CMZ

CodeME S.A.R.L.
27 March 1998

Quick Guide

CMZ is an advanced, interactive and machine-independent source code management and version control system.

The present Quick Guide reproduces only partly the documentation "A Source Code Management System; CMZ User's Guide and Reference Manual". Programmers with little or no experience in code management are advised to read the previously mentioned manual for a complete view of the subject.

Notation and Conventions

In this guide, commands and other strings that must be typed in literally are shown in a typewriter font, like `PURGE`. Source code directives, file, patch, deck and other names are also in a typewriter font but surrounded by apostrophes, like '`+PATCH,AAP`'. Placeholders such as command argument names are in a slanted font, like *argument*. Square brackets ('`[ ]`') enclose optional items in command descriptions. A vertical bar ('|') OR's items in a list of choices, and brackets ('`( )`') group items in a list of choices.

In commands uppercase and lowercase characters are generally equivalent, so `SHELL` is the same as `shell`. Commands may be abbreviated until they become ambiguous, e.g. `select` may be abbreviated to `sel`, not, however, to `se` (ambiguous with `sequences`). In case of an ambiguous command a list of possible commands is shown. In the guide the part of the command names in upper case indicates the minimum abbreviation.

Whenever you have to specify a filename in CMZ you can use the standard Unix filename syntax, like '`/user/rdm/test.f`'. The '~' as in '`~/delphi/delgra.car`' are interpreted in the usual Unix way. Environment variables at the beginning of a filename are substituted by their associated value, e.g. '`$HOME/pipo.f`' will become '`/u/xx/rdm/pipo.f`' when '`$HOME=/u/xx/rdm`'.

The DECKLIST Argument

A number of CMZ commands allow as an argument a DECKLIST. The DECKLIST argument allows you to specify one or more sets of decks. Some examples:

`//top/patch2/deck1`	'deck1' in 'patch2' in file 'top'
`deck1`	'deck1' in CMZ Working Directory, CWD
`deck3.deck6`	from 'deck3' to 'deck6' included in CWD (the order is as from the listing)
`*`	all decks in the CWD or all decks in all patches if the CWD is the top directory

Starting CMZ

UNIX SYSTEMS

To invoke CMZ just type `cmz` at the shell. The allowed CMZ arguments are:

'`-n`'	Do not read cmzlogon file `cmzlogon.kumac`.
'`-l` *logon*'	Read logon commands from file '*logon*`.kumac`'. The '`-l`' and '`-n`' options are mutually exclusive.
'`-r [`*restore*`]`'	Restore the environment of the previous CMZ session. By default the file '`cmzsave.dat`' will be read. This option is mutually exclusive with all the others.
'`-b` *batchjob*'	Run CMZ in batch mode. The commands CMZ has to execute are read from the file '`batchjob.kumac`'. The '`-b`' option implies also the '`-n`' option.
'`-install` *file* `[`*opts*`]`'	Execute the macro '*install*' in file '*file*`.cmz`' or '*file*`.car`'
'`-x` *file* `[`*mname*`[`*opts*`]]`'	Execute the default macro '*flogon*' (or the macro '*mname*' with option '*opts*') in file '*file*`.cmz`' or '*file*`.car`'

VMS SYSTEMS	**IBM VM/CMS**
`cmz`	`cmz`
'`/NOLOG`'	'`(NOLog`'
'`/LOGON=`*logon*'	'`(LOGon=`*logon*'
'`/RESTORE[=`*restore*`]`'	'`(REStore[=`*restore*`]`'
'`/BATCH=`*batchjob*'	'`(BATch=`*batchjob*'
'`/INSTALL=`*file* `[,`*opts*`]`'	'`(INStall=`*file* `[,`*opts*`]`'
'`/X=`*file* `[,`*mname*`[,`*opts*`]]`'	'`(X=`*file* `[,`*mname*`[,`*opts*`]]`'

Exiting CMZ

To terminate CMZ type `EXIT`.

CMZ Language Independent Source Code Directives

```
+TITLE.
+PATCH, pname [,T=language] [,IF=...].
+DECK, dname [,T=language] [,IF=...].
+KEEP, sname [,IF...].
+SEQ, sname [,s2, ...]  [,IF=...].
+SELF [,IF=...].
+IF,(a,b) & -c.
+ELSEIF, d.
+ELSE.
+ENDIF.
"," or "|" = OR , "&" = AND , "-" = NOT
```


Example of a CMZ Ascii Readable file: demo.car

```
+TITLE
DEMO 1.34/00 900806 17.04 CMZ demonstration file

+PATCH,DCDES.
+KEEP,DCONST.
      INTEGER K1,K2
      COMMON/DCONST/K1,K2

+KEEP,DSTORE.
      PARAMETER (NWORDS=50000)
      INTEGER NWORDS,LPOINT,LHITS
      REAL WS
      COMMON/DSTORE/WS(NWORDS),LPOINT,LHITS

+PATCH,INIT.
+DECK,START.
      SUBROUTINE START
+SEQ,DCONST.
      K1=1
      K2=2
      ...
      END

+DECK,FOPEN.
      SUBROUTINE FOPEN
+SEQ,DSTORE.
+IF,IBM.
      OPEN(1,FILE='/FNAME FTYPE',STATUS='OLD')
+ELSE.
      OPEN(1,FILE='FNAME.FTYPE',STATUS='OLD')
+ENDIF.
      READ(1,1000)WS
      ...
      END

+PATCH,WORK.
+DECK,COMPUT.
      SUBROUTINE COMPUT
+SEQ,DCONST,DSTORE.
      DIMENSION SUM(NWORDS)
      SUM(1)=WS(1)
      DO 10 I=2,NWORDS
         SUM(I)=SUM(I-1)+WS(I)
10    CONTINUE
      END

+DECK,RESULT.
      SUBROUTINE RESULT
+SEQ,DCONST,DSTORE.
      PRINT *,'WS(K1)=',WS(K1),'WS(K2)=',WS(K2)
      END

+PATCH,CWORK,T=C.
+DECK,CUPAD,IF=UNIX.
#include <string.h>
void cupad_(char *name ,int len_name)
{
  char fname[256];
  strncpy( fname, name, len_name );
  fname[len_name] = '\0';

  ku_pad(fname,1);
}
```

Setting default file names and options

SET fname ftype lang
Change the default filenames for the **editor**, **extract** (or source) (Fortran, C, text, etc.), **library, execution, template** or **save** files. The Filecase may be changed via the command FILECASE. The language selection is based on the 'T=' directive on the +DECK and/or +PATCH line. If the name of the language follows the options -F, -C, -COM or -INC the properties referring to this language can be modified without changing the current language.

`SET`	show the current defaults
`SET C -LAN`	set C the current language, all new edited decks will have the directive 'T=CC' on the +DECK line
`SET TEXT -LAN`	set TEXT the current language, all new edited decks will have the directive 'T=TEXT' on the +DECK line
`SET ? -LAN`	list all defined languages
`SET * -LAN`	all defined languages selected for compiling
`SET . -LAN`	no languages selected for compiling
`SET F C C++ -LAN`	languages Fortran77, C and C++ are selected for compiling
`SET * -C++ -LAN`	all defined languages except C++ selected for compiling
`SET go.com -X`	set execution filename which contains compile and/or archive commands created by the commands `CCompile`, `CLib` and `CMake`. With the option 'X' additional options can be specified as shown below
`SET go.com -XA`	idem as above and in addition the execution file 'go.com' will be automatically executed by the command `CCompile`, `CLib` or `CMake`.
`SET go.com -XACDL`	idem as above and in addition a Comment (C) directive is added before each compiler directive, the Librarian (L) will be invoked after all calls to the compiler and the object file(s) will be Deleted (D) once they have been replaced in the library (the default)
`SET go.com -XACD2L`	idem as above and in addition the extract file(s) will also be Deleted (D2)
`SET cmlib.a -L`	set the object library filename which will be created by `CLib` or `CMake`
`SET $cmzfile.a -L`	$CMZFILE will be replaced by the file name corresponding to the current directory. For instance, if the current file is 'project.cmz', then the library file will be set to 'project.a'
`SET cmedt.edt -E`	set the editor filename to which command EDIT writes
`SET $cmzfile.edt -E`	$CMZFILE will be replaced by the file name corresponding to the current directory
`SET cmzsave.dat -S`	set the file on which the current environment is saved when exiting from CMZ
`SET . -S`	unset the save file, no environment will be saved

• Use option -COM to define the comment character.

`SET % -COM LaTeX`	set % as comment character for LaTeX language

• Use option -F[E] to define for each language the name of the extract files to which commands `CXtract`, `CCompile`, `CLib` and `CMake` will write. If 'fname' contains the character "*", then the commands `CXtract`, `CCompile`, `CLib` and `CMake` will automatically write as many files as there are decks in the DECKLIST of these commands. The "*" will be replaced by the deck names. The $CMZDIR cmz variable will be replaced by the current cmz directory.

```
SET src/new_*.f -F F
SET src/$CMZDIR_*.c -F C
SET latex/*.tex -F TEX
```

• If option -C is given, 'fname' is a string between quotes containing the compiler directive for the currently selected language. The compiler directive must at least contain one $COMPFILE string which will be replaced at execution time by the name of the file to be compiled.

```
SET 'f77 -c -g +ppu $compfile' -C F77
SET 'cc -g -c $compfile' -C C
```

`SET . -C EIFFEL`	Remove compiler directives for EIFFEL language mode.

• Use option -INC to select the include option. When this option is active, the sequences are written onto include files instead of being developed inside the extract file. General form :

```
SET 'directory_path/$SEQ.exth' -INC language_name
```

The string $SEQ is replaced by the name of the sequence when writing the deck onto the default extract file. If the directory_path is not set, the include files will be written in the same directory as the one of the extract files. For example:

```
SET src/*.c -F C
SET '#include "$SEQ.h"' -INC C
```

The CMZ directive '+SEQ,PROJ' will be replaced by the include statement '#include "proj.h"' and the file 'proj.h' will be written in the directory '**src**', if it does not exist or if the sequence PROJ has been modified.
The command 'SET . -INC' disables the include directive and sequences will be expanded in the extract files.
• Use option -T[D] to set the template filename which contains a deck header template. The deck header template will be inserted by EDIT every time a new deck is created. The strings $DECK, $AUTHOR, $DATE and $TIME are reserved and will be replaced by the deck name, the author name (as set by the AUTHOR command), the date and the time, respectively.

`SET temp.cmt -T`	set the template filename
`SET . -T`	unset the template filename

Example of a macro 'install'

```
+PATCH,//DEMO/$KUMACS
+DECK,INSTALL.
*CMZ :  1.44/19 14/10/93  10.25.57  by  M. Baker
*-- Author :    F. Smith   24/04/91
macro  install  machine=x
*
if ([machine]=x) then
  NAME = $MACHINE
else
  NAME = $upper([machine])
endif
case [NAME] in
 (HPUX,SUN)  set src/*.f  -F f77
             set src/*.c  -F c
             set libdemo.a   -L
 (VMS)  sel . [name]
        set '[.SRC]demo.for' -F f77
        set '[.SRC]demo.c' -F c
        set demo.olb -l
        set '$ for/obj=$compfile.obj $compfile' -C f77
        set '$ cc/obj=$compfile.obj $compfile'  -C c
 (?,HELP,X,*)
   mess To install DEMO on HPUX: cmz -install demo HPUX
   mess To install DEMO on SUN : cmz -install demo SUN
   mess To install DEMO on VMS : cmz /install=demo,VMS
   exitm
endcase
case [NAME] in
  (HPUX) sel . [name] unix
         set 'cc -g -c -Ae $compfile' -C c
         set 'f77 -K +ppu -g -c $compfile' -C f77
  (SUN)  sel . [name] unix bslash
         set 'cc -c -ansiposix -g -G 0 $compfile' -C c
         set 'f77 -c -g $compfile' -C f77
endcase

mess Installation on [NAME] starting at $DATE $TIME...
set F C -lan
seq //
clib *
mess Installation on [NAME] finished at $DATE $TIME...
return
```

Summary of macro functions (see help functions)

`$MACHINE`	Hardware or Unix brand, e.g. HPUX or VAX
`$OS`	Operating system name, e.g. UNIX or VMS
`$DATE`	Current date in format DD/MM/YY
`$TIME`	Current time in format HH.MM.SS
`$CWD()`	Current CMZ working directory
`$CMZFILE()`	Name of the current CMZ file
`$VERSION(fname)`	Version number of the file fname
`$NDECKS('?')`	Number of decks in the CWD
`$NDIRS('?')`	Number of directories in the CWD
`$NFILES('?')`	Number of connected files
`$DECKNAME(id)`	Name of the IDth deck
`$DIRNAME(id)`	Name of the IDth directory
`$FILENAME(id)`	Name of the IDth file

Interface to Data Base manager

CReate [-C] fname [nrecs]
Create a new CMZ file named fname.cmz

FILe fname [opt exec]
Attach an existing CMZ file fname.cmz

`FILE hbook`	connect in update mode
`FILE hbook -R`	connect in readonly mode
`FILE aa -X`	connect 'aa.cmz' and execute the default macro 'flogon'
`FILE aa -X load`	connect 'aa.cmz' and execute macro 'load'

RELease fname
Close and disconnect file 'fname' from the CMZ session.

MKDir newdir
Create a new directory in current CMZ file.

RMDir dirnam
Delete a directory in current CMZ file.

CDir [path]
Change the current working directory (CWD) to 'path'. If no pathname is given print the pathname of the CWD.

DIRectory [opt path] (alias LS)
List the contents of a directory.

`DIR dir1`	list contents of directory 'dir1'
`DIR A?*`	list all decks starting with A

COPy source target (alias CP)
Copy deck(s) from the source pathname to the target pathname.

`COPY deck1 deck2`	copy 'deck1' to 'deck2'

CPTree topd1 topd2
Copy file 'topd1' to 'topd2'.

`CPTREE //top1 //top`	copy all directories, decks and cycles from '//top1' to '//top2'

MOve obj opt ref [type] (alias MV)
Move an object OBJ before (-B) or after (-A) object 'ref'.

`MV deck8 -B deck1 -D`	move 'deck8' before 'deck1'

DELete dklist (alias RM)
Delete decks from a CMZ file.

PUrge [keep opt]
Purge all but the highest cycle of all decks in the current directory.

Interface to Local Editor

Edit dklist
Edit one or more decks using the local editor.

`E deck1`	only 'deck1'
`E *`	all decks in patch
`E -B`	all decks in the decklist buffer
`E -V 1.00/01 *`	all decks belonging to version 1.00/01
`E $keep1`	edit sequence 'keep1'

Read dklist
Browse one or more decks with local browser or pager.

`READ deck`	browse 'deck'
`READ -S deck`	idem as above, but with all `+IF` and `+SEQ` directives resolved

HOST_Editor [name]
Set the host command to invoke the editor. The EDIT command will invoke this editor. If NAME='?' the current host editor command is shown. On Unix workstations it is possible to do asynchronous editing via the edit server, i.e. to start an editor in a separate window while CMZ can continue to receive commands. In order to do that, the following conditions must be fulfilled:

- The edit server 'kuesvr' must be found in the search path.
- NAME must end with an ampersand ('&').
- The environment variable 'DISPLAY' must be set.

If the edit server cannot be used, the ampersand will be ignored. No other CMZ commands can be issued until the editor terminates if either the 'kuesvr' is not available or 'DISPLAY' is undefined. When using the edit server the editor command is expected to create its own window. The startup value can be defined by the environment variable 'EDITOR'. Otherwise it is set to a system dependent default: 'DM' (Apollo), 'EDT' (VMS), 'XEDIT' (VM/CMS), 'vi' (Unix).

`HOST_E TPU/DISP=DECW`	DECwindow interface to EVE
`HOST_E 'emacs -geometry 80x48'`	
`HOST_E 'xterm -e vi &'`	Editor will start `vi` in a new `xterm`

HOST_Pager [pager]
Set the host command to view a file in read-only mode. The startup value can be defined by the environment variables 'KUIPPAGER' or 'PAGER'. If neither of them is defined the value set by the HOST_EDITOR command is used. On VAX/VMS the startup value is 'TYPE/PAGE'.

```
HOST_P more
HOST_P 'xterm -e view &'
HOST_P 'ved &'
```


Example of a small interactive session

(hpcmz) #[321] **cmz**

```
**** CMZ Code Manager version  1.45/17  (01/03/94) ****
```

CMZ [1] **AUTHOR Louis Daniel**
CMZ [2] **CREATE -C analysis**
analysis [3] **MKDIR code**
analysis [4] **CD code**
analysis/code [5] **DIR**

```
 Current Working Directory = //ANALYSIS/CODE
   00_PATCH;1
 Total:       1 Decks      1 Cycles
```

analysis/code [6] **CIMPORT analysis.f**

```
 Number of DECKS   read   :  4
 Number of RECORDS written:  428
```

analysis/code [7] **DIR**

```
 Current Working Directory = //ANALYSIS/CODE
   00_PATCH;1   HWABEG;1     ANALIS;1     HWAEND;1
   FILLHIST;1
 Total:       5 Decks      5 Cycles
```

analysis/code [8] **HOST_ED 'vi &'**
analysis/code [9] **VERSION -S 1.00/00**

```
 VERSION: Directory $VERSION created
 VERSION: Creating new TITLE deck
```

analysis/code [10] **SET *.f -F F77**
analysis/code [11] **EDIT new**

```
 EDIT: New deck new
```

analysis/code [12] **DIR**

```
 Current Working Directory = //ANALYSIS/CODE
   00_patch;1   hwabeg;1     analis;1     hwaend;1
   fillhist;1   NEW;1
 Total:       6 Decks      6 Cycles
```

analysis/code [13] **ALPHA_ORDER**
analysis/code [14] **DIR -A**

```
 Current Working Directory = //ANALYSIS/CODE
 DECKNAME     CYCLE    VERSION    DATE/TIME  BYTES  LINES
 00_PATCH        1*    1.00/00 940119/1145     68      2
 ANALIS          1*    1.00/00 920402/ 119   7472    226
 FILLHIST        1*    1.00/00 920402/ 124    952     36
 HWABEG          1*    1.00/00 920402/ 118   3032    116
 HWAEND          1*    1.00/00 920402/ 123   1592     54
 NEW             1             940119/1151    112    153
 Total:       6 Decks      6 Cycles       437 Lines
```

analysis/code [15] **CCOMPILE new**

```
 Number of DECKS   written:  1
 Number of RECORDS written:  206
```

analysis/code [16] **SHELL ls *.f**

```
 analysis.f new.f
```

analysis/code [17] **EXIT**

```
 All connected CMZ files are now released

 **** CMZ normal termination ****
```

(hpcmz) #[322]

CMZ Basic Commands

AUthor [name]
Make yourself known to CMZ.

`AUTHOR Louis Daniels`	next modified decks will be stamped with Louis Daniels

SELect [olist]
Select control options (used to test `IF=` conditionals on CMZ directives).

`SELECT`	show active options
`SELECT .`	clear option list
`SELECT vax gks debug`	set options
`SELECT -gks`	remove option 'gks' from option list

SEQuences [dirnam]
Store into memory all sequences definitions (`+KEEP` directives) in directory DIRNAM. Always set the control options with command SELECT before loading the sequence definitions. The control options are used to test `IF=` conditionals on the `+KEEP` directives. By default the corrections referenced in the USE bank are taken into account, they will be ignored if the option -U is set.

`SEQ`	show all stored sequences definitions
`SEQ .`	clear all sequences from memory
`SEQ hcdes`	store into memory all sequences in patch 'hcdes'
`SEQ //file1 //file2`	store into memory all sequences in both file 'file1' and 'file2'
`SEQ //`	store into memory all sequences in all connected files
`SEQ -U //`	idem as above, but corrections referenced in the USE bank are ignored
`SEQ -O hcdes`	overwrite the sequence definitions of the sequences in memory with the ones in patch 'hcdes'
`SEQ -V 1.00/02 hcdes`	store all sequences in patch 'hcdes' belonging to version 1.00/02
`SEQ -E`	view the full sequence definitions in the local pager or browser
`SEQ -S`	idem as above, but with all `+SEQ` directives resolved
`SEQ -F file`	write sequences to 'file'
`SEQ -F *.h`	write sequences on separate files (∗ replaced by the sequence name)

Lmods [author frdate todate]
List all decks modified by 'author' between 'frdate' and 'todate'.

`LM Smith 13.15`	list all mods by user Smith since today from 13.15 pm
`LM -Smith -V 1.28/03`	list all mods made by all users except Smith after the version 1.28/03
`LM -O`	write a 'file.history' file which contains the date and time of the last deck changed. 'file' is the name of the connected file
`LM -O hbook.lastmod`	idem as above but write on 'hbook.lastmod'
`LM H.Louis 11/01/94`	list full history of all the decks modified by user Louis after 11/01/94

CXtract dklist
Write one or more decks onto the default extract file. Their corresponding languages have to be selected via the command SET .... -LAN.

`CX deck1`	write 'deck1' to the default extract file
`CX *`	write all decks in the CWD to the extract file
`CX deck2.deckN`	write all decks from 'deck2' to 'deckN'
`CX deck2.`	idem as above but from 'deck2' to the last deck in the CWD
`CX .deckN`	idem as above but from the first deck to 'deckN' in the CWD
`CX GC?*`	idem as above but for all decks starting with GC
`CX -B`	write all decks in the decklist buffer
`CX //`	write all decks in all patches in all files
`CX -V 1.23/01 *`	write version 1.23/01 to the default extract file
`CX $USE`	process all decks referenced in the USE bank (see command `USE`)
`CX -R *`	process all decks using corrections then remove decks from the USE bank
`CX -N pat1/new2`	write deck 'new2' in patch 'pat1' in the USE bank to the extract file
`CX -U *`	process all decks in the CWD without using the corrections

CCompile dklist
Write one or more decks onto the default extract file and compile the file with the local compiler of the chosen language.

`CC *`	compile all decks in the CWD
`CC -H *`	idem as above but no CMZ history records
`CC -B`	compile all decks in the decklist buffer
`CC //`	all decks in all patches in all files
`CC -V 2.01/05 *`	compile all decks from version 2.01/05
`CC $USE`	compile all decks in the USE bank(see command `USE`)
`CC -U *`	compile all decks in the CWD, do not use corrections in USE bank
`CC -R *`	idem as above but use corrections then remove decks from the USE bank
`CC -N pat1/new2`	compile deck 'new2' in patch 'pat1' in the USE bank

CLib dklist
Write one or more decks onto the default file, compile and archive. Use the local compiler and librarian (archiver). The extract file name and the library file name can be defined by the command: SET ... -F or -L. The options -U, -R, -H, -N and -V must precede the 'dklist' or any other option.

`CL deck1 deck2`	compile and archive 'deck1' and 'deck2'
`CL *`	compile and archive all decks in the CWD
`CL -P`	compile and archive all patches selected by the `PILOT` command
`CL -HP`	idem as above but without CMZ history records
`CL -P deck1`	compile and archive 'deck1', items in PILOT bank (see command `PILOT`) are used together with SELECT items for testing IF= sequences
`CL -B`	compile and archive all decks in the decklist buffer
`CL //`	all decks in all patches in all files
`CL -V 1.44/01 *`	compile and archive all decks of version 1.44/01
`CL $USE`	process all decks in the USE bank
`CL -UR deck1`	compile and archive 'deck1' without using corrections and remove 'deck1' from the USE bank (see command `USE`)
`CL -N pat1/new2`	compile and archive deck 'new2' in patch 'pat1' in the USE bank
`CL -FR f.car $USE`	process decks in the file 'f.car' then remove them from the USE bank and process the remaining decks referenced in the USE bank

CMake [opt]
Recompile decks that are out of date compared to the date and time written on the 'file.history' file (written by the command 'LMODS -O'). `CMAKE` compare the time of last change in the history file (or of the date and time of the default library if the history file does not exist) with the time of last change of all the decks in the CWD and its sub-directories. Decks that are more recent than the time of the last change will be written to the default extract file, recompiled and replaced in the library.

`CM`	recompile all decks in the CWD and its sub-directories that are out of date
`CM -O`	recompile all decks that are out of date compared to their default object files
`CM -N`	list decks that need to be recompiled
`CM -A`	recompile all decks in all connected files that are out of date
`CM -V 2.03/00`	recompile all decks modified since the version 2.03/00 of the file
`CM -V`	recompile all decks modified since the version number on the history file
`CM -I user.history`	recompile all decks that are out of date compared to the date written on 'user.history' file

COR orig [desc corfil]
Generate a correction file by comparing two decks, patches, files or versions.

`COR deck1 deck2`	compare two decks
`COR //file1 //file2`	compare two files
`COR pat1 pat2 -F file.cor`	compares two patches and writes the differences to FILE.COR.
`COR // //userf`	Compare decks available in USERF with decks in all connected files
`COR -V 1.19/01 -VD 1.23/03`	compare version 1.23/03 with 1.19/01
`COR -O -V 1.19/01 //file1 //file2`	compare only the decks available in 'file2' with the decks belonging to the version 1.19/01 in 'file1'

USE [opt corlis]
Read the contents of correction files into memory (the USE bank).

`USE`	list patches and decks referenced in USE bank
`USE .`	clear all corrections from the USE bank
`USE fil1.cra fil2.cra`	load corrections from files into USE bank
`USE -I fil1.cra fil2.cra`	idem as above, but in case of clashes corrections are taken from 'fil1.cra'

UPdate dklist
Apply the corrections stored in the USE bank to the decks specified by 'dklist'.

`UP -E deck1 deck2`	apply the corrections in the USE bank to 'deck1' and 'deck2' and view the result in the editor
`UP $USE`	process all decks referenced in the USE bank
`UP -REF //ref //myfil`	apply the corrections in the USE bank to the file 'ref' and save the result in the file 'myfil'

Version [versn]
Mark highest cycles of all decks in the current file as belonging to one version.

`VERSION -S 1.00/00`	set base version number on current file
`VERSION +`	mark current file as version x.y/z+1
`VERSION +0`	mark current file as version x.y+1/00
`VERSION`	list the current highest version number

CMZ Import Commands

ARC fname [dklist]
Convert a CMZ Ascii Readable file to a CMZ file.

`ARC a.car`	read complete file A.CAR
`ARC a.car deck1`	read only DECK1 from A.CAR

CImport [fname]
Import one or more source files into a CMZ file. The file becomes a DECK with the type depending on the current language (+DECK followed by T= 'type of the current language') in the current working directory.

`CI foo.tex`	becomes deck 'FOO'
`CI -T latex foo.tex`	becomes deck 'FOO,T=LATEX'
`CI -A hello.tex`	idem as above but author field is filled in
`CI -S hello.cdf`	becomes deck hello and matching sequences in memory will be replaced by a '+SEQ' line
`CI ?*.f`	import all files ∗.f (on Apollo and UNIX systems only)
`CI h?*.c`	import all files h∗.c (on Apollo and UNIX systems only)

TTOC [fname]
Import one or more plain TEXT files into a CMZ file. The TEXT file becomes a DECK of type DATA. If the TEXT file is very long, it is split in several decks. A suffix "_id" is added to the deck name.

`TTOC foo.cdf a1.txt`	become decks FOO and A1
`TTOC longfile.txt`	become decks longfi_1, longfi_2 etc...

CMZ Export Commands

CAR fname [dklist]
Convert a CMZ file to a CMZ Ascii Readable file.

`cd //file;CAR test.car`	write the CAR file corresponding to //file
`CAR deck1.car deck1`	write only DECK1 onto DECK1.CAR

CLIst fname [dklist]
Write a listing of a CMZ file or directory on 'fname'.

`CLIST test.lis`	list file or directory on list file 'test.lis'
`CLIST test.lis -E`	idem as above and view result in editor

AXtract fname [dklist]
Convert a CMZ Ascii Readable file directly to an extract file. This command is the combination of `ARC` and `CXtract`.

CTOT [dklist]
Write one or more decks as plain text onto the default text file.

CTOY fname [dklist]
Convert a CMZ file to a CAR file compatible with Patchy.

CMZ Utilities Commands

Alpha_order
Order all decks in the current working directory in alphabetical order.

Buffer [optlis]
Show or change the decklist buffer.

`BUFFER`	show current decklist buffer
`B .`	clear decklist buffer
`B //file/pat1/dec2`	add '//file/pat1/dec2' to buffer
`B -deck1`	remove 'deck1' from decklist buffer

DIFferences deckref [deck]
Find the differences between two decks.

`DIF deck1 deck2`	compare 'deck1' and 'deck2'
`DIF -H deck1 deck2`	compare 'deck1' and 'deck2' ignoring CMZ history records
`DIF patch1 patch2`	compare 'patch1' and 'patch2'
`DIF deck`	compare 'deck;max-1' and 'deck;max'

FINd dname [opt]
Scan all connected files for deck 'dname'.

`FIND deck`	search for 'deck' in all connected files and print out its path
`FIND cmd?* -E`	find all decks whose name start with 'cmd' and put them in the editor

LIbrary [opt fname]
Add, replace or delete (option -D) the object modules in the current object library with the modules in 'fname'.

`LIB foo`	put modules in 'foo.[bin, o, obj]' in the default library
`LIB -D foo.o`	delete module 'foo.o' from the default library

TOuch [-T TIME] dklist
Update the time of the decks in DKLIST (see DECKLIST).

`TOUCH deck1 deck2`	update the date and time of DECK1 and DECK2 to the current date and time.
`TO -T 15/12/95.14.12 *`	update the time of all decks in the current CMZ working directory to the specified time.

PIlot [pilpat]
Store all PATCHY `+USE` directives in directory 'pilpat' in memory. Command `PILOT` is mainly added for PATCHY backward compatibility.

`PILOT *apollo`	scan patch '∗apollo' for `+USE` directives and all patches referred to via the `+USE` directives

REGexp

This is a summary of the regular expression syntax used for pattern matching in CMZ (e.g., by GREP).
A text pattern is a concatenation of the following elements:

`c`	literal character c
`?`	any character except newline
`%`	beginning of line
`$`	end of line (null string before newline)
`[...]`	character class (any one of these characters (e.g. [a-z])
`[^...]`	negated character class (all but these characters)
`*`	closure (zero or more occurrences of previous pattern)

Grep pattern [dklist]

Scan ‘dklist’ for the occurrence of the text pattern ‘pattern’.

`GREP string`	search for "string" in all decks and sub-directories
`GREP string -I`	search for "string" as well as "STRING"
`GREP ab?*`	search for all occurences of strings starting with "ab".

EXEC mname [args]

Execute the command lines contained in the macro ‘mname’. Macros may be stored in a deck or in a normal text file. If a deck or file contains several macros, the character ‘#’ is used to select a particular macro inside the deck or file. If MNAME is of the form DECK_NAME#MACRO_NAME, the macro MACRO_NAME in DECK_NAME is executed. If MNAME does not contain the character ‘#’, the first macro found will be executed.

If MNAME is a full CMZ pathname the macro in that deck will be executed. IF MNAME is not a full pathname it will be searched for in the deck MNAME in the CWD. If it cannot be found in the CWD it will be searched for in the CMZ directory $KUMACS. If MNAME cannot be found in the CMZ file, it will be searched in the file ‘*mname.kumac*’.

A macro consists of a sequence of CMZ commands. The first line in a macro must be the line MACRO MACRO_NAME and the last line of the macro must be RETURN. If the macro contains +IF directives they will be resolved before the macro will be executed (only for macros in CMZ decks).

`EXEC install`	execute first macro in deck ‘install’ (in the CWD, or in $KUMACS or in file INSTALL.kumac)
`EXEC install apollo`	idem as above but give argument APOLLO to the macro
`EXEC mdeck#dialog`	execute macro DIALOG in deck MDECK

Fortran77 Utilities

The following commands can only be used on Fortran77 decks.

FToc fname [dklist]

Import a Fortran source file into a CMZ file. Each subroutine or function becomes a DECK in the current CMZ directory.

`FTOC fname`	read all routines from file ‘fname’
`FTOC fname routine1`	read only ‘routine1’

Indent [dklist]

Indent one or more decks.

`INDENT deck1`	indent ‘deck1’

LAbel dklist

Change label numbers in one or more decks.

`LABEL deck1`	re-label ‘deck1’

STEps [ilone ilstep ifone ifstep ilend]

Change default parameters to be used by the command LABEL.

`STEPS 5 5`	change ‘ilone’ and ‘ilstep’
`STEPS`	show current settings

TRee [dklist]

Draw the calling tree of a routine.

`TREE deck`	draw calling tree of routine ‘deck’
`TREE -F deck`	same as above but functions are also processed

UNdefined [dklist]

Find variables which are not initialized in a list of decks. UNDEFINED cannot detect undefined variables in COMMON blocks. UNDEFINED can also check the consistency of routine calling sequences (number of parameters and if they are input and/or output). With argument LEVEL the level of information printed (1 to 5) is given. The new value will remain valid until it is redefined.

LEVEL=1:	(default) minimum output.
LEVEL=2:	UNDEF prints variables, commons and sequences which are not used in the current routine.
LEVEL=3:	UNDEF prints global variables.
LEVEL=4:	UNDEF prints local variables.
LEVEL=5:	same as 1+2+3+4.

`UNDEF deck1`	check ‘deck1’ for undefined variables
`UNDEF $USE`	check all decks referenced in USE bank (see comand USE
`UNDEF 3`	set default output level to 3 (may be in CMZLOGON)

Example of a Batch session

When running cmz in batch mode with the argument ‘-b batchfile’, the trace is written on the batchfile.log in the working directory.

Example of batchfile.kumac file:

```
*
* Connect general project files.
* Load the appropriate corrections.
* Extract decks referred in the user file (without
* taking care of possible corrections (-U)),
* then after removing these decks from the USE bank,
* process all remaining decks in the USE bank ($USE).
*
* $GROUP environment variable defining the access path
* of general files.
*

file $GROUP/project1 -r
file $GROUP/project2 -r
file $GROUP/project3 -r
file userfile
*
sel $MACHINE $OS DEBUG
seq //
*
use $GROUP/cproject1.corr
use $GROUP/cproject2.corr
use $GROUP/cproject3.corr
*
CXtract -UR //userfile $USE
*
exit
```

Summary of macro language

`MACRO mname par1=val1 ...`	begin macro mname
`RETURN`	end of macro mname
`label:`	label (must terminate with a colon)
`GOTO label`	jump to label
`ON ERROR GOTO label`	resume at label on error condition
`IF logical_exp. GOTO label`	conditional statement
`CASE, ENDCASE`	macro flow control
`IF-THEN, ELSEIF, ELSE, ENDIF`	
`DO, ENDDO`	
`FOR, ENDFOR`	
`EXITM`	macro termination
`par = arithmetic_expression`	assignment statement

`Arithmetic operators`	+ , - , * , /
`Logical operators`	= , < , <= , > , >= , <>