Appendix A

User Interface Reference

This appendix provides reference material for the features of the CMAX Converter's user interface. These are:

- The **`cmax`** command
- The CMAX converter directives
- The CMAX library
- Some sample EMACS utilities

Copyright © 1994 Thinking Machines Corporation

A.1 The `cmax` Command

The **`cmax`** command creates and operates upon *packages*. A package is a specialized subdirectory containing the Fortran source files that the CMAX Converter treats as a complete program. Table 2 summarizes the command's actions and, when the action is translation of a package, its translation options.

The **`cmax`** command can be invoked in several modes (as detailed in Section A.1.1). In summary, the modes are:

- Information only

  `cmax` *information-operation*

- Package operation without translation

  `cmax` *packname package-operation*

- Package translation with or without other package operation

  `cmax` *packname* [*translation-option*s]
  `cmax` *packname* **`-T`** [*translation-options*]
  `cmax` *packname* **`-Add=`** *sourcefile-list* **`-T`** [*translation-options*]

  packname can be up to 14 characters, including any alphanumeric as well as underscore, hyphen, or period. The first character must be alphanumeric; case is ignored.

The following rules apply when specifying any **`cmax`** command option:

- Case is ignored.
- Any nonambiguous abbreviation is accepted. The abbreviations shown in Table 2 are recommended, since they are unlikely to conflict with the names of internal options and possible future options.
- Where an = sign appears after an option name, it is required syntax. For options that take a list of arguments (**`-Add`**=, **`-Del=`**, and **`-Define=`**), a space after = is optional. For options that take a single argument, no space after = is permitted.
- Binary switches are specified with or without a prepended **`no`**. The brackets shown in Table 2 are a documentation convention only.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

Table 2. `cmax` command options, recommended abbreviations, and default values, if any.

Option	Abbreviation	Default
Information Operations		
`-Help`	`-Help`	
`-PackagesList`	`-Pack`	
Package Operations		
`-AddFiles=` *sourcefile-list*	`-Add=`	
`-ContentsList`	`-Cont`	
`-DeleteFiles=` *sourcefile-list*	`-Del=`	
`-RemovePackage`	`-Rem`	
`-TranslatePackage`	`-T`	
Package Translation Options		
Input decisions		
`-[no]CMFortran`	`-[no]CMF`	`-noCMF`
`-EntryPoint=`*program-unit-name*	`-E=`	
`-Interface Routines=` *program-unit-name-list*	`-Interface=`	
`-LineWidth=`*number*	`-LineWidth`	`72`
`-[no]PermitArraySyntax`	`-[no]PermitArr`	`-noPermitArr`
`-[no]PermitAutomaticArrays`	`-[no]PermitAuto`	`-noPermitAuto`
`-StatementBufferSize=`*number*	`-StatementBuffer`	`6700`
Output decisions		
`-CharForContinuation=`*char*	`-Char=`	`&`
`-[no]LineMapping`	`-[no]LineMap`	`-LineMap`
`-[no]ListingFile[=`*filename*`]`	`-[no]List`	`-noList`
	`-List[=]`	*packname*`.lis`
`-OutputExtension=`*string*	`-OutputE=`	`fcm`
`-OutputFile=`*filename*	`-OutputF=`	
`-Verbose=`*number*	`-Verb=`	`1`
Conversion decisions		
`-DefineSymbols=` *name-list*	`-Define=`	
`-[no]Dependence`	`-[no]Dep`	`-Dep`
`-[no]PermitKeywordsCMF`	`-[no]PermitKey`	`-noPermitKey`
`-[no]Permutation`	`-[no]Perm`	`-noPerm`
`-[no]Push`	`-[no]Push`	`-noPush`
`-[no]RestructureCode`	`-[no]Restruct`	`-Restruct`
`-[no]UnknownRoutinesSafe`	`-[no]Unknown`	`-Unknown`
`-ShortVectorLength=`*number*	`-ShortV=`	`8`
`-ShortLoopLength=`*number*	`-ShortL=`	`8`
`-[no]Vectorize`	`-[no]Vec`	`-Vec`
`-[no]ZeroArrays`	`-[no]Zero`	`-noZero`

Copyright © 1994 Thinking Machines Corporation

A.1.1 Invoking the `cmax` Command

The general rules for invoking the **`cmax`** command are:

- Options that specify operations (the first two categories in Table 2) are mutually incompatible, *except* that **`-T`** (translate) can be used with **`-Add=`**.
- If a *packname* but no operation is specified on the command line, **`-T`** is assumed. Any translation option may appear.
- Translation options are incompatible with all operations *except* **`-T`** (which may be assumed) and the combination of **`-Add=`** and **`-T`**. If **`-Add=`** is specified along with any translation option, **`-T`** must appear.

From these rules derive the following **`cmax`** command modes:

- Getting information

  `%` **`cmax`** *information-operation*

  The *information-operation* may be:

  - display a help message: **`-Help`**
  - display a list of the packages in the current directory: **`-Pack`**

- Creating and manipulating packages (without translation)

  `%` **`cmax`** *packname package-operation*

  The *package-operation* may be:

  - create a package with specified source files: **`-Add=`** *sourcefile-list*
  - add files to an existing package: **`-Add=`** *sourcefile-list*
  - delete files from a package: **`-Del=`** *sourcefile-list*
  - display a list of the files in a package: **`-Cont`**
  - remove a package: **`-Rem`**

  sourcefile-list is a space-delimited list of filenames with extensions **`.f`**, **`.fcm`**, **`.F`**, or **`.FCM`**.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

Packages contain pointers to source files, not copies. Removing a package or deleting files from it removes pointers only; source files are not affected.

- Translating packages

  % **cmax** *packname* [*translation-option*s]
  % **cmax** *packname* **-T** [*translation-options*]
  % **cmax** *packname* **-Add=** *sourcefile-list* **-T** [*translation-options*]

  The *translation-options* may be any of the options listed under that category in Table 2 and described below.

A.1.2 cmax Translation Options

Input Decisions

The following options control certain decisions the CMAX Converter makes concerning acceptable input.

-[no]CMFortran Default: **-noCMFortran**

When enabled, accept input files with the filename extension **.fcm** or **.FCM**. In this case, CMAX parses all files in the package as CM Fortran files, regardless of their extension. When disabled, CMAX exits with an error when it encounters an **.fcm** source file. The positive form should be used with **-OutputExtension** or **-OutputFile**, since CMAX exits rather than overwriting an input file with output of the same name.

-EntryPoint= *program-unit-name*

Use the specified program unit as the program's entry point (root node).

-Interface Routines = *program-unit-name-list*

Convert the subtree(s) beginning at the specified program unit(s) in "firewalls" mode, which maintains data consistency between converted and unconverted program units. If any program unit in the subtree to be converted is also called elsewhere in the program, then create and convert a variant of that program unit. The option **-OutputExtension=** should be used to give distinct names to the output files of the two procedure variants.

Copyright © 1994 Thinking Machines Corporation

-LineWidth=*number* Default: **-LineWidth=72**

Accept source files with line width (in characters) up through the specified *number*. The *number* argument is an integer less than or equal to 255.

-[no]PermitArraySyntax Default: **-noPermitArraySyntax**

When enabled, accept whole arrays and array sections in expressions in **.f** source files.

-[no]PermitAutomaticArrays Default: **-noPermitAutomatic**

When enabled, accept automatic arrays in **.f** source files. An automatic array is an explicit-shape local array with a bound that depends on the values of one or more variables.

-StatementBufferSize=*number* Default: **-StatementBuffer=6700**

Accept source files with line length (in characters and including continuations) up through the specified *number*. There is no aribtrary limit on the value of the integer *number* argument.

Output Decisions

The following options control certain decisions the CMAX Converter makes concerning output from the translation process.

-CharForContinuation=*char* Default: **-CharForContinuation=&**

Specify the continuation character used in generated code.

-[no]LineMapping Default: **-LineMapping**

When enabled, generate files named *sourcefile* **.ttab** that contain translation tables. These **.ttab** files are used by the Prism development environment to relate the source code lines in the input and output files.

-[no]ListingFile Default: **-noListingFile**
-ListingFile[**=***filename*] Default: *packname* **.lis**

When enabled, generate a listing file for the translation operation. This file reports the homes of all arrays, the subprogram variants the converter has generated, and statement-level efficiency notes. The listing file is named *packname* **.lis** if no other name is supplied.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

-OutputExtension=*string* Default: **-OutputExtension=fcm**

Write converted output files to the present working directory with filenames constructed of the sourcefile names and the extension *string*. The *string* argument may contain periods, such as **cmaxed.fcm**.

-OutputFile=*filename*

Write the converted program to a single file *filename* in the present working directory.

-Verbose=*number* Default: **-Verbose=1**

Set the level of informational messages issued to **stdout** during program analysis and conversion. The levels are:

0	General startup messages only
1	0 + report of actions
2	1 + messages at start of "passes" over program
3	2 + message at start of transformation of each subprogram
4	3 + message at start of transformation of each **DO** loop

Conversion Decisions

The following options control certain decision rules that the CMAX Converter applies in the course of translating a package.

-DefineSymbols= *name-list*

Treat the specified *name* or *names* as defined symbols when preprocessing source files. Code that is conditional upon an undefined symbol is ignored during translation and does not appear in the output program.

-[no]Dependence Default: **-Dependence**

When enabled, asserts that **DO** constructs in the package may exhibit loop-carried dependences. (This option is the global scope variant of the CMAX directive **[NO]DEPENDENCE**.)

-[no]PermitKeywordsCMF Default: **-noPermitKeywordsCMF**

When enabled, accept CM Fortran reserved keywords in **.f** source files and treat them as intrinsic procedure names. If disabled, CMAX changes user-supplied names for variables and procedures when they conflict with CM Fortran keywords.

Copyright © 1994 Thinking Machines Corporation

-[no]Permutation Default: **-noPermutation**

When enabled, asserts that index arrays used in **DO** constructs to perform indirect addressing do not contain duplicate index values. (This option is the global scope variant of the CMAX directive **[NO]PERMUTATION**.)

-[no]Push Default: **-noPush**

Enables or disables "loop pushing" as a technique of making vectorization possible. Loop pushing refers to rewriting a **DO** construct that contains a subroutine call as a call to a subroutine that contains the **DO** construct. Loop pushing is not currently implemented for functions. (This option is the global scope variant of the CMAX directive **[NO]PUSH**.)

-[no]RestructureCode Default: **-RestructureCode**

Enables or disables the conversion of **IF/GOTO** constructions to block **IF/ENDIF** constructs where possible. Block **IF/ENDIF** constructs inside **DO** constructs may be translatable to masked array assignments (**WHERE** or **FORALL**).

-ShortLoopLength=*number* Default: **ShortLoopLength=8**

Do not vectorize **DO** constructs with explicit loop iteration counts (product of extents in loop nest) of less than *number*.

-ShortVectorLength=*number* Default: **ShortVectorLength=8**

Do not vectorize **DO** constructs on arrays whose size (product of dimension extents) is less than *number*.

-[no]UnknownRoutinesSafe Default: **-UnknownRoutinesSafe**

When enabled, asserts that unseen program units (those outside the package being translated) do not contain code that would constrain any array in the package to a front-end home.

-[no]Vectorize Default: **-Vectorize**

Enables or disables the translation of **DO** constructs into array operations. (This option is the global scope variant of the CMAX directive **[NO]VECTORIZE**.)

-[no]ZeroArrays Default: **-noZeroArrays**

When enabled, generate code that initializes local CM arrays to zero. Scalar variables, front-end arrays, and arrays in common are not affected.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

A.2 CMAX Converter Directives

CMAX Converter directives are specialized code comments that control certain translation actions or decision rules. A converter directive may:

- Direct the converter not to attempt to vectorize a loop or loops
- Direct the converter not to attempt "loop pushing" on a loop or loops
- Assert that a particular loop or loops do not exhibit loop-carried data dependences
- Assert that the index array(s) used in a loop or loops to perform indirect addressing do not contain duplicate index values

The scope of a converter directive can be either a particular **DO** construct or a program unit. Comparable **cmax** command options apply the directives' actions or assertions globally throughout the package during program translation. Typically, in-line directives are used to override the default command action for particular loops or for particular program units.

A.2.1 Directive Syntax

A directive is a comment line of the form

CMAX$*directive-name* [*scope-spec*]

directive-name is one of:

[NO]VECTORIZE	enable/disable vectorization
[NO]PUSH	enable/disable loop pushing
[NO]DEPENDENCE	assert/deny possibility of data dependence
[NO]PERMUTATION	assert/deny non-repetition of index values

scope-spec is one of:

L [the default]	the subsequent loop (but not loops inside it)
R	all subsequent loops in the program unit

The directive prefix **CMAX$** must begin in column one, and one or more spaces must separate the directive name from the scope specifier (if supplied).

Copyright © 1994 Thinking Machines Corporation

A.2.2 Synopsis of Directives

This section summarizes the forms and the behavior of the CMAX Converter directives and the corresponding command options.

Loop Vectorization

The positive form enables conversion of loop(s) to array operations.

CMAX$VECTORIZE	applies to subsequent loop
CMAX$VECTORIZE L	applies to subsequent loop
CMAX$VECTORIZE R	applies to all subsequent loops in routine
-Vectorize	applies to all loops in package

The negative form disables conversion of loop(s) to array operations.

CMAX$NOVECTORIZE	applies to subsequent loop
CMAX$NOVECTORIZE L	applies to subsequent loop
CMAX$NOVECTORIZE R	applies to all subsequent loops in routine
-noVectorize	applies to all loops in package

The default global setting is **-Vectorize**.

Loop Pushing

The positive form enables the "pushing" of loop(s) that contain subroutine calls. In this transformation, a variant of the subroutine is created whose body contains the loop that had formerly called it. The loop within the subroutine may then be vectorizable.

CMAX$PUSH	applies to subsequent loop
CMAX$PUSH L	applies to subsequent loop
CMAX$PUSH R	applies to all subsequent loops in routine
-Push	applies to all loops in package

The negative form disables the "pushing" of loop(s).

CMAX$NOPUSH	applies to subsequent loop
CMAX$NOPUSH L	applies to subsequent loop
CMAX$NOPUSH R	applies to all subsequent loops in routine
-noPush	applies to all loops in package

The default global setting is **-noPush.**

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

Loop-Carried Data Dependence

The positive form asserts that data dependence may exist in the loop(s). The converter analyzes such loops to determine whether dependence does exist and whether it inhibits vectorization.

CMAX$DEPENDENCE	applies to subsequent loop
CMAX$DEPENDENCE L	applies to subsequent loop
CMAX$DEPENDENCE R	applies to all subsequent loops in routine
-Dependence	applies to all loops in package

The negative form asserts that there is no data dependence in the loop(s). The converter does not perform dependence analysis on such loops.

CMAX$NODEPENDENCE	applies to subsequent loop
CMAX$NODEPENDENCE L	applies to subsequent loop
CMAX$NODEPENDENCE R	applies to all subsequent loops in routine
-noDependence	applies to all loops in package

The default global setting is **-Dependence**

Uniqueness of Values in an Index Array

The positive form asserts that array elements used as indices into another array on the left hand side of an assignment in the specified loop(s) are each unique. Repetition of an index value indicates multiple assignments of an array element and thus a particular type of loop-carried data dependence.

CMAX$PERMUTATION	applies to subsequent loop
CMAX$PERMUTATION L	applies to subsequent loop
CMAX$PERMUTATION R	applies to all subsequent loops in routine
-Permutation	applies to all loops in package

The negative form asserts that array index values may not be unique.

CMAX$NOPERMUTATION	applies to subsequent loop
CMAX$NOPERMUTATION L	applies to subsequent loop
CMAX$NOPERMUTATION R	applies to all subsequent loops in routine
-noPermutation	applies to all loops in package

The default global setting is **-noPermutation**

Copyright © 1994 Thinking Machines Corporation

A.3 The CMAX Library

The CMAX library provides canonical ways to express actions that have no straightforward expression in Fortran 77. These procedures are fully portable to any Fortran 77 platform. The CMAX Converter recognizes them and translates them into the corresponding CM Fortran feature.

The library presently provides utilities for expressing the intent of dynamic array allocation and circular shifting of array elements.

A.3.1 Using the CMAX Library

Header File

Place the library header file in the Fortran 77 program with an **INCLUDE** line:

```
INCLUDE '/usr/include/cm/cmax.h'
```

Linking Fortran 77 Programs

To compile and link a Fortran 77 program, the serial link library must be available on the system and specified on the link line as:

```
% f77 myprogram.f libdirectory/libcmax.a
```

A version of this library compiled for a Sun-4 computer is available in the standard CM *libdirectory:*

CM-5:	**/usr/lib/libcmax.a**
CM-2/200:	**/usr/local/lib/libcmax.a**

To compile the serial library for other systems, obtain the library source code from the CMAX on-line examples directory:

CM-5:	**/usr/examples/cmax/libcmax/libcmax.f** **libcmax.c**
CM-2/200:	**/usr/cm/examples/cmax/libcmax/libcmax.f** **libcmax.c**

See your site system adminstrator for locations if any of these files have been moved from their default locations.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

Linking CM Fortran Programs

You must link CMAX-generated **.fcm** files with a CMAX library if they contain:

- Converter output from the dynamic allocation utility, since CMAX generates its own dynamic allocation library routines rather than CM Fortran features
- Calls to the CM Fortran Utility Library procedure **CMF_SEND_***combiner*, since CMAX uses its own library routine **CMAX_GRID_TO_SEND_ADDR** to construct send addresses

NOTE: No explicit linking is needed for converted programs that use *only* the circular shift utility from the CMAX library.

Where explicit linking is needed, it is most convenient with CM Fortran Version 2.2 or later. Simply place **-lcmax** on the **cmf** command line, and the compiler driver will find the CMAX library that is appropriate for the target platform.

In any earlier version of CM Fortran, you must specify on the **cmf** or linker command line the appropriate link library for the target CM platform and execution model:

```
% cmf myprogram.fcm library
```

where *library* is one of:

libdirectory/**libcmax_cm2.a**
libdirectory/**libcmax_cm200.a**
libdirectory/**libcmax_cm5_sparc.a**
libdirectory/**libcmax_cm5_vu.a**
libdirectory/**libcmax_cm5_cmsim.a**

and *libdirectory* is either:

CM-5 systems:	**/usr/lib/**
CM-2/200 systems:	**/usr/local/lib/**

See your site system adminstrator for locations if any of these files have been moved from their default locations.

Copyright © 1994 Thinking Machines Corporation

A.3.2 The Dynamic Allocation Utility

This utility uses a common array **CMAX_MEMORY**. A call to the subroutine **CMAX_ALLOCATE_***rank* associates an index variable (a scalar or a front-end array element) with a pointer into that memory. Passing the argument **CMAX_MEMORY(INDEX_VAR)** to a subroutine or function makes the memory available as an array. The corresponding dummy argument in the procedure must match the array indicated by **CMAX_MEMORY(INDEX_VAR)** in shape and type.

The converter transforms these subroutines into CM Fortran dynamic array allocation and deallocation. The dynamic arrays may be CM arrays or front-end arrays, depending on how they are treated in the subprogram scope.

```
CMAX_ALLOCATE_1(INDEX_VAR, ELT_TYPE, N1)
CMAX_ALLOCATE_2(INDEX_VAR, ELT_TYPE, N1, N2)
CMAX_ALLOCATE_3(INDEX_VAR, ELT_TYPE, N1, ..., N3)
CMAX_ALLOCATE_4(INDEX_VAR, ELT_TYPE, N1, ..., N4)
CMAX_ALLOCATE_5(INDEX_VAR, ELT_TYPE, N1, ..., N5)
CMAX_ALLOCATE_6(INDEX_VAR, ELT_TYPE, N1, ..., N6)
CMAX_ALLOCATE_7(INDEX_VAR, ELT_TYPE, N1, ..., N7)

CMAX_DEALLOCATE(INDEX_VAR)
```

where,

INDEX_VAR An integer variable *or* an element of an integer front-end array

ELT_TYPE A predefined integer constant, one of **CMAX_LOGICAL**, **CMAX_INTEGER**, **CMAX_REAL**, **CMAX_DOUBLE**, **CMAX_COMPLEX**, or **CMAX_DOUBLE_COMPLEX**

N1 ..., N7 Integer extents for array dimensions; the number of extent arguments must correspond to the rank specified in the procedure name

NOTE: An index variable can be used for multiple arrays with different dimension extents, but they must all be of the same rank and layout. Also, all the arrays referenced by the elements of an array of index variables must have the same rank and layout.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

Link the output program with the appropriate CMAX library for the target CM system and execution model, as shown in Section A.3.1. The converter translates the subroutines listed above into the following:

```
CMAX_I_ALLOCATE_1
CMAX_I_ALLOCATE_2
CMAX_I_ALLOCATE_3
CMAX_I_ALLOCATE_4
CMAX_I_ALLOCATE_5
CMAX_I_ALLOCATE_6
CMAX_I_ALLOCATE_7
CMAX_I_DEALLOCATE
```

These routines are internal versions of the CMAX allocation/deallocation routines. In addition to the user-supplied arguments, CMAX inserts a bit-mask argument to the allocation routines, which specifies the layout of the dynamic array.

The user can opt to change these CMAX routines into CM Fortran dynamic allocation features. No special linking is then required.

Copyright © 1994 Thinking Machines Corporation

A.3.3 The Circular Shift Utility

These subroutines shift the elements on dimension **DIM** of the source array by **SHIFT** element positions and store the result in the destination array. Elements that shift off the end of a dimension wrap around to the other end of that dimension.

```
CMAX_CSHIFT_1(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1)
CMAX_CSHIFT_2(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, N2)
CMAX_CSHIFT_3(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, ..., N3)
CMAX_CSHIFT_4(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, ..., N4)
CMAX_CSHIFT_5(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, ..., N5)
CMAX_CSHIFT_6(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, ..., N6)
CMAX_CSHIFT_7(DEST, SOURCE, DIM, SHIFT, ELT_TYPE, N1, ..., N7)
```

where,

DEST	The destination array; must not overlap with the source array and must be of the same type and shape
SOURCE	The source array; must be of the specified type and shape
DIM	An integer between 1 and the rank of the array, indicating the dimension along which to shift elements
SHIFT	An integer indicating the distance (number of element positions) to shift
ELT_TYPE	A predefined integer constant, one of **CMAX_LOGICAL**, **CMAX_INTEGER**, **CMAX_REAL**, **CMAX_DOUBLE**, **CMAX_COMPLEX**, or **CMAX_DOUBLE_COMPLEX**
N1 ..., N7	Integer extents for array dimensions; the number of extent arguments must correspond to the rank specified in the procedure name

The converter transforms these subroutines into references to the CM Fortran intrinsic function **CSHIFT**. The converter uses argument keywords in the output code so that the order of the **DIM** and **SHIFT** arguments is compatible with any version of CM Fortran.

No special linking is required for a CMAX-generated program that performs circular shifts, since the CM Fortran intrinsic function is generated.

Version 2.0, May 1994
Copyright © 1994 Thinking Machines Corporation

A.4 Gnu EMACS Utilities for CMAX (Unsupported)

EMACS editors can be customized in a number of ways to assist with the conversion process. CMAX provides a sample implementation for Gnu Emacs of the utilities described in this section:

- **cmax-viewer** command
- **cmax-subprogram** command
- **cmax-buffe**r command

The implementation is in the file **cmax.el**. Customize this file to suit your needs and your local variant of EMACS. (It is not officially supported by Thinking Machines.)

The file is on-line in:

- CM-5 systems: **/usr/examples/cmax/emacs/cmax.el**
- CM-2/200 systems: **/usr/cm/examples/cmax/emacs/cmax.el**

See your system administrator for the location if this file has been moved.

A.4.1 The CMAX Viewer

The CMAX viewer is an EMACS mode lets you view and move through CMAX input and output files at the same time. Unlike Prism, which requires executable files, this utility loads ASCII source files. By allowing you to compare input and output source, along with efficiency notes, the utility helps you determine what CMAX has done and why.

To activate the CMAX viewer:

1. Translate a **.f** file using the **cmax** option **-LineMap** (the default).
2. Place the cursor within a buffer containing either the **.f** or **.fcm** file.
3. Invoke the EMACS command **M-x cmax-viewer**.

The utility finds the line-mapping (**.ttab**) file and both Fortran files associated with it. It displays the input and output file in two adjoining windows, allowing you to scroll or otherwise move around in both files in synchronized fashion. For programs translated with **-list**, invoke the EMACS command bound to **C-c v** to display efficiency notes for the variable or subprogram name at the cursor.

Copyright © 1994 Thinking Machines Corporation

Type **C-h m** for mode help, and you will see the following message.

CMAX Viewer Mode:

This mode can be entered by calling **cmax-viewer** interactively.

Turn on **cmax-viewer-mode**, a mode for looking at **.f** files and cmax output simultaneously, based on a **.ttab** file.

Moving the cursor in either the input source or output source window will move the cursor in both windows.

If the cursor is in the input source window, any efficiency notes for the pointed to line are displayed at the bottom of the screen. You must run CMAX with **−list** to generate a **.ttab** with efficiency notes.

You can use **C-C v** (below) to display efficiency notes for variables or subprograms when pointing to them in either window.

These special commands operate in both source windows in synchronized fashion:

C-n	Move point to next line.
C-p	Move point to previous line.
C-v	Scroll source windows up.
M-v	Scroll source windows down.
M-<	Go to beginning of buffer.
M->	Go to end of buffer.
C-L	Recenter.
C-c v	Display efficiency notes for the variable or subprogram name at the point.
g	Go to line number of buffer.
q	Quit cmax−viewer mode.

NOTE: If you prefer side-by-side source windows to top-and-bottom source windows, add this line to the EMACS code:

```
(setq cmax-viewer-split-function
     'split-window-horizontally)
```


Copyright © 1994 Thinking Machines Corporation

A.4.2 Command cmax−subprogram

The command **M−x cmax−subprogram** sends the subprogram surrounding the point to a CMAX process and displays the translated output in the buffer named CMAX.

If invoked with a prefix argument (**C−u**), the command prompts for additional switch arguments and passes them to the **cmax** command. Alternatively, set the variable **cmax−default−switches** (it is set to **−verb=4** by default).

A.4.3 Command cmax−buffer

The command **M−x cmax−buffer** sends the contents of the current buffer to a CMAX process and displays the translated output in the buffer named CMAX.

If invoked with a prefix argument (**C−u**), the command prompts for additional switch arguments and passes them to the **cmax** command. Alternatively, set the variable **cmax−default−switches** (it is set to **−verb=4** by default).

Copyright © 1994 Thinking Machines Corporation