The
Connection Machine
System

CM-5 C* Release Notes

Version 7.1
May 1993

Thinking Machines Corporation
Cambridge, Massachusetts

First printing, May 1993

****

The information in this document is subject to change without notice and should not be construed as a commitment by Thinking Machines Corporation. Thinking Machines Corporation reserves the right to make changes to any products described herein to improve functioning or design. Although the information in this document has been reviewed and is believed to be reliable, Thinking Machines Corporation does not assume responsibility or liability for any errors that may appear in this document. Thinking Machines Corporation does not assume any liability arising from the application or use of any information or product described herein.

****

Thinking Machines Corporation
245 First Street
Cambridge, Massachusetts 02142-1264
(617) 234-1000

Customer Support

Thinking Machines Customer Support encourages customers to report errors in Connection Machine operation and to suggest improvements in our products.

When reporting an error, please provide as much information as possible to help us identify and correct the problem. A code example that failed to execute, a session transcript, the record of a backtrace, or other such information can greatly reduce the time it takes Thinking Machines to respond to the report.

If your site has an applications engineer or a local site coordinator, please contact that person directly for support. Otherwise, please contact Thinking Machines' home office customer support staff:

Internet
Electronic Mail: `customer-support@think.com`

uucp
Electronic Mail: `ames!think!customer-support`

U.S. Mail:
Thinking Machines Corporation
Customer Support
245 First Street
Cambridge, Massachusetts 02142-1264

Telephone: (617) 234-4000

CM-5 C* Version 7.1
Release Notes

1 About CM-5 C* Version 7.1

CM-5 C* Version 7.1 is a new release of the CM-5 C* compiler. CM-5 C* is an implementation of the C* language, as described in the *C* Programming Guide*. Version 7.1 works with CMOST Version 7.2 S2 or later. CMOST Version 7.2 Beta Patch 3 is required to remove some restrictions in support for CMFS calls on CM-5s with vector units.

The release notes are organized as follows:

- Section 2 lists changes from the Beta release of Version 7.1.
- Section 3 lists differences between CM-5 C* and CM-200 C*.
- Section 4 discusses issues in porting CM-200 C* programs to the CM-5.

To learn about restrictions in this release, see the on-line bug update report, which is by default in the file **/usr/doc/cstar-7.1.bugupdate**; if this file doesn't exist on your system, check with your system administrator.

2 Changes from the Beta Release

The final release of Version 7.1 adds the features discussed in this section to CM-5 C*.

2.1 Routines for Manipulating Pointers to Parallel Variables

A functional interface has been added that gives you access to the memory address and stride for a pointer to parallel variable, and lets you create pointers to parallel variables using this information. For a description of this interface, see Appendix C of the *C* Programming Guide*, May 1993 edition.

2.2 Interface for Calling C* Routines from CM Fortran

An interface is now available that lets you call C* routines from a CM Fortran program. For complete information and sample programs, see Chapter 2 of the *CM-5 C* User's Guide*.

2.3 Increased Performance

Several areas of the compiler have been made more efficient for the official release of Version 7.1.

2.4 CMFS_dlseek Supported as of CMOST 7.2 Beta 2

As of CMOST 7.2 Beta 2, CM-5 C* will support **`CMFS_dlseek`**. **`CMFS_dlseek`** is the same as **`CMFS_lseek`**, except that it takes a **`double`** as the argument specifying the number of bytes. A new C* CMFS library will be shipped with CMOST 7.2 Beta 2 to support this enhancement.

3 Differences from CM-200 C*

This section lists differences between CM-5 C* and C* for the CM-2 and CM-200 (referred to as *CM-200 C**). For further information on these differences, see the *C* Programming Guide*, May 1993 edition.

3.1 Restriction on Shape Sizes Removed

The CM-200 C* restrictions on shape extents are not present in CM-5 C*. The sizes of a shape's dimensions need not be powers of 2, and the total number of positions in the shape need not be a multiple of the number of physical processors that the C* program is using. The only restriction is that the size of each dimension must be greater than 0.

3.2 Different Size for Parallel bools

On the CM-5, parallel `bool`s occupy 1 byte of storage, not 1 bit, as on the CM-2 and CM-200. (This change is necessary because CM-5 memory is not bit-addressable.) The semantics of using `bool`s remain the same; you need not change an existing program to deal with the new size. Memory usage will go up on the CM-5, however. Also note that on the CM-5, **`boolsizeof`** gives a size in bytes, and is therefore exactly like **`sizeof`**. See Section 5.4 of the *C* Programming Guide* for more information.

3.3 Programs Can't Call Paris

CM-5 C* programs can't call Paris routines (because there is no Paris on the CM-5). CM-2-specific header files such as **`<cm/paris.h>`** are not available on the CM-5.

3.4 Improved Performance of Parallel Right Indexing

Parallel indexing into parallel arrays performs better in CM-5 C* than it does in CM-200 C*.

3.5 New *= and /= Reduction Operators

CM-5 C* implements the `*=` and `/=` parallel-to-scalar reduction operators.

As a binary reduction operator, `*=` multiplies the values of the active elements of the parallel RHS by the value of the scalar LHS and assigns it to the LHS. As a unary operator, it returns the product of the active elements of the parallel variable.

As a binary reduction operator, `/=` divides the value of the scalar LHS by the product of the parallel RHS and assigns the result to the scalar LHS. When it is used as a unary operator, it returns the reciprocal of the product of all active positions in the parallel variable.

3.6 ANSI Compliance

The CM-5 C* compiler is generally compliant with the ANSI standard. This means that the CM-5 C* compiler will reject some programs that previously compiled without error.

3.7 Parallel enums Are Supported

Unlike the CM-200 C* compiler, CM-5 C* supports parallel `enums`. See Section 5.6 of the *C* Programming Guide* for more information.

3.8 Limitations on Parallel Unions Removed

The limitations on parallel unions discussed on page 60 of the *C* Programming Guide*, Version 6.0.2, are removed in CM-5 C*. Note, however, that taking advantage of the removal of these limitations may make your program nonportable. See Section 5.5 of the *C* Programming Guide*, May 1993 edition.

3.9 New Versions of read_from_pvar and write_to_pvar

CM-5 C* overloads the communication functions `read_from_pvar` and `write_to_pvar` for parallel data of any length. Using these versions of `read_from_pvar` and `write_to_pvar` for aggregate data may make your program nonportable. See Section 14.4 of the *C* Programming Guide* for more information.

3.10 New allocated_detailed_shape Function

CM-5 C* has its own version of `allocate_detailed_shape`. For a description of it, see Appendix B of the *C* Programming Guide*.

4 Porting CM-200 C* Programs to the CM-5

Most CM-200 C* programs should port without difficulty to the CM-5. You must recompile and relink using the CM-5 C* compiler. This list summarizes the changes that you must make (when applicable) to ensure portability:

- Remove all Paris calls.
- Remove all calls to libraries not supported on the CM-5.

- Remove all include files not supported on the CM-5 (for example, `<cm/paris.h>`).

- If you express lengths in terms of bits in a function (for example, in the overloaded versions of the grid communication functions or the `get` or `send` function), rewrite the code to express the size with `boolsizeof` and the appropriate parallel type.

- Change calls to `allocate_detailed_shape` to use the new format.

- The CM-5 C* compiler disallows casts between scalar types and pointers to parallel variables. If you call `palloc()` in a CM-200 C* program without including `<stdlib.h>` (which properly declares its return type) and cast the result, the code won't compile on the CM-5. Thus, this code won't work:

  ```
  /* No included stdlib.h file */

  int:current *p = (int:current *)palloc(current,
                boolsizeof(int:current));
  ```

  Change it to this so that it will work in CM-5 C*:

  ```
  #include <stdlib.h>

  int:current *p = palloc(current,
                     boolsizeof(int:current));
  ```