CA-Clipper
Version 5.3
Release Notes

June 1995

This document contains the latest information regarding *CA-Clipper.* This information is provided as an update to the written documentation.

Using Write to View This Document

This document will be easiest to read if the Write window is at its maximum size and the printer orientation is set to portrait. To enlarge the Write window, click the Maximize button in the upper right corner of the window, or open the Control menu in the upper left corner of the Write window (press Alt+Spacebar) and choose Maximize. To ensure portrait orientation, choose File, Print Setup, and select Portrait for the Orientation choice.

To move through the document, press Page Up or Page Down or click the arrow at the top or bottom of the scroll bar along the right side of the Write window.

To print the document, choose the Print command from the File menu.

For help using Write, press F1.

To read other On-Line documents, choose the Open command from the File menu.

Contents

This document contains information on the following topics:

File Updates

The Blinker Order Form and CDXLOCK.OBJ files are new files that are not referenced in the written documentation. Please see the text below for a description of these files. LLIBG.LIB and DBINFO.CH have been updated since the written documentation went to press. This updated information is included in the text below.

BLINKER Order Form
In addition to the ASCII file order form, BLINKER.TXT, there is another file, BLINKER.WRI in Write format for Windows users.

CDXLOCK.OBJ
Use the CDXLOCK.OBJ driver in order to use your CA-Clipper database concurrently with FoxPro databases. The CDXLOCK.OBJ driver is necessary to maintain compatible locking schemes between CA-Clipper and FoxPro.

LLIBG.LIB
In the CA-Clipper Reference Guide and the CA-Clipper Technical Reference Guide, the LLIBG.LIB library is shown as being part of the default libraries. This library is NOT part of the default libraries and must be added to your Exospace link script in order to use graphics mode with CA-Clipper.

DBINFO.CH
Several symbols have been removed from the header file DBINFO.CH and now reside in ORD.CH or BLOB.CH.

Command and Function Changes

The following is an update to the CA-Clipper Reference Guide. These commands and functions have been updated since the written documentation went to press. Please see the text below for the most recent updates to these commands and functions.

COLORSELECT()
The 4th parameter is used as the hotkey color for all @ GET controls.

DBINFO()
The following constants are used with the DBINFO() function to specify the type of information requested:

DBI_GETDELIMITER
This information is related to the type of RDD used. This setting always returns NIL.

DBI_LOCKOFFSET
This setting is not supported and will always return a value of NIL.

DBI_VALIDBUFFER
The DBI_VALIDBUFFER setting of DBINFO() will return .T. after an add/delete/update of a record. In all other cases, the return value will be .F..

DBRECORDINFO()
Valid *<infotype>* values should be listed as:

DBRI_DELETED
DBRI_LOCKED
DBRI_RECNO
DBRI_RECSIZE
DBRI_UPDATED

DIRMAKE()
The DIRMAKE() function has been renamed to MAKEDIR().

DISPBEGIN()
The DISPBEGIN() function is ignored when the application is in graphics mode.

DISPEND()
The DISPEND() function is ignored when the application is in graphics mode.

LLG_VIDEO_TXT
The LLG_VIDEO_TXT constant is incorrectly spelled as LLG_VIDEO_TEXT in the CA-Clipper Reference Guide. For a list of all LLG constants, please see the LLIGB.CH include file in the CLIP53\INCLUDE directory.

MEMOREAD()
The MEMOREAD() function only searches for a text file in the current directory. It does not search the DOS PATH for the file as is stated in the CA-Clipper Reference Guide.

MSETBOUNDS()
When using the SETMODE() function, the mouse boundaries set by MSETBOUNDS() are automatically reset to the entire screen. After using the SETMODE() function, you must call the MSETBOUNDS() function to reset the mouse boundaries.

MUPDATE()
Syntax: MUPDATE() -> NIL
Description: This function is used to refresh the mouse cursor. Use this function to avoid the flashing of the mouse cursor in your applications.
Example:
```
? "Press Alt-C to break this loop"
DO WHILE .T.
        MUPDATE()
ENDDO
```

ORDNUMBER(<cOrderName>)
If <cOrderName> is not found, ORDNUMBER() does not raise a recoverable runtime error. Instead, ORDNUMBER() will return a value of zero.

PushButton Class
ColorSpec - The ColorSpec instance variable for PushButtons must consist of either 4 or 5 colors. The 5th ColorSpec is used as the text color in graphics mode only and is ignored in text mode.

RESTSCREEN()
In graphics mode, the 5th parameter is an array, not a character string. In text mode, it remains a character string.

SAVESCREEN()
In graphics mode, the return value of SAVESCREEN() is an array, not a character string. In text mode, it remains a character string.

SETCOLOR()
The 4th parameter is used as the hotkey color for all @ GET controls.

SETMODE()
When using the SETMODE() function, the mouse boundaries set by MSETBOUNDS() are automatically reset to the entire screen. After using the SETMODE() function, you must call the MSETBOUNDS() function to reset the mouse boundaries.

TBrowse Class
The return value of the TBrowse Class exported method hitTest() should be <*nHitStatus*> instead of *self*.

API Changes

The following is an update to the CA-Clipper Technical Reference Guide. The following have been updated since the written documentation went to press. Please see the text below for the most recent updates.

_mUpdate()
Syntax: void _mUpdate(void);

Description: This function is used to refresh the mouse cursor. Use this function to avoid the flashing of the mouse cursor in your applications.

LLG_VIDEO_TXT
The LLG_VIDEO_TXT constant is incorrectly spelled as LLG_VIDEO_TEXT in the CA-Clipper Reference Guide. For a list of all LLG constants, please see the LLIGB.CH include file in the CLIP53\INCLUDE directory.

LLIBG.LIB
In the CA-Clipper Reference Guide and the CA-Clipper Technical Reference Guide, the LLIBG.LIB library is shown as being part of the default libraries. This library is NOT part of the default libraries and must be added to your Exospace link script in order to use graphics mode with CA-Clipper.

Linker Templates

A link template provides the CA-Clipper Workbench with information necessary to invoke a utility other than Exospace, the default linker. The link template is used, along with application and module information, to construct a link batch file and a link script file. Application and module data are available via symbols which may be used in the link template. Symbols are replaced by actual application or module data during the generation of the link batch and link script files.

Link template sections

Batch Section

The batch section of the link template, delimited by template commands #batch/#endbatch, controls the generation of a batch file which is executed under COMMAND.COM.

Script Section

Unconditional Script Section(s) - An undelimited set template and script commands which are processed unconditionally.

Conditional Script Section(s) - A set of template and script commands, delimited by #if/#endif, which are processed conditionally.

Generated Script Section(s) - A set of template and script commands, delimited by #gen/#endgen, which permits a list to be expanded into one or more script commands.

Template symbols are predefined identifiers used to represent application- and module-specific data in the generated batch or script files. Symbols may represent string, string list, numeric or Boolean data. Although the following symbols are shown in mixed case for clarity, interpretation of symbols is not case sensitive.

String symbols

__ScriptFile	Link script file specification
__LogFile	Log file specification
__OutFile	Output file specification
__MapFile	Map file specification
__MapSort	Map sort specification ("A" or "N")
__ObjMain	Application main module object file
__ClipperName	Alternate name for the CLIPPER environment variable
__ClipperEnv	Default value of the CLIPPER environment variable

String list symbols

__Obj	Other application object files
__ObjNew	Objects (except main) created in current build
__ObjExtra	Additional object files declared via Link Dialog
__Lib	Libraries declared via Link Dialog
__Packages	List of selected ExoSpace packages
__Mod	Object, Library pairs from Link Dialog <obj> FROM <lib> list. O1, L1, O2, L2, ...

Numeric symbols

__StackSize	CA-Clipper Runtime stack allocation
__ProcDepth	CA-Clipper Runtime stack allowance

Boolean symbols (numeric symbols with value 0 or 1):

__ClipperTools	Include CA-Clipper Tools
__NoDefLibs	Ignore default library declarations
__ClipperOver	Allow runtime CLIPPER environment to override default
__IgnoreError	If possible, produce output in spite of errors
__Map	Create link map
__MapSegs	Include segments with assigned addresses

Environment variables: Any environment variable may be used as a string symbol. For example, the word PATH can be used as a string symbol in the template file.

Runtime symbols: A symbol may be created by assignment and augmented by reflexive assignment. Runtime symbols may be string or string-list symbols.

Creation by assignment examples (a string or string-list is created)

```
#assign MySymbol   = "alpha"
#assign MySymbol   = "$(CLIPPER)"
#assign MySymbol   = __LogFile
#assign MySymbol   = __ObjExtra
```

Creation and augmentation example (A string list is created)

```
#assign MyObjs     = __ObjMain
#assign MyObjs    += __Obj
#assign MyObjs    += __ObjExtra
```

Diagnostic symbols: The symbols __Warning and __Error are set by #warning and #error commands. In addition, certain syntactic errors in the template cause these symbols to be set.

Symbol usage

In the following, _Str, _Lst and _Num represent string, string-list and numeric symbols, respectively:

Values:

$(_Str)	Replaced by the value of the symbol (null if undefined)
$(_Lst)	Outside a generated script section, replaced by a delimited list. The separator is provided by a #sep command, the default being ", ".
$(_Lst)	Inside a generated script section, replaced by a list element.
$(_Num)	Replaced by the decimal ASCII value of the symbol

#if operand:

_Str	True if not null, otherwise false
_Lst	True if not empty, otherwise false
_Num	True if greater than zero, otherwise false

File specification components (_Str is a symbol. The value of which is a file specification):

$D(_Str)	Drive + directory of $(_Str)
$B(_Str)	Base filename of $(_Str)
$F(_Str)	Base filename + extension of $(_Str)
$R(_Str)	Drive + directory + base filename of $(_Str)

Characters such as $, above; " in a quoted string; etc. can be "escaped". The default escape character is ^, however, the #escape command will assign an alternate character. e.g. ^$, ^", ^^. Escaped characters will be processed as normal characters. Within quoted strings, carriage return and new line (ASCII line feed) may be symbolized as ^r and ^n, respectively.

Template Commands

Template commands provide information, delimit template sections and specify conditional processing. The first character of a template command must be the pound (#) character or the character specified by the #template command.

#template <c> - This command is used when the syntax of the output script file permits # to be the first character of a script command. <c> will be recognized instead of # in subsequent template commands.

#escape <c> - <c> replaces the default escape character, ^. The characters \ and # are not allowed.

#comment "<string>" - This command declares the character or string which introduces a comment line. If #comment appears, the template commands and spacing will be passed to the output script file as comments.

#message <string> - Pass MESSAGE message to log file.

#warning <string> - Pass WARNING message to log file - set __Warning symbol

#error <string> - Pass ERROR message to log file - set __Error symbol

#write "<string>" - This command provides a string which is passed to the output script file without any interpretation. It is not automatically terminated with a new line. The escape sequence, \n, inside the string is interpreted as a new line character.

#sep "<string>" - This command, outside of a generated script section, supplies the separator used when a string list value is produced via $(<symbol>).

#batch - Begins the link batch section

#endbatch - Terminates the link batch section

#assign - This command identifies a template assignment of string assignment.

#if <symbol> - Begins a link script conditional section which is processed if <symbol> is "true", see above. The conditional section may be empty which allows the following to be used as an #ifnot construct:

```
#if <symbol>
#else
        script command
        ...
        script command
#endif
```

#else - Terminates a link script conditional section introduced by #if and begins a conditional section which is processed if the operand of the #if condition was NOT true.

#endif - Terminates a link script conditional section.

#gen <symbol> - Begins a Generated script section. <symbol> must represent a string list. The section is processed if the list is not empty. In the body of the script section, $(symbol) is expanded into a series of elements from the members of <symbol>. A sufficient number of script lines will be generated to exhaust the list.

#max <n> - This command specifies the maximum length allowed for a generated script command. Default value is 80. If insertion of a single list item plus any #more (see below) string exceeds the maximum length, the string is not truncated but written to the output file.

#sep "<string>" - This command, inside a generated script section, implies that multiple list elements separated by the specified string may occur on a single script line. If addition of another list element would cause a script line (with a continuation string) to exceed the maximum length, a new script line is started.

#more "<string>" - This command, permitted only in a generated script section, specifies a string to be added to a script line when data is to be continued on a new line.

#endgen - Terminates a generated script section.

Error Messages

The following error messages were omitted from the CA-Clipper Error Guide:

BASE/1022 - ALLTRIM() Argument Error
Explanation: A character parameter was not passed to the ALLTRIM() function
Action: Provide a character variable as the parameter to ALLTRIM().

DBFNTX/1019 - Workarea not indexed
Explanation: The workarea does not have a currently active index.
Action: Use DBCREATEINDEX() or the INDEX ON command to create an index for this workarea.
Note: This error is the same for DBFCDX, DBFMDX, DBFNDX, and all other RDDs.

Reserved Words

The following words are reserved by CA-Clipper and should be added to the Reserved Word list in the documentation:

PUBLIC
SYSINIT
TEXTB - abbreviation for TEXTBLOCK

In addition, please see the file RESERVED.CH in the CLIP53\INCLUDE directory for a complete list of the new reserved words for CA-Clipper 5.3.

Workbench User Guide Additions

The following is an update to the CA-Clipper Workbench User Guide.

DBServer Editor

A CDX file which contains memo fields (files with an extension of .FPT) cannot be loaded into the DBServer Editor.