

[Harbour Reference Guide](#)

[Index Harbour core](#)

[Clipper 5.3Clipper Tools 3](#)

[gtwwwbcthbcupshbgdhhbgthbmischbnfhbxpphziparcrrddads](#)

[Harbour core indexClipper 5.3 indexClipper Tools 3 indexgtwww](#)

[indexhbct indexhbcups indexhbgd indexhbgd indexhbmisc indexhbnf](#)

[indexhbxpp indexhbziparc indexrddads index](#)

Harbour core

Document

[Welcome to Harbour](#) A starter's guide

Compiler

[Compiler Options](#)

[Macro compiler](#)

[Harbour Extensions](#)

[The Garbage Collector](#) Readme for Harbour Garbage Collect Feature

[The idle states](#) Readme file for Idle States

API

Application

[Do\(\)](#) Calls a procedure or a function

`hb_PValue()` Retrieves the value of an argument.

`PCount()` Retrieves the number of arguments passed to a function.

`ProcFile()` This function always returns an empty string.

`ProcLine()` Gets the line number of the current function on the stack.

`ProcName()` Gets the name of the current function on the stack

Array

`AAdd()` Dynamically add an element to an array

`AClone()` Duplicate a multidimensional array

`ACopy()` Copy elements from one array to another

`ADel()` Delete an element from an array.

`AEval()` Evaluates the subscript element of an array

`AFill()` Fill an array with a specified value

`AIns()` Insert a NIL value at an array subscript position.

`Array()` Create an uninitialized array of specified length

`AScan()` Scan array elements for a specified condition

`ASize()` Adjust the size of an array

`ASort()` Sort an array

`ATail()` Returns the last element of an array

`hb_ADel()` Delete an element from an array.

`hb_AIns()` Inserts a value at an array subscript position and optionally increases the length of array.

`hb_AScan()` Scan array elements for a specified condition

Classes

`HBClass()` `HBClass()` is used in the creation of all classes

Conversion

[Bin2I\(\)](#) Convert signed short encoded bytes into Harbour numeric
[Bin2L\(\)](#) Convert signed long encoded bytes into Harbour numeric
[Bin2W\(\)](#) Convert unsigned short encoded bytes into Harbour numeric
[Descend\(\)](#) Inverts an expression of string, logical, date or numeric type.
[I2Bin\(\)](#) Convert Harbour numeric into signed short encoded bytes
[L2Bin\(\)](#) Convert Harbour numeric into signed long encoded bytes
[Word\(\)](#) Converts double to integer values.

Database

[__dbCopyStruct\(\)](#) Create a new database based on current database structure
[__dbCopyXStruct\(\)](#) Copy current database structure into a definition file
[__dbCreate\(\)](#) Create structure extended file or use one to create new file
[__dbDelim\(\)](#) Copies the contents of a database to a delimited text file or appends the contents of a delimited text file to a database.
[__dbSDF\(\)](#) Copies the contents of a database to an SDF text file or appends the contents of an SDF text file to a database.
[__dbStructFilter\(\)](#) Filter a database structure array
[__FLedit\(\)*](#) Filter a database structure array
[AFields\(\)*](#) Fills referenced arrays with database field information
[Alias\(\)](#) Returns the alias name of a work area
[Bof\(\)](#) Test for the beginning-of-file condition
[dbAppend\(\)](#) Appends a new record to a database file.
[dbClearFilter\(\)](#) Clears the current filter condition in a work area
[dbCloseAll\(\)](#) Close all open files in all work areas.
[dbCloseArea\(\)](#) Close a database file in a work area.

`dbCommit()` Updates all index and database buffers for a given work area

`dbCommitAll()` Flushes the memory buffer and performs a hard-disk write

`dbCreate()` Creates an empty database from a array.

`dbDelete()` Mark a record for deletion in a database.

`dbFilter()` Return the filter expression in a work area

`dbGoBottom()` Moves the record pointer to the bottom of the database.

`dbGoto()` Position the record pointer to a specific location.

`dbGoTop()` Moves the record pointer to the top of the database.

`dbRecall()` Recalls a record previously marked for deletion.

`dbRLock()` This function locks the record based on identity

`dbRLockList()` This function return a list of locked records in the database work area

`dbRUnlock()` Unlocks a record based on its identifier

`dbSeek()` Searches for a value based on an active index.

`dbSelectArea()` Change to another work area

`dbSetDriver()` Establishes the RDD name for the selected work area

`dbSetFilter()` Establishes a filter condition for a work area.

`dbSkip()` Moves the record pointer in the selected work area.

`dbStruct()` Creates a multidimensional array of a database structure.

`dbUnlock()` Unlock a record or release a file lock

`dbUnlockAll()` Unlocks all records and releases all file locks in all work areas.

`dbUseArea()` Opens a work area and uses a database file.

Dbf() Alias name of a work area

Deleted() Tests the record's deletion flag.

Eof() Test for end-of-file condition.

FCount() Counts the number of fields in an active database.

FieldGet() Obtains the value of a specified field

FieldName() Return the name of a field at a numeric field location.

FieldPos() Return the ordinal position of a field.

FieldPut() Set the value of a field variable

FLock() Locks a file

Found() Determine the success of a previous search operation.

Header() Return the length of a database file header

IndexExt() Returns the file extension of the index module used in an application

IndexKey() Yields the key expression of a specified index file.

IndexOrd() Returns the numeric position of the controlling index.

LastRec() Returns the number of records in an active work area or database.

LUpdate() Yields the date the database was last updated.

NetErr() Tests the success of a network function

ordBagExt() Returns the Order Bag extension

ordBagName() Returns the Order Bag Name.

ordCondSet() Set the Condition and scope for an order

ordCreate() Create an Order in an Order Bag

ordDestroy() Remove an Order from an Order Bag

ordFor() Return the FOR expression of an Order

`ordKey()` Return the key expression of an Order

`RecCount()` Counts the number of records in a database.

`RecNo()` Returns the current record number or identity.

`RecSize()` Returns the size of a single record in an active database.

`RLock()` Lock a record in a work area

`Select()` Returns the work area number for a specified alias.

`Used()` Checks whether a database is in use in a work area

Date/Time

`CDoW()` Converts a date to the day of week

`CMonth()` Return the name of the month.

`CToD()` Converts a character string to a date expression

`Date()` Return the Current OS Date

`Day()` Return the numeric day of the month.

`Days()` Convert elapsed seconds into days

`DoW()` Value for the day of week.

`DToC()` Date to character conversion

`DToS()` Date to string conversion

`ElapTime()` Calculates elapsed time.

`hb_DToT()` Create a `tDateTime` value from a `dDate` parameter

`hb_Week()` Returns the week number of year.

`Month()` Converts a date expression to a month value

`Seconds()` Returns the number of elapsed seconds past midnight.

`Secs()` Return the number of seconds from the system date.

`Time()` Returns the system time as a string

`Year()` Extracts the year designator of a given date as a numeric value

Environment

[__Run\(\)](#) Run an external program.

[__SetCentury\(\)](#) Set the Current Century

[GetE\(\)](#) Obtains a system environmental setting.

[GetEnv\(\)](#) Obtains a system environmental setting.

[hb_eol\(\)](#) Returns the newline character(s) to use with the current OS

[hb_GetEnv\(\)](#) Obtains a system environmental setting.

[OS\(\)](#) Return the current operating system.

[RUN](#) Run an external program.

[Set\(\)](#) Changes or evaluated environmental settings

[SetMode\(\)](#) Change the video mode to a specified number of rows and columns

[SetTypeahead\(\)](#) Sets the typeahead buffer to given size.

[Tone\(\)](#) Sound a tone with a specified frequency and duration.

[Version\(\)](#) Returns the version of Harbour compiler

Error

[Break\(\)](#) Exits from a BEGIN SEQUENCE block

[ErrorSys\(\)](#) Install default error handler

Events

[__Quit\(\)](#) Terminates an application.

[__SetFunction\(\)](#) Assign a character string to a function key

[__Wait\(\)](#) Stops the application until a key is pressed.

[hb_SetKeyCheck\(\)](#) Implements common hot-key activation code

[hb_SetKeyGet\(\)](#) Determine a set-key code block and condition-block

`hb_SetKeySave()` Returns a copy of internal set-key list, optionally overwriting

`SetKey()` Assign an action block to a key

Execute and Execution

`dbEval()` Performs a code block operation on the current Database

`Eval()` Evaluate a code block

FileSys

`__Dir()*` Display listings of files

`ADir()` Fill pre-defined arrays with file/directory information

`CurDir()` Returns the current OS directory name.

`DirChange()` Changes the directory

`DirRemove()` Attempt to remove an directory

`DiskSpace()` Get the amount of space available on a disk

`FClose()` Closes an open file

`FCreate()` Creates a file.

`FErase()` Erase a file from disk

`FError()` Reports the error status of low-level file functions

`File()` Tests for the existence of file(s)

`FOpen()` Open a file.

`FRead()` Reads a specified number of bytes from a file.

`FReadStr()` Reads a string from a file.

`FRename()` Renames a file

`FSeek()` Positions the file pointer in a file.

`FWrite()` Writes characters to a file.

`hb_DiskSpace()` Get the amount of space available on a disk

`hb_FEOF()` Check for end-of-file.

`hb_FLock()` Locks part or all of any file

`hb_FUnlock()` Unlocks part or all of any file

`IsDisk()` Verify if a drive is ready

`MakeDir()` Create a new directory

Garbage Collector

`hb_gcAll()` Scans the memory and releases all garbage memory blocks.

`hb_gcAlloc()` Allocates memory that will be collected by the garbage collector.

`hb_gcCollectAll()` Scans all memory blocks and releases the garbage memory.

`hb_gcFree()` Releases the memory that was allocated with `hb_gcAlloc()`.

`hb_gcItemRef()` Marks the memory to prevent deallocation by the garbage collector.

Hash table

`hb_Hash()` Returns a hash table

`hb_HAlloc()` Preallocates a hash table

`hb_HAutoAdd()` Sets the 'auto add' flag for the hash table

`hb_HBinary()` Sets the 'binary' flag for the hash table

`hb_HCaseMatch()` Sets the 'case match' flag for the hash table

`hb_HClone()` Creates a copy of a hash table

`hb_HCopy()` Adds entries from the source hash table to the destination hash table

`hb_HDefault()` Returns/sets a default value for a hash table.

`hb_HDel()` Removes a key/value pair from a hash table

`hb_HDelAt()` Removes an entry from a hash table based on its index position

`hb_HEval()` Evaluate a code block across the contents of a hash table

`hb_HFill()` Fills a hash table with a value

`hb_HGet()` Returns a hash value

`hb_HGetDef()` Returns a hash value, or a default value if the key is not present

`hb_HHasKey()` Determines whether a hash table has an entry with a give key

`hb_HKeyAt()` Gets a hash table key at a given position

`hb_HKeys()` Returns an array of the keys of a hash table

`hb_HMerge()` Merges a source hash table into a destination hash table

`hb_HPairAt()` Returns a two-dimensional array of a hash table entry key/value pair

`hb_HPos()` Locates the index of a key within a hash table

`hb_HScan()` Scans a hash table

`hb_HSet()` Sets a hash value

`hb_HSort()` Reorganizes the internal list of the hash table to be sorted

`hb_HValueAt()` Gets/sets a hash value at a given position

`hb_HValues()` Returns an array of the values of a hash table

Idle states

`hb_idleAdd()` Adds the background task.

`hb_idleDel()` Removes the background task from the list of tasks.

`hb_idleState()` Evaluates a single background task and calls the garbage collector.

Internal

[__mvDbgInfo\(\)](#) This function returns the information about the variables for debugger

[__SetHelpK\(\)](#) Set F1 as the default help key

[__TextRestore\(\)](#) Restore console output settings as saved by [__TextSave\(\)](#)

[__TextSave\(\)](#) Redirect console output to printer or file and save old settings

[__XHelp\(\)](#) Determines whether a HELP() user defined function exists.

[CLIPINIT\(\)](#) Initialize various Harbour sub-systems

INET

[hb_inetAccept\(\)](#) Wait until a socket is ready

[hb_inetAddress\(\)](#) Get a remote server address

[hb_inetCleanup\(\)](#) Terminate Harbour INET support

[hb_inetClearError\(\)](#) Clear the socket error value

[hb_inetClearPeriodCallback\(\)](#) Clear the periodic callback value of a socket

[hb_inetClearTimeLimit\(\)](#) Clear the time limit value of a socket

[hb_inetClearTimeout\(\)](#) Clear the timeout value of a socket

[hb_inetClose\(\)](#) Close an INET socket

[hb_inetConnect\(\)](#) Connect a socket to a remote server by IP address or name

[hb_inetConnectIP\(\)](#) Connect to a remote server by IP address

[hb_inetCount\(\)](#) Get the number of bytes last read or sent

[hb_inetCreate\(\)](#) Create an INET socket

[hb_inetCRLF\(\)](#) Get a CRLF sequence for internet protocols

`hb_inetDataReady()` Get whether there is data ready in a socket

`hb_inetDGram()` Create a datagram socket

`hb_inetDGramBind()` Create a bound datagram socket

`hb_inetDGramRecv()` Get data from a datagram socket

`hb_inetDGramSend()` Send data to a datagram socket

`hb_inetErrorCode()` Get the last INET error code

`hb_inetErrorDesc()` Get the last INET error code description

`hb_inetFD()` ?

`hb_inetGetAlias()` Get an array of aliases of a server

`hb_inetGetHosts()` Get an array of IP addresses of a host

`hb_inetGetRcvBufSize()` Get the socket receive buffer size

`hb_inetGetSndBufSize()` Get the socket send buffer size

`hb_inetInit()` Activate Harbour INET support

`hb_inetIsSocket()` Get whether a variable is a socket

`hb_inetPeriodCallback()` Get or change the periodic callback value of a socket

`hb_inetPort()` Get the port a socket is bound to.

`hb_inetRecv()` Read from a socket

`hb_inetRecvAll()` Read from a socket without blocking

`hb_inetRecvEndblock()` Read a block from a socket

`hb_inetRecvLine()` Read a line from a socket

`hb_inetstatus()` Get the status of a socket

`hb_inetSend()` Sent data through a socket

`hb_inetSendAll()` Send data through a socket with blocking

`hb_inetServer()` Create a socket bound to a port

[hb_inetSetRcvBufSize\(\)](#) Set the receive buffer size of a socket

[hb_inetSetSndBufSize\(\)](#) Set the send buffer size of a socket

[hb_inetTimeLimit\(\)](#) Get or change the time limit value of a socket

[hb_inetTimeout\(\)](#) Get or change the timeout value of a socket

Language and Nation

[hb_cdpSelect\(\)](#) Select the active code page by language ID

[hb_langErrMsg\(\)](#) Description of an error code using current language

[hb_langMessage\(\)](#) Returns international strings messages and errors

[hb_langName\(\)](#) Return the name of the language module

[hb_langSelect\(\)](#) Select a specific nation message module

[hb_Translate\(\)](#) Translate a string from one code page to the other

[IsAffirm\(\)](#) Checks if passed char is an affirmation char

[IsNegative\(\)](#) Checks if passed char is a negation char.

[NationMsg\(\)](#) Returns international strings messages.

Macro

[hb_SetMacro\(\)](#) Enable/disable the macro compiler runtime features.

Math

[Abs\(\)](#) Return the absolute value of a number.

[Exp\(\)](#) Calculates the value of e raised to the passed power.

[hb_matherBlock\(\)](#) Set/Get math error handling codeblock

[hb_matherMode\(\)](#) Set/Get math error handling mode

[Int\(\)](#) Return the integer port of a numeric value.

[Log\(\)](#) Returns the natural logarithm of a number.

[Max\(\)](#) Returns the maximum of two numbers or dates.

[Min\(\)](#) Determines the minimum of two numbers or dates.

`Mod()` Return the modulus of two numbers.

`Round()` Rounds off a numeric expression.

`Sqrt()` Calculates the square root of a number.

Objects

`__objAddData()` Add a VAR to an already existing class

`__objAddInline()` Add an INLINE to an already existing class

`__objAddMethod()` Add a METHOD to an already existing class

`__objDelData()` Delete a VAR (instance variable) from class

`__objDelInline()` Delete a METHOD INLINE from class

`__objDelMethod()` Delete a METHOD from class

`__objDerivedFrom()` Determine whether a class is derived from another class

`__objGetMethodList()` Return names of all METHOD for a given object

`__objGetMsgList()` Return names of all VAR or METHOD for a given object

`__objGetValueList()` Return an array of VAR names and values for a given object

`__objHasData()` Determine whether a symbol exist in object as VAR

`__objHasMethod()` Determine whether a symbol exist in object as METHOD

`__objModInline()` Modify (replace) an INLINE method in an already existing class

`__objModMethod()` Modify (replace) a METHOD in an already existing class

`__objSetValueList()` Set object with an array of VAR names and values

RDD

FieldBlock() Return a code block that sets/gets a value for a given field

FieldWBlock() Return a sets/gets code block for field in a given work area

Strings

AllTrim() Removes leading and trailing blank spaces from a string

Asc() Returns the ASCII value of a character

At() Locates the position of a substring in a main string.

Chr() Converts an ASCII value to its character value

hb_At() Locates the position of a substring in a main string.

hb_AtI() Locates the position of a substring in a main string.

hb_ATokens() Parses a complex string (e.g. a sentence or multi-line text) into individual tokens (words or other string chunks depending on delimiter used).

hb_LeftEq() Checks if a sub-string matches to leftmost part of a string.

hb_LeftEqI() Checks if a sub-string matches to leftmost part of a string.

hb_MemoRead() Return the text file's contents as a character string

hb_MemoWrit() Write a memo field or character string to a text file on disk

hb_ntoc() Converts a numeric value to string

hb_ntos() Converts a numeric value to string.

hb_RAt() Searches for last occurrence a substring of a string.

hb_ValToStr() Converts any scalar type to a string.

HardCR() Replace all soft carriage returns with hard carriage returns.

IsAlpha() Checks if leftmost character in a string is an alphabetic character

IsDigit() Checks if leftmost character is a digit character

IsLower() Checks if leftmost character is an lowercased letter.

IsUpper() Checks if leftmost character is an uppercased letter.

Left() Extract the leftmost substring of a character expression

Lower() Universally lowercases a character string expression.

LTrim() Removes leading spaces from a string

MemoRead() Return the text file's contents as a character string

MemoTran() Converts hard and soft carriage returns within strings.

MemoWrit() Write a memo field or character string to a text file on disk

PadC() Centers an expression for a given width

PadL() Left-justifies an expression for a given width

PadR() Right-justifies an expression for a given width

RAt() Searches for last occurrence a substring of a string.

Replicate() Repeats a single character expression

Right() Extract the rightmost substring of a character expression

RTrim() Remove trailing spaces from a string.

Space() Returns a string of blank spaces

Str() Convert a numeric expression to a character string.

StrTran() Translate substring value with a main string

StrZero() Convert a numeric expression to a character string, zero padded.

SubStr() Returns a substring from a main string

Transform() Formats a value based on a specific picture template.

`Trim()` Remove trailing spaces from a string.

`Upper()` Converts a character expression to uppercase format

`Val()` Convert a number from a character type to numeric

Terminal

`__TypeFile()` Show the content of a file on the console and/or printer

`Col()` Returns the current screen column position

`DevOutPict()` Displays a value to a device using a picture template

`hb_ColorIndex()` Extract one color from a full color-spec string.

`MaxCol()` Returns the maximum number of columns in the current video mode

`MaxRow()` Returns the current screen row position

`RESTORE SCREEN` Restore screen image and coordinate from an internal buffer

`Row()` Returns the current screen row position

`SAVE SCREEN` Save whole screen image and coordinate to an internal buffer

User interface

`__AtPrompt()` Display a menu item on screen and define a message

`__Input()` Stops application

`__Keyboard()` Use `hb_keyPut()` instead

`__MenuTo()` Invoked a menu defined by set of `@...PROMPT`

`__NoNoAlert()` Override `//NOALERT` command-line switch

`__XRestScreen()` Restore screen image and coordinate from an internal buffer

`__XSaveScreen()` Save whole screen image and coordinate to an internal buffer

`AChoice()` Allows selection of an element from an array

`Alert()` Display a dialog box with a message

`Browse()` Browse a database file

`dbEdit()*` Browse records in a table

`hb_keyPut()` Put an inkey code to the keyboard buffer.

`Inkey()` Extracts the next key code from the Harbour keyboard buffer.

`LastKey()` Get the last key extracted from the keyboard buffer.

`MCol()` Returns the mouse cursor column position.

`MENU TO` Invoked a menu defined by set of @. . . PROMPT

`MRow()` Returns the mouse cursor row position.

`NextKey()` Get the next key code in the buffer without extracting it.

`OutErr()` Write a list of values to the standard error device

`OutStd()` Write a list of values to the standard output device

`ReadKey()*` Determine which key terminated a READ.

`ReadVar()` Return variable name of current GET or MENU

`TBrowseDB()` Create a new TBrowse object to be used with database file

Variable management

`__mvClear()` This function releases all PRIVATE and PUBLIC variables

`__mvExist()` Determine if a given name is a PUBLIC or PRIVATE memory variable

`__mvGet()` This function returns value of memory variable

`__mvPrivate()` This function creates a PRIVATE variable

`__mvPublic()` This function creates a PUBLIC variable

[__mvPut\(\)](#) This function set the value of memory variable

[__mvRelease\(\)](#) This function releases PRIVATE variables

[__mvScope\(\)](#) If variable exists then returns its scope.

[__mvXRelease\(\)](#) This function releases value stored in PRIVATE or PUBLIC variable

[Empty\(\)](#) Checks if the passed argument is empty.

[hb_PlsByRef\(\)](#) Determine if a parameter is passed by reference.

[Len\(\)](#) Returns size of a string or size of an array.

[MemVarBlock\(\)](#) Returns a codeblock that sets/gets a value of memvar variable

[Type\(\)](#) Retrieves the type of an expression

[ValType\(\)](#) Retrieves the data type of an expression

C level API

Environment

[hb_setListenerAdd\(\)](#)

[hb_setListenerNotify\(\)](#)

[hb_setListenerRemove\(\)](#)

Idle states

[hb_idleState\(\)](#) Evaluates a single background task and calls the garbage collector.

Math

[hb_mathGetErrMode\(\)](#) get math error handling mode

[hb_mathGetHandler\(\)](#) get current Harbour math error handler

[hb_mathGetLastError\(\)](#) get the last math lib error

`hb_mathIsMathErr()` Check if Harbour math error handling is available
`hb_mathResetError()` Reset the internal math error information structure
`hb_mathSetErrMode()` set math error handling mode
`hb_mathSetHandler()` set the Harbour math handler

Class

Data

`CLASS VAR` Define a CLASS VAR variable for a class (NOT for an Object!)
`VAR` Alternate syntax for VAR: instance variable for the objects.

Definition

`CLASS` Define a Class for Object Oriented Programming
`ENDCLASS` End the declaration of a class.

Method

`ERROR HANDLER` Designate a method as an error handler for the class
`MESSAGE` Route a method call to another Method
`METHOD` Declare a METHOD for a class in the class header
`ON ERROR` Designate a method as an error handler for the class

Command

Database

`COPY STRUCTURE` Create a new database based on current database structure
`COPY STRUCTURE EXTENDED` Copy current database structure into a definition file
`CREATE` Create empty structure extended file

CREATE FROM Create new database file from a structure extended file

PACK Remove records marked for deletion from a database

ZAP Remove all records from the current database file

Environment

SET ALTERNATE Toggle and echos output to an alternate file

SET BELL Toggle the bell to sound once a GET has been completed.

SET CENTURY Toggle the century digits in all dates display

SET CONSOLE Toggle the console display

SET DATE Assigns a date format or chooses a predefined date data set.

SET DECIMALS Toggle the console display

SET DEFAULT Establishes the Harbour search drive and directory.

SET DEVICE Directs all @...SAY output to a device.

SET EPOCH Specify a base year for interpreting dates

SET FIXED Set the number of decimal position to be displayed

SET FUNCTION Assign a character string to a function key

SET INTENSITY Toggles the enhanced display of PROMPTs and GETs.

SET KEY Assign an action block to a key

SET MESSAGE Establishes a message row for @...PROMPT command

SET PATH Specifies a search path for opening files

SET PRINTER Toggles the printer and controls the printer device

SET WRAP Toggle wrapping the PROMPTs in a menu.

FileSys

COPY FILE Copies a file.

DELETE FILE Remove a file from disk

DIR Display listings of files

ERASE Remove a file from disk

RENAME Changes the name of a specified file

TYPE Show the content of a file on the console, printer or file

Legacy

LABEL FORM Displays labels to the screen or an alternate device

REPORT FORM Display a report

Printer

EJECT Issue an command to advance the printer to the top of the form

User interface

@...GET Creates a GET object and displays it to the screen

@...PROMPT Display a menu item on screen and define a message

@...SAY Displays data at specified coordinates of the current device.

KEYBOARD Stuffs the keyboard with a string.

Runtime errors

RTE BASE/1003 Attempt to access non-existing or hidden variable

RTE BASE/1068 Bound error in array access

RTE BASE/1069 Bound error in array access

RTE BASE/1070 Invalid type of arguments

RTE BASE/1072 Invalid type of arguments

RTE BASE/1073 Invalid type of arguments

RTE BASE/1074 Invalid type of arguments

RTE BASE/1075 Invalid type of arguments

RTE BASE/1076 Invalid type of arguments

RTE BASE/1077	Invalid type of arguments
RTE BASE/1078	Invalid type of arguments
RTE BASE/1079	Invalid type of arguments
RTE BASE/1081	Invalid type of arguments
RTE BASE/1082	Invalid type of arguments
RTE BASE/1085	Invalid argument passed to function
RTE BASE/1086	Invalid type of arguments
RTE BASE/1089	Invalid argument passed to function
RTE BASE/1090	Invalid argument passed to function
RTE BASE/1092	Invalid argument passed to function
RTE BASE/1093	Invalid argument passed to function
RTE BASE/1094	Invalid argument passed to function
RTE BASE/1095	Invalid argument passed to function
RTE BASE/1096	Invalid argument passed to function
RTE BASE/1097	Invalid argument passed to function
RTE BASE/1098	Invalid argument passed to function
RTE BASE/1099	Invalid argument passed to function
RTE BASE/1100	Incorrect type of argument
RTE BASE/1101	Incorrect type of argument
RTE BASE/1102	Invalid argument passed to function
RTE BASE/1103	Invalid argument passed to function
RTE BASE/1104	Incorrect type of argument
RTE BASE/1105	Invalid argument passed to function
RTE BASE/1106	Invalid argument passed to function
RTE BASE/1107	Incorrect type of argument

RTE BASE/1108	Incorrect type of argument
RTE BASE/1109	Invalid type of arguments
RTE BASE/1110	Invalid argument passed to function
RTE BASE/1111	Invalid argument passed to function
RTE BASE/1112	Invalid argument passed to function
RTE BASE/1113	Invalid argument passed to function
RTE BASE/1114	Invalid argument passed to function
RTE BASE/1115	Invalid argument passed to function
RTE BASE/1116	Invalid argument passed to function
RTE BASE/1117	Invalid argument passed to function
RTE BASE/1120	Invalid argument passed to function
RTE BASE/1122	Incorrect type of argument
RTE BASE/1124	Incorrect type of argument
RTE BASE/1126	Invalid argument passed to function
RTE BASE/1132	Bound error in array access
RTE BASE/1133	Bound error in array element assignment
RTE BASE/2010	Incorrect arguments type
RTE BASE/2012	File error
RTE BASE/2017	Invalid argument passed to a function
RTE BASE/2020	Invalid argument passed to function
RTE BASE/3001	Incorrect argument type
RTE BASE/3002	Super class does not return an object
RTE BASE/3003	Cannot find super class
RTE BASE/3004	Cannot modify a DATA item in a class
RTE BASE/3005	Incorrect arguments type

- [RTE BASE/3007](#) Invalid type of argument
- [RTE BASE/3008](#) Invalid type of argument
- [RTE BASE/3009](#) Incorrect argument passed to __mvGet() function
- [RTE BASE/3010](#) Incorrect argument passed to __mvPut() function
- [RTE BASE/3012](#) Invalid argument passed to a function
- [RTE BASE/3101](#) Invalid argument passed to an object/class function
- [RTE BASE/3102](#) A symbol should be modified or deleted from a class, but the symbol doesn't exist.
- [RTE BASE/3103](#) A symbol should be added to a class, but the symbol already exists.
- [RTE TERM/2013](#) Create error

Statement

RDD

[FIELD](#) Declares a list of database field names.

Variable management

[LOCAL](#) Initializes a local memory variable or array

[MEMVAR](#) Declares private and public variables and arrays.

TBrowse class

[TBrowseNew\(\)](#) Create a Browse Object

TBrowse Method

[TBrowse\(\):AddColumn\(\)](#) Add an New Column to an TBrowse Object

`TBrowse():ApplyKey()` Evaluates an code block associated with an specific key

`TBrowse():SetKey()` Get an optionally Set an new Code block associated to a inkey value

Clipper 5.3

`<` Less than—binary (Relational)

`<=` Less than or equal—binary (Relational)

`<> != #` Not equal—binary (Relational)

`= (assign)` Simple assign—binary (Assignment)

`= (compound assign)` Compound assignment—binary (Assignment)

`= (equality)` Equal—binary (Relational)

`==` Exactly equal—binary (Relational)

`>` Greater than—binary (Relational)

`>=` Greater than or equal—binary (Relational)

`-` Subtraction, unary negative, concatenation (Math, Character)

`->` Alias assignment—binary (Special)

`--` Decrement—unary (Mathematical)

`:` Send—binary (Object)

`:=` Inline assign—binary (Assignment)

`?|??` Display one or more values to the console

`/` Division—binary (Mathematical)

`.AND.` Logical AND—binary (Logical)

`.NOT.` Logical NOT—unary (Logical)

`.OR.` Logical OR—binary (Logical)

- () Function or grouping indicator (Special)
- [] Array element indicator (Special)
- { } Literal array and code block delimiters (Special)
- @ Pass-by-reference—unary (Special)
- @...BOX Draw a box on the screen
- @...CLEAR Clear a rectangular region of the screen
- @...GET Create a new Get object and display it to the screen
- @...GET CHECKBOX Create a new check box Get object and display it to the screen
- @...GET LISTBOX Create a new list box Get object and display it to the screen
- @...GET PUSHBUTTON Create a new push button Get object and display it to the screen
- @...GET RADIOGROUP Create a new radio button group Get object and display it to the screen
- @...GET TBROWSE Create a new TBrowse Get object and display it to the screen
- @...PROMPT Paint a menu item and define a message
- @...SAY Display data at a specified screen or printer row and column
- @...TO Draw a single- or double-line box
- \$ Substring comparison—binary (Relational)
- * Multiplication—binary (Mathematical)
- ** Exponentiation—binary (Mathematical)
- & Macro evaluation—unary (Special)

`#command` | `#translate` Specify a user-defined command or translation directive

`#define` Define a manifest constant or pseudofunction

`#error` Generate a compiler error and display a message

`#ifdef` Compile a section of code if an identifier is defined

`#ifndef` Compile a section of code if an identifier is undefined

`#include` Include a file into the current source file

`#stdout` Send literal text to the standard output device

`#undef` Remove a `#define` macro definition

`#xcommand` | `#xtranslate` Specify a user-defined command or translation directive

`%` Modulus—binary (Mathematical)

`+` Addition, unary positive, concatenation (Math, Character)

`++` Increment—unary (Mathematical)

`AAdd()` Add a new element to the end of an array

`Abs()` Return the absolute value of a numeric expression

`ACCEPT*` Place keyboard input into a memory variable

`AChoice()` Execute a pop-up menu

`AClone()` Duplicate a nested or multidimensional array

`ACopy()` Copy elements from one array to another

`ADel()` Delete an array element

`ADir()*` Fill a series of arrays with directory information

`AEval()` Execute a code block for each element in an array

`AFields()*` Fill arrays with the structure of the current database file

`AFill()` Fill an array with a specified value

Alns() Insert a NIL element into an array

Alert() Display a simple modal dialog box

Alias() Return a specified work area alias

AllTrim() Remove leading and trailing spaces from a character string

AltD() Invoke the CA-Clipper debugger

ANNOUNCE Declare a module identifier

APPEND BLANK Add a new record to the current database file

APPEND FROM Import records from a database (.dbf) file or ASCII text file

Array() Create an uninitialized array of specified length

Asc() Convert a character to its ASCII value

AScan() Scan an array for a value or until a block returns true (.T.)

ASize() Grow or shrink an array

ASort() Sort an array

At() Return the position of a substring within a character string

ATail() Return the highest numbered element of an array

AVERAGE Average numeric expressions in the current work area

BEGIN SEQUENCE Define a sequence of statements for a BREAK

Bin2I() Convert a 16-bit signed integer to a numeric value

Bin2L() Convert a 32-bit signed integer to a numeric value

Bin2W() Convert a 16-bit unsigned integer to a numeric value

BLOBDIRECTEXPORT() Export the contents of a binary large object (BLOB) pointer to a file

BLOBDIRECTGET() Retrieve data stored in a BLOB file without referencing a specific field

BLOBDIRECTIMPORT() Import a file into a BLOB file and return a pointer to the data

BLOBDIRECTPUT() Put data in a BLOB file without referencing a specific field

BLOBEXPORT() Copy the contents of a BLOB, identified by its memo field number, to a file

BLOBGET() Get the contents of a BLOB, identified by its memo field number

BLOBIMPORT() Read the contents of a file as a BLOB, identified by a memo field number

BLOBROOTGET() Retrieve the data from the root area of a BLOB file

BLOBROOTLOCK() Obtain a lock on the root area of a BLOB file

BLOBROOTPUT() Store data in the root area of a BLOB file

BLOBROOTUNLOCK() Release the lock on a BLOB file's root area

Bof() Determine when beginning of file is encountered

Break() Branch out of a BEGIN SEQUENCE...END construct

Browse()* Browse records within a window

CALL* Execute a C or Assembler procedure

CANCEL* Terminate program processing

CDoW() Convert a date value to a character day of the week

CheckBox class Create check boxes, which are controls that can be toggled on or off by a user

Chr() Convert an ASCII code to a character value

CLEAR ALL* Close files and release public and private variables

CLEAR GETS Release Get objects from the current GetList array

CLEAR MEMORY Release all public and private variables

CLEAR SCREEN Clear the screen and return the cursor home

CLEAR TYPEAHEAD Empty the keyboard buffer

CLOSE Close a specific set of files

CMonth() Convert a date to a character month name

Col() Return the screen cursor column position

ColorSelect() Activate attribute in current color settings

COMMIT Perform a solid-disk write for all active work areas

CONTINUE Resume a pending LOCATE

COPY FILE Copy a file to a new file or to a device

COPY STRUCTURE Copy the current .dbf structure to a new database (.dbf) file

COPY STRUCTURE EXTENDED Copy field definitions to a .dbf file

COPY TO Export records to a new database (.dbf) file or ASCII text file

COUNT Tally records to a variable

CREATE Create an empty structure extended (.dbf) file

CREATE FROM Create a new .dbf file from a structure extended file

CToD() Convert a date string to a date value

CurDir() Return the current DOS directory

dbAppend() Append a new record to the database open in the current work area

dbClearFilter() Clear a filter condition

dbClearIndex() Close all indexes for the current work area

dbClearRelation() Clear active relations

dbCloseAll() Close all occupied work areas

`dbCloseArea()` Close a work area

`dbCommit()` Flush pending updates

`dbCommitAll()` Flush pending updates in all work areas

`dbCreate()` Create a database file from a database structure array

`dbCreateIndex()` Create an index file

`dbDelete()` Mark a record for deletion

`dbEdit()` Browse records in a table layout

`dbEval()` Evaluate a code block for each record matching a scope and condition

`dbFieldInfo()` Return and optionally change information about a field

`dbFileGet()` Insert the contents of a field into a file

`dbFilePut()` Insert the contents of a file into a field

`dbFilter()` Return the current filter expression as a character string

`dbGoBottom()` Move to the last logical record

`dbGoto()` Position record pointer to a specific identity

`dbGoTop()` Move to the first logical record

`dbInfo()` Return and optionally change information about a database file opened in a work area

`dbOrderInfo()` Return and optionally change information about orders and index files

`dbRecall()` Reinstate a record marked for deletion

`dbRecordInfo()` Return and optionally change information about a record

`dbReindex()` Recreate all active indexes for the current work area

`dbRelation()` Return the linking expression of a specified relation

`dbRLock()` Lock the record at the current or specified identity

`dbRLockList()` Return an array of the current lock list

`dbRSelect()` Return the target work area number of a relation

`dbRUnlock()` Release all or specified record locks

`dbSeek()` Move to the record having the specified key value

`dbSelectArea()` Change the current work area

`dbSetDriver()` Return the default database driver and optionally set a new driver

`dbSetFilter()` Set a filter condition

`dbSetIndex()` Empty orders from an order bag into the order list

`dbSetOrder()` Set the controlling order

`dbSetRelation()` Relate two work areas

`dbSkip()` Move relative to the current record

`dbStruct()` Create an array containing the structure of a database file

`dbUnlock()` Release all locks for the current work area

`dbUnlockAll()` Release all locks for all work areas

`dbUseArea()` Use a database file in a work area

`Date()` Return the system date as a date value

`Day()` Return the day of the month as a numeric value

`Dbf()*` Return current alias name

`Deleted()` Return the deleted status of the current record

`Descend()` Create a descending index key value

`DevOut()` Write a value to the current device

`DevOutPict()` Write a value to the current device using a picture clause

DevPos() Move the cursor or printhead to a new position depending on the current device

DECLARE* Create and initialize private memory variables and arrays

DELETE Mark records for deletion

DELETE FILE Remove a file from disk

DELETE TAG Delete a tag

DirChange() Change the current DOS directory

Directory() Create an array of directory and file information

DirMake() Create a directory

DirRemove() Remove a directory

DiskChange() Change the current DOS disk drive

DiskName() Return the current DOS drive

DiskSpace() Return the space available on a specified disk

DispBegin() Begin buffering screen output

DispBox() Display a box on the screen

DispCount() Return the number of pending DispEnd() requests

DispEnd() Display buffered screen updates

DispOut() Write a value to the display

DIR* Display a listing of files from a specified path

DISPLAY Display records to the console

DosError() Return the last DOS error number

DoW() Convert a date value to a numeric day of the week

DO CASE Execute one of several alternative blocks of statements

DO WHILE Execute a loop while a condition is true (.T.)

DO* Call a procedure

DToC() Convert a date value to a character string

DToS() Convert a date value to a character string formatted as
yyyymmdd

EJECT Advance the printhead to top of form

Empty() Determine if the result of an expression is empty

Eof() Determine when end of file is encountered

Error class Provides objects containing information about runtime errors

ErrorBlock() Post a code block to execute when a runtime error occurs

ErrorLevel() Set the CA-Clipper return code

ERASE Remove a file from disk

Eval() Evaluate a code block

Exp() Calculate e^{**x}

EXIT PROCEDURE Declare an exit procedure

EXTERNAL* Declare a list of procedure or user-defined function names
to the linker

FClose() Close an open binary file and write DOS buffers to disk

FCount() Return the number of fields in the current .dbf file

FCreate() Create and/or truncate a binary file to zero-length

FErase() Delete a file from disk

FError() Test for errors after a binary file operation

FieldBlock() Return a set-get code block for a given field

FieldGet() Retrieve the value of a field using the ordinal position of the
field in the database structure

FieldName()/Field() Return a field name from the current database (.dbf)
file

FieldPos() Return the position of a field in a work area

FieldPut() Set the value of a field variable using the ordinal position of the field in the database structure

FieldWBlock() Return a set-get code block for a field in a given work area

FIELD Declare database field names

FILE() Determine if files exist in the CA-Clipper default directory or path

FIND* Search an index for a specified key value

FKLabel()* Return function key name

FKMax()* Return number of function keys as a constant

FLock() Lock an open and shared database file

Found() Determine if the previous search operation succeeded

FOpen() Open a binary file

FOR Execute a block of statements a specified number of times

FRead() Read characters from a binary file into a buffer variable

FReadStr() Read characters from a binary file

FRename() Change the name of a file

FSeek() Set a binary file pointer to a new position

FUNCTION Declare a user-defined function name and formal parameters

FWrite() Write to an open binary file

GBMPDISP() Display a bitmap (.BMP) file on screen

GBMPLOAD() Load a bitmap (.bmp) or icon (.ico) file into memory

Get class Provides objects for interactive editing of database fields and variables

[GetActive\(\)](#) Return the currently active Get object

[GetApplyKey\(\)](#) Apply a key to a Get object from within a reader

[GetDoSetKey\(\)](#) Process SET KEY during GET editing

[GetEnv\(\)](#) Retrieve the contents of a DOS environment variable

[GetPostValidate\(\)](#) Postvalidate the current Get object

[GetPreValidate\(\)](#) Prevalidate a Get object

[GetReader\(\)](#) Execute standard READ behavior for a Get object

[GELLIPSE\(\)](#) Draw an ellipse or circle

[GFENTERASE\(\)](#) Erase a font from memory

[GFNTLOAD\(\)](#) Load a font file into memory

[GFNTSET\(\)](#) Set an already loaded font as active

[GFRAME\(\)](#) Draw a frame with a 3-D look

[GGETPIXEL\(\)](#) Get color information for a pixel

[GLINE\(\)](#) Draw a line in graphic mode

[GMODE\(\)](#) Switch video mode

[GO](#) Move the pointer to the specified identity

[GPOLYGON\(\)](#) Draw a polygon on screen

[GPUTPIXEL\(\)](#) Draw a pixel on the screen

[GRECT\(\)](#) Draw a rectangle in graphic mode

[GSETCLIP\(\)](#) Define the allowed display area

[GSETEXCL\(\)](#) Define a screen region to be excluded from display

[GSETPAL\(\)](#) Change components of a color

[GWRITEAT\(\)](#) Draw graphic text without background

[HardCR\(\)](#) Replace all soft carriage returns in a character string with hard carriage returns

Header() Return the current database file header length

I2Bin() Convert a CA-Clipper numeric to a 16-bit binary integer

IF Execute one of several alternative blocks of statements

IF() Return the result of an expression based on a condition

IIF() Return the result of an expression based on a condition

IndexExt() Return the default index extension based on the database driver currently linked

IndexKey() Return the key expression of a specified index

IndexOrd() Return the order position of the controlling index

Inkey() Extract a character from the keyboard buffer or a mouse event

INDEX Create an index file

INIT PROCEDURE Declare an initialization procedure

INPUT* Enter the result of an expression into a variable

INT() Convert a numeric value to an integer

IsAlpha() Determine if the leftmost character in a string is alphabetic

IsColor() Determine if the current computer has color capability

IsDigit() Determine if the leftmost character in a character string is a digit

IsLower() Determine if the leftmost character is a lowercase letter

IsPrinter() Determine whether LPT1 is ready

IsUpper() Determine if the leftmost character in a string is uppercase

JOIN Create a new database file by merging records/fields from two work areas

KEYBOARD Stuff a string into the keyboard buffer

L2Bin() Convert a CA-Clipper numeric value to a 32-bit binary integer

LastKey() Return the Inkey() value of the last key extracted from the keyboard buffer

LastRec() Determine the number of records in the current .dbf file

LABEL FORM Display labels to the console

Left() Extract a substring beginning with the first character in a string

Len() Return the length of a character string or the number of elements in an array

ListBox class Create a list box

LIST List records to the console

Log() Calculate the natural logarithm of a numeric value

Lower() Convert uppercase characters to lowercase

LOCAL Declare and initialize local variables and arrays

LOCATE Search sequentially for a record matching a condition

LTrim() Remove leading spaces from a character string

LUpdate() Return the last modification date of a database (.dbf) file

Max() Return the larger of two numeric or date values

MaxCol() Determine the maximum visible screen column

MaxRow() Determine the maximum visible screen row

MCol() Determine the mouse cursor's screen column position

MDbIClk() Determine the double-click speed threshold of the mouse

MemoEdit() Display or edit character strings and memo fields

MemoLine() Extract a line of text from a character string or memo field

Memory() Determine the amount of available free pool memory

MemoRead() Return the contents of a disk file as a character string

MemoTran() Replace carriage return/linefeeds in character strings

MemoWrit() Write a character string or memo field to a disk file

MemVarBlock() Return a set-get code block for a given memory variable

MenuItem class Create a menu item

MenuModal() Activate a top bar menu

MEMOSETSUPER() Set an RDD inheritance chain for the DBFMEMO database driver

MEMVAR Declare private and public variable names

MENU TO Execute a lightbar menu for defined PROMPTs

MHide() Hide the mouse pointer

Min() Return the smaller of two numeric or date values

MLCount() Count the number of lines in a character string or memo field

MLCToPos() Return the byte position of a formatted string based on line and column position

MLeftDown() Determine the press status of the left mouse button

MLPos() Determine the position of a line in a character string or memo field

Mod()* Return the dBASE III PLUS modulus of two numbers

Month() Convert a date value to the number of the month

MPosToLC() Return line and column position of a formatted string based on a specified byte position

MPresent() Determine if a mouse is present

MRestState() Re-establish the previous state of a mouse

MRightDown() Determine the status of the right mouse button

MRow() Determine a mouse cursor's screen row position

`MSaveState()` Save the current state of a mouse

`MSetBounds()` Set screen boundaries for the mouse cursor

`MSetCursor()` Determine a mouse's visibility

`MSetPos()` Set a new position for the mouse cursor

`MSETCLIP()` Define an inclusion region

`MShow()` Display the mouse pointer

`MSTATE()` Return the current mouse state

`NetErr()` Determine if a network command has failed

`NetName()` Return the current workstation identification

`NextKey()` Read the pending key in the keyboard buffer

`NoSnow()` Toggle snow suppression

NOTE* Place a single-line comment in a program file

`ordBagExt()` Return the default order bag RDD extension

`ordBagName()` Return the order bag name of a specific order

`ordCondSet()` Set the condition and scope for an order

`ordCreate()` Create an order in an order bag

`ordDescend()` Return and optionally change the descending flag of an order

`ordDestroy()` Remove a specified order from an order bag

`ordFor()` Return the FOR expression of an order

`ordIsUnique()` Return the status of the unique flag for a given order

`ordKey()` Return the key expression of an order

`ordKeyAdd()` Add a key to a custom built order

`ordKeyCount()` Return the number of keys in an order

`ordKeyDel()` Delete a key from a custom built order

`ordKeyGoto()` Move to a record specified by its logical record number in the controlling order

`ordKeyNo()` Get the logical record number of the current record

`ordKeyVal()` Get the key value of the current record from the controlling order

`ordListAdd()` Add orders to the order list

`ordListClear()` Clear the current order list

`ordListRebuild()` Rebuild all orders in the order list of the current work area

`ordName()` Return the name of an order in the order list

`ordNumber()` Return the position of an order in the current order list

`ordScope()` Set or clear the boundaries for scoping key values in the controlling order

`ordSetFocus()` Set focus to an order in an order list

`ordSetRelation()` Relate a specified work area to the current work area

`ordSkipUnique()` Move the record pointer to the next or previous unique key in the controlling order

`ORDCOND()` Specify conditions for ordering

`OS()` Return the operating system name

`OutErr()` Write a list of values to the standard error device

`OutStd()` Write a list of values to the standard output device

`Pad()` Pad character, date, and numeric values with a fill character

`PACK` Remove deleted records from a database file

`PARAMETERS` Create private parameter variables

`PCol()` Return the current column position of the printhead

PCount() Determine the position of the last actual parameter passed

PopUpMenu class Create a pop-up menu

ProcLine() Return the source line number of the current or previous activation

ProcName() Return the name of the current or previous procedure or user-defined function

PRIVATE Create and initialize private memory variables and arrays

PRow() Return the current row position of the printhead

PROCEDURE Declare a procedure name and formal parameters

PushButton class Create a push button

PUBLIC Create and initialize public memory variables and arrays

QOut() Display a list of expressions to the console

QUIT Terminate program processing

rddList() Return an array of the available Replaceable Database Drivers (RDDs)

rddName() Return the name of the RDD active in the current or specified work area

rddSetDefault() Set or return the default RDD for the application

RadioButto class Create a radio button

RadioGroup class Create a radio button group

RAt() Return the position of the last occurrence of a substring

ReadExit() Toggle Up arrow and Down arrow as READ exit keys

ReadFormat() Return and optionally, set the code block that implements a format (.fmt) file

ReadInsert() Toggle the current insert mode for READ and MemoEdit()

ReadKey()* Determine what key terminated a READ

ReadKill() Return, and optionally set, whether the current READ should be exited

ReadModal() Activate a full-screen editing mode for a GetList

ReadUpdated() Determine whether any GET variables changed during a READ and optionally change the ReadUpdated() flag

ReadVar() Return the current GET/MENU variable name

RecCount()* Determine the number of records in the current database (.dbf) file

RecNo() Return the identity at the position of the record pointer

RecSize() Determine the record length of a database (.dbf) file

Replicate() Return a string repeated a specified number of times

RestScreen() Display a saved screen region to a specified location

READ Activate full-screen editing mode using Get objects

RECALL Restore records marked for deletion

REINDEX Rebuild open indexes in the current work area

RELEASE Delete public and private memory variables

RENAME Change the name of a file

REPLACE Assign new values to field variables

REPORT FORM Display a report to the console

REQUEST Declare a module request list

RESTORE Retrieve memory variables from a memory (.mem) file

RESTORE SCREEN* Display a saved screen

RETURN Terminate a procedure, user-defined function, or program

Right() Return a substring beginning with the rightmost character

RLock() Lock the current record in the active work area

Round() Return a numeric value rounded to a specified number of digits

Row() Return the screen row position of the cursor

RTrim() Remove trailing spaces from a character string

RUN Execute a DOS command or program

SaveScreen() Save a screen region for later display

SAVE Save variables to a memory (.mem) file

SAVE SCREEN* Save the current screen to a buffer or variable

Scroll() Scroll a screen region up or down, right or left

Scrollbar class Creates a scroll bar

Seconds() Return the number of seconds elapsed since midnight

Select() Determine the work area number of a specified alias

Set() Inspect or change a system setting

SetBlink() Toggle asterisk (*) interpretation in the SetColor() string between blinking and background intensity

SetCancel() Toggle Alt+C and Ctrl+Break as program termination keys

SetColor() Return the current colors and optionally set new colors

SetCursor() Set the cursor shape

SetKey() Assign an action block to a key

SetMode() Change display mode to a specified number of rows and columns

SetPos() Move the cursor to a new position

SetPRC() Set PRow() and PCol() values

SEEK Search an order for a specified key value

SELECT Change the current work area

SET ALTERNATE Echo console output to a text file

SET BELL Toggle automatic sounding of the bell during full-screen operations

SET CENTURY Modify the date format to include or omit century digits

SET COLOR* Define screen colors

SET CONFIRM Toggle required exit key to terminate GETs

SET CONSOLE Toggle console display to the screen

SET CURSOR Toggle the screen cursor on or off

SET DATE Set the date format for input and display

SET DECIMALS Set the number of decimal places to be displayed

SET DEFAULT Set the CA-Clipper default drive and directory

SET DELETED Toggle filtering of deleted records

SET DELIMITERS Toggle or define GET delimiters

SET DESCENDING Change the descending flag of the controlling order

SET DEVICE Direct @...SAYs to the screen or printer

SET EPOCH Control the interpretation of dates with no century digits

SET ESCAPE Toggle Esc as a READ exit key

SET EVENTMASK Specify events to be returned by the Inkey() function

SET EXACT* Toggle exact matches for character strings

SET EXCLUSIVE* Establish shared or exclusive USE of database files

SET FILTER Hide records not meeting a condition

SET FIXED Toggle fixing of the number of decimal digits displayed

SET FORMAT* Activate a format when READ is executed

SET FUNCTION Assign a character string to a function key

SET INDEX Open one or more order bags in the current work area

SET INTENSITY Toggle enhanced display of GETs and PROMPTs

SET KEY Assign a procedure invocation to a key

SET MARGIN Set the page offset for all printed output

SET MEMOBLOCK Change the block size for memo files

SET MESSAGE Set the @...PROMPT message line row

SET OPTIMIZE Change the setting that determines whether to optimize using the open orders when processing a filtered database file

SET ORDER Select the controlling order

SET PATH Specify the CA-Clipper search path for opening files

SET PRINTER Toggle echo of console output to the printer or set the destination of printed output

SET PROCEDURE* Compile procedures and functions into the current object (.OBJ) file

SET RELATION Relate two work areas by a key value or record number

SET SCOPE Change the top and/or bottom boundaries for scoping key values in the controlling order

SET SCOPEBOTTOM Change the bottom boundary for scoping key values in the controlling order

SET SCOPETOP Change the top boundary for scoping key values in the controlling order

SET SCOREBOARD Toggle the message display from READ or MemoEdit()

SET SOFTSEEK Toggle relative seeking

SET TYPEAHEAD Set the size of the keyboard buffer

SET UNIQUE* Toggle inclusion of non-unique keys into an index

SET VIDEOMODE Change the current video mode of the current application

SET WRAP* Toggle wrapping of the highlight in menus

SKIP Move the record pointer to a new position

SoundEx() Convert a character string to "soundex" form

SORT Copy to a database (.dbf) file in sorted order

Space() Return a string of spaces

Sqrt() Return the square root of a positive number

Str() Convert a numeric expression to a character string

StrTran() Search and replace characters within a character string or memo field

Stuff() Delete and insert characters in a string

STATIC Declare and initialize static variables and arrays

STORE* Assign a value to one or more variables

SubStr() Extract a substring from a character string

SUM Sum numeric expressions and assign results to variables

TBColumn class Provide column objects for TBrowse objects

TBrowse class Provide objects for browsing table-oriented data

TEXT* Display a literal block of text

Time() Return the system time

Tone() Sound a speaker tone for a specified frequency and duration

TopBarMenu class Create a top bar menu

TOTAL Summarize records by key value to a database (.dbf) file

Transform() Convert any value into a formatted character string

Trim() Remove trailing spaces from a character string

[Type\(\)](#) Determine the type of an expression

[TYPE](#) Display the contents of a text file

[UNLOCK](#) Release file/record locks set by the current user

[Updated\(\)](#) Determine whether a GET changed during a READ

[Upper\(\)](#) Convert lowercase characters to uppercase

[UPDATE](#) Update current database file from another database file

[Used\(\)](#) Determine whether a database file is in USE

[USE](#) Open an existing database (.dbf) and its associated files

[Val\(\)](#) Convert a character number to numeric type

[ValType\(\)](#) Determine the data type returned by an expression

[Version\(\)](#) Returns CA-Clipper version

[WAIT*](#) Suspend program processing until a key is pressed

[Word\(\)*](#) Convert CALL command numeric parameters from double to integer values

[Year\(\)](#) Convert a date value to the year as a numeric value

[ZAP](#) Remove all records from the current database file

Copyright

[CA-Clipper 5.3](#) Copyright

Tables

[ASCII Chart](#) ASCII Chart

[Box Characters](#) Box Characters

[Colors](#) Colors

[Inkey Codes](#) Inkey Codes

[Picture Codes](#) [Picture Codes](#)

[Reserved Words](#) [Reserved Words](#)

[Special Characters](#) [Special Characters](#)

[Unsupported dBASE Items](#) [Unsupported dBASE Items](#)

Clipper Tools 3

Intro

[Broadcast Messages](#) [Introduction](#)

[Capture Services](#) [Introduction](#)

[Connection Services](#) [Introduction](#)

[Database Functions](#) [Introduction](#)

[Date / Time Functions](#) [Introduction](#)

[Devices and Forms](#) [Introduction](#)

[Disk Utilities](#) [Introduction](#)

[Drive Mapping](#) [Introduction](#)

[Extended Drivers](#) [Introduction](#)

[Functions For IBM PC-LAN](#) [Introduction](#)

[Get / Read Functions](#) [Introduction](#)

[Installing CA-Clipper Tools](#) [Installing CA-Clipper Tools](#)

[Introduction](#) [Introduction](#)

[Low Level Bindery Access](#) [Introduction](#)

[Mathematical Functions](#) [Introduction](#)

[Miscellaneous Functions](#) [Introduction](#)

[Miscellaneous Functions](#) [Introduction](#)

[Number / Bit Manipulation](#) Introduction

[Peek / Poke Functions](#) Introduction

[Point-To-Point Communication](#) Introduction

[Print Job Definitions](#) Introduction

[Print Queue Management](#) Introduction

[Print Server Access](#) Introduction

[Printer Functions](#) Introduction

[Semaphores](#) Introduction

[Serial Communication](#) Introduction

[Set Status](#) Introduction

[String Manipulation](#) Introduction

[System Informations](#) Introduction

[Transition Tracking](#) Introduction

[User and Group Management](#) Introduction

[Using CA-Clipper Tools](#) Using CA-Clipper Tools

[Video Functions](#) Introduction

[Volumes, Directories, ...](#) Introduction

[Window Functions](#) Introduction

[Acos\(\)](#) Computes the cosine arc

[AddAscii\(\)](#) Adds a value to each ASCII code in a string

[Additional Programs](#) Additional Programs

[AddMonth\(\)](#) Adds or subtracts months to/from a date

[AfterAtNum\(\)](#) Returns the remainder of the string after the nth appearance of a sequence

[AlloFree\(\)*](#) Determines the maximum memory size allocation

AsciiSum() Finds the sum of the ASCII values of all the characters of a string

AscPos() Determines the ASCII value of a character at a particular position within a string

Asin() Computes the sine arc

Atan() Computes the tangent arc

AtAdjust() Adjusts the beginning position of a sequence within a string

Atn2() Computes the angle size from the sine and cosine

AtNum() Determines the starting position of a sequence within a string

AtRepl() Searches for a sequence within a string and replaces it

AtToken() Finds the position of a token within a string

BeforeAtNum() Returns the string segment before the nth occurrence of a sequence

BitToC() Converts position-dependent bits into characters

BIOSDATE() Determines the system BIOS date

Blank() Creates a blank value for each data type

BoM() Determines the date of the first day of a month

BoQ() Determines the date for the beginning of a quarter

BoY() Determines the date for the beginning of a year

BOOTCOLD() Triggers a cold boot

BOOTWARM() Triggers a warm start of the system

com_Break() Creates a break on a transmission line

com_Close() Clears the receiving buffer and closes the com port

com_Count() Counts the number of characters in the input buffer

com_CRC() Computes a Cyclic Redundancy Check (CRC) for the string

`com_CTS()` Queries the Clear To Send (CTS) status

`com_DCD()` Queries the Data Carrier Detect (DCD) status

`com_DosCon()` Provides screen output through DOS (ANSI.SYS terminal emulation)

`com_DSR()` Queries the Data Set Ready (DSR) status

`com_DTR()` Queries/sets the Data Terminal Ready (DTR) status

`com_ErrChr()` Defines the replacement character for a character that is not received correctly

`com_Event()` Designates which event at the port triggered a key trap

`com_Flush()` Clears the receiving buffer

`com_GetIO()` Determines the base address of a port

`com_GetIRQ()` Determines the interrupt request for a port

`com_Hard()` Turns the hardware handshake (automatic CTS) on/off

`com_Init()` Initializes the port parameters

`com_Key()` Monitors the port using key traps

`com_LSR()` Reads the Line Status Register (LSR)

`com_MCR()` Reads or sets the Modem Control Register (MCR)

`com_MSR()` Reads the Modem Status Register (MSR)

`com_Num()` Gives the number of the highest available serial interface port

`com_Open()` Opens the port and initializes the buffer

`com_Read()` Reads characters from the receiving buffer

`com_Remote()` Determines the clear character for the receiving buffer

`com_Ring()` Queries the ring line

`com_RTS()` Queries or sets the Request To Send (RTS)

`com_SCount()` Counts the number of characters in the background sending buffer

`com_Send()` Transmits data directly or in the background

`com_SetIO()` Changes the base address for a port

`com_SetIRQ()` Changes the interrupt request for a port

`com_SFlush()` Deletes the sending background buffer

`com_SKey()` Monitors the port using key traps during background transmission

`com_SMode()` Determines the current status of a background transmission

`com_Soft_R()` Tests to see if an XOFF character has been received

`com_Soft_S()` Tests to see if the buffer has automatically sent an XOFF character

`com_Soft()` Queries or sets the software handshake (automatic XON/XOFF)

`Ceiling()` Rounds up to the next integer

`Celsius()` Converts a Fahrenheit temperature value into Celsius

`Center()` Centers a string using pad characters

`CGA40()` Switches to 40-column mode (color or monochrome)

`CGA80()` Switches to 80-column mode (color or monochrome)

`CharAdd()` Adds the corresponding ASCII codes of two strings

`CharAnd()` Links corresponding ASCII codes of paired strings with an AND operation

`CharEven()` Returns characters in the even positions of a string

`CharList()` Lists each character in a string

CharMirr() Mirrors characters within a string

CharMix() Mixes two strings together

CharNoList() Lists the characters that do not appear in a string

CharNot() Complements each character in a string

CharOdd() Returns characters in the odd positions of a string

CharOne() Reduces adjoining duplicate characters in a string to one character

CharOnly() Determines the common denominator between two strings on the basis of individual characters

CharOr() Joins the corresponding ASCII code of paired strings with an OR operation

CharPack() Compresses (packs) a string

CharPix() Returns the number of pixel lines per character

CharRelA() Correlates the character positions in paired strings

CharRelRep() Replaces characters in a string depending on their correlation

CharRem() Removes particular characters from a string

CharRepl() Replaces certain characters with others

CharSort() Sorts sequences within a string

CharSpread() Expands a string at the tokens

CharSwap() Exchanges all adjoining characters in a string

CharUnpack() Decompresses (unpacks) a string

CharWin() Exchanges particular characters in a screen area.

CharXor() Joins corresponding ASCII codes of paired strings using an exclusive OR operation

Checksum() Calculates the checksum for a character string (algorithm)

ClearBit() Clears one or more bits within a number to zero

ClearEol() Clears from the cursor position to the end of line

ClearSlow() Deletes a screen area from the outside in with a delay

ClearWin() Clears a screen area

ClEol() Clears characters and attributes to the end of a line

ClWin() Clears character and attribute from a screen area

CLOSESOCK() Closes the socket and all connected IPX/SPX handles

ColorRepl() Exchanges particular screen attributes

ColorToN() Converts NN/NN or CC/CC color values into numeric values

ColorWin() Exchanges particular attributes in a screen area

Complement() Forms the complement value of a data type

Cos() Computes the cosine

Cot() Computes the cotangent

CountGets() Determines the number of posted GET fields

CountLeft() Counts a particular character at the beginning of a string

CountRight() Counts a particular character at the end of a string

CPUType() Determines what type of microprocessor in use

Crypt() Encrypts and decrypts a string

CSetAtMupa() Determines the setting of the multi-pass mode for ATXXX() functions

CSetKey()* Queries the setting for SET KEY TO

CSetRef() Determines whether or not reference sensitive functions return a value

CSetSafety() Queries/sets the safety mode switch

CSETALL() Saves all ON/OFF switch settings

CSETCLIP() Determines the content of CA-Clipper environmental variables

CSETDATE()* Queries the SET DATE setting

CSETDECI()* Queries the setting for SET DECIMALS TO

CSETDEFA()* Queries the setting for SET DEFAULT

CSETFUNC()* Queries the setting for SET FUNCTION TO

CSETLDEL()* Queries the setting for the left delimiters

CSETMARG()* Queries the setting for SET MARGIN TO

CSETPATH()* Queries the setting for SET PATH TO

CSETRDEL()* Queries the setting for the right delimiter

CSETRDONLY()* Queries/sets the read-only mode switch

CSETSNOW() Helps prevent snow on the screen

CSETxxxx()* Queries the SET position of a particular ON/OFF switch and provides the option to set it

CToBit() Converts a character string into a bit pattern

CToDoW() Converts the name of the day of the week into a corresponding number

CToF() Converts a special 8-byte string into a floating point number

CToMonth() Converts the name of the month into a corresponding number

CToN() Converts a numeric string into a different base

CurrentGet() Determines the number of the currently active GET field

DATATYPE()* Determines the data type of a variable or UDF

DbfSize() Determines the size of a selected (.dbf) file in memory

DBFDSKSIZE() Determines the size of a selected (.dbf) file on a disk drive

DeleteFile() Deletes an error-tolerant file

DirChange() Changes the current directory

DirMake() Creates a directory

DirName() Determines the name of the current directory

DirRemove() Removes a directory

DiskChange() Changes the current disk drive

DiskFree() Determines the space available on a floppy or hard disk

DiskName() Determines the drive designator for the current drive

DiskTotal() Determines the total capacity of a floppy or hard disk

DISKCHECK() Creates a checksum for a disk

DISKFORMAT() Formats disks, controlled through a UDF

DISKREADY() Tests to see if a disk drive is ready

DISKREADYW() Queries whether you can write to a drive

DISKSPEED() Determines a comparison value for the drive speed

DISKSTAT() Determines the status of a drive.

DISKTYPE() Determines the type of data carrier

DMY() Returns a date in "DD Month YY" format

DosParam() Retrieves the DOS command line as a string

DoY() Determines the day of the year for a specific date

DriveType() Determines the drive type

DSetKBIOS() Turns the extended keyboard mode on or off through BIOS
(additional keys F11, F12)

DSETNOLINE() Ignores the next line feed sent to the screen

DSETQFILE() Creates a protocol file when the program ends normally

DSETTYPE() Determines the size of the keyboard buffer (SET TYPEAHEAD TO)

DSETWINDOW() Reroutes external functions and programs to a window

DTOR() Converts from a degree to radian measure

EGA43() Switches to the 43-line EGA mode

EGAPALETTE() Changes EGA palette colors

Enhanced() Selects the enhanced color value for SET COLOR TO output

EnvParam() Reads the entire DOS environment table into a string

EoM() Determines the date for the last day of a month

EoQ() Determines the date for the end of a quarter

EoY() Determines the date for the end of the year

ERRORACT() Recommends an action for a DOS error that has occurred previously

ERRORBASE() Source of the most-recent DOS error

ERRORCODE() Identifies a DOS error that has occurred previously

ERRORORG() Origin of the most-recent DOS error

ExeName() Returns name and directory of the current CA-Clipper program

Expand() Expands a string by inserting characters

Exponent() Determines the exponent of a floating point number (base 2)

Fact() Computes the factorial

Fahrenheit() Converts a temperature value from Celsius into Fahrenheit

FieldDeci() Determines the number of decimal places in a field

FieldNum() Determines the field number for a specific field in a database file

FieldSize() Determines the size of a field

FieldType() Determines the data type for a field

FileAppend() Appends data to a file

FileAttr() Determines a file's attributes

FileCClose() Closes a file after backup mode

FileCCont() Copies sections of a file in backup mode

FileCDaTi() Determines whether a source file's date and time stamp or the system's date and time stamp, should be used to create or copy a file. Use original or

FileCopy() Copies files normally or in backup mode

FileCOpen() Tests to see if the file is still open in the backup mode

FileDate() Determines the file date

FileDelete() Deletes file(s) by name and attribute

FileMove() Moves files to another directory

FileScreen() Reads screen content from a file

FileSeek() Searches for files by name and attribute

FileSize() Determines the size of a file

FileSMax() Specifies the maximum number of files that can be open at one time

FileStr() Reads a portion of a file into a string

FileTime() Determines a file's time

FILECHECK() Calculates/computes/determines a checksum for a file

FILESFREE() Specifies the number of files you can open

FILEVALID() Tests whether a string has a valid file name

FIRSTCOL() Sets the first visible column of a virtual screen

FIRSTROW() Sets the first visible line of a virtual screen

Floor() Rounds down to the next integer

FLOPPYTYPE() Determines the exact type of floppy drive

FONTLOAD() Loads EGA/VGA fonts from another file

FONTRESET() Resets all font and palette changes to the ROM defaults

FONTROTATE() Rotates and mirrors images within a font string

FONTSELECT() Determines font areas for normal- and high-intensity output

FToC() Converts a floating point number into a special 8-byte string

FV() Computes future value of capital

GetClearA() Queries the current attribute for the clearing functions

GetClearB() Queries the default character for the clearing functions

GetFldCol() Determines the screen column of a GET field

GetFldRow() Determines the row of a GET field on the screen

GetFldVar() Determines the name of a GET field

GetInput() Keyboard input function similar to a GET field

GetKXLat() Determines the current key code table

GetKXTab() Retrieves the entire key code table

GetPrec() Determines the level of precision that is set

GetSecret() Keyboard input function for hidden input (similar to a GET field)

GETBOXGROW() Gets the time delay with which boxes are opened

GETCOUNTRY() Queries country setting for the operating system

[GETCURSOR\(\)](#)* Determines the setting for the cursor form

[GETFONT\(\)](#) Queries the current font

[GETLINES\(\)](#) Determines the number of lines after which the screen display pauses

[GETMODE\(\)](#) Uses the current screen mode as a function name

[GETPAGE\(\)](#) Determines the current screen page

[GETPBIOS\(\)](#) Determines if printing is through DOS or the BIOS

[GETPXLAT\(\)](#) Retrieves the current printer table

[GETSCRMODE\(\)](#) Determines the number of the active video mode

[GETSCRSTR\(\)](#) Queries screen output that was redirected by SETSCRSTR()

[GETSHARE\(\)](#) Determines the file open (share) mode

[GETTAB\(\)](#) Retrieves tab values for CA-Clipper screen output

[GETTIC\(\)](#) Determines the number of timer ticks

[GETVGAPAL\(\)](#) Determines the palette settings on a VGA card

[HexToStr\(\)](#) Converts a hexadecimal string into a byte sequence

[Infinity\(\)](#) Creates the largest number possible (21023)

[IntNeg\(\)](#) Converts an unsigned integer into a signed integer

[IntPos\(\)](#) Converts a signed integer into an unsigned integer

[InvertAttr\(\)](#) Inverts the foreground and background of an attribute

[InvertWin\(\)](#) Inverts all attributes in an area of the screen

[INBYTE\(\)](#) Reads an 8 byte from a port

[INKEYTRAP\(\)](#) Behaves like Inkey() with support for key traps

[INPUTMODE\(\)](#) Determines the previously active or the currently active input mode

[INWORD\(\)](#) Reads in a 16-bit word from a port

IPXERROR() Determines the error code of the last IPX/SPX call

IPXOPEN() Opens the IPX sending and receiving buffer

IsAt() Determines if a program is running on an AT

IsBit() Tests the bits in a number

IsCGA() Tests for presence of a CGA card or if one can be emulated

IsEGA() Determines if an EGA card is present or can be emulated

IsHercules() Determines if a HERCULES card is present or can be emulated

IsLeap() Tests if a specific year is a leap year

IsMath() Determines if a math coprocessor is installed

IsMCGA() Determines if an MCGA card is present or can be emulated

IsMono() Determines if a monochrome card is present or can be emulated

IsPGA() Determines if a PGA card is present or can be emulated

IsVGA() Determines if a VGA card is present

ISANSI() Tests to see if the ANSI screen driver is installed

ISDBT() Determines if a memo file (.dbt) is present

ISDEBUG() Determines if the debugger is available in the application

ISIPX() Checks for IPX support

ISNETBIOS() Checks for NetBIOS support

ISSPX() Checks for SPX support

JustLeft() Moves characters from the beginning to the end of a string

JustRight() Moves characters from the end of a string to the beginning

KbdStat() Tests for key shift state status, such as Ctrl and Shift

KBDDISABLE() Locks/unlocks the keyboard

KBDEMULATE() Inserts characters into the BIOS keyboard buffer to emulate keyboard input

KBDSPEED() Sets keyboard auto repeat speed

KBDTYPE() Determines the type of keyboard in use

KeySec() Triggers a key trap after a time delay

KeyTime() Triggers a key trap at a specific clock time

KEYREAD() Reads already processed CA-Clipper keyboard buffer input

KEYSEND() Simulates CA-Clipper keyboard buffer input

KSetCaps() Queries/sets the system setting for CAPS LOCK

KSetIns() Queries/sets the system setting for "INSERT"

KSetNum() Queries/sets the system setting for NUMLOCK

KSetScroll() Queries/sets the system setting for SCROLL LOCK

LastDayOM() Determines the number of days in a month

LASTKFUNC() Returns the function name that created the most-recent key trap

LASTKLINE() Returns the line number of the most-recent key trap

LASTKPROC() Returns the procedure name that was interrupted

Like() Compares character strings using wildcard characters

Log10() Computes the common logarithm

LToC() Converts a logical value into a character

LToN() Converts a logical value into a numeric value

Mantissa() Determines the mantissa of a floating point number (base2)

MaxCol() Extends the CA-Clipper MaxCol() function

MaxLine() Finds the longest line within a string

MaxRow() Extends the CA-Clipper MaxRow() function

MAXFONT() Determines the number of available fonts

MAXPAGE() Determines the number of available screen pages

MDY() Returns a date in the "Month DD, YY" format

MEMSIZE() Determines size of conventional or extended memory

Millisec() Time delay in milliseconds

MONISWITCH() Switches between monochrome and color screen

MONOCHROME() Switches to the monochrome mode

NBADDGROUP() Adds a NetBIOS group name to the local name table

NBADDNAME() Adds a NetBIOS station name to the local name table

NBDELNAME() Deletes a NetBIOS name from the local name table

NBDOPEN() Opens the NetBIOS datagram sending and receiving buffer

NBERROR() Determines the error code of the most recent NetBIOS call

NBNAME() Reads a NetBIOS name from the name table of a workstation

NBNAMECNT() Determines the number of NetBIOS names on a workstation

NBNAMENUM() Determines the number of a NetBIOS name

NBNAMESTAT() Determines the status of a NetBIOS name

NBRESET() Resets the NetBIOS adapter

NBSCALL() Opens a NetBIOS session sending and receiving buffer and attempts a connection setup

NBSCONTARG() Determines the target name of a NetBIOS communication session

NBSLISTCON() Opens the NetBIOS session sending and receiving buffer and waits for connection in the background

NetCancel() Releases the redirection of a local device

NetDisk() Determines whether a drive is local or resident on the server

NetPrinter() Determines whether the current printer is a local or network printer

NetRedir() Redirects a local device to a server

NetRmtname() Determines the name of a server device for a local device

Network() Tests to see if a network is active

NETLOCNAME() Determines the name of a local redirected device

NNetwork() Determines whether or not a Novell network is active

NNETACCDIS() Sets or queries an access account lock

NNETADDGRP() Adds one or more users to a group

NNETADDQOP() Adds operators to a print queue

NNETADDQSV() Adds a print server

NNETADDQUS() Adds a queue user

NNETADDSET() Adds a bindery object to a set property

NNETADR() Determines the internal netaddress of a user

NNETAEXPD() Determines or sets the expiration date of an access account

NNETATTACH() Attaches a file server to a workstation

NNETBANNER() Sets or queries the current banner name

NNETBINACC() Determines the bindery security access level

NNETBINCL() Closes a bindery

NNETBINOP() Opens a bindery

NNETBRDCST() Determines or queries the receiving mode for broadcast messages

NNETCAPACT() Determines if the capture mode is active for an LPT device

NNETCAPBEG() Activates the capture mode for an LPT device

NNETCAPCAN() Deactivates the capture mode and aborts the print job

NNETCAPEND() Terminates the capture mode for an LPT device

NNETCAPFLU() Closes the current print job and dispatches it

NNETCAPJOB() Determines the job number of the most recently opened capture job

NNETCAPSSF() Sets the Novell capture mode flags

NNETCCNSRV() Determines the number of attached file servers

NNETCLRCON() Clears a connection number

NNETCOPRIV() Tests for console privileges under Novell

NNETCOPY() Copies files within a Novell file server

NNETCPASS() Modifies or sets the password for a bindery object

NNETCRTGRP() Creates a new group

NNETCRTOBJ() Creates a new bindery object

NNETCRTPRP() Creates a new bindery property

NNETCRTQ() Creates a new print queue

NNETCRTUSR() Creates a new user

NNETCVSSRV() Determines the number of visible file servers in a network

NNETDELFAK() Deletes the fake root mapping for a drive

NNETDELOBJ() Deletes a bindery object

NNETDELPRP() Deletes an object property

NNETDELQ() Deletes a print queue

NNETDELSET() Deletes a bindery object from a set property

NNETDIRFRE() Determines the number of free directory entries in a volume

NNETDIRMAX() Determines the maximum number of directory entries in a volume

NNETDIRS() Determines the extended information about subdirectories

NNETDISLOG() Disables a file server for logins

NNETDISTTS() Turns off the Transaction Tracking System (TTS)

NNETDOWN() Shuts down the file server

NNETDSKUSE() Determines the server disk capacity occupied by a user

NNETENLOG() Enables a file server for logins

NNETENTTS() Activates the Transaction Tracking System (TTS)

NNETERROR() Returns the error most recently encountered by a Novell function

NNETEXTATT() Sets or queries extended file attributes

NNETFAKDEP() Determines the relative directory depth under a fake root

NNETFILES() Determines the extended information about subdirectories

NNETFSLST() Determines the names of all visible file servers within a network

NNETGETMSG() Reads the broadcast buffer on a file server

NNETGROUPS() Queries the user groups on a file server

NNETGRPMEM() Checks to see if a user is a member of a group

NNETINFO() Returns the information string of the Netware in use

NNETINGRPS() Determines the groups to which a user belongs

NNETINSET() Determines if a bindery object belongs to a set property

NNETINUSE() Returns the number of objects currently logged into the network

NNETISTTS() Checks for the availability of the Transaction Tracking System (TTS)

NNETJBAN() Determines if a banner page is printed for a job

NNETJBFILE() Sets or determines the file names in the banner text of a job

NNETJBNAME() Sets or determines the banner name for a job

NNETJCNT() Determines the number of jobs in a print queue

NNETJCOPY() Sets or determines the number of copies for a job

NNETJCPL() Sets or determines the number of characters per line for a job

NNETJDEL() Deletes a job from a print queue

NNETJDESC() Reads and modifies the job description

NNETJEDATE() Determines the entry date of a job

NNETJETIME() Determines the entry time of a job

NNETJFILE() Determines the name of a job file

NNETJFLAGS() Reads and sets the job control flag

NNETJFORM() Sets or determines the form name for a job

NNETJLIST() Determines a list of job numbers in a queue

NNETJLPP() Sets or determines the number of lines per page for a job

NNETJOWNER() Determines the login name of a job owner

NNETJPOINT() Determines if a job is serviced at an interrupt

NNETJPOS() Determines or changes the position of a job in a queue

NNETJSDATE() Queries and sets a job start date

NNETJSIZE() Determines the size of a job file

NNETJSRV() Sets or determines the target or the processing print server

NNETJSTIME() Queries or sets the job start time

NNETJSUPFF() Checks to see if the form feed at the end of a job is suppressed

NNETJTABS() Sets or determines the tab setting for a job

NNETJTXT() Determines if the contents of a job file are interpreted as text or a binary byte sequence

NNETJWORK() Checks to see if a job is currently in service on a print server

NNETLGUSER() Determines the users that are currently logged in on a file server

NNETLOGGED() Determines if a station has logged in using LOGIN

NNETLOGIN() Log in a user on a file server

NNETLOGOUT() Log out a user of one or all file servers

NNETMAP() Maps and unmaps drives

NNETMAPINF() Determines the mapping for a drive

NNETMAPMOD() Determines the mapping mode for a drive

NNETMAXCON() Sets or determines the maximum number of connections

NNETMAXRGH() Sets or queries the maximum and inherited rights for a directory

NNETMKDIR() Creates a directory on a volume

NNETMSGCLO() Reads the message buffer and uninstalls the broadcast system

NNETMSGCLR() Clears the broadcast buffer

NNETMSGCNT() Determines the number of messages in the broadcast buffer

NNETMSGKEY() Defines the key code for incoming broadcast messages

NNETMSGOPN() Initializes the interrupt-controlled broadcast system

NNETMSGRD() Reads the message from broadcast buffer

NNETMSGSIZ() Determines the size of the broadcast buffer

NNETNAME() Determines the user name for an internal netaddress

NNETNQSRVS() Determines the number of print servers that service a print queue

NNETNUMMEM() Determines the number of members of a set property

NNETNXTFRE() Returns the next available drive designator

NNETOBJID() Determines the object ID with the object name and the object type

NNETOBJNAM() Determines the object name of an object ID

NNETOBJSEC() Sets the security of a bindery object

NNETOBJTYP() Determines the object type of an object ID

NNETOCNUMS() Determines all of the stations where a user is logged in

NNETPDEV() Determines the names of defined devices

NNETPEXPD() Sets or determines the password expiration date

NNETPEXPI() Sets or determines the password expiration interval

NNETPFLEN() Determines the page length of a form definition

NNETPFNUM() Determines the number of the form

NNETPFORMS() Determines the names of the defined forms

NNETPFUNCO() Determines the control sequences of a function definition

NNETPFUNCS() Determines the function name for a device

NNETPFWDTH() Determines the form width of a form definition

NNETPJADD() Adds a new print job definition

NNETPJBAN() Sets or queries the print of a banner page

NNETPJBNAM() Sets or queries the banner name of a print job

NNETPJCAP() Sets or queries the capture mode for a print job

NNETPJCAPF() Starts the capture mode with print job settings

NNETPJCNT() Determines the number of jobs in a job file

NNETPJCOPY() Sets or queries the number of copies for a print job

NNETPJCRT() Creates an empty print job file

NNETPJDEF() Sets or queries the default print job for a user

NNETPJDEL() Deletes a print job definition

NNETPJDEV() Sets or queries the print job device

NNETPJFORM() Sets or queries the print job form

NNETPJFSRV() Sets or queries the file server for a print job

NNETPJLIST() Determines the print job list for a user

NNETPJLPT() Sets the LPT device for a print job definition

NNETPJMODE() Sets or queries the device mode for a print job

NNETPJNOTY() Sets or queries the notify mode for a print job

NNETPJPSRV() Sets or queries the target print server for a print job

NNETPJQ() Sets or queries the print queue for a print job

NNETPJSUPF() Sets or queries the form feed mode after a print job

NNETPJABS() Sets or queries the tab widths for a print job

NNETPJTXT() Sets or queries the text or binary mode for a print job

NNETPJUSR() Sets or determines the user name of a print job

NNETPJWAIT() Sets or queries the timeout for a print job

NNETPMODES() Determines the mode name for a device

NNETPMODFU() Determines the functions of a mode definition

NNETPRPSEC() Sets the property security

NNETPSACC() Determines the access rights to a print server

NNETPSADDQ() Adds a queue to a print server printer

NNETPSAJOB() Aborts the print job that is currently being printed

NNETPSANO() Defines an object that is notified for the print server printer

NNETPSARP() Determines the available remote printers of a print server

NNETPSCAND() Cancels the down request of a print server

NNETPSCHGQ() Changes the queue priority

NNETPSCNO() Changes the settings for the object that is notified

NNETPSDELQ() Deletes queue access for a print server printer

NNETPSDNO() Deletes an object that is notified for a print server printer

NNETPSDOWN() Deinitializes a print server

NNETPSERR() Determines the error code of the most recent print server call

NNETPSFF() Initializes a form feed on the print server printer

NNETPSJBYT() Determines the number of bytes already printed for an active job

NNETPSJDSC() Determines the description of the active job

NNETPSJFRM() Determines the form number for the active job

NNETPSJFS() Determines the file server of the active job

NNETPSJNO() Determines the number of an active job

NNETPSJPCO() Determines the number of copies printed for the active job

NNETPSJQ() Determines the queue of the active job

NNETPSJSIZ() Determines the size of an active job

NNETPSJTCO() Determines the total number of copies for an active job

NNETPSJTXT() Checks to see if an active job is a text job

NNETPSLIN() Logs in a print server to a file server

NNETPSLOUT() Logs out a print server from a file server

NNETPSMTOF() Marks the top of the page on the print server printer

NNETPSNMOD() Determines the number of queue service modes

NNETPSNOTL() Determines the list of notified objects for the print server printer

NNETPSNPRN() Determines the number of print server printers

NNETPSPFRM() Sets or determines the form for the print server printer

NNETPSPJOB() Checks to see if the print server printer is processing a job

NNETPSPMOD() Sets or determines the queue service mode

NNETPSPNAM() Determines the name of a print server printer

NNETPSPROB() Determines the cause of an error for print server printer

NNETPSPSQ() Determines the list of printers that service a queue

NNETPSPSTA() Queries the status of a print server printer

NNETPSQL() Determines the queues that are serviced by a print server printer

NNETPSRRP() Requests a remote printer's configuration information

NNETPSSRM() Sets the remote printer mode

NNETPSSRVL() Determines the file server for the print server

NNETPSSTAT() Determines a print server's status

NNETPSTART() Starts a print server printer

NNETPSTOP() Stops a print server printer

NNETPSTYPE() Determines a print server's type

NNETPSVER() Determines a print server's version

NNETPURGE() Removes the files that are marked for deletion from the server

NNETPWGRCE() Sets or queries the number of grace logins

NNETPWLEN() Sets or determines the minimum password length

NNETQDIR() Determines the queue directory

NNETQOPS() Determines the list of queue operators

NNETQSRVS() Determines the list of print servers that service a print queue

NNETQSTAT() Sets or reads a print queue status

NNETQUSERS() Determines the list of queue users

NNETRDITM() Reads an item property segment

NNETRDSET() Reads a set property

NNETREMGRP() Removes one or more users from a group

NNETREMQOP() Removes a queue operator

NNETREMQSV() Removes a queue server

NNETREMQUS() Removes a queue user

NNETRENDIR() Renames a directory on a volume

NNETRENOBJ() Renames a bindery object

NNETRGHMSK() Returns the rights mask, dependent on the Netware version

NNETRRIGHTS() Tests for access rights to a Novell file server directory

NNETRMDIR() Removes a directory from a volume

NNETSALLST() Determines the list of recoverable files (Netware 3.x only)

NNETSALVAG() Recovers the files marked for deletion

NNETSCNBIN() Scans a bindery for an object

NNETSCNPRP() Scans a bindery object for a property

NNETSDATE() Queries the Novell file server date

NNETSEARCH() Determines the index of a search drive in the path variable

NNETSEMCLO() Closes a semaphore

NNETSEMOPC() Determines the number of stations that have opened a particular semaphore

NNETSEMOPN() Opens and initializes a semaphore

NNETSEMSGN() Increments a semaphore

NNETSEMVAL() Determines a semaphore's value

NNETSEMWAI() Decrements a semaphore

NNETSERNO() Determines the Netware serial number

NNETSETQ() Sets a print queue for the capture mode

NNETSETSRV() Sets an attached server as the default server

NNETSFTLVL() Determines the System Fault Tolerance (SFT) level of the Netware in use

NNETSLIST() Returns a list of all the attached file servers

NNETSNAME() Returns the name of the default file server

NNETSNDALL() Sends a message to all users on a file server

NNETSNDCON() Sends a message to the server console

NNETSNDGRP() Sends a message to all group members

NNETSNDLOG() Sends a message to the log file of a file server

NNETSNDUSR() Sends a message to a user

NNETSPRVSR() Determines if the logged in user has supervisor rights

NNETSPSTAT() Determines the status of a server printer

NNETSTAID() Determines the ID of a station (network card)

NNETSTANUM() Queries the station number on a file server

NNETSTIME() Queries the time on a Novell file server

NNETTRSLST() Determines the trustee directories and files

NNETTRUST() Grants or removes trustee rights

NNETTTTSAB() Aborts a running transaction

NNETTTSBEG() Starts an explicit transaction

NNETTTSEND() Terminates a transaction

NNETTTSSTA() Determines the transaction status after completion

NNETUSERID() Queries the established network user IDs

NNETUSERS() Determines the users on a file server

NNETUSINGR() Determines the users that are members of a group

NNETUSRFRE() Determines the server disk space still available to a user

NNETVER() Determines the version of the Netware in use

NNETVOLFRE() Determines the available capacity of a server volume

NNETVOLMAX() Determines the maximum capacity of a server volume

NNETVOLNAM() Determines the name of a server volume

NNETVOLNUM() Determines the maximum number of available server volumes

NNETVPASS() Verifies a password for a bindery object

NNETWHOAMI() Determines the station LOGIN name

NNETWRTITM() Writes a segment of an item property

NToC() Converts the numbers in a digit string into a different number base

NToCDoW() Changes the number of a weekday into a weekday name

NToCMonth() Changes the number of a month into a month name

NToColor() Converts a numeric value into a color value in the NN/NN or CC/CC form

Nul() Converts the value returned by a function into a null string

NumAnd() Performs a 16-bit "AND" of a list of numbers

NumAt() Counts the number of occurrences of a sequence within a string

NumCount() Uses the internal CA-Clipper Tools counter

NumDiskL() Determines the number of available logical drives

NumHigh() Returns the higher value byte in a 16-bit number

NumLine() Determines the number of lines required for string output

NumLow() Returns the lower value byte in a 16-bit number

NumMirr() Mirrors 8-bit or 16-bit values

NumNot() Performs a 16-bit "NOT" of a number

`NumOr()` Performs a 16-bit "OR" of a list of numbers

`NumRol()` Performs a 16-bit left rotation of a number

`NumToken()` Determines the number of tokens in a string

`NumXor()` Performs a 16-bit "XOR" of two numbers

`NUMBUFFERS()` Determines the BUFFERS= setting

`NUMCOL()` Restores the number of available screen columns

`NUMDISKF()` Determines the number of installed disk drives

`NUMDISKH()` Determines the number of hard disks

`NUMFILES()` Determines the maximum number of files you can open simultaneously

`NUMFKEY()` Determines the number of function keys

`NUMPRINTER()` Returns the number of parallel ports

`OPENSOCK()` Opens the socket for IPX or SPX communication

`OSVER()` Returns the DOS version number

`OUTBYTE()` Sends a byte to a port

`OUTWORD()` Sends a 16-bit word to a port

`PadLeft()` Pads a string on the left to a particular length

`PadRight()` Pads a string on the right to a particular length

`Payment()` Computes the periodic payment amount

`PAGECOPY()` Copies one screen page to another

`PCType()` Returns the type of computer in use

`Periods()` Computes the number of payment periods necessary to repay a loan

`PEEKBYTE()` Reads a byte from memory

`PEEKSTR()` Reads a byte sequence from memory

PEEKWORD() Reads a 16-bit word from memory

Pi() Returns pi with the highest degree of accuracy

PosAlpha() Determines the position of the first alphabetic character in a string

PosChar() Replaces an individual character at a particular position within a character string

PosDel() Deletes characters at a particular position in a string

PosDiff() Finds the first position from which two strings differ

PosEqual() Finds the first position at which two strings are the same

PosIns() Inserts characters at a particular position within a string

PosLower() Finds the position of the first lower case alphabetic character

PosRange() Determines the position of the first character within a given ASCII code range

PosRepl() Replaces one or more characters from a certain position

PosUpper() Finds the position of the first upper case, alphabetic character

POKEBYTE() Writes a byte to memory

POKEWORD() Writes a 16-bit word to memory

PPCBUFTYP() Determines the communication buffer type

PPCCANCEL() Closes the communication handle and the connected buffers

PPCCONACT() Checks for an active communication connection

PPCEVENT() Queries the event code that most recently triggered a key trap

PPCKEY() Supervises the communication buffer with key traps

PPCREAD() Reads data from a communication receiving buffer

PPCRECCNT() Determines the number of characters in a communication receiving buffer

PPCRECDISC() Determines the number of discarded packets

PPCRECERR() Determines the error code of the most recent receiving process

PPCRECFAIL() Determines the number of packets received with a failure

PPCRECFLSH() Clears a communication receiving buffer

PPCRECFREE() Determines the number of free characters in a communication receiving buffer

PPCRECSIZE() Determines the size of a communication receiving buffer

PPCRECTOT() Determines the total number of received packets

PPCSNDCNT() Determines the number of characters in a communication sending buffer

PPCSNDERR() Determines the error code of the most recent sending process

PPCSNDFAIL() Determines the number of packets sent with failure

PPCSNDFLSH() Clears a communication sending buffer

PPCSNDFREE() Determines the number of free characters in a communication sending buffer

PPCSNDSIZE() Determines the size of a communication sending buffer

PPCSNDTOT() Determines the total number of packets that have been sent

PPCWRITE() Writes data to a communication sending buffer

PrintlINIT() Initializes one of the printers

`PrintReady()` Determines if a particular printer is ready

`PrintSend()` Sends characters directly to a printer

`PrintStat()` Determines the status of a parallel port

`PRINTERERROR()` Returns the error code for the last printer output

`PRINTFILE()` Prints out ASCII files; clears high bits

`PRINTSCR()` Prints screen contents

`PRINTSCRX()` Prints screen contents while it exchanges specific characters

`PV()` Computes the cash present value after interest charges

`Quarter()` Determines the quarter in which a specific date lies

`Rand()` Generates random numbers

`Random()` Generates random numbers

`RangeRem()` Deletes characters that are within a specified ASCII code range

`RangeRepl()` Replaces characters within a specified ASCII code range with a particular character

`Rate()` Computes the interest rate for a loan

`RemAll()` Removes particular characters from the beginning and end of a string

`RemLeft()` Removes particular characters from the beginning of a string

`RemRight()` Removes particular characters at the end of a string

`RenameFile()` Fault tolerant renaming of a file.

`ReplAll()` Exchanges particular characters at the beginning and end of a string

`ReplLeft()` Exchanges particular characters at the beginning of a string

`ReplRight()` Exchanges particular characters at the end of a string

`RestCursor()` Restores a saved cursor position and form

`RestGets()` Restores GET settings from an array

`RestSetKey()` Restores SET KEY..TO settings from an array

`RestToken()` Recreates an incremental tokenizer environment

`RESTFSEEK()` Restores the FILESEEK environment

`RToD()` Converts from a radian to degree measure

`SaveCursor()` Saves current cursor position and form

`SaveGets()` Saves the GET settings of the active environment

`SaveSetKey()` Saves SET KEY..TO settings in an array

`SaveToken()` Saves the incremental tokenizer environment to a variable

`SayDown()` Displays screen output downward and vertically

`SayMoveIn()` Displays screen output with a "move in" effect

`SayScreen()` Output to the screen without changing the attribute

`SaySpread()` Displays screen output with "spread" effect

`SAVEFSEEK()` Saves the current FILESEEK environment

`ScreenAttr()` Determines the attribute at a particular position

`ScreenFile()` Writes screen content to a file

`ScreenMark()` Searches for a string on the screen and marks it with an attribute

`ScreenMix()` Mixes characters and attributes of a screen

`ScreenStr()` Reads a string, including attributes, from the screen

`SCANKEY()` Queries scan code of keyboard input

`SCREENSIZE()` Queries the number of characters that can be displayed on a screen or window

`SecToTime()` Converts seconds into a time string

`SetAtLike()` Provides an additional search mode for all AT functions

`SetBit()` Sets one or more bits in a number

`SetClearA()` Changes the default attribute for screen clear

`SetClearB()` Changes the default character for screen clear

`SetCursor()` Sets the cursor form

`SetDate()` Sets the system date

`SetFAttr()` Sets a file's attributes

`SetFCreate()` Default attribute for creating with CA-Clipper Tools functions

`SetFDaTi()` Sets the date and time of a file

`SetFont()` Loads the font directly out of a string

`SetKXLat()` Redefines key codes or lock keys

`SetKXTab()` Installs key tables

`SetLastKey()` Sets the value for LastKey()

`SetPrec()` Sets the precision level for trigonometric functions

`SetRC()` Sets line and column for the CA-Clipper cursor

`SetTime()` Sets the system clock

`SETBELL()` Sets the tone frequency and duration for Chr(7) or CA-Clipper's "beep"

`SETBOXGROW()` Opens boxes with a time delay

`SETLINES()` Determines the number of lines after which the screen display pauses

`SETMAXCOL()` Sets the number of columns for a virtual screen

`SETMAXROW()` Sets the number of lines for a virtual screen

SETPAGE() Selects a new screen page

SETPBIOS() Redirects print output to the BIOS or DOS, and establishes the timeout

SETPXLAT() Establishes translation tables for printer output

SETQNAME() Changes the file and path name for the QUIT file

SETSCRMODE() Establishes a new video mode

SETSCRSTR() Redirects screen output into a string

SETSHARE() Sets the default opening mode (share mode) for CA-Clipper Tools file functions

SETTAB() Sets the tab widths for CA-Clipper screen outputs

SETTIC() Increases number of time ticks to produce a more precise time measurement

ShowTime() Continuously displays the time at desired screen position

SHOWKEY() Continuously displays the INSERT and LOCK status

Sign() Determines the mathematical sign of a number

Sin() Computes the sine of a radian value

SOUND() Creates tones (melodies) by designating frequency and duration

SPEED() A comparison value used to determine the processor speed

SPOOLACTIV() Determines if the DOS PRINT program is installed

SPOOLADD() Appends a file to the print queue.

SPOOLCOUNT() Determines the number of entries in the print spool queue

SPOOLDEL() Deletes files from the print queue

SPOOLENTRY() Determines the name and path of a print job

SPOOLFLUSH() Completely empty the print queue

SPXCONTARG() Determines the target internet address of an SPX communication

SPXESTBCON() Opens the SPX sending and receiving buffer and tries a connection setup

SPXLISTCON() Opens the SPX sending and receiving buffer and waits for a connection in the background

SSETBREAK() Sets and checks the DOS BREAK switch

SSETVERIFY() Sets and checks the DOS VERIFY switch

Standard() Selects the standard color value for SET COLOR TO output

StrDiff() Finds similarity between two strings (Levenshtein Distance)

StrFile() Writes a string to a file

StrScreen() Displays a string with characters and attributes on the screen

StrSwap() Interchanges two strings

StrToHex() Converts a byte sequence into hexadecimal form

STACKFREE() Determines the remaining stack space

SToD() Converts an ANSI date string into CA-Clipper format

TabExpand() Converts tabs to spaces

TabPack() Converts spaces in tabs

Tan() Computes the tangent of a radian value

TempFile() Creates a file for temporary use

TimeToSec() Calculates the seconds since midnight

TimeValid() Determines whether a specified time is valid

TokenAt() Determines the most recent TokenNext() position within a string

TokenEnd() Determines if more tokens are available in TokenNext()

TokenInit() Initializes a string for TokenNext()

TokenLower() Converts the initial alphabetic character of a token into lower case

TokenNext() Provides an incremental tokenizer

TokenSep() Provides the separator before or after the token most recently retrieved by TOKEN()

TokenUpper() Converts the initial letter of a token into upper case

ToolVer() Queries the version number of the CA-Clipper Tools in use

TOF() Determines if CA-Clipper is at top of form (TOF)

TOKEN() Selects the nth token from a string

TrueName() Standardizes the path designation

TRAPANYKEY() Calls a procedure with any keyboard input

TRAPINPUT() Allows supervision of CA-Clipper input commands

TRAPSHIFT() Calls a procedure that depends on switching keys

Unselected() Selects the unselected color value for SET COLOR TO output

UnTextWin() Replaces an area of characters from a region of the screen

ValPos() Determines the numerical value of the character at a particular position

VGA28() Switches to 28-line VGA mode

VGA50() Switches to 50-line VGA mode.

VGAPalette() Changes VGA palette colors

VideoType() Returns bit-coded information about available video modes

VIDEOINIT() Reinitializes a video system after a RUN

VIDEOSETUP() Queries video mode at system start

VolSerial() Determines the DOS disk serial number

Volume() Establishes a volume label for a floppy or hard disk

WaitPeriod() Pauses a specified time in increments of 1/100 seconds

WAClose() Closes all windows

WBoard() Allocates allowable screen area for windows

WBox() Places a frame around the active window

WCenter() Returns a window to the visible screen area, or centers it

WClose() Closes the active window

WCol() Returns the position of the leftmost column of the active window

Week() Returns the calendar week for a date

WFCol() Returns the position of the leftmost column of the formatted area of a window

WFLastCol() Returns the position of the rightmost column of the formatted area of a window

WFLastRow() Returns the position of the bottom row of the formatted area of a window

WFormat() Determines the usable area within a window

WFRow() Returns the position of the top row of the formatted area of a window

WLastCol() Returns the position of the rightmost column of the active window

[WLastRow\(\)](#) Returns the position of the bottom row of the active window

[WMode\(\)](#) Turns the screen border overstep mode on or off

[WMove\(\)](#) Moves a window

[WNum\(\)](#) Determines the highest window handle

[WoM\(\)](#) Returns the week within a month.

[WordOne\(\)](#) Reduces the multiple appearances of particular double characters to one

[WordOnly\(\)](#) Finds the common denominator between two strings on the basis of double characters

[WordRepl\(\)](#) Replaces particular double characters with others

[WordSwap\(\)](#) Exchanges double characters that lie beside each other in a string

[WordToChar\(\)](#) Exchanges double characters for individual ones

[WOpen\(\)](#) Opens a new window

[WRow\(\)](#) Returns the position of the top row of the active window

[WSelect\(\)](#) Activates one of the open windows

[WSetMove\(\)](#) Turns the interactive movement mode on or off

[WSetShadow\(\)](#) Sets the window shadow colors

[WStep\(\)](#) Determines the step width of interactive window movement

[XMoBlock\(\)](#) Generates a block for XMODEM transmission

[XMoCheck\(\)](#) Tests a received XMODEM block

[XToC\(\)](#) Converts an expression of any data type into a string

[ZeroInsert\(\)](#) Inserts a 0-bit after every fifth 1-bit

[ZeroRemove\(\)](#) Removes 0-bits in a file block

Appendix

[Appendix A - Key Codes](#) A - Key Codes

[Appendix B - MS/PC-DOS Extended Errors](#) B - MS/PC-DOS Extended Errors

[Appendix C - Novell Error Codes](#) C - Novell Error Codes

[Appendix D - IPX and SPX Error Codes](#) D - IPX and SPX Error Codes

[Appendix E - NetBIOS Error Codes](#) E - NetBIOS Error Codes

[Appendix F - Print Server Error Codes](#) F - Print Server Error Codes

Copyright

[CA-Clipper Tools](#) Copyright

gtwww lib

API

[www__MakeDlgTemplate\(\)](#)

[www_AddRows\(\)](#)

[www_AppendMenu\(\)](#)

[www_cbAddString\(\)](#)

[www_cbCreate\(\)](#)

[www_cbDestroy\(\)](#)

[www_cbEnable\(\)](#)

[www_cbFindString\(\)](#)

[www_cbGetCurText\(\)](#)

www_cbGetIndex()
www_cbIsDropped()
www_cbIsFocused()
www_cbSetCodeblock()
www_cbSetCurSel()
www_cbSetFont()
www_cbSetIndex()
www_cxCreate()
www_cxDestroy()
www_cxEnable()
www_cxGetCheck()
www_cxSetCheck()
www_cxSetCodeblock()
www_cxSetFocus()
www_CenterWindow()
www_ChooseColor()
www_ChooseFont()
www_ClientToScreen()
www_CreateDialogDynamic()
www_CreateDialogModal()
www_CreateFont()
www_CreateMenu()
www_CreatePopupMenu()
www_DeleteMenu()
www_DestroyMenu()

www_DlgSetIcon()
www_DrawBoxGet()
www_DrawBoxGroup()
www_DrawBoxRaised()
www_DrawBoxRecessed()
www_DrawButton()
www_DrawColorRect()
www_DrawEllipse()
www_DrawFocusRect()
www_DrawGridHorz()
www_DrawGridVert()
www_DrawImage()
www_DrawLabel()
www_DrawLabelEx()
www_DrawLabelObj()
www_DrawLine()
www_DrawLineEx()
www_DrawMenuBar()
www_DrawOutline()
www_DrawOutlineEx()
www_DrawPicture()
www_DrawProgressBar()
www_DrawRectangle()
www_DrawRoundRect()
www_DrawScrollButton()

www_DrawScrollThumbHorz()

www_DrawScrollThumbVert()

www_DrawShadedRect()

www_DrawStatusBar()

www_DrawTextBox()

www_DrawToolButtonState()

www_EnableMaximize()

www_EnableMenuItem()

www_EnableShortcuts()

www_FillRectangle()

www_GetClipboard()

www_GetCursorPos()

www_GetFontInfo()

www_GetLastMenuEvent()

www_GetMenu()

www_GetPaintRect()

www_GetPalette()

www_GetRGBColor()

www_GetRowColFromXY()

www_GetScreenHeight()

www_GetScreenWidth()

www_GetTitle()

www_GetToolTipBkColor()

www_GetToolTipTextColor()

www_GetToolTipWidth()

wwv_GetWindowHandle()
wwv_GetXYFromRowCol()
wwv_InvalidateRect()
wwv_IsLButtonPressed()
wwv_KillTimer()
wwv_LbAddString()
wwv_LbSetCurSel()
wwv_LCloseWindow()
wwv_LoadFont()
wwv_LoadPen()
wwv_LoadPicture()
wwv_Maximize()
wwv_MaxMaxCol()
wwv_MaxMaxRow()
wwv_MessageBox()
wwv_Minimize()
wwv_nColOfs()
wwv_nNumWindows()
wwv_nOpenWindow()
wwv_nRowOfs()
wwv_nSetCurWindow()
wwv_NoClose()
wwv_NoStartupSubWindow()
wwv_NumBMCache()
wwv_pbCreate()

www_pbDestroy()
www_pbEnable()
www_pbSetCodeblock()
www_pbSetFocus()
www_pbSetFont()
www_pbSetStyle()
www_pgCreate()
www_pgDestroy()
www_pgGetPos()
www_pgSetPos()
www_PasteFromClipboard()
www_ProcessMessages()
www_Restore()
www_RestScreen()
www_sbAddPart()
www_sbCreate()
www_sbDestroy()
www_sbGetParts()
www_sbGetText()
www_sbRefresh()
www_sbSetText()
www_SaveScreen()
www_SetAltF4Close()
www_SetAsNormal()
www_SetBrush()

www_SetClipboard()
www_SetCodepage()
www_SetDefCentreWindow()
www_SetDefHCentreWindow()
www_SetDefLineSpacing()
www_SetDefLSpaceColor()
www_SetDefVCentreWindow()
www_SetFont()
www_SetIcon()
www_SetLastMenuEvent()
www_SetLineSpacing()
www_SetLSpaceColor()
www_SetMainCoord()
www_SetMaxBMCache()
www_SetMenu()
www_SetMenuKeyEvent()
www_SetMouseMove()
www_SetMousePos()
www_SetOnTop()
www_SetPaintRefresh()
www_SetPalette()
www_SetPen()
www_SetPointer()
www_SetPopupMenu()
www_SetTimer()

www_SetTitle()
www_SetToolTip()
www_SetToolTipActive()
www_SetToolTipBkColor()
www_SetToolTipMargin()
www_SetToolTipText()
www_SetToolTipTextColor()
www_SetToolTipTitle()
www_SetToolTipWidth()
www_SetVertCaret()
www_SetWindowCentre()
www_SetWindowPos()
www_SetWinStyle()
www_ShowWindow()
www_tbAddButton()
www_tbButtonCount()
www_tbCmd2Index()
www_tbCreate()
www_tbDelButton()
www_tbDestroy()
www_tbEnableButton()
www_tbIndex2Cmd()
www_TrackPopupMenu()
www_UnreachedBr()
www_UpdateWindow()

[www_xbCreate\(\)](#)
[www_xbDestroy\(\)](#)
[www_xbEnable\(\)](#)
[www_xbInfo\(\)](#)
[www_xbUpdate\(\)](#)
[www_XReposWindow\(\)](#)

Copyright

[GTWVW](#) Copyright

[hbct lib](#)

CT3 date and time functions

[AddMonth\(\)](#) add months to a date
[BoM\(\)](#) Begin of Month
[BoQ\(\)](#) Begin of Quarter
[BoY\(\)](#) Begin of Year
[CToDoW\(\)](#) convert name of day of the week to its ordinal number
[CToMonth\(\)](#) convert name of month to its ordinal number
[DMY\(\)](#) Returns the date as a string in DD Month YY format
[DoY\(\)](#) Determines the day of the year for a specific date
[EoM\(\)](#) End of Month
[EoQ\(\)](#) End of Quarter
[EoY\(\)](#) End of Year

`IsLeap()` determines if year of date is a leap year

`LastDayOM()` Returns the the number of days in the month.

`MDY()` Returns the date as a string in Month DD, YY Or Month DD, YYYY

`NToCDoW()` (num of day) → day name

`NToCMonth()` (num of month) → Month Name

`Quarter()` Returns a number equal to the quarter in which a date falls

`SetDate()` Sets the system date

`SetTime()` Sets the system clock

`TimeValid()` Determines whether a specified time is valid

`WaitPeriod()` Pauses a specified time in increments of 1/100 seconds

`Week()` Returns the calendar week a number

CT3 general functions

`CSetArgErr()` Sets argument error behaviour

CT3 math functions

`Acos()` Arcus cosine of the argument

`Asin()` Arcus sine of the argument

`Atan()` Arcus tangent of the argument

`Atn2()` Arcus tangent a sine and a cosine argument

`Ceiling()` Rounds up a number to the next integer

`Cos()` Cosine of the argument

`Cosh()` Hyperbolic Cosine of the argument

`Cot()` Cotangent of the argument

`DToR()` Convert degree to radiant

`Fact()` Calculates faculty

`Floor()` Rounds down a number to the next integer

`FV()` Future value of a capital

`GetPrec()` Get precision of math functions

`Log10()` Decadic logarithm of a number

`Payment()` Payments for a loan

`Periods()` Number of periods for a loan

`Pi()` Returns Pi, the perimeter-to-diameter-ratio of a circle

`PV()` Present value of a loan

`Rate()` Estimate rate of interest for a loan

`RToD()` Convert radiant to degree

`SetPrec()` Set precision of math functions

`Sign()` Sign of a number

`Sin()` Sine of the argument

`Sinh()` Hyperbolic Sine of the argument

`Tan()` Tangent of the argument

`Tanh()` Hyperbolic Tangent of the argument

CT3 miscellaneous functions

`XToC()`

CT3 number and bit manipulation functions

`BitToC()`

`CToBit()`

`CToF()`

CToN()

Exponent() Evaluate the exponent of a floating point number

FToC()

Mantissa() Evaluate the mantissa of a floating point number

NToC()

CT3 numeric functions

Celsius() Temperature conversion Fahrenheit to Celsius

Fahrenheit() Temperature conversion Celsius to Fahrenheit

Infinity() Returns the largest floating point number available in the system

CT3 printer functions

PrintReady()

PrintStat()

CT3 string functions

AddAscii() Add an integer value to an ASCII value of a string

AfterAtNum() Returns string portion after nth occurrence of substring

AsciiSum() calculate the sum of the ASCII values of the characters in a string

AscPos() ASCII value of a character at a certain position

AtAdjust() Adjusts a sequence within a string to a specified position

AtNum() Returns the start position of the nth occurrence of a substring in a string

AtRepl() Search and replace sequences in a string

AtToken() Position of a token in a string

BeforAtNum() Returns string portion before nth occurrence of substring

CharAdd() Adds corresponding ASCII value of two strings

CharAnd() Combine corresponding ASCII value of two strings with bitwise AND

CharEven() Returns the characters on the even positions in a string

CharHist() Generates a character histogram of a string

CharList() Generates a list of all characters in a string

CharMirr() Mirror a string

CharMix() Mix two strings

CharNoList() Generates a list of all characters not contained in a string

CharNot() Process each character in a string with bitwise NOT operation

CharOdd() Returns the characters on the odd positions in a string

CharOne() Reduce multiple occurrences of a character to one

CharOnly() Intersectional set of two strings based on characters

CharOr() Combine corresponding ASCII value of two strings with bitwise OR

CharRelA() Character relation of two strings

CharRelRep() Relation dependent character replacement

CharRem() Removes characters from a string

CharRepl() Replacement of characters

CharRll() Process each character in a string with bitwise ROLL LEFT operation

CharRlr() Process each character in a string with bitwise ROLL RIGHT operation

CharShl() Process each character in a string with bitwise SHIFT LEFT operation

CharShr() Process each character in a string with bitwise SHIFT RIGHT operation

CharSList() Generates a sorted list of all characters in a string

CharSort() Sort sequences within a string.

CharSub() Subtracts corresponding ASCII value of two strings

CharSwap() Swap neighbouring characters in a string

CharXor() Combine corresponding ASCII value of two strings with bitwise XOR

CountLeft() Count a certain character at the beginning of a string

CountRight() Count a certain character at the end of a string

CSetAtMupa() Determine "multi-pass" behaviour in some string functions

CSetRef() Determine return value of reference sensitive CT3 string functions

JustLeft() Move characters from the beginning to the end of a string

JustRight() Move characters from the end to the beginning of a string

NumAt() Number of occurrences of a sequence in a string

NumToken() Retrieves the number of tokens in a string

PadLeft() Fills string to a certain length on the left

PadRight() Fills string to a certain length on the right

PosAlpha() Left-most position of a letter in a string

`PosChar()` Replace character at a certain position within a string

`PosDel()` Delete characters at a certain position within a string

`PosDiff()` The left-most position where two strings differ

`PosEqual()` The left-most position where two strings begin to be equal

`PosIns()` Insert characters at a certain position within a string

`PosLower()` Left-most position of a lowercase letter in a string

`PosRange()` Left-most position of a character from a set in a string

`PosRepl()` Replace characters at a certain position within a string

`PosUpper()` Left-most position of an uppercase letter in a string

`RangeRem()` Remove characters within a certain ASCII range from a string

`RANGEREPL` Replace characters within a certain ASCII range from a string

`RemAll()` Remove certain characters at the left and right of a string

`RemLeft()` Remove certain characters at the left of a string

`RemRight()` Remove certain characters at the right of a string

`ReplAll()` Replace certain characters at the left and right of a string

`ReplLeft()` Replace certain characters at the left of a string

`ReplRight()` Replace certain characters at the right of a string

`RestToken()` Restore global token environment

`SaveToken()` Save the global token environment

`SetAtLike()` Determine scan behaviour in some string functions

`StrDiff()` Evaluate the "Edit (Levenshtein) Distance" of two strings

`StrSwap()` Swap the contents of two strings

`TabExpand()` Replace tabulator control characters with fill characters

[TabPack\(\)](#) Pack fill characters to appropriate tab characters

[Token\(\)](#) Tokens of a string

[TokenAt\(\)](#) Get start and end positions of tokens in a token environment

[TokenEnd\(\)](#) Check whether additional tokens are available with

[TokenNext\(\)](#)

[TokenExit\(\)](#) Release global token environment

[TokenInit\(\)](#) Initializes a token environment

[TokenLower\(\)](#) Change the first letter of tokens to lower case

[TokenNext\(\)](#) Successively obtains tokens from a string

[TokenNum\(\)](#) Get the total number of tokens in a token environment

[TokenSep\(\)](#) Retrieves the token separators of the last Token() call

[TokenUpper\(\)](#) Change the first letter of tokens to upper case

[ValPos\(\)](#) Numerical value of a character at a certain position

[WordOne\(\)](#) Reduce multiple occurrences of a double character to one

[WordOnly\(\)](#) Intersectional set of two strings based on double characters

[WordRem\(\)](#) Removes characters from a string

[WordRepl\(\)](#) Replacement of double characters

[WordSwap\(\)](#) Swap neighbouring double characters in a string

[WordToChar\(\)](#) Replace double with single characters

CT3 switch and state functions

[KSetCaps\(\)](#)

[KSetIns\(\)](#)

[KSetNum\(\)](#)

[KSetScroll\(\)](#)

CT3 video functions

`CharPix()` Gets the number of scan lines per character.

`CharWin()`

`ColorRepl()`

`ColorToN()`

`ColorWin()`

`Enhanced()` Select the "ENHANCED" color value for output

`InvertAttr()`

`InvertWin()`

`NToColor()`

`SayScreen()`

`ScreenAttr()`

`ScreenMix()`

`SetFont()` Loads font from a string.

`Standard()` Select the "STANDARD" color value for output

`Unselected()` Select the "UNSELECTED" color value for output

`UnTextWin()`

`VGAPalette()` Changes VGA palette colors

`VideoType()` Detects supported video adapter modes

CT3 video functions (Harbour extension)

`ScreenText()`

hbcups lib

API

[cupsGetDefault\(\)](#) Returns the CUPS name of the default printer.

[cupsGetDests\(\)](#) Returns a list of all CUPS printers.

[cupsPrintFile\(\)](#) Prints the named file on the named printer.

hbgd lib

Document

[The GD Library](#) Read me file for GD Library

hbgd

[gdImageArc\(\)](#) Draws a partial ellipse centered at a given point.

[gdImageCreate\(\)](#) Create a palette-based image in memory with no more than 256 colors.

[gdImageCreateFromGd\(\)](#) Load a GD image file.

[gdImageCreateFromGif\(\)](#) Load a GIF image file.

[gdImageCreateFromJpeg\(\)](#) Load a JPEG image file.

[gdImageCreateFromPng\(\)](#) Load a PNG image file.

[gdImageCreateFromWBmp\(\)](#) Load a WBMP image file.

[gdImageCreateTrueColor\(\)](#) Create a true color image in memory.

[gdImageDashedLine\(\)](#) Draws a dashed line between two end points (x1, y1 and x2, y2) with a particular color index.

[gdImageFill\(\)](#) floods a portion of the image with the specified color.

`gdImageFilledArc()` Draws a partial filled ellipse centered at a given point.

`gdImageFilledEllipse()` Draws a filled ellipse centered at a given point.

`gdImageFilledPolygon()` Draws a filled polygon with vertices (at least 3) with a particular color index.

`gdImageFilledRectangle()` Draws a filled rectangle with a particular color index.

`gdImageFillToBorder()` floods a portion of the image with the specified color.

`gdImageGD()` Save a GD image.

`gdImageGif()` Save a GIF image.

`gdImageJpeg()` Save a JPEG image.

`gdImageLine()` Draws a line between two end points (x1, y1 and x2, y2) with a particular color index.

`gdImageOpenPolygon()` Draws an open polygon with vertices (at least 3) with a particular color index.

`gdImagePng()` Save a PNG image.

`gdImagePolygon()` Draws a closed polygon with vertices (at least 3) with a particular color index.

`gdImageRectangle()` Draws a rectangle with a particular color index.

`gdImageSetAntiAliased()` specify the actual foreground color to be used when drawing anti-aliased lines.

`gdImageSetAntiAliasedDontBlend()` indicate the special color that the foreground should stand out more clearly against.

[gdImageSetBrush\(\)](#) A "brush" is an image used to draw wide, shaped strokes in another image.

[gdImageSetPixel\(\)](#) Set a pixel to a particular color index.

[gdImageSetStyle\(\)](#) set any desired series of colors to be repeated during the drawing of a line.

[gdImageSetThickness\(\)](#) determines the width of lines drawn in pixels.

[gdImageSetTile\(\)](#) A "tile" is an image used to fill an area with a repeated pattern.

[gdImageWBmp\(\)](#) Save a WBMP image.

hbgt lib

General

[gt_ClrFlag\(\)](#) Set a number of flags to FALSE in a bit flag string.

[gt_IsFlag\(\)](#) Test the setting of a flag in a bit flag string.

[gt_NewFlag\(\)](#) Create a new bit flag string.

[gt_SetFlag\(\)](#) Set a number of flags to TRUE in a bit flag string.

String Tools

[gt_AsciiSum\(\)](#) Sum the ASCII values in a string.

[gt_AscPos\(\)](#) Return the ASCII value of a specified character in a string

[gt_AtDiff\(\)](#) Return the position where two strings begin to differ

[gt_CharEven\(\)](#) Return a string of all the characters in even positions

[gt_CharMix\(\)](#) Amalgamate two strings to form the return value

[gt_CharOdd\(\)](#) Return a string of all the characters in odd positions

`gt_ChrCount()` Count the number of times a character appears in a string

`gt_ChrFirst()` Find which character occurs first in a string

`gt_ChrTotal()` Find number of times a set of characters appears in a string

`gt_StrCount()` Count the number of times a substring appears in a string

`gt_StrCSPN()` Return length of prefix in string of chars NOT in set.

`gt_StrDiff()` Return a string where it begins to differ from another

`gt_StrExpand()` Insert fillers between characters in a passed string

`gt_StrLeft()` Find length of prefix of a string

`gt_StrPBRK()` Return string after 1st char from a set

`gt_StrRight()` Find length of a suffix of a string

hbmisc lib

Conversion Tools

`BinToDec()` Converts a binary value to decimal

`DecToBin()` Converts a decimal value to binary

`DecToHexa()` Converts a decimal value to hexadecimal

`DecToOctal()` Converts a decimal value to octal

`HexaToDec()` Converts a hexadecimal value to decimal

`IsBin()` Check if the value is a binary number

`IsDec()` Check if the value is a Decimal Number

`IsHexa()` Check if the value is a hexadecimal number

`IsOctal()` Check if the value is a octal number

[OctalToDec\(\)](#) Converts a octal value to decimal

Date

[ADays\(\)](#) Returns an array with the days names.

[AMonths\(\)](#) Returns an array with the months names.

[hbmisc_DaysInMonth\(\)](#) Gets the days in a month.

[IsLeapYear\(\)](#) Checks if the given date is a leap year.

[WoY\(\)](#) Gets the week number of the year.

File

[TFileRead\(\)](#) Read a file one line at a time

hbnf lib

Array

[ft_AAddition\(\)](#) Add elements unique of source array to target array

[ft_AAvg\(\)](#) Average numeric values in an array

[ft_ADesSort\(\)](#) Sort an array in descending order

[ft_AEMaxLen\(\)](#) Find longest element within an array

[ft_AEMinLen\(\)](#) Find shortest element within an array

[ft_AMedian\(\)](#) Find middle value in array, or average of two middle values

[ft_ANoMatches\(\)](#) Find the number of array elements meeting a condition

[ft_ArEdit\(\)](#) 2 dimensional array editing function using TBrowse

`ft_ASum()` Sum the elements of an array
`ft_RestArr()` Restore a Clipper array from a disc file
`ft_SaveArr()` Save Clipper array to a disc file.

Conversion

`ft_Byt2Bit()` Convert byte to string of 1's and 0's
`ft_Byt2Hex()` Convert byte to hexadecimal version of its binary value
`ft_D2E()` Convert decimal to scientific notation
`ft_Dec2Bin()` Convert decimal to binary
`ft_Descend()` Create a descending index key value
`ft_E2D()` Convert scientific notation string to a decimal
`ft_EscCode()` Convert Lotus style escape codes
`ft_Hex2Dec()` Convert a hex number to decimal
`ft_InvClr()` Get the inverse of a color
`ft_NToW()` Translate numeric value to words
`ft_Sqzn()` Compress a numeric value into a character string
`ft_SToD()` Convert a date string to a Clipper date data type
`ft_Unsqzn()` Uncompress a numeric compressed by `ft_Sqzn()`
`ft_XToY()` Convert from any data type to any other data type

Date/Time

`ft_AcctAdj()` Adjust beginning or ending fiscal pd. dates to acctg. dates
`ft_AcctMonth()` Return accounting month data
`ft_AcctQtr()` Return accounting quarter data
`ft_AcctWeek()` Return accounting week data

`ft_AcctYear()` Return accounting year data

`ft_AddWkDy()` Return true number of days to add given number of workdays

`ft_Calendar()` Display date/time calendar, find a date, return calendar data.

`ft_Civ2Mil()` Convert usual civilian format time to military time.

`ft_DateCnfg()` Set beginning of year/week for `ft_*`() date functions

`ft_DayOfYr()` Return calendar, fiscal or accounting day data

`ft_DayToBoW()` Calculate no. of days between date and beginning of week

`ft_DoY()` Find number of day within year

`ft_Easter()` Return the date of Easter

`ft_ElapMin()` Return difference, in minutes, between two mil format times.

`ft_Elapsed()` Return elapsed time between two days and/or times

`ft_ElTime()` Compute difference between times in hours, minutes, seconds.

`ft_FDay()` Return first day of the month

`ft_LDay()` Return last day of the month

`ft_MAdd()` Add or subtract months to/from a date

`ft_Mil2Civ()` Convert time in military format to civilian format.

`ft_Mil2Min()` Convert time in military format to number of minute of day.

`ft_Min2Dhm()` Convert numeric minutes to days, hours and minutes.

`ft_Min2Mil()` Convert minute of day to military format time.

`ft_Month()` Return Calendar or Fiscal Month Data

`ft_Qtr()` Return Calendar or Fiscal Quarter Data.
`ft_Sys2Mil()` Convert system time to military time format.
`ft_Week()` Return calendar or fiscal week data
`ft_Workdays()` Return number of work days between two dates
`ft_WoY()` Find number of week within year
`ft_Year()` Return calendar or fiscal year data

DOS/BIOS

`ft_ChDir()` Change the current directory
`ft_Default()` Retrieve and optionally change the current default drive
`ft_DosVer()` Return the current DOS major and minor version as a string
`ft_DskFree()` Return the amount of available disk space
`ft_DskSize()` Return the maximum capacity of a fixed disk
`ft_FlopTst()` Test diskette drive status
`ft_inp()` Retrieve a byte from a specified I/O port
`ft_int86()` Execute a software interrupt
`ft_IAMIdle()` Inform the operating system that the application is idle.
`ft_IsPrint()` Check printer status
`ft_IsShare()` Determine if DOS "Share" is installed
`ft_MkDir()` Create a subdirectory
`ft_outp()` Write a byte to a specified I/O port
`ft_Peek()` Retrieve a byte from a specified memory location.
`ft_Poke()` Write a byte to a specified memory location
`ft_Reboot()` Force a warm or cold boot
`ft_RmDir()` Delete a subdirectory

`ft_SetDate()` Set the DOS system date
`ft_SetTime()` Set the DOS system time
`ft_SysMem()` Determine the amount of conventional memory installed
`ft_TempFil()` Create a file with a unique name

Environment

`ft_GetE()` Return the entire current environment
`ft_Linked()` Determine if a function was linked in
`ft_Origin()` Report the drive, path and filename of the current program
`ft_RestSets()` Restore status of all SET command settings
`ft_SaveSets()` Save the status of all the SET command settings
`ft_SetCentury()` Check/Set the CENTURY Setting

Event

`ft_Idle()` Generate an idle event to allow incremental garbage collection.
`ft_OnIdle()` Evaluate a designated code block during idle states.
`ft_OnTick()` Evaluate a designated code block at a designated interval.

File I/O

`ft_DFClose()` Close file displayed by `ft_DispFile()`
`ft_DFSetup()` Set up parameters for `ft_DispFile()`
`ft_DispFile()` Browse a text file
`ft_FAppend()` Appends a line to the currently selected text file
`ft_FBoF()` Determine if attempt to skip past beginning of text file
`ft_FDelete()` Deletes a line from the currently selected text file

`ft_FEOF()` Determine if end of text file has been encountered

`ft_FError()` Return the error code for a text file operation

`ft_FGoBot()` Go to the last record in a text file

`ft_FGoto()` Move record pointer to specific record in a text file

`ft_FGoTop()` Go to the first record in a text file

`ft_FInsert()` Inserts a line in the currently selected text file

`ft_FLastRe()` Get the no. of records in the currently selected text file

`ft_FReadLn()` Read a line from the currently selected text file

`ft_FRecNo()` Return the current record number of a text file

`ft_FSelect()` Select a text file workarea

`ft_FSkip()` Move the record pointer to a new position in a text file

`ft_FUse()` Open or close a text file for use by the `ft_F*()` functions

`ft_FWriteLn()` Write a line to the currently selected text file

Game

`ft_Pegs()` PEGS GAME (all work and no play...)

Keyboard/Mouse

`ft_Alt()` Determine status of the Alt key

`ft_CapLock()` Determine and optionally change the status of CapLock key

`ft_Ctrl()` Determine status of the Ctrl key

`ft_LastKey()` Force LastKey() to return a programmer-defined value.

`ft_MButPrs()` Retrieve button press status

`ft_MButRel()` Get mouse button release information

`ft_MCOFF()` Turn mouse cursor off if in specified region

`ft_MCursor()` Set the mouse cursor

`ft_MDbIClk()` Return true if a double click was detected

`ft_MDefCrs()` Define the mouse cursor

`ft_MGetCoord()` Get mouse cursor position (text coord.) and button status

`ft_MGetPage()` Get the display page for the mouse pointer

`ft_MGetPos()` Get mouse cursor position and button status

`ft_MGetSens()` Get the mouse sensitivity parameters

`ft_MGetX()` Get mouse cursor row position

`ft_MGetY()` Get mouse cursor column position

`ft_MHideCrs()` Decrement internal mouse cursor flag and hide mouse cursor

`ft_MInit()` Initialize the mouse driver, vars and return status of mouse

`ft_MInRegion()` Test if the mouse cursor is in the passed region

`ft_MMickeys()` Get mickeys

`ft_MReset()` Reset mouse driver and return status of mouse

`ft_MSetCoord()` Position the mouse cursor using text screen coordinates

`ft_MSetPage()` Set the display page for the mouse pointer

`ft_MSetPos()` Position the mouse cursor using virtual screen coordinates

`ft_MSetSens()` Set the mouse sensitivity parameters

`ft_MShowCrs()` Increment internal cursor flag and display mouse cursor

`ft_MVersion()` Get the mouse driver version

`ft_MXLimit()` Set vertical bounds of mouse using virtual screen coord.

`ft_MYLimit()` Set horizontal bounds of mouse using virtual screen coordinates

`ft_NumLock()` Return status of NumLock key

`ft_PrtScr()` Enable or disable the Print Screen key

`ft_PutKey()` Stuff a keystroke into the keyboard buffer

`ft_ScanCode()` Wait for key-press and return keyboard scan code

`ft_SetRate()` Set the keyboard delay and repeat rate on PC/AT & PS/2

`ft_Shift()` Determine status of shift key

`ft_SInkey()` Replacement for `Inkey()` that tests for SET KEY procedures

Math

`ft_GCD()` Calculate greatest common divisor of two numbers

`ft_NetPV()` Calculate net present value

`ft_Rand1()` Generate a random number

`ft_Round()` Rounds a number to a specific place

Menus/Prompts

`ft_Adder()` Pop up a simple calculator

`ft_Blink()` Display a blinking message on the screen

`ft_BrwsWhl()` Browse an indexed database limited to a while condition

`ft_ClrSel()` User Selectable Color Routine

`ft_DisPMsg()` Display a message and optionally waits for a key-press

`ft_Fill()` Declare menu options for `ft_Menu1()`

`ft_Menu1()` Pulldown menu system

`ft_Menu2()` Vertical lightbar menu

`ft_MenuTo()` Execute light bar menu using prompts created with
`@...PROMPT`

`ft_Pending()` Display same-line pending messages after a wait.

`ft_PickDay()` Picklist of days of week

`ft_Prompt()` Define a menu item for use with `ft_MenuTo()`

`ft_Sleep()` Wait for a specified amount of time

`ft_XBox()` Display a self-sizing message box and message

NetWare

`ft_NWLStat()` Return the current Novell NetWare logical station number

`ft_NWSemClose()` Close a NetWare semaphore

`ft_NWSemEx()` Examine a NetWare semaphore's value and open count

`ft_NWSemLock()` Perform a semaphore "lock"

`ft_NWSemOpen()` Open or create a NetWare semaphore

`ft_NWSemSig()` Signal a NetWare semaphore (increment)

`ft_NWSemUnlock()` "Unlock" a semaphore locked by `ft_NWSemLock()`

`ft_NWSemWait()` Wait on a NetWare semaphore (decrement)

`ft_NWUID()` Return the current Novell NetWare User ID

String

`ft_At2()` Find position of the nth occurrence of a substring

`ft_BitClr()` Clear (reset) selected bit in a byte

`ft_BitSet()` Set selected bit in a byte

`ft_ByteAnd()` Perform bit-wise AND on two ASCII characters (bytes)

`ft_ByteNeg()` Perform bit-wise negation on an ASCII character

[ft_ByteNot\(\)](#) Perform bit-wise NOT on an ASCII character (byte)

[ft_ByteOr\(\)](#) Perform bit-wise OR on two ASCII characters (bytes)

[ft_ByteXor\(\)](#) Perform bit-wise XOR on two ASCII characters (bytes)

[ft_Color2N\(\)](#) Returns the numeric complement of a Clipper color string

[ft_FindIth\(\)](#) Find the "ith" occurrence of a substring within a string

[ft_IsBit\(\)](#) Test the status of an individual bit

[ft_IsBitOn\(\)](#) Determine the state of individual bits in a number

[ft_Metaph\(\)](#) Convert a character string to MetaPhone format

[ft_N2Color\(\)](#) Returns the string complement of a Clipper color number

[ft_NoOccur\(\)](#) Find the number of times one string occurs in another

[ft_PChr\(\)](#) Convert printer control codes

[ft_Proper\(\)](#) Convert a string to proper-name case

[ft_RAt2\(\)](#) Find position of the reversed nth occurrence of a substring

Video

[ft_Adapter\(\)](#) Report the type of video adapter installed

[ft_CLS\(\)](#) Clear screen

[ft_GetMode\(\)](#) Get the video mode

[ft_GetVCur\(\)](#) Return info about the cursor on a specified video page

[ft_GetVPg\(\)](#) Get the currently selected video page

[ft_PopVid\(\)](#) Restore previously saved video states.

[ft_PushVid\(\)](#) Save current video states on internal stack.

[ft_RestAtt\(\)](#) Restore the attribute bytes of a specified screen region.

[ft_RevAttr\(\)](#) Reverse colors of specified screen coordinates

[ft_RevChr\(\)](#) Reverse the color of a single character on the screen

`ft_RgnStack()` Push or pop a saved screen region on or off the stack
`ft_RstRgn()` Restore region of the screen saved with `ft_SavRgn()`
`ft_SaveAtt()` Save the attribute bytes of a specified screen region.
`ft_SavRgn()` Save a screen region for later display
`ft_SetAttr()` Change color attributes of screen region
`ft_SetMode()` Set the video mode
`ft_SetVCur()` Set the cursor position on a specified video page
`ft_SetVPg()` Set the current video page
`ft_Shadow()` Draw a non-destructive shadow on the screen
`ft_VidStr()` Display string on screen in specified attribute
`ft_WrtChr()` Display character on screen

hbxpp lib

API

`Bin2U()` Convert unsigned long encoded bytes into Harbour numeric
`U2Bin()` Convert Harbour numeric into unsigned long encoded bytes
`W2Bin()` Convert Harbour numeric into unsigned short encoded bytes

User interface

`dbSkipper()` Helper function to skip a database

hbziparc lib

Zip Functions

`hb_GetZipComment()` Return the comment of an zip file

[hb_SetBuffer\(\)](#)

[hb_SetDiskZip\(\)](#) Set a codeblock for disk changes

[hb_SetZipComment\(\)](#) Set an Zip archive Comment

[hb_UnzipFile\(\)](#) Unzip a compressed file

[hb_ZipDeleteFiles\(\)](#) Delete files from an zip archive

[hb_ZipFile\(\)](#) Create a zip file

[hb_ZipFileByPKSpan\(\)](#) Create a zip file on removable media

[hb_ZipFileByTDSpan\(\)](#) Create a zip file

[hb_ZipTestPK\(\)](#) Test pkSpanned zip files

rddads lib

Document

[ADS Overview](#) Advantage Database Server RDD

Advantage Database RDD

[AdsBlob2File\(\)](#) Write a Binary (memo) field's contents to a file

[AdsCacheOpenCursors\(\)](#) Provides caching of open cursors

[AdsCacheOpenTables\(\)](#) Provides caching of open tables

[AdsClearAOF\(\)](#) Clears an Advantage Optimized Filter in the current workarea.

[AdsCloseCachedTables\(\)](#) Close all cached tables on the given connection

[AdsClrCallback\(\)](#) Clears the callback set by AdsRegCallback().

[AdsConnect60\(\)](#) Connect to a local/remote/internet server

[AdsCustomizeAOF\(\)](#) Add or remove records from an existing AOF

[AdsDDAddTable\(\)](#) Add a new table to a Data dictionary

[AdsEvalAOF\(\)](#) Evaluate a filter expression to determine its optimization level

[AdsFile2Blob\(\)](#) Save a Binary file to a field

[AdsGetAOF\(\)](#) Retrieve the filter expression used in the call to

[AdsSetAOF\(\)](#)

[AdsGetAOFNoOpt\(\)](#) Return the non-optimized portion of the current filter expression

[AdsGetAOFOptLevel\(\)](#) Returns optimization level of the current AOF filter

[AdsGetConnectionType\(\)](#) Returns the type of Server used by the given connection handle.

[AdsGetLastError\(\)](#) Returns any error code generated by the most recent ADS API call

[AdsGetRelKeyPos\(\)](#) Estimated key position of current record within the current index

[AdsGetTableConType\(\)](#) Returns the type of Server used by current workarea.

[AdsIsExprValid\(\)](#) Determine if the ADS server can parse an expression

[AdsIsIndexed\(\)](#) Fast determination for if current workarea has a selected index

[AdsIsRecordInAOF\(\)](#) Determine if a record is in the current AOF

[AdsKeyCount\(\)](#) Retrieve the number of keys in a specified index

[AdsKeyNo\(\)](#) Get the logical key number of the current record in the given index

[AdsLocking\(\)](#) Turns on/off the Advantage proprietary locking mode

[AdsRefreshAOF\(\)](#) Update the filter snapshot

[AdsRegCallback\(\)](#) For Progress displays: Sets a codeblock to be called during indexing

[AdsRightsCheck\(\)](#) Sets the "rights checking" setting for opening files

[AdsSetAOF\(\)](#) Create an Advantage Optimized Filter

[AdsSetCharType\(\)](#) Sets the type of character data expected when opening tables.

[AdsSetDefault\(\)](#) * Get/Set function for ADS's DEFAULT setting.

[AdsSetDeleted\(\)](#) * Get/Set function for ADS's AdsShowDeleted() setting.

[AdsSetExact\(\)](#) * Get/Set function for ADS's AdsSetExact() setting.

[AdsSetRelKeyPos\(\)](#) Go to the given estimated key position within the current index

[AdsSetSearchPath\(\)](#) * Get/Set function for ADS's AdsSetSearchPath() setting.

[AdsTestRecLocks\(\)](#) Turn On or Off a "debug mode" for trapping missed record locks.

Generated by hbdoc on 2017-10-28 10:02 UTC

Based on revision [f6a35f2](#)

Content processing, layout & design © 2017 [vszakats](#)