 Cellular Automaton, described in Stephen Wolfram, ''Random Sequence Generation by Cellular Automata,'' *Advances in Applied Mathematics* **7**, pp. 123 – 69 (1986). This paper may be more readily available as a reprint in Stephen Wolfram, *Theory and Application of Cellular Automata*, World Scientific (1986).

Copyright © 1994 Thinking Machines Corporation

The cellular automaton is run on a finite string of bits, i=0,...,$n-1$, with periodic boundary conditions (so that site n is equivalent to site 0). In the CM implementation $n = 59$.

- For integer values, **CMF_RANDOM** generates the random numbers by running the automaton for 32 generations. (If **DEST** is of type **INTEGER*8**, only 32 of the 64 bits are filled.) **CMF_RANDOM_LONG_S_INTEGER** runs the automaton for 64 generations.

- For real, double-precision real, complex, or double-precision complex values, the random numbers are generated by running the automaton for s generations (where s is the mantissa length), and setting the exponent bits and sign bit so that the values are uniformly distributed between 2.0 and 1.0. Then 1.0 is subtracted from the results to yield numbers that are uniformly distributed between 0.0 and 1.0 (including 0.0, but excluding 1.0).

NOTE

The performance of **CMF_RANDOM** and **CMF_RANDOM_LONG_S_INTEGER** is significantly improved if the program is linked with the CM Scientific Software Library, **-lcmssl**.

3.1 Language Comparison

The behavior and performance of the random-number subroutines are comparable to the intrinsic function **RANDOM_NUMBER**. Both are enhanced when the program is linked with the CMSSL library, **-lcmssl**.

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_RANDOMIZE

Initializes the random number generator with a seed.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_RANDOMIZE( SEED )
```

Arguments

SEED	An **INTEGER** scalar (*4 only) specifying the seed value with which to initialize the random number generator.

Result

None.

Description

This subroutine uses **SEED** to initialize the random number generator used when **CMF_RANDOM** or **CMF_RANDOM_LONG_S_INTEGER** is called.

The random number generator algorithm used by **CMF_RANDOM** and **CMF_RANDOM_LONG_S_INTEGER** is Wolfram's Rule 30 Cellular Automaton. For more information, see the descriptions of those subroutines.

Restriction

- The Wolfram random number generator cannot be seeded with −1 or 0.

Copyright © 1994 Thinking Machines Corporation

See Also

```
CMF_RANDOM
CMF_RANDOM_LONG_S_INTEGER
```


Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_RANDOM

Places a different pseudo random number in each element of an array **DEST**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_RANDOM( DEST, LIMIT )
```

Arguments

DEST	A CM array of one of the following types: INTEGER*4 INTEGER*8 REAL DOUBLE PRECISION COMPLEX DOUBLE COMPLEX
LIMIT	A scalar **INTEGER*4** specifying the exclusive upper bound for the range of random numbers generated. For floating-point values this number should be 1.0. For integers only, to specify no upper bound, **LIMIT** should be 0.

Result

None.

Description

This subroutine places a pseudo random number in each element of the **DEST** array.

Copyright © 1994 Thinking Machines Corporation

The random number generator algorithm used by this procedure is Wolfram's Rule 30 Cellular Automaton. For **INTEGER** values the random numbers are generated by simply running the automaton for 32 generations. (If **DEST** is of type **INTEGER*8**, only 32 of the 64 bits are filled.) For **REAL**, **DOUBLE PRECISION**, **COMPLEX**, and **DOUBLE COMPLEX** values, the random numbers are generated by running the automaton for **s** generations (where **s** is the mantissa length), and setting the exponent bits and sign bit so that the result is uniformly distributed between 2.0 and 1.0. Then 1.0 is subtracted from the result to yield numbers that are uniformly distributed between 0.0 and 1.0 (including 0.0, but excluding 1.0). This automaton is run on a finite string of bits, i=0,...,*n*-1, with periodic boundary conditions (so that site *n* is equivalent to site 0). In the CM implementation $n = 59$.

The primary reference for the Rule 30 Cellular Automaton is Stephen Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics* **7**, pp. 123−69 (1986). This paper may be more readily available as a reprint in Stephen Wolfram, *Theory and Application of Cellular Automata* (including selected papers 1983−1986), World Scientific (1986).

Note

The performance of this subroutine is significantly improved if the program is linked with the CM Scientific Software Library, **-lcmssl**.

Restrictions

- The CM Fortran utility library procedures do not operate on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.
- The **INTEGER*8** type is supported only on CM-5 systems with vector units.

See Also

CMF_RANDOMIZE
CMF_RANDOM_LONG_S_INTEGER

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation

CMF_RANDOM_LONG_S_INTEGER

Places a different pseudo random number in each element of a 64-bit integer array **DEST**.

Syntax

```
INCLUDE 'CMF_defs.h'

CALL CMF_RANDOM_LONG_S_INTEGER( DEST, LIMIT )
```

Arguments

DEST	A CM array of type **INTEGER*8**.
LIMIT	An **INTEGER*8** specifying the exclusive upper bound for the range of random numbers generated. To specify no upper bound, **LIMIT** should be 0.

Result

None.

Description

This subroutine places a pseudo random number in each element of the **DEST** array.

The random number generator algorithm used by this procedure is Wolfram's Rule 30 Cellular Automaton. The random numbers are generated by simply running the automaton for 64 generations. This automaton is run on a finite string of bits, i=0,...,*n*-1, with periodic boundary conditions (so that site *n* is equivalent to site 0). In the CM implementation *n*= 59.

The primary reference for the Rule 30 Cellular Automaton is Stephen Wolfram, "Random Sequence Generation by Cellular Automata," *Advances in Applied Mathematics* **7**, pp. 123−69 (1986). This paper may be more readily available as a reprint in Stephen Wolfram, *Theory and Application of Cellular Automata* (including selected papers 1983−1986), World Scientific (1986).

Copyright © 1994 Thinking Machines Corporation

Note

The performance of this subroutine is significantly improved if the program is linked with the CM Scientific Software Library, **`-lcmssl`**.

Restrictions

- The CM Fortran utility library procedures do not operate on arrays that have been aligned with other arrays of greater rank or with other arrays of the same rank but with offsets for the individual axes.
- The **`CMF_RANDOM_LONG_S_INTEGER`** procedure is supported only on CM-5 systems with vector units.

See Also

`CMF_RANDOM`
`CMF_RANDOMIZE`

Version 2.2, October 1994
Copyright © 1994 Thinking Machines Corporation