Welcome to Harbour^ | ☰ | Improve this doc

A starter's guide
Description

Clipper is a trademark of Computer Associates and will often be referred to as CA-Cl*pper within Harbour documents. Regardless of this variant, Clipper is recognized as Computer Associates' trademark.

Harbour is a free software compiler for the xBase superset language often referred to as Clipper (the language that is implemented by the compiler Clipper). The goal of Harbour is to produce a cross platform CA-Cl*pper compatible compiler.

The Harbour website is at https://vszakats.github.io/harbour-core/. If you have any problem with this copy of Harbour please visit our web site and ensure that you are using the latest release.

If you are reading this file as part of a source distribution of Harbour you probably want to start by reading `doc/dirstruc.txt` because this is your map to the Harbour source directories.

Harbour is a superset of Clipper and is backwards compatible with nearly 100% of all Clipper 5.2x or 5.3 code. Most Clipper S'87 code will also compile and run fine, but may require some modifications to run well.

Platforms
Available on all platforms
Tags
Document, Intro

Compiler Options^ | ☰ | Improve this doc

Description

Invoking the Harbour compiler:

```
harbour <file[.prg]> [options]
```

or

```
harbour [options] <file[.prg]>
```

or

```
harbour [options] <file[.prg]> [options]
```

The command-line options have to be separated by at least one space. The option can start with either - character or / character.

The Harbour command-line options:

`-a` automatic memvar declaration

This causes all variables declared by `PARAMETER`, `PRIVATE` or `PUBLIC` statements to be automatically declared as `MEMVAR` variables.

`-b` debug info

The compiler generates all information required for debugging

---

`-build` display detailed version info

`-credits` display credits

`-d<id>[=<val>]` `#define <id>`

`-es[<level>]` set exit severity

`-es` or `-es0` - all warnings are ignored and exit status returned

by the compiler is equal to 0 if there are no errors in compiled source file.

`-es1` - any warnings generate a non-zero exit status, but

output is still created.

`-es2` - all warnings are treated as errors and no output

file is created. The exit status is set to a non-zero value.

`-fn[:[l|u]|-]` set file name casing (`l`=lower `u`=upper)

`-fd[:[l|u]|-]` set directory casing (`l`=lower `u`=upper)

`-fp[:<char>]` set path separator

`-fs[-]` turn file name space trimming on or off (default)

`-g<type>` output type generated is `type`

-gc[<type>] output type: C source .c (default)

type: 0=compact (default) 1=normal 2=verbose

3=generate real C code

-gh output type: Harbour Portable Object .hrb

-gd[.<destext>] generate dependencies list into .d file

-ge[<mode>] error output mode: 0=Clipper (default)

1=IDE friendly

-i<path> #include file search path

-i[-|+] disable/enable support for INCLUDE envvar

-j[<file>] generate i18n gettext file .pot

-k<mode> compilation mode (type -k? for more data)

-kc clear all flags (strict Clipper mode)

-kh Harbour mode (default)

-ko allow operator optimizations

-ki enable support for HB_INLINE (default)

-kr runtime settings enabled

-ks allow indexed assignment on all types

-kx extended Xbase++ mode (default)

-ku strings in user encoding

-kd accept macros with declared symbols

-km turn off macro-text substitution

-kj turn off jump optimization in pcode

-k? this info

-l suppress line number information

The compiler does not generate the source code line numbers in the output file. The `ProcLine()` function will return 0 for modules compiled using this option.

`-m` compile module only

`-n[<type>]` no implicit starting procedure

`type`: `0`=no implicit starting procedure

`1`=no starting procedure at all `2`=add starting procedure if necessary

The compiler does not create a procedure with the same name as the compiled file. This means that any declarations placed before the first `PROCEDURE` or `FUNCTION` statement have file- wide scope and can be accessed/used in all functions/procedures defined in the compiled source file. All executable statements placed at the beginning of the file and before the first `PROCEDURE/FUNCTION` statement are ignored.

`-o<path>` object file drive and/or path

`-p` generate pre-processed output `.ppo` file

The compiler only creates the file that contains the result of pre-processing the source file.

`-p+` generate pre-processor trace `.ppt` file

`-q` quiet

The compiler does not print any messages during compiling (except the copyright info).

`-q0` quiet and don't display program header

`-q2` disable all output messages

`-ql` suppress line number information

`-r[<lib>]` request linker to search `lib` (or none)

Currently not supported in Harbour.

`-r=<max>` sets maximum number of preprocessor iterations

This set the maximum number of preprocessor iterations during processing the source code. If this switch is not used then the preprocessor stops after 1024 iterations. This value is used to stop processing of infinite loops, for example: `#command ( => (,7`

`-s[m]` syntax check only [minimal for dependencies list]

The compiler checks the syntax only. No output file is generated.

`-t<path>` path for temp file creation

Currently not used in Harbour (the Harbour compiler does not create any temporary files).

`-u[<file>]` use command def set in `file` (or none)

`-u+<file>` add command def set from `file`

`-undef:<id>` `#undef <id>`

`-v` variables are assumed `M->`

All undeclared or unaliased variables are assumed `MEMVAR` variables (private or public variables). If this switch is not used then the scope of such variables is checked at runtime.

`-w[<level>]` set warning level number (0..3, default 1)

`-w0` - no warnings

`-w` or `-w1` - CA-Cl*pper compatible warnings

`-w2` - some useful warnings missed in CA-Cl*pper

`-w3` - warnings generated for Harbour language extensions

and also enables strong type checking but only warns against declared types, or types which may be calculated at compile time

---

`-x[<prefix>]` set symbol init function name prefix (for `.c` only)

Sets the prefix added to the generated symbol init function name (in C output currently). This function is generated automatically for every PRG module compiled. This additional prefix can be used to suppress problems with duplicated symbols during linking an application with some third party libraries.

`-z` suppress shortcutting (`.AND.` & `.OR.`)

Compilation in batch mode.

`@file` compile list of modules in `file`

Not supported yet.

Known incompatibilities between Harbour and CA-Cl*pper compilers

---

Note:

If you want a 100% compatible runtime libraries then you have to define `HB_CLP_STRICT`, using `HB_USER_CFLAGS=-DHB_CLP_STRICT`, then rebuild.

Passing an undeclared variable by the reference

The CA-Cl*pper compiler uses the special opcode `PUSHP` to pass a reference to an undeclared variable (`@` operator). The type of passed variable is checked at runtime (field or memvar). However, field variables cannot be passed by reference. This means that CA-Cl*pper checks the memvar variable only and doesn't look for a field. This is the reason why the Harbour compiler uses the usual `PUSHMEMVARREF` opcode in such cases. Notice that the runtime behavior is the same in CA-Cl*pper and in Harbour - only the generated opcodes are different.

Handling of object messages

The `HB_CLP_STRICT` setting determines the way chained send messages are handled.

For example, the following code:

```
a:b( COUNT() ):c += 1
```

will be handled as:

```
a:b( COUNT() ):c := a:b( COUNT() ):c + 1
```

in strict CA-Cl*pper compatibility mode and

```
temp := a:b( COUNT() ), temp:c += 1
```

in non-strict mode.

In practice, CA-Cl*pper will call the `COUNT()` function two times: the first time before addition and the second one after addition. In Harbour, `COUNT()` will be called only once, before addition.

The Harbour (non-strict) method is:

1) faster

2) it guarantees that the same instance variable of the same object will be changed

(See also: include/hbexpra.c, include/hbexprb.c)

Initialization of static variables

There is a difference in the initialization of static variables that are initialized with a codeblock that refers to a local variable. For example:

```
LOCAL MyLocalVar
STATIC s_MyStaticVar := {|| MyLocalVar }

MyLocalVar := 0
? Eval( s_MyStaticVar )
```

The above code compiles fine in CA-Cl*pper, but it generates a runtime error `Error/BASE 1132 Bound error: array access Called form (b)STATICS$(0)`

In Harbour this code generates a compile time error: `Error E0009 Illegal variable (b) initializer: 'MyLocalVar'`

Both CA-Cl*pper and Harbour are handling all local variables used in a codeblock in a special way: they are detached from the local stack of function/procedure where they are declared. This allows access to these variables after the exit from a function/procedure. However, all static variables are initialized in a separate procedure (`STATICS$` in CA-Cl*pper and `(_INITSTATICS)` in Harbour) before the main procedure and before all `INIT` procedures. The local variables don't exist on the eval stack when static variables are initialized, so they cannot be detached.

Platforms
Available on all platforms
Tags
Document, Compiler

Macro compiler^ | ☰ | **Improve this doc**

Description

Invoking the macro compiler:

```
&variable
```

or

```
&( expression )
```

or

```
&variable.text
```

Platforms
Available on all platforms

Tags
Document, Compiler

Harbour Extensions^ | ☰ | **Improve this doc**

Description

Language extensions:

---

* Class generation and management.

CA-Cl*pper only allowed creation of objects from a few standard classes.

In Harbour, you can create your own classes—complete with Methods, Instance Variables, Class Variables and Inheritance. Entire applications can be designed and coded in Object Oriented style.

* `@<FunctionName>()`

Returns the pointer (address) to a function.

The returned value is not useful to application-level programming, but is used at a low-level to implement object oriented coding. (Internally, a class method is a static function and there is no symbol for it, so it is accessed via its address).

* Class `HBGetList()`

Object oriented support for GetLists management.

* `ProcName()` support for class Method names.

Class Methods can be retrieved from the call stack.

* `Memory()` has new return values.

See `hbmemory.ch`

* `Transform()` → new function in format string

@0 Make a zero padded string out of the number.

* `SToD()` → dDate

New function that converts a `yyyymmdd` string to a Date value.

* Optional Compile Time **strong type** declaration (and compile time

type mismatch warnings)

Example: `LOCAL/STATIC` Var `AS` ...

* The Harbour debugger provides new interesting classes:

- Class `HBDbWindow()` could be the foundation for a generic multi-platform

- Class `HBDbInput()`

- Class `HBDbMenu()` implement both pull-down and popup menus.

RTL enhanced functionality:

- `hb_vfDirSpace( <nDir>, <nType> )`

The second parameter is a Harbour (optional) parameter and indicates the type of disk info being requested. See doc/en/diskspac.txt for info.

Platforms
Available on all platforms
Tag
Document

The Garbage Collector^ | ☰ | Improve this doc

Readme for Harbour Garbage Collect Feature
Description

The garbage collector uses the following logic: - first collect all memory allocations that can cause garbage; - next scan all variables if these memory blocks are still referenced.

Notice that only arrays, objects and codeblocks are collected because these are the only datatypes that can cause self-references `a[ 1 ] := a` or circular references `a[ 1 ] := b; b[ 1 ] := c; c[ 1 ] := a` that cannot be properly deallocated by simple reference counting.

Since all variables in Harbour are stored inside some available tables (the eval stack, memvars table and array of static variables) then checking if the reference is still alive is quite easy and doesn't require any special treatment during memory allocation. Additionally the garbage collector is scanning some internal data used by Harbour objects implementation that also stores some values that can contain memory references. These data are used to initialize class instance variables and are stored in class shared variables.

In special cases when the value of a Harbour variable is stored internally in some static area (at C or assembler level), the garbage collector will be not able to scan such values since it doesn't know their location. This could cause some memory blocks to be released prematurely. To prevent the premature deallocation of such memory blocks the static data have to store a pointer to the value created with `hb_itemNew()` function. Example:

```
static HB_ITEM s_item; /* this item can be released by the GC */
```

```
static PHB_ITEM pItem; /* this item will be maintained correctly */
pItem = hb_itemNew( hb_param( 1, IT_BLOCK ) );
```

However, scanning of all variables can be a time consuming operation. It requires that all allocated arrays have to be traversed through all their elements to find more arrays. Also all codeblocks are scanned for detached local variables they are referencing. For this reason, looking for unreferenced memory blocks is performed during the idle states.

The idle state is a state when there is no real application code executed. For example, the user code is stopped for 0.1 of a second during `Inkey( 0.1 )` - Harbour is checking the keyboard only during this time. It leaves however quite enough time for many other background tasks. One such background task can be looking for unreferenced memory blocks.

Allocating memory

The garbage collector collects memory blocks allocated with `hb_gcAlloc()` function calls. Memory allocated by `hb_gcAlloc()` should be released with `hb_gcFree()` function.

The garbage collecting

During scanning of unreferenced memory the GC is using a mark & sweep algorithm. This is done in three steps:

1) mark all memory blocks allocated by the GC with unused flag;

2) sweep (scan) all known places and clear unused flag for memory blocks that are referenced there;

3) finalize collecting by deallocation of all memory blocks that are still marked as unused and that are not locked.

To speed things up, the mark step is simplified by swapping the meaning of the unused flag. After deallocation of unused blocks all still alive memory blocks are marked with the same 'used' flag so we can reverse the meaning of this flag to 'unused' state in the next collecting. All new or unlocked memory blocks are automatically marked as 'unused' using the current flag, which assures that all memory blocks are marked with the same flag before the sweep step will start. See `hb_gcCollectAll()` and `hb_gcItemRef()`

Calling the garbage collector from Harbour code

The garbage collector can be called directly from the Harbour code. This is useful in situations where there is no idle states available or the application is working in the loop with no user interaction and there is many memory allocations. See `hb_gcAll()` for explanation of how to call this function from your Harbour code.

Platforms
Available on all platforms
Tag
Document
See also
hb_gcAll(), hb_gcAlloc(), hb_gcCollectAll(), hb_gcFree(), hb_gcItemRef(), hb_idleState()

The idle states^ | ☰ | Improve this doc

Readme file for Idle States
Description

The idle state is the state of the Harbour virtual machine when it waits for the user input from the keyboard or the mouse. The idle state is entered during Inkey() calls currently. All applications that don't use Inkey() function call can signal the idle states with the call of hb_idleState() function (or hb_idleState() on C level).

During idle states the virtual machine calls the garbage collector and it can call user defined actions (background tasks). It also releases the CPU time slices for some poor platforms that are not smart enough to detect it automatically.

For defining the background tasks see the hb_idleAdd() and hb_idleDel() functions.

For direct call for background actions see hb_idleState() function.

For signaling the idle state from C code see the hb_idleState() API function.

Platforms
Available on all platforms
Tag
Document
See also
hb_idleAdd(), hb_idleDel()

Do() Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Calls a procedure or a function
Syntax

```
Do( <xFuncProc> [, <xArguments...>] ) → xRetVal
```

Arguments
xFuncProc = Either a string with a function/procedure name to be called or a codeblock to evaluate.
xArguments = arguments passed to a called function/procedure or to a codeblock.
Returns
xRetVal A value that was returned from called function.
Description

This function can be called either by the Harbour compiler or by user. The compiler always passes the item of HB_IT_SYMBOL type that stores the name of procedure specified in DO <proc> WITH ... statement.

If called procedure/function doesn't exist then a runtime error is generated.

This function can be used as replacement of macro operator. It is also used internally to implement `DO <proc> WITH <args...>` In this case `xFuncProc` is of type `HB_SYMB`.

Examples

```
LOCAL cFunction := "MyFunc"

? Do( cFunction, 1 )  // Old style
DO &cFunction WITH 2  // Old style with macro

? Do( {| n | MyFunc( n ) }, 3 )
? Do( @MyFunc(), 4 )

FUNCTION MyFunc( n )  /* must be a public function for old style calls */
   ? n
   RETURN n + 1
```

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Application

`hb_PValue()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Retrieves the value of an argument.

Syntax

```
hb_PValue( <nArg> ) → xExp
```

Arguments

A number that indicates the argument to check.

Returns

`xExp` Returns the value stored by an argument.

Description

This function is useful to check the value stored in an argument.

Examples

```
Test( 123, "abc" )
STATIC PROCEDURE Test( nValue, cString )
   IF PCount() == 2
      ? hb_PValue( 1 ), nValue
      ? hb_PValue( 2 ), cString
   ENDIF
   RETURN
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Application
See also
PCount() Clipper 5.3

PCount() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Retrieves the number of arguments passed to a function.
Syntax
```
PCount() → nArgs
```
Arguments
None
Returns
nArgs A number that indicates the number of arguments passed to a function or procedure.
Description
This function is useful to check if a function or procedure has received the required number of arguments.
Examples
```
Test()
Test( "abc" )
STATIC PROCEDURE Test( xExp )
   IF PCount() == 0
      ? "This function needs a parameter"
   ELSE
      ? xExp
   ENDIF
   RETURN
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Application
See also
hb_PValue()

ProcFile() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function always returns an empty string.
Syntax
```
ProcFile( <xExp> ) → cEmptyString
```
Arguments
xExp is any valid type.
Returns
cEmptyString Return an empty string
Description
This function is added to the RTL for full compatibility. It always returns an empty string.
Examples
```
? ProcFile()
? ProcFile( NIL )
? ProcFile( 2 )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Application
See also
`ProcLine()` Clipper 5.3, `ProcName()` Clipper 5.3

`ProcLine()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Gets the line number of the current function on the stack.
Syntax
```
ProcLine( <nLevel> ) → nLine
```
Arguments
`nLevel` is the function level required.
Returns
`nLine` The line number of the function that it is being executed.
Description
This function looks at the top of the stack and gets the current line number of the executed function if no arguments are passed. Otherwise it returns the line number of the function or procedure at `nLevel`.
Examples
```
? ProcLine( 0 )
? ProcName( 2 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Application
See also
`ProcFile()`, `ProcName()` Clipper 5.3

`ProcName()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Gets the name of the current function on the stack
Syntax
```
ProcName( <nLevel> ) → cProcName
```
Arguments
`nLevel` is the function level required.
Returns
`cProcName` The name of the function that it is being executed.

Description
This function looks at the top of the stack and gets the current executed function if no arguments are passed. Otherwise it returns the name of the function or procedure at `nLevel`.
Examples
```
// This test will show the functions and procedures in stack.
// before executing it.
LOCAL n := 1
DO WHILE ! Empty( ProcName( n ) )
   ? ProcName( n++ )
ENDDO
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Application
See also
`ProcFile()`, `ProcLine()` Clipper 5.3

AAdd() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Dynamically add an element to an array
Syntax
```
AAdd( <aArray>, [<xValue>] ) → xValue
```
Arguments
`aArray` The name of an array
`xValue` Element to add to array `aArray`
Returns
`xValue` if specified `xValue`, `xValue` will be returned, otherwise this function returns a `NIL` value.
Description

This function dynamically increases the length of the `aArray` by adding one new element to the end of the array and optionally stores the value `xValue` to that newly created element.

`xValue` may be of an data type, including an array reference pointer, which in turn may be stored to an array's subscript position.

Examples
```
LOCAL aArray := {}, tmp
AAdd( aArray, 10 )
FOR tmp := 1 TO 10
   AAdd( aArray, tmp )
NEXT
? hb_ValToExp( aArray )  // --> { 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
AIns() Clipper 5.3, ASize() Clipper 5.3

AClone() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Duplicate a multidimensional array
Syntax
```
AClone( <aSource> ) → aDuplicate
```
Arguments
aSource Name of the array to be cloned.
Returns
aDuplicate A new array pointer reference complete with nested array values.
Description
This function makes a complete copy of the array expressed as aSource and return a cloned set of array values. This provides a complete set of arrays values for all dimensions within the original array aSource
Examples
```
LOCAL aOne := { "Harbour", " is ", "POWER" }
LOCAL aTwo := AClone( aOne )
? hb_ValToExp( aTwo )  // --> { "Harbour", " is ", "POWER" }
aOne[ 1 ] := "The Harbour Compiler"
? hb_ValToExp( aOne )  // --> { "The Harbour Compiler", " is ", "POWER" }
? hb_ValToExp( aTwo )  // --> { "Harbour", " is ", "POWER" }
```
Status
Ready
Compliance
CA-Cl*pper will return NIL if the parameter is not an array.
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
ACopy() Clipper 5.3, ADel() Clipper 5.3, AIns() Clipper 5.3, ASize() Clipper 5.3

ACopy() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Copy elements from one array to another
Syntax
```
ACopy( <aSource>, <aTarget>, [<nStart>], [<nCount>], [<nTargetPos>] ) →
aTarget
```
Arguments
aSource is the array to copy elements from.
aTarget is the array to copy elements to.
nStart is the beginning subscript position to copy from aSource

`nCount` the number of subscript elements to copy from `aSource`.
`nTargetPos` the starting subscript position in `aTarget` to copy elements to.
Returns
`aTarget` an array pointer reference
Description

This function copies array elements from `aSource` to `aTarget`. `nStart` is the beginning element to be copied from `aSource`; the default is 1.

`nCount` is the number of elements to be copied from `aSource`; the default is the entire array. `nTargetPos` is the subscript number in the target array, `aTarget`, to which array elements are to be copied; the default is 1.

This function will copy all data types in `aSource` to `aTarget`. If an array element in `aSource` is a pointer reference to another array, that array pointer will be copied to `aTarget`; not all subdimensions will be copied from one array to the next. This must be accomplished via the `AClone()` function.

Note: If array `aSource` is larger then `aTarget`, array elements will start copying at `nTargetPos` and continue copying until the end of array `aTarget` is reached. The `ACopy()` function doesn't append subscript positions to the target array, the size of the target array `aTarget` remains constant.

Examples
```
LOCAL nCount := 2, nStart := 1
LOCAL aOne := { "Harbour", " is ", "Power" }
LOCAL aTwo := { "Clipper", " was ", "Power" }
ACopy( aOne, aTwo, nStart, nCount )
? hb_ValToExp( aTwo )  // --> { "Harbour", " is ", "Power" }
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AClone()` Clipper 5.3, `ADel()` Clipper 5.3, `AEval()` Clipper 5.3, `AFill()` Clipper 5.3, `AIns()` Clipper 5.3, `ASort()` Clipper 5.3

`ADel()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Delete an element from an array.
Syntax
```
ADel( <aArray>, <nPos> ) → aArray
```
Arguments
`aArray` Name of array from which an element is to be removed.
`nPos` Subscript of the element to be removed.
Returns

`aArray` an array pointer reference.
Description

This function deletes the element found at `nPos` subscript position in the array `aArray`. All elements in the array `aArray` below the given subscript position `nPos` will move up one position in the array.

In other words, what was formerly the sixth subscript position will become the fifth subscript position. The length of the array `aArray` will remain unchanged, as the last element in the array will become a `NIL` data type.

Note: To completely remove an element and decrease the length of array you can use `hb_ADel()` that supports auto-sizing.

Examples

```
LOCAL aArray := { "Harbour", "is", "Power" }
ADel( aArray, 2 )
? hb_ValToExp( aArray )  // --> { "Harbour", "Power", NIL }
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`ACopy()` Clipper 5.3, `AFill()` Clipper 5.3, `AIns()` Clipper 5.3, `hb_ADel()`

`AEval()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Evaluates the subscript element of an array
Syntax

```
AEval( <aArray>, <bBlock>, [<nStart>], [<nCount>] ) → aArray
```

Arguments
`aArray` Is the array to be evaluated.
`bBlock` Is a code block to evaluate for each element processed.
`nStart` The beginning array element index to evaluate.
`nCount` The number of elements to process.
Returns
`aArray` an array pointer reference.
Description

This function evaluates the elements of `aArray`, and executes for each of them the processing that's defined with `bBlock`. By default, all the elements of array are being processed, starting from 1st and up to the last element.

If `nStart` and/or `nCount` parameters are given, then the processing starts from `nStart` element and continues for the next `nCount` elements (if defined) or up to the last element of the array.

The `bBlock` code block receives two parameters: the element value and element's index (position) into the array. i.e. `{| xValue, nIndex | ... }` Worth to note that elements are passed to code block 'by value', thus any change being made to this value doesn't affects the value of element in the array.

Examples

```
LOCAL a := { 10, 20, 30 }
AEval( a, {| e, n | QOut( e * n + 1 , a[ n ] ) } )
? a[ 1 ], a[ 2 ], a[ 3 ]  // array elements unchanged
? "----"
AEval( a, {| e, n | QOut( e * n + 1, a[ n ] *= n + 1 ) }, 2, 1 )
/* Here the 2nd element been changed, because we've explicitly used
   its pointer 'a[ n ] *= ...' into array */
? a[ 1 ], a[ 2 ], a[ 3 ]
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`dbEval()` Clipper 5.3, `Eval()` Clipper 5.3

`AFill()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Fill an array with a specified value
Syntax

```
AFill( <aArray>, <xValue>, [<nStart>], [<nCount>] ) → aArray
```

Arguments
`aArray` Name of array to be filled.
`xValue` Expression to be globally filled in `aArray`
`nStart` Subscript starting position
`nCount` Number of subscript to be filled
Returns
`aArray` pointer to the array.
Description

This function will fill each element of an array named `aArray` with the value `xValue`. If specified, `nStart` denotes the beginning element to be filled and the array elements will continue to be filled for `nCount` positions.

If neither `nStart`/`nCount` specified, the value of `nStart` will be 1, and the value of `nCount` will be the value of `Len( <aArray> )`; thus, all subscript positions in the array `aArray` will be filled with the value of `xValue`.

This function will work on only a single dimension of `aArray`. If there are array pointer references within a subscript `aArray`, those values will be lost, since this function will overwrite those values with new values.

Examples

```
LOCAL aArray := { NIL, 0, 1, 2 }
AFill( aArray, 5 )
? hb_ValToExp( aArray )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AAdd()` Clipper 5.3, `AEval()` Clipper 5.3, `dbStruct()` Clipper 5.3

`AIns()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Insert a NIL value at an array subscript position.
Syntax

```
AIns( <aArray>, <nPos> ) → aArray
```

Arguments
`aArray` Array name.
`nPos` Subscript position in `aArray`
Returns
a reference to `aArray`.
Description

This function inserts a `NIL` value in the array named `aArray` at the `nPos`th position.

All array elements starting with the `nPos`th position will be shifted down one subscript position in the array list and the last item in the array will be removed completely. In other words, if an array element were to be inserted at the fifth subscript position, the element previously in the fifth position would now be located at the sixth position. The length of the array `aArray` will remain unchanged.

Note: To avoid loosing last element, you can use `hb_AIns()` which supports auto-sizing of array.

Examples

```
LOCAL aArray := { "Harbour", "is", "Power!", "!!!" }
AIns( aArray, 4 )
? hb_ValToExp( aArray )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
AAdd() Clipper 5.3, ACopy() Clipper 5.3, ADel() Clipper 5.3, AEval() Clipper 5.3, AFill() Clipper 5.3, ASize() Clipper 5.3, hb_AIns()

Array() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create an uninitialized array of specified length
Syntax
```
Array( <nElements>[, <nElements>...] ) → aArray
```
Arguments
`nElements` is the number of elements in the specified dimension.
Returns
`aArray` an array of specified dimensions.
Description

This function returns an uninitialized array with the length of `nElements`.

Nested arrays are uninitialized within the same array pointer reference if additional parameters are specified.

Establishing a memory variable with the same name as the array may destroy the original array and release the entire contents of the array. This depends, of course, on the data storage type of either the array or the variable with the same name as the array.

CA-Cl*pper v5.x compliant except that arrays in Harbour can have an unlimited number of elements.

Examples
```
LOCAL aArray := Array( 10 ), tmp
FOR tmp := 1 TO Len( aArray )
   aArray[ tmp ] := Array( tmp )
NEXT
? hb_ValToExp( aArray )  // --> { { NIL }, { NIL, NIL }, ... }
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
AAdd() Clipper 5.3, ADel() Clipper 5.3, AFill() Clipper 5.3, AIns() Clipper 5.3

AScan() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Scan array elements for a specified condition
Syntax
```
AScan( <aArray>, <xSearch>, [<nStart>], [<nCount>] ) → nStoppedAt
```
Arguments
`aArray` Array to be scanned.
`xSearch` Expression to search for in `aTarget`
`nStart` Beginning subscript position at which to start the search.
`nCount` Number of elements to scan with `aTarget`.
Returns
`nStoppedAt` A numeric value of subscript position where `xSearch` was found, or 0 if `xSearch` is not found.
Description

This function scan the content of array named `aArray` for the value of `xSearch`. The return value is the position in the array `aArray` in which `xSearch` was found. If it was not found, the return value will be 0.

`nStart` is the position from which to start scanning. The default is 1. (1st element) `nCount`, if specified, is the number of array elements to be scanned. The default is all elements in the array `aArray`.

If `xSearch` is a code block, the operation of the function is slightly different. Each array subscript pointer reference is passed to the code block to be evaluated. The scanning routine will continue until the value obtained from the code block is a logical true (`.T.`) or until the end of the array has been reached.

Examples
```
#include "directry.ch"
LOCAL aDir := hb_vfDirectory( "*.prg" )
? AScan( aDir,,, ;
   {| aFile, nPos | HB_SYMBOL_UNUSED( nPos ), aFile[ F_NAME ] == "test.prg"
} )
```
Status
Ready
Compliance
This function is not CA-Cl*pper compatible. CA-Cl*pper `AScan()` is affected by the `SET EXACT ON/OFF` Condition
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AEval()` Clipper 5.3

ASize() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Adjust the size of an array
Syntax

```
ASize( <aArray>, <nLen> ) → aArray
```
Arguments
`aArray` Name of array to be dynamically altered
`nLen` Numeric value representing the new size (i.e. number of elements) of `aArray`
Returns
The function returns a reference to `aArray`.
Description

This function will dynamically increase or decrease the size of `aArray` by adjusting the length of the array to `nLen` subscript positions.

If the length of the array `aArray` is shortened, the redundant elements are removed from the end of array. If the length of the array is lengthened the new elements are added to the end of array and they are assigned a `NIL` value.

Examples
```
LOCAL aArray := { 1 }
? hb_ValToExp( aArray )  // --> { 1 }
ASize( aArray, 3 )
? hb_ValToExp( aArray )  // --> { 1, NIL, NIL }
ASize( aArray, 1 )
? hb_ValToExp( aArray )  // --> { 1 }
```
Status
Ready
Compliance
If `HB_COMPAT_C53` is defined, the function generates an Error, else it will return the array itself.
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AAdd()` Clipper 5.3, `ADel()` Clipper 5.3, `AFill()` Clipper 5.3, `AIns()` Clipper 5.3

`ASort()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sort an array
Syntax
```
ASort( <aArray>, [<nStart>], [<nCount>], [<bSort>] ) → aArray
```
Arguments
`aArray` Array to be sorted.
`nStart` The first element to start the sort from, default is 1.
`nCount` Number of elements starting from `nStart` to sort, default is all elements.
`bSort` Code block for sorting order, default is ascending order `{| x, y | x < y }`. The code block should accept two parameters and must return `.T.` if the sort is in order, `.F.` if not.
Returns
`aArray` reference to the now sorted `aArray` or `NIL` if the passed `aArray` is not an array.
Description

`ASort()` sort all or part of a given array. If `bSort` is omitted, the function expect `aArray` to be one dimensional array containing single data type (one of: Character, Date, Logical, Numeric) and sort this array in ascending order: Character are sorted by their ASCII value, Dates are sorted chronologically, Logical put `.F.` values before `.T.`, Numeric are sorted by their value. If `bSort` is specified, it is used to handle the sorting order. With each time the block is evaluate, two array elements are passed to the code block, and `bSort` must return a logical value that state if those elements are in order (`.T.`) or not (`.F.`). Using this block you can sort multidimensional array, descending orders or even (but why would you want to do that) sort array that contain different data type.

Codeblock calling frequency and order differs from CA-Cl*pper, since Harbour uses a different (faster) sorting algorithm (quicksort).

Examples

```
LOCAL aKeys, bSort, aPair

// Sort numeric values in ascending order
aKeys := { 3, 1, 4, 42, 5, 9 }
ASort( aKeys )
? hb_ValToExp( aKeys )  // --> { 1, 3, 4, 5, 9, 42 }

// Sort character strings in descending lexical order
aKeys := { "Ctrl", "Alt", "Delete" }
bSort := {| x, y | Upper( x ) > Upper( y ) }
ASort( aKeys,,, bSort )
? hb_ValToExp( aKeys )  // --> { "Delete", "Ctrl", "Alt" }

// Sort two-dimensional array according to 2nd element of each pair
aPair := { { "Sun", 8 }, { "Mon", 1 }, { "Tue", 57 }, { "Wed", -6 } }
ASort( aPair,,, {| x, y | x[ 2 ] < y[ 2 ] } )
? hb_ValToExp( aPair )  // --> { { "Wed", -6 }, { "Mon", 1 }, { "Sun", 8 },
{ "Tue", 57 } }
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Array

See also

`AScan()` Clipper 5.3, `Eval()` Clipper 5.3, `SORT`

`ATail()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the last element of an array

Syntax

```
ATail( <aArray> ) → xValue
```

Arguments

`aArray` is the array.

Returns

`xValue` the value of the last element in the array.

Description
This function return the value of the last element in the array named `aArray`. Same as `xValue := aArray[ Len( aArray ) ]`
Examples

```
LOCAL aArray := { "Harbour", "is", "Supreme", "Power" }
? ATail( aArray )  // --> "Power"
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AAdd()` Clipper 5.3, `Array()` Clipper 5.3, `ASize()` Clipper 5.3, `Len()` Clipper 5.3

`hb_ADel()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Delete an element from an array.
Syntax

```
hb_ADel( <aArray>, [<nPos>], [<lAutoSize>] ) → aArray
```

Arguments
`aArray` Name of array from which an element is to be removed.
`nPos` Subscript of the element to be removed. Default value: 1.
`lAutoSize` Boolean flag specifying if the array will be resized or not.
Default value: `.F.` (no resize).
Returns
`aArray` an array pointer reference.
Description

This function deletes the element value (not the element itself!) stored in position `nPos` and shifts all the following values, one position up.

If `lAutoSize` is `.T.`, then the last element is removed and the size of the array is decreased by one, otherwise the length of the array remains unchanged and a `NIL` value will be stored in the last element, just like in `ADel()`.

Examples

```
LOCAL aArray := { "Harbour", "is", "Power" }
hb_ADel( aArray, 2 )
? hb_ValToExp( aArray )  // --> { "Harbour", "Power", NIL } - length unchanged
hb_ADel( aArray, 2, .T. )
? hb_ValToExp( aArray )  // --> a{ "Harbour", "Power" } - length decreased
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms

File
Library is core
Tags
API, Array
See also
ACopy() Clipper 5.3, ADel() Clipper 5.3, AFill() Clipper 5.3, AIns() Clipper 5.3

hb_AIns() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Inserts a value at an array subscript position and optionally increases the length of array.
Syntax
```
hb_AIns( <aArray>, [<nPos>], [<xValue>], [<lAutoSize>] ) → aArray
```
Arguments
`aArray` The array name into which the value `xValue` will be inserted.
`nPos` Subscript position in `aArray`. Default: 1st position
`xValue` Value to be inserted
`lAutoSize` Boolean flag to increase or not the size of `aArray`.
Default value: `.F.`
Returns
A reference to array `aArray`
Description

This function inserts `xValue` in the `nPos` position of the array, moving all the items one position down in the array list. If `lAutoSize` is `.T.`, a new element will be added at the end of array, making room for the previous last element, which means the length of array will be increased by 1.

If `lAutoSize` is `.F.` (or is not passed) the function behaves like `AIns()`, that is, the size of `aArray` won't change and the last item of `aArray` will be lost.

Examples
```
LOCAL aArray := { "Harbour", "Power!" }
hb_AIns( aArray, 2, "is", .F. )
? hb_ValToExp( aArray )  // --> { "Harbour", "is" }
hb_AIns( aArray, 2, "is", .T. )
? hb_ValToExp( aArray )  // --> { "Harbour", "is", "Power!" }
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
AAdd() Clipper 5.3, ADel() Clipper 5.3, AFill() Clipper 5.3, AIns() Clipper 5.3, ASize() Clipper 5.3

hb_AScan() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Scan array elements for a specified condition
Syntax
`hb_AScan( <aArray>, <xSearch>, [<nStart>], [<nCount>], [<lExact>] ) → nPosition`
Arguments
`aArray` Array to be scanned.
`xSearch` Expression to search for in `aArray`
`nStart` Beginning subscript position at which to start the search.
Default value: 1
`nCount` Number of elements to be scanned within `aArray`.
Default value: All elements.
`lExact` Boolean flag specifying if an "Exact" search will be performed or not. Default value: `.F.`.
Returns
`nPosition` A numeric value > 0 indicating the array position where `xSearch` was found, or 0 if nothing found.
Description

The function scans (left to right) for `xSearch` into `aArray` and returns `nPosition` of `aArray` in which `xSearch` was found or 0 (zero) if nothing found.

If `lExact` is `.T.`, then an exact search will be performed. When `xSearch` is a code block, the operation of the function is slightly different. See `AScan()` for details.

Examples
```
LOCAL a := { "there", "here" }
Set( _SET_EXACT, .F. )
? AScan( a, "he" )             // --> 2
? hb_AScan( a, "he",,, .F. )   // --> 2
? hb_AScan( a, "he",,, .T. )   // --> 0
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Array
See also
`AEval()` Clipper 5.3, `AScan()` Clipper 5.3

`HBClass()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

HBClass() is used in the creation of all classes
Syntax
```
oClass := HBClass():New( "TMyClass" )

  -or-

HBClass() is usually accessed by defining a class with the commands
defined in hbclass.ch:
```

```
CREATE CLASS HBGetList  // Calls HBClass() to create the HBGetList class
   ...
ENDCLASS
```

Returns

An instance of the `HBClass()` Class. This special object's `:New()` method can then create the classes you define.

Description

`HBClass()` is a class that ... The class methods are as follows:

`:New()` Create a new instance of the class

Examples

```
LOCAL oObject := HBClass():New( "TMyClass" )

oObject:End()
```

Status

Ready

Compliance

Object Oriented syntax in Harbour is compatible with CA-Cl*pper.
However CA-Cl*pper only allowed creation of objects from a few standard classes, and did not let the programmer create new classes.
In Harbour, you can create your own classes—complete with Methods, Instance Variables, Class Variables and Inheritance. Entire applications can be designed and coded in Object Oriented style.

Platforms

Available on all platforms

File

Library is core

Tags

API, Classes

See also

`__objHasData()`, `CLASS`

`Bin2I()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert signed short encoded bytes into Harbour numeric

Syntax

```
Bin2I( <cBuffer> ) → nNumber
```

Arguments

`cBuffer` is a character string that contains 16-bit encoded signed short integer (least significant byte first). The first two bytes are taken into account, the rest if any are ignored.

Returns

`Bin2I()` return numeric integer (or 0 if `cBuffer` is not a string).

Description

`Bin2I()` is one of the low-level binary conversion functions, those functions convert between Harbour numeric and a character representation of numeric value. `Bin2I()` take two bytes of encoded 16-bit signed short integer and convert it into standard Harbour numeric value.

You might ask what is the need for such functions, well, first of all it allow you to read/write information from/to a binary file (like extracting information from DBF header), it is also a useful way to share information from source other than Harbour (C for instance).

`Bin2I()` is the opposite of `I2Bin()`

Examples

```
// Show DBF last update date
#include "fileio.ch"
LOCAL hFile, cYear, cMonth, cDay
IF ( hFile := hb_vfOpen( "test.dbf", FO_READ ) ) != NIL
   hb_vfSeek( hFile, 1 )
   cYear := cMonth := cDay := hb_BChar( 0 )
   hb_vfRead( hFile, @cYear , hb_BLen( cYear ) )
   hb_vfRead( hFile, @cMonth, hb_BLen( cMonth ) )
   hb_vfRead( hFile, @cDay  , hb_BLen( cDay ) )
   ? "Last update:", Bin2I( cYear ), Bin2I( cMonth ), Bin2I( cDay )
   hb_vfClose( hFile )
ELSE
   ? "Cannot open file"
ENDIF
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Conversion

See also

`Bin2L()` Clipper 5.3, `Bin2U()`, `Bin2W()` Clipper 5.3, `I2Bin()` Clipper 5.3, `L2Bin()` Clipper 5.3, `U2Bin()`, `W2Bin()`, `Word()` Clipper 5.3

`Bin2L()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert signed long encoded bytes into Harbour numeric

Syntax

```
Bin2L( <cBuffer> ) → nNumber
```

Arguments

`cBuffer` is a character string that contains 32-bit encoded signed long integer (least significant byte first). The first four bytes are taken into account, the rest if any are ignored.

Returns

`Bin2L()` return numeric integer (or 0 if `cBuffer` is not a string).

Description

`Bin2L()` is one of the low-level binary conversion functions, those functions convert between Harbour numeric and a character representation of numeric value. `Bin2L()` take four bytes of encoded 32-bit signed long integer and convert it into standard Harbour numeric value.

You might ask what is the need for such functions, well, first of all it allow you to read/write information from/to a binary file (like extracting information from DBF header), it is also a useful way to share information from source other than Harbour (C for instance).

`Bin2L()` is the opposite of `L2Bin()`

Examples

```
// Show number of records in DBF
#include "fileio.ch"
LOCAL hFile, cBuffer := Space( 4 )
IF ( hFile := hb_vfOpen( "test.dbf", FO_READ ) ) != NIL
   hb_vfSeek( hFile, 4 )
   hb_vfRead( hFile, @cBuffer, hb_BLen( cBuffer ) )
   ? "Number of records in file:", Bin2L( cBuffer )
   hb_vfClose( hFile )
ELSE
   ? "Cannot open file"
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Conversion
See also
`Bin2I()` Clipper 5.3, `Bin2U()`, `Bin2W()` Clipper 5.3, `I2Bin()` Clipper 5.3, `L2Bin()` Clipper 5.3, `U2Bin()`, `W2Bin()`, `Word()` Clipper 5.3

`Bin2W()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert unsigned short encoded bytes into Harbour numeric
Syntax

```
Bin2W( <cBuffer> ) → nNumber
```

Arguments
`cBuffer` is a character string that contains 16-bit encoded unsigned short integer (least significant byte first). The first two bytes are taken into account, the rest if any are ignored.
Returns
`Bin2W()` return numeric integer (or 0 if `cBuffer` is not a string).
Description

`Bin2W()` is one of the low-level binary conversion functions, those functions convert between Harbour numeric and a character representation of numeric value. `Bin2W()` take two bytes of encoded 16-bit unsigned short integer and convert it into standard Harbour numeric value.

You might ask what is the need for such functions, well, first of all it allow you to read/write information from/to a binary file (like extracting information from DBF header), it is also a useful way to share information from source other than Harbour (C for instance).

`Bin2W()` is the opposite of `W2Bin()`

Examples

```
// Show header length of a DBF
#include "fileio.ch"
```

```
LOCAL hFile, cBuffer := Space( 2 )
IF ( hFile := hb_vfOpen( "test.dbf", FO_READ ) ) != NIL
   hb_vfSeek( hFile, 8 )
   hb_vfRead( hFile, @cBuffer, hb_BLen( cBuffer ) )
   ? "Length of DBF header in bytes:", Bin2W( cBuffer )
   hb_vfClose( hFile )
ELSE
   ? "Cannot open file"
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Conversion
See also
Bin2I() Clipper 5.3, Bin2L() Clipper 5.3, Bin2U(), I2Bin() Clipper 5.3, L2Bin() Clipper 5.3, U2Bin(), W2Bin(), Word() Clipper 5.3

Descend() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Inverts an expression of string, logical, date or numeric type.
Syntax
```
Descend( <xExp> ) → xExpInverted
```
Arguments
xExp is any valid expression.
Returns
Inverted value of the same type as passed.
Description
This function converts an expression in his inverted form. It is useful to build descending indexes.
Examples
```
// FIXME
// Seek for Smith in a descending index
dbSeek( Descend( "SMITH" ) )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Conversion
See also
INDEX, SEEK

I2Bin() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert Harbour numeric into signed short encoded bytes

Syntax

`I2Bin( <nNumber> ) → cBuffer`

Arguments

`nNumber` is a numeric value to convert (decimal digits are ignored).

Returns

`I2Bin()` return two bytes character string that contains 16-bit encoded signed short integer (least significant byte first).

Description

`I2Bin()` is one of the low-level binary conversion functions, those functions convert between Harbour numeric and a character representation of numeric value. `I2Bin()` take a numeric integer value and convert it into two bytes of encoded 16-bit signed short integer.

You might ask what is the need for such functions, well, first of all it allow you to read/write information from/to a binary file (like extracting information from DBF header), it is also a useful way to share information from source other than Harbour (C for instance).

`I2Bin()` is the opposite of `Bin2I()`

Examples

```
// Update DBF "last update" date
#include "fileio.ch"
LOCAL hFile, cYear, cMonth, cDay
USE test
? "Original update date is:", LUpdate()
dbCloseArea()
IF ( hFile := hb_vfOpen( "test.dbf", FO_READWRITE ) ) != NIL
   hb_vfSeek( hFile, 1 )
   cYear  := I2Bin( 68 )
   cMonth := I2Bin(  8 )
   cDay   := I2Bin(  1 )
   hb_vfWrite( hFile, cYear , 1 )  // write only the first byte
   hb_vfWrite( hFile, cMonth, 1 )
   hb_vfWrite( hFile, cDay  , 1 )
   hb_vfClose( hFile )
   USE test
   ? "New update date is:", LUpdate()
   dbCloseArea()
ELSE
   ? "Cannot open file"
ENDIF
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Conversion

See also

`Bin2I()` Clipper 5.3, `Bin2L()` Clipper 5.3, `Bin2U()`, `Bin2W()` Clipper 5.3, `L2Bin()` Clipper 5.3, `U2Bin()`, `W2Bin()`, `Word()` Clipper 5.3

L2Bin() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert Harbour numeric into signed long encoded bytes
Syntax
```
L2Bin( <nNumber> ) → cBuffer
```
Arguments
`nNumber` is a numeric value to convert (decimal digits are ignored).
Returns
`L2Bin()` return four bytes character string that contains 32-bit encoded signed long integer (least significant byte first).
Description

`L2Bin()` is one of the low-level binary conversion functions, those functions convert between Harbour numeric and a character representation of numeric value. `L2Bin()` take a numeric integer value and convert it into four bytes of encoded 32-bit signed long integer.

You might ask what is the need for such functions, well, first of all it allow you to read/write information from/to a binary file (like extracting information from DBF header), it is also a useful way to share information from source other than Harbour (C for instance).

`L2Bin()` is the opposite of `Bin2L()`

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Conversion
See also
`Bin2I()` Clipper 5.3, `Bin2L()` Clipper 5.3, `Bin2U()`, `Bin2W()` Clipper 5.3, `I2Bin()` Clipper 5.3, `U2Bin()`, `W2Bin()`, `Word()` Clipper 5.3

Word() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts double to integer values.
Syntax
```
Word( <nDouble> ) → nInteger
```
Arguments
`nDouble` is a numeric double value.
Returns
`Word()` return an integer in the range +-32767
Description
This function converts double values to integers to use within the `CALL` command
Status
Ready
Compliance

The CA-Cl*pper NG states that `Word()` will only work when used in `CALL` commands parameter list, otherwise it will return `NIL`, in Harbour it will work anywhere.
Platforms
Available on all platforms
File
Library is core
Tags
API, Conversion
See also
`CALL`

`__dbCopyStruct()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a new database based on current database structure
Syntax

```
__dbCopyStruct( <cFileName>, [<aFieldList>] )
```

Arguments
`cFileName` is the name of the new database file to create. `.dbf` is the default extension if none is given.
`aFieldList` is an array where each element is a field name. Names could be specified as uppercase or lowercase.
Description

`__dbCopyStruct()` create a new empty database file with a structure that is based on the currently open database in this work-area. If `aFieldList` is empty, the newly created file would have the same structure as the currently open database. Else, the new file would contain only fields that exactly match `aFieldList`.

`__dbCopyStruct()` can be use to create a sub-set of the currently open database, based on a given field list.

`COPY STRUCTURE` command is preprocessed into `__dbCopyStruct()` function during compile time.

Examples

```
// Create a new file that contain the same structure
USE test
__dbCopyStruct( "mycopy.dbf" )

// Create a new file that contain part of the original structure
LOCAL aList
USE test
aList := { "NAME" }
__dbCopyStruct( "onlyname.dbf", aList )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags

API, Database
See also
__dbCopyXStruct(), __dbCreate(), __dbStructFilter(), COPY STRUCTUREClipper 5.3, COPY STRUCTURE EXTENDEDClipper 5.3, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

__dbCopyXStruct() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Copy current database structure into a definition file
Syntax
```
__dbCopyXStruct( <cFileName> ) → lSuccess
```
Arguments
cFileName is the name of target definition file to create. .dbf is the default extension if none is given.
Returns
__dbCopyXStruct() returns .F. if no database is **used** in the current work-area, .T. on success, or a run-time error if the file create operation had failed.
Description

__dbCopyXStruct() create a new database named cFileName with a pre-defined structure (also called "structure extended file"):

Field name	Type	Length	Decimals
FIELD_NAME	C	10	0
FIELD_TYPE	C	1	0
FIELD_LEN	N	3	0
FIELD_DEC	N	3	0

Each record in the new file contains information about one field in the original file. CREATE FROM could be used to create a database from the structure extended file.

For prehistoric compatibility reasons, Character fields which are longer than 255 characters are treated in a special way by writing part of the length in the FIELD_DEC according to the following formula (this is done internally):

```
FIELD->FIELD_DEC := Int( nLength / 256 )
FIELD->FIELD_LEN :=      nLength % 256
```

Later if you want to calculate the length of a field you can use the following formula:

```
nLength := iif( FIELD->FIELD_TYPE == "C", ;
                FIELD->FIELD_DEC * 256 + FIELD->FIELD_LEN, ;
                FIELD->FIELD_LEN )
```

COPY STRUCTURE EXTENDED command is preprocessed into __dbCopyXStruct() function during compile time.

Examples
```
// Open a database, then copy its structure to a new file,
// Open the new file and list all its records
USE test
__dbCopyXStruct( "teststru" )
USE teststru
LIST
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
__dbCopyStruct(), __dbCreate(), COPY STRUCTUREClipper 5.3, COPY STRUCTURE EXTENDEDClipper 5.3, CREATEClipper 5.3, CREATE FROMClipper 5.3, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

__dbCreate() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create structure extended file or use one to create new file
Syntax

```
__dbCreate( <cFileName>, [<cFileFrom>], [<cRDDName>], [<lNew>], [<cAlias>]
) → lUsed
```

Arguments
cFileName is the target file name to create and then open. .dbf is the default extension if none is given.
cFileFrom is an optional structure extended file name from which the target file cFileName is going to be built. If omitted, a new empty structure extended file with the name cFileName is created and opened in the current work-area.
cRDDName is RDD name to create target with. If omitted, the default RDD is used.
lNew is an optional logical expression, (.T.) opens the target file name cFileName in the next available unused work-area and makes it the current work-area. (.F.) opens the target file in the current work-area. Default value is (.F.). The value of lNew is ignored if cFileFrom is not specified.
cAlias is an optional alias to USE the target file with. If not specified, alias is based on the root name of cFileName.
Returns
__dbCreate() returns (.T.) if there is database **used** in the current work-area (this might be the newly selected work-area), or (.F.) if there is no database **used**. Note that on success a (.T.) would be returned, but on failure you probably end up with a run-time error and not a (.F.) value.
Description

__dbCreate() works in two modes depending on the value of cFileFrom:

1) If cFileFrom is empty or not specified a new empty structure extended file with the name cFileName is created and then opened in the current work-area (lNew is ignored). The new file has the following structure:

Field name	Type	Length	Decimals
FIELD_NAME	C	10	0
FIELD_TYPE	C	1	0
FIELD_LEN	N	3	0

FIELD_DEC N 3 0

The CREATE command is preprocessed into the __dbCopyStruct() function during compile time and uses this mode.

2) If cFileFrom is specified, it is opened and assumed to be a structure extended file where each record contains at least the following fields (in no particular order): FIELD_NAME, FIELD_TYPE, FIELD_LEN and FIELD_DEC. Any other field is ignored. From this information the file cFileName is then created and opened in the current or new work-area (according to lNew), if this is a new work-area it becomes the current.

For prehistoric compatibility reasons, structure extended file Character fields which are longer than 255 characters should be treated in a special way by writing part of the length in the FIELD_DEC according to the following formula:

```
FIELD->FIELD_DEC := Int( nLength / 256 )
FIELD->FIELD_LEN :=      nLength % 256
```

CREATE FROM command is preprocessed into __dbCopyStruct() function during compile time and use this mode.

Examples

```
// CREATE a new structure extended file, append some records and
// then CREATE FROM this file a new database file

__dbCreate( "template" )
dbAppend()
FIELD->FIELD_NAME := "CHANNEL"
FIELD->FIELD_TYPE := "N"
FIELD->FIELD_LEN  := 2
FIELD->FIELD_DEC  := 0
dbAppend()
FIELD->FIELD_NAME := "PROGRAM"
FIELD->FIELD_TYPE := "C"
FIELD->FIELD_LEN  := 20
FIELD->FIELD_DEC  := 0
dbAppend()
FIELD->FIELD_NAME := "REVIEW"
FIELD->FIELD_TYPE := "C"      // this field is 1000 char long
FIELD->FIELD_LEN  := 232      // 1000 % 256 = 232
FIELD->FIELD_DEC  := 3        // 1000 / 256 = 3
dbCloseArea()
__dbCreate( "TV_Guide", "template" )
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is rdd

Tags

API, Database

See also

__dbCopyStruct(), __dbCopyXStruct(), COPY STRUCTUREClipper 5.3, COPY STRUCTURE EXTENDEDClipper 5.3, CREATEClipper 5.3, CREATE FROMClipper 5.3, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

__dbDelim() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Copies the contents of a database to a delimited text file or appends the contents of a delimited text file to a database.

Syntax

```
__dbDelim( <lExport>, <xcFile>, [<xcDelim>], [<aFields>],
[<bFor>], [<bWhile>], [<nNext>], [<nRecord>], <lRest>  )
```

Arguments

`lExport` If set to `.T.`, copies records to a delimited file. If set to `.F.`, append records from a delimited file.

`xcFile` The name of the text file to copy to or append from. If a file extension is not specified, ".txt" is used by default.

`xcDelim` Either the character to use as the character field delimiter (only the first character is used). or `"BLANK"` (not case sensitive), which eliminates the character field delimiters and sets the field separator to a single space instead of a comma.

`aFields` An array of field names to limit the processing to. If not specified, or if empty, then all fields are processed.

`bFor` An optional code block containing a `FOR` expression that will reduce the number of records to be processed.

`bWhile` An optional code block containing a `WHILE` expression that will reduce the number of records to be processed.

`nNext` If present, but nRecord is not present, specifies to process this number of records, starting with the current record. A value of 0 means to process no records.

`nRecord` If present, specifies the only record to process. A value of 0 means to process no records. Overrides `nNext` and `lRest`.

`lRest` If `lExport` is `.T.`, then if `lRest` is set to `.T.` and there are no `nRecord`, `nNext`, or `bWhile` arguments, processes all records from current to last.

Description

`__dbDelim()` copies all or selected contents of a database table to an SDF text file or appends all or selected contents of an SDF text file to a database table.

Examples

```
// Copy delinquent accounts into a delimited text file.
USE accounts NEW
COPY TO overdue DELIMITED FOR ! Empty( accounts->duedate ) ;
   .AND. Date() - accounts->duedate > 30
// Import new customer records.
USE customer NEW
APPEND FROM customer DELIMITED
```

Status

Started

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

API, Database

See also

__dbSDF(), APPEND FROM, COPY TO

__dbSDF() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Copies the contents of a database to an SDF text file or appends the contents of an SDF text file to a database.

Syntax

```
__dbSDF( <lExport>, <xcFile>, [<aFields>],
   [<bFor>], [<bWhile>], [<nNext>], [<nRecord>], <lRest> )
```

Arguments

`lExport` If set to `.T.`, copies records to an SDF file. If set to `.F.`, append records from an SDF file.

`xcFile` The name of the text file to copy to or append from. If a file extension is not specified, ".txt" is used by default.

`aFields` An array of field names to limit the processing to. If not specified, or if empty, then all fields are processed.

`bFor` An optional code block containing a `FOR` expression that will reduce the number of records to be processed.

`bWhile` An optional code block containing a `WHILE` expression that will reduce the number of records to be processed.

`nNext` If present, but `nRecord` is not present, specifies to process this number of records, starting with the current record. A value of 0 means to process no records.

`nRecord` If present, specifies the only record to process. A value of 0 means to process no records. Overrides `nNext` and `lRest`.

`lRest` If `lExport` is `.T.`, then if `lRest` is set to `.T.` and there are no `nRecord`, `nNext`, or `bWhile` arguments, processes all records from current to last.

Description

`__dbSDF()` copies all or selected contents of a database table to an SDF text file or appends all or selected contents of an SDF text file to a database table.

Examples

```
// Copy delinquent accounts into an SDF text file.
USE accounts NEW
COPY TO overdue SDF FOR ;
   ! Empty( accounts->duedate ) .AND. ;
   Date() - accounts->duedate > 30
// Import new customer records.
USE customer NEW
APPEND FROM customer SDF
```

Status

Started

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

API, Database

See also

`__dbDelim()`, `APPEND FROM`, `COPY TO`

__dbStructFilter() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Filter a database structure array

Syntax

```
__dbStructFilter( <aStruct>, [<aFieldList>] ) → aStructFiltered
```

Arguments

`aStruct` is a multidimensional array with database fields structure, which is usually the output from `dbStruct()`, where each array element has the following structure:

Position	Description	dbstruct.ch
1	cFieldName	DBS_NAME
2	cFieldType	DBS_TYPE
3	nFieldLength	DBS_LEN
4	nDecimals	DBS_DEC

`aFieldList` is an array where each element is a field name. Names could be specified as uppercase or lowercase.

Returns

`__dbStructFilter()` return a new multidimensional array where each element is in the same structure as the original `aStruct`, but the array is built according to the list of fields in `aFieldList`. If `aFieldList` is empty, `__dbStructFilter()` return reference to the original `aStruct` array.

Description

`__dbStructFilter()` can be use to create a sub-set of a database structure, based on a given field list.

Note that field names in `aStruct` *must* be specified in uppercase or else no match would be found.

`SET EXACT` has no effect on the return value.

Examples

```
// FIXME
LOCAL aStruct, aList, aRet
aStruct := { ;
   { "CODE",  "N",  4, 0 }, ;
   { "NAME",  "C", 10, 0 }, ;
   { "PHONE", "C", 13, 0 }, ;
   { "IQ",    "N",  3, 0 } }
aList := { "IQ", "NAME" }
aRet := __dbStructFilter( aStruct, aList )
                  // { { "IQ", "N", 3, 0 }, { "NAME", "C", 10, 0 } }

aRet := __dbStructFilter( aStruct, {} )
? aRet == aStruct  // .T.

aList := { "iq", "NOTEXIST" }
aRet := __dbStructFilter( aStruct, aList )
                  // { { "IQ", "N", 3, 0 } }

aList := { "NOTEXIST" }
aRet := __dbStructFilter( aStruct, aList )  // --> {}

// Create a new file that contain part of the original structure
LOCAL aStruct, aList, aRet
USE test
aStruct := dbStruct()
aList := { "NAME" }
dbCreate( "onlyname.dbf", __dbStructFilter( aStruct, aList ) )
```

Status

Ready

Compliance

__dbStructFilter() is a Harbour extension. CA-Cl*pper has an internal undocumented function named __FLedit() that does exactly the same thing. The new name gives a better description of what this function does.

Platforms

Available on all platforms

Files

Header file is dbstruct.ch

Library is rdd

Tags

API, Database

See also

__dbCopyStruct(), __FLedit()*, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

__FLedit()* → __dbStructFilter() Source code in Repository in Discussions

^ | ☰ | Improve this doc

Filter a database structure array

Syntax

```
__FLedit( <aStruct>, [<aFieldList>] ) → aStructFiltered
```

Arguments

aStruct is a multidimensional array with database fields structure, which is usually the output from dbStruct(), where each array element has the following structure:

Position	Description	dbstruct.ch
1	cFieldName	DBS_NAME
2	cFieldType	DBS_TYPE
3	nFieldLength	DBS_LEN
4	nDecimals	DBS_DEC

aFieldList is an array where each element is a field name. Names could be specified as uppercase or lowercase.

Returns

__FLedit() return a new multidimensional array where each element is in the same structure as the original aStruct, but the array is built according to the list of fields in aFieldList. If aFieldList is empty, __FLedit() return reference to the original aStruct array.

Description

__FLedit() can be use to create a sub-set of a database structure, based on a given field list.

Note that field names in aStruct *must* be specified in uppercase or else no match would be found.

SET EXACT has no effect on the return value.

__FLedit() is a compatibility function and it is synonym for __dbStructFilter() which does exactly the same.

Examples

```
// FIXME
LOCAL aStruct, aList, aRet
aStruct := { ;
   { "CODE",  "N",  4, 0 }, ;
   { "NAME",  "C", 10, 0 }, ;
   { "PHONE", "C", 13, 0 }, ;
```

```
   { "IQ",    "N",  3, 0 } }
aList := { "IQ", "NAME" }
aRet := __FLedit( aStruct, aList )
                  // { { "IQ", "N", 3, 0 }, { "NAME", "C", 10, 0 } }

aRet := __FLedit( aStruct, {} )
? aRet == aStruct  // .T.

aList := { "iq", "NOTEXIST" }
aRet := __FLedit( aStruct, aList )
                  // { { "IQ", "N", 3, 0 } }

aList := { "NOTEXIST" }
aRet := __FLedit( aStruct, aList )  // {}

// Create a new file that contain part of the original structure
LOCAL aStruct, aList, aRet
USE test
aStruct := dbStruct()
aList := { "NAME" }
dbCreate( "onlyname.dbf", __FLedit( aStruct, aList ) )
```

Status

Ready

Compliance

CA-Cl*pper has internal undocumented function named `__FLedit()`, in Harbour we name it `__dbStructFilter()`. The new name gives a better description of what this function does. In Harbour `__FLedit()` simply calls `__dbStructFilter()` and therefor the latter is the recommended function to use.

This function is only visible if `src/rdd/dbstrux.prg` was compiled with the `HB_CLP_UNDOC` flag.

Platforms

Available on all platforms

Files

Header file is `dbstruct.ch`

Library is rdd

Tags

API, Database

See also

`__dbCopyStruct()`, `__dbStructFilter()`, `dbCreate()` Clipper 5.3, `dbStruct()` Clipper 5.3

`AFields()` *Source code in Repository in Discussions ^ | ☰ | Improve this doc

Fills referenced arrays with database field information

Syntax

```
AFields( <aNames>, [<aTypes>], [<aLen>], [<aDecs>] ) → nFields
```

Arguments

`aNames` Array of field names

`aTypes` Array of field names

`aLens` Array of field names

`aDecs` Array of field names

Returns

`nFields` Number of fields in a database or work area

Description

This function will fill a series of arrays with field names, field types, field lengths, and number of field decimal positions for the currently selected or designed database. Each array parallels the different descriptors of a file's structure. The first array will consist of the names of the fields in the current work area. All other arrays are optional and will be filled with the corresponding data. This function will return zero if no parameters are specified or if no database is available in the current work area. Otherwise, the number of fields or the length of the shortest array argument, whichever is smaller, will be returned.

`AFields()` is a compatibility function, it is superseded by `dbStruct()` which returns one multidimensional array.

NOTE: The destination arrays must be initialized to a given size,

usually `FCount()`, before calling this function.

Examples

```
PROCEDURE Main()

   LOCAL nCount

   USE test

   nCount := FCount()
   ? "Number of fields:", nCount
   PrintFields( nCount )  // Information for all fields
   PrintFields( 4 )       // Information for first 4 fields

   RETURN

STATIC PROCEDURE PrintFields( nCount )

   LOCAL aNames  := Array( nCount )
   LOCAL aTypes  := Array( nCount )
   LOCAL aLens   := Array( nCount )
   LOCAL aDecs   := Array( nCount )

   LOCAL nFields := AFields( aNames, aTypes, aLens, aDecs ), i

   ? "Number of items:", nFields
   FOR tmp := 1 TO nFields
      ? tmp, ;
         PadR( aNames[ tmp ], 12 ), ;
         aTypes[ tmp ], ;
         aLens[ tmp ], ;
         aDecs[ tmp ]
   NEXT

   RETURN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags

API, Database
See also
dbStruct() Clipper 5.3

Alias() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the alias name of a work area
Syntax
```
Alias( [<nWorkArea>] ) → cWorkArea
```
Arguments
nWorkArea Number of a work area
Returns
cWorkArea Name of alias
Description
This function returns the alias of the work area indicated by nWorkArea. If nWorkArea is not provided, the alias of the current work area is returned.
Examples
```
USE test
SELECT 0
? iif( Alias() == "", "No Name", Alias() )
? test->( Alias() )
? Alias( 1 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
Dbf() Clipper 5.3

Bof() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Test for the beginning-of-file condition
Syntax
```
Bof() → lBegin
```
Returns
Bof() Logical true (.T.) or false (.F.)
Description
This function determines if the beginning of the file marker has been reached. If so, the function will return a logical true (.T.); otherwise, a logical false (.F.) will be returned. By default, Bof() will apply to the currently selected database unless the function is preceded by an alias
Examples
```
USE test NEW
? "Is Bof()", Bof()
dbGoTop()
DO WHILE ! Bof()
   dbSkip( -1 )
```

```
ENDDO
? "Is Bof()", Bof()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Eof()` Clipper 5.3, `Found()` Clipper 5.3, `LastRec()` Clipper 5.3

`dbAppend()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Appends a new record to a database file.
Syntax
```
dbAppend( [<lLock>] ) → NIL
```
Arguments
`lLock` Toggle to release record locks
Returns
`dbAppend()` always returns `NIL`
Description

This function add a new record to the end of the database in the selected or aliased work area. All fields in that database will be given empty data values - character fields will be filled with blank spaces, date fields with `hb_SToD()`, numeric fields with 0, logical fields with `.F.`, and memo fields with `NULL` bytes. The header of the database is not updated until the record is flushed from the buffer and the contents are written to the disk.

Under a networking environment, `dbAppend()` performs an additional operation: It attempts to lock the newly added record. If the database file is currently locked or if a locking assignment is made to `LastRec() + 1`, `NetErr()` will return a logical true (`.T.`) immediately after the `dbAppend()` function. This function does not unlock the locked records.

If `lLock` is passed a logical true (`.T.`) value, it will release the record locks, which allows the application to maintain multiple record locks during an appending operation. The default for this parameter is a logical false (`.F.`).

Examples
```
LOCAL cName := "Harbour", nAge := 15
USE test
test->( dbAppend() )
test->first := cName
test->age   := nAge
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbUnlock()` Clipper 5.3, `dbUnlockAll()` Clipper 5.3

`dbClearFilter()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Clears the current filter condition in a work area
Syntax
`dbClearFilter() → NIL`
Returns
`dbClearFilter()` always returns `NIL`
Description
This function clears any active filter conduction for the current or selected work area.
Examples
```
USE test
SET FILTER TO hb_LeftEq( test->first, "An" )
dbGoTop()
dbEval( {|| QOut( test->first ) } )
dbClearFilter()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbFilter()` Clipper 5.3, `dbSetFilter()` Clipper 5.3

`dbCloseAll()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Close all open files in all work areas.
Syntax
`dbCloseAll() → NIL`
Returns
`dbCloseAll()` always return `NIL`
Description
This function close all open databases and all associated indexes. In addition, it closes all format files and moves the work area pointer to the first position
Examples
```
USE test NEW
dbEdit()
USE test1 NEW
dbEdit()
dbCloseAll()
```
Status

Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbCloseArea()` Clipper 5.3, `dbUseArea()` Clipper 5.3

`dbCloseArea()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Close a database file in a work area.
Syntax
`dbCloseArea()`
Description
This function will close any database open in the selected or aliased work area.
Examples
```
USE test
dbEdit()
test->( dbCloseArea() )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbCloseAll()` Clipper 5.3, `dbUseArea()` Clipper 5.3

`dbCommit()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Updates all index and database buffers for a given work area
Syntax
`dbCommit()`
Description
This function updates all of the information for a give, selected, or active work area. This operation includes all database and index buffers for that work area only. This function does not update all open work areas.
Examples
```
LOCAL GetList := {}
LOCAL cName := Space( 40 )
LOCAL nAge := 0

USE test EXCLUSIVE NEW

@ 10, 10 GET cName
```

```
@ 11, 10 GET nAge
READ

IF Updated()
   dbAppend()
   test->first := cName
   test->age   := nAge
   dbCommit()
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbCloseAll()` Clipper 5.3, `dbCommitAll()` Clipper 5.3, `dbUnlock()` Clipper 5.3

`dbCommitAll()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Flushes the memory buffer and performs a hard-disk write
Syntax
`dbCommit()`
Description
This function performs a hard-disk write for all work areas. Before the disk write is performed, all buffers are flushed. open work areas.
Examples

```
// FIXME
LOCAL GetList := {}
LOCAL cName := Space( 40 )
LOCAL nId := 0

USE test EXCLUSIVE NEW
USE testid NEW INDEX testid

@ 10, 10 GET cName
@ 11, 10 GET nId
READ

IF Updated()
   dbAppend()
   test->first := cName
   test->Id    := nId
   IF ! testid->( dbSeek( nId ) )
      testid->( dbAppend() )
      testid->Id := nId
   ENDIF
ENDIF
dbCommitAll()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbCloseAll() Clipper 5.3, dbCommit() Clipper 5.3, dbUnlock() Clipper 5.3

dbCreate() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Creates an empty database from a array.
Syntax

```
dbCreate( <cFile>, <aStruct>, [<cRDD>], [<lKeepOpen>], [<cAlias>],
          [<cDelimArg>], [<cCodePage>], [<nConnection>] )
```

Arguments

`cFile` Name of database file to be create

`aStruct` Name of a multidimensional array that contains the database structure

`cRDD` Name of the RDD

`lKeepOpen` 3-way toggle to Open the file in New or Current work area:

NIL	The file is not opened.
True	It is opened in a New area.
False	It is opened in the current area.

`cAlias` Name of database Alias

Description

This function creates the database file specified as `cDatabase` from the multidimensional array `aStruct`. If no file extension is use with `cDatabase` the .dbf extension is assumed. The array specified in `aStruct` must follow a few guidelines when being built prior to a call to `dbCreate()`:

- All subscripts values in the second dimension must be set to proper values

- The fourth subscript value in the second dimension - which contains

the decimal value-must he specified. even 1kw non-numeric fields.

- The second subscript value in the second dimension-which contains

the field data type-must contain a proper value: C, D, L, M or N It is possible to use additional letters for clarity (e.g., 'Numeric' for 'N'): however, the first letter of this array element must be a proper value.

The `dbCreate()` function does not use the decimal field to calculate the length of a character held longer than 256. Values up to the maximum length of a character field (which is 65519 bytes) are stored directly in the database in the length attribute if that database was created via this function. However, a file containing fields longer than 256 bytes is not compatible with any interpreter.

The cRDD parameter specifies the name of the Replaceable Database Driver to use to create the database. If it is not specified, then the Replaceable Database Driver in the current work area is used.

The lKeepOpen parameter specifies if the already created database is to be opened, and where. If NIL, the file is not opened. If True, it is opened in a New area, and if False it is opened in the current area (closing any file already occupying that area). The cAlias parameter specifies the alias name for the new opened database.

Examples

```
LOCAL aStruct := { ;
   { "CHARACTER", "C", 25, 0 }, ;
   { "NUMERIC",   "N",  8, 0 }, ;
   { "DOUBLE",    "N",  8, 2 }, ;
   { "DATE",      "D",  8, 0 }, ;
   { "LOGICAL",   "L",  1, 0 }, ;
   { "MEMO1",     "M", 10, 0 }, ;
   { "MEMO2",     "M", 10, 0 } }

REQUEST DBFCDX

? dbCreate( "testdbf", aStruct, "DBFCDX", .T., "MYALIAS" )
```

Status
Ready
Compliance
This function is not CA-Cl*pper compliant
Platforms
Available on all platforms
Files
Library is rdd
Header is dbstruct.ch
Tags
API, Database
See also
AFields()*Clipper 5.3, dbStruct()Clipper 5.3

dbDelete()Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Mark a record for deletion in a database.
Syntax

```
dbDelete()
```

Description

This function marks a record for deletion in the selected or aliased work area. If the DELETED setting is on, the record will still be visible until the record pointer in that work area is moved to another record.

In a networking situation, this function requires that the record be locked prior to issuing the dbDelete() function.

Examples

```
LOCAL nAge := 50
USE test NEW
INDEX ON field->age TO test
```

```
IF dbSeek( nAge ) .AND. RLock()
   dbDelete()
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbRecall() Clipper 5.3

dbFilter() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the filter expression in a work area
Syntax
```
dbFilter() → cFilter
```
Returns
`dbFilter()` returns the filter expression.
Description
This function return the expression of the `SET FILTER TO` command for the current or designated work area. If no filter condition is present, a null string will be returned.
Examples
```
USE test INDEX test NEW
SET FILTER TO field->first = "Harbour"
USE testid INDEX testid NEW
SET FILTER TO field->age == 25
SELECT test

? dbFilter()
? testid->( dbFilter() )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbRelation(), dbRSelect()

dbGoBottom() Source code in Repository in Discussions ^ | ☰ | Improve this doc

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

Description
This function moves the record pointer in the selected or aliased work area to the end of the file. The position of the record pointer is affected by the values in the index key or by an active `FILTER` condition. Otherwise, if no index is active or if no filter condition is present, the value of the record pointer will be `LastRec()`.
Examples
```
USE test
dbGoTop()
? RecNo()
dbGoBottom()
? RecNo()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoTop()` Clipper 5.3, `dbSeek()` Clipper 5.3, `dbSkip()` Clipper 5.3, `Eof()` Clipper 5.3

`dbGoto()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Position the record pointer to a specific location.
Syntax
```
dbGoto( <xRecordNumber> )
```
Arguments
`xRecordNumber` Record number or unique identity
Description

This function places the record pointer, if working with a .dbf file, in selected or aliased work area at the record number specified by `xRecordNumber`. The position is not affected by an active index or by any environmental `SET` condition.

The parameter `xRecordNumber` may be something other than a record number. In some data formats, for example, the value of `xRecordNumber` is a unique primary key while in other formats, `xRecordNumber` could be an array offset if the data set was an array.

Issuing a `dbGoto( RecNo() )` call in a network environment will refresh the database and index buffers. This is the same as a `dbSkip( 0 )` call.

Examples
```
// The following example uses dbGoto() to iteratively process
// every fourth record:

dbUseArea( .T., "DBFNTX", "sales", "sales", .T. )

// toggle every fourth record
DO WHILE ! Eof()
```

```
   dbGoto( RecNo() + 4 )
   sales->Group := "Bear"
ENDDO
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoBottom()` Clipper 5.3, `dbGoTop()` Clipper 5.3, `dbSeek()` Clipper 5.3, `dbSkip()` Clipper 5.3, `Eof()` Clipper 5.3

`dbGoTop()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Moves the record pointer to the top of the database.
Syntax
`dbGoTop()`
Description
This function moves the record pointer in the selected or aliased work area to the top of the file. The position of the record pointer is affected by the values in the index key or by an active `FILTER` condition. Otherwise, if no index is active or if no filter condition is present, the value of `RecNo()` will be 1.
Examples

```
USE test
dbGoTop()
? RecNo()
dbGoBottom()
? RecNo()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoBottom()` Clipper 5.3, `dbSeek()` Clipper 5.3, `dbSkip()` Clipper 5.3, `Eof()` Clipper 5.3

`dbRecall()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

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

This function unmarks those records marked for deletion and reactivates them in the aliased or selected work area. If a record is `DELETED` and the `DELETED` setting is on, the record will still be visible for a `dbRecall()` provided that the database record pointer has not been skipped. Once a record marked for deletion with the `DELETE` setting `ON` has been skipped, it no longer can be brought back with `dbRecall()`.

Examples

```
USE test NEW
dbGoto( 10 )
dbDelete()
? Deleted()
dbRecall()
? Deleted()
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is rdd

Tags

API, Database

See also

`dbDelete()` Clipper 5.3

`dbRLock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function locks the record based on identity

Syntax

```
dbRLock( [<xIdentity>] ) → lSuccess
```

Arguments

`xIdentity` Record identifier

Returns

`dbRLock()` returns a logical true (`.T.`) if lock was successful

Description

This function attempts to lock a record which is identified by `xIdentity` in the active data set. If the lock is successful the function will return a logical true (`.T.`) value; otherwise a logical false (`.F.`) will be returned. If `xIdentity` is not passed it will be assumed to lock the current active record/data item.

Examples

```
LOCAL nRecNo
USE test NEW
FOR nRecNo := 1 TO LastRec()
   IF ! dbRLock()
      dbUnlock()
   ENDIF
NEXT
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File
Library is rdd
Tags
API, Database
See also
dbUnlock() Clipper 5.3, dbUnlockAll() Clipper 5.3, FLock() Clipper 5.3, RLock() Clipper 5.3

dbRLockList() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function return a list of locked records in the database work area
Syntax
```
dbRLockList() → aRecordLocks
```
Returns
aRecordList is an array of lock records
Description
This function will return an array of locked records in a given and active work area. If the return array is an empty array (meaning no elements in it), then there are no locked records in that work area.
Examples
```
LOCAL nRecNo
USE test NEW
dbGoto( 10 )
? RLock()
dbGoto( 100 )
? RLock()
FOR EACH nRecNo IN dbRLockList()
   ? nRecNo
NEXT
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbRLock() Clipper 5.3, dbRUnlock() Clipper 5.3, RLock() Clipper 5.3

dbRUnlock() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Unlocks a record based on its identifier
Syntax
```
dbRUnlock( [<xIdentity>] )
```
Arguments
xIdentity Record identifier, typically a record number
Description
This function will attempt to unlock the record specified as xIdentity, which in a .dbf format is the record number. If not specified, them the current active record/data item will be unlocked

Examples

```
USE test NEW
dbGoto( 10 )
IF RLock()
   ? test->age
   dbRUnlock()
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbRLock() Clipper 5.3, dbRLockList() Clipper 5.3, RLock() Clipper 5.3

dbSeek() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Searches for a value based on an active index.
Syntax

```
dbSeek( <expKey>, [<lSoftSeek>], [<lFindLast>] ) → lFound
```

Arguments
expKey Any expression
lSoftSeek Toggle SOFTSEEK condition
lFindLast is an optional logical value that set the current record position to the last record if successful
Returns
dbSeek() returns logical true (.T.) if found, otherwise false
Description

This function searches for the first record in a database file whose index key matches expKey. If the item is found, the function will return a logical true (.T.), the value of Found() will be a logical true (.T.), and the value of Eof() will be a logical false (.F.). If no item is found. then the function will return a logical false, the value of Found() will be a logical false (.F.), and the value of Eof() will be a logical true (.T.).

This function always "rewinds" the database pointer and starts the search from the top of the file.

If the SOFTSEEK flag is on or if lSoftSeek is set to a logical true (.T.) the value of Found() will be a logical false and Eof() will be false if there is an item in the index key with a greater value than the key expression expKey; at this point the record pointer will position itself on that record. However, if there is no greater key in the index, Eof() will return a logical true (.T.) value. If lSoftSeek is not passed, the function will look to the internal status of SOFTSEEK before performing the operation. The default of lSoftSeek is a logical false (.F.)

Examples

```
PROCEDURE Main()
```

```
   LOCAL nAge

   USE test NEW
   INDEX ON field->age TO test
   dbGoto( 10 )
   nAge := test->age
   dbGoTop()
   IF dbSeek( nAge )
      ? test->first
   ENDIF

   RETURN

STATIC PROCEDURE EmployeeLookup()

   LOCAL cName

   ACCEPT "Employee name: " TO cName
   IF Employee->( dbSeek( cName ) )
      Employee->( ViewRecord() )
   ELSE
      ? "Not found"
   ENDIF

   RETURN

STATIC PROCEDURE ViewRecord()

   ? field->name

   RETURN
```

Status
Started
Compliance
`dbSeek()` is Compatible with CA-Cl*pper 5.3
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoBottom()` Clipper 5.3, `dbGoTop()` Clipper 5.3, `dbSkip()` Clipper 5.3, `Eof()` Clipper 5.3, `Found()` Clipper 5.3

`dbSelectArea()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Change to another work area
Syntax
`dbSelectArea( <xArea> ) → NIL`
Arguments
`xArea` Alias or work area
Description

This function moves the Harbour internal primary focus to the work area designated by `xArea`. If `xArea` is numeric, then it will select the numeric work area; if `xArea` is character, then it will select the work area with the alias name.

`dbSelectArea( 0 )` will select the next available and unused work area. Up to 65534 work areas are supported. Each work area has its own alias and record pointer, as well as its own `Found()`, `dbFilter()`, `dbRSelect()` and `dbRelation()` function values.

Examples

```
LOCAL nAge
USE test NEW
COPY TO test1
USE test1 NEW
INDEX ON field->age TO test1
dbSelectArea( "test" )
dbGoto( 100 )
? nAge := field->age
dbSelectArea( "test1" )
IF dbSeek( nAge )
   ? field->first
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbUseArea()` Clipper 5.3, `Select()` Clipper 5.3

`dbSetDriver()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Establishes the RDD name for the selected work area
Syntax

```
dbSetDriver( [<cDriver>] ) → cCurrentDriver
```

Arguments
`cDriver` Optional database driver name
Returns
`dbSetDriver()` returns the name of active driver
Description
This function returns the name of the current database driver for the selected work area. The default will be "DBFNTX". If specified, `cDriver` contains the name of the database driver that should be used to activate and manage the work area. If the specified driver is not available, this function will have no effect.
Examples

```
? dbSetDriver( "DBFNSX" )
```

Status
Ready
Compliance

CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbUseArea() Clipper 5.3

dbSetFilter() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Establishes a filter condition for a work area.
Syntax
```
dbSetFilter( <bCondition>, [<cCondition>] )
```
Arguments
`bCondition` Code block expression for filtered evaluation.
`cCondition` Optional character expression of code block.
Description
This function masks a database so that only those records that meet the condition prescribed by the expression in the code block `bCondition` and literally expressed as `cCondition` are visible. If `cCondition` is not passed to this function, then the `dbFilter()` function will return an empty string showing no filter in that work area which in fact, would be not correct.
Examples
```
USE test NEW
dbSetFilter( {|| test->age > 30 }, "test->age > 30" )
dbGoTop()
? RecNo()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbClearFilter() Clipper 5.3, dbFilter() Clipper 5.3

dbSkip() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Moves the record pointer in the selected work area.
Syntax
```
dbSkip( [<nRecords>] )
```
Arguments
`nRecords` Numbers of records to move record pointer.
Description
This function moves the record pointer `nRecords` in the selected or aliased work area. The default value for `nRecords` will be 1. A `dbSkip( 0 )` will flush and refresh the internal

database buffer and make any changes made to the record visible without moving the record pointer in either direction.
Examples
```
USE test NEW
dbGoTop()
DO WHILE ! Eof()
   ? test->age, test->first
   dbSkip()
ENDDO
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoBottom()` Clipper 5.3, `dbGoTop()` Clipper 5.3, `dbSeek()` Clipper 5.3, `Eof()` Clipper 5.3

`dbStruct()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Creates a multidimensional array of a database structure.
Syntax
`dbStruct() → aStruct`
Returns
`dbStruct()` returns an array pointer to database structure
Description
This function returns a multidimensional array. This array has array pointers to other arrays, each of which contains the characteristic of a field in the active work area. The length of this array is based in the number of fields in that particular work area. In other words, `Len( dbStruct() )` is equal to the value obtained from `FCount()`. Each subscript position
Examples
```
#include "dbstruct.ch"
LOCAL field
USE test NEW
FOR EACH field IN dbStruct()
   ? field[ DBS_NAME ]
NEXT
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Files
Library is rdd
Header is `dbstruct.ch`
Tags
API, Database

See also
AFields()*Clipper 5.3

dbUnlock() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Unlock a record or release a file lock
Syntax
```
dbUnlock()
```
Description
This function releases the file or record lock in the currently selected or aliased work area. It will not unlock an associated lock in a related databases.
Examples
```
LOCAL nAge := 30
USE test NEW
INDEX ON field->age TO test
IF test->( dbSeek( nAge ) )
   IF test->( RLock() )
      dbDelete()
   ELSE
      dbUnlock()
   ENDIF
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbUnlockAll()Clipper 5.3, FLock()Clipper 5.3, RLock()Clipper 5.3

dbUnlockAll() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Unlocks all records and releases all file locks in all work areas.
Syntax
```
dbUnlockAll()
```
Description
This function will remove all file and record locks in all work area.
Examples
```
LOCAL nAge := 50
USE test NEW
INDEX ON field->age TO test
IF test->( dbSeek( nAge ) )
   IF test->( RLock() )
      dbDelete()
   ELSE
      dbUnlock()
   ENDIF
ELSE
   dbUnlockAll()
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbUnlock()` Clipper 5.3, `FLock()` Clipper 5.3, `RLock()` Clipper 5.3

`dbUseArea()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Opens a work area and uses a database file.
Syntax

```
dbUseArea( [<lNewArea>], [<cDriver>], <cName>, [<xcAlias>],
[<lShared>], [<lReadonly>] )
```

Arguments
`lNewArea` A optional logical expression for the new work area
`cDriver` Database driver name
`cName` File Name
`xcAlias` Alias name
`lShared` Shared/exclusive status flag
`lReadonly` Read-write status flag.
Description

This function opens an existing database named `cName` in the current work area. If `lNewArea` is set to a logical true (`.T.`) value, then the database `cName` will be opened in the next available and unused work area. The default value of `lNewArea` is a logical false (`.F.`). If used, `cDriver` is the name of the database driver associated with the file `cName` that is opened. The default for this will be the value of `dbSetDriver()`.

If used, `xcAlias` contains the alias name for that work area, If not specified, the root name of the database specified in `cName` will be used.

If `lShared` is set to a logical true (`.T.`) value, the database that is specified in `cName` will be opened by the user **exclusively**. Thus locking it from all other nodes or users on the network. If `lShared` is set to a logical false (`.F.`) value, then the database will be in `SHARED` mode. If `lShared` is not passed, then the function will turn to the internal setting of `SET EXCLUSIVE` to determine a setting.

If `lReadOnly` is specified, the file will be set to **read only** mode. If it is not specified, the file will he opened in normal read-write mode.

Examples

```
? dbUseArea( .T.,, "test" )
```

Status
Ready
Compliance

CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbCloseArea()` Clipper 5.3, `dbSetDriver()` Clipper 5.3, `Select()` Clipper 5.3, `Set()` Clipper 5.3

`Dbf()` Source code □ in Repository in Discussions ^ | ☰ | Improve this doc

Alias name of a work area
Syntax
`Dbf() → cWorkArea`
Returns
`cWorkArea` Name of alias
Description
This function returns the same alias name of the currently selected work area.
Examples

```
USE test
SELECT 0
? iif( Dbf() == "", "No Name", Dbf() )
? test->( Dbf() )
? Alias( 1 )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Alias()` Clipper 5.3

`Deleted()` Source code □ in Repository in Discussions ^ | ☰ | Improve this doc

Tests the record's deletion flag.
Syntax
`Deleted() → lDeleted`
Arguments
(This command has no arguments)
Returns
`Deleted()` return a logical true (`.T.`) or false (`.F.`).
Description
This function returns a logical true (`.T.`) if the current record in the selected or designated work area has been marked for deletion. If not, the function will return a logical false (`.F.`).
Examples

```
USE test NEW
dbGoto( 10 )
dbDelete()
? "Is Record Deleted", test->( Deleted() )
dbRecall()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbDelete()` Clipper 5.3

`Eof()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Test for end-of-file condition.
Syntax
`Eof() → lEnd`
Arguments
(This command has no arguments)
Returns
`lEnd` A logical true (`.T.`) or false (`.F.`)
Description
This function determines if the end-of-file marker has been reached. If it has, the function will return a logical true (`.T.`); otherwise a logical false (`.F.`) will be returned
Examples

```
USE test NEW
dbGoTop()
? "Is Eof()", Eof()
dbGoBottom()
DO WHILE ! Eof()
   dbSkip()
ENDDO
? "Is Eof()", Eof()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `Found()` Clipper 5.3, `LastRec()` Clipper 5.3

`FCount()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Counts the number of fields in an active database.
Syntax
`FCount() → nFields`
Returns
`nFields` Return the number of fields
Description
This function returns the number of fields in the current or designated work area. If no database is open in this work area, the function will return 0.
Examples
```
USE test NEW
? "This database has", hb_ntos( test->( FCount() ) ), "field(s)"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FieldName()` Clipper 5.3, `Type()` Clipper 5.3

`FieldGet()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Obtains the value of a specified field
Syntax
`FieldGet( <nField> ) → ValueField`
Arguments
`nField` Is the numeric field position
Returns
`ValueField` Any expression
Description
This function returns the value of the field at the `nField`th location in the selected or designed work area. If the value in `nField` does not correspond to n available field position in this work area, the function will return a `NIL` data type.
Examples
```
USE test NEW
? test->( FieldGet( 1 ) )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FieldPut()` Clipper 5.3

FieldName() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the name of a field at a numeric field location.
Syntax
```
FieldName()/Field( <nPosition> ) → cFieldName
```
Arguments
`nPosition` Field order in the database.
Returns
`cFieldName` returns the field name.
Description
This function return the name of the field at the `nPosition`th position. If the numeric value passed to this function does not correspond to an existing field in the designated or selected work area, this function will return a `NULL` byte.
Examples
```
LOCAL nField
USE test NEW
FOR nField := 1 TO test->( FCount() )
   ? "Field Name:", FieldName( nField )
NEXT
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbStruct()` Clipper 5.3, `FCount()` Clipper 5.3, `Len()` Clipper 5.3, `ValType()` Clipper 5.3

FieldPos() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the ordinal position of a field.
Syntax
```
FieldPos( <cFieldName> ) → nFieldPos
```
Arguments
`cFieldName` Name of a field.
Returns
`nFieldPos` is ordinal position of the field.
Description
This function return the ordinal position of the specified field `cField` in the current or aliased work area. If there isn't field under the name of `cField` or of no database is open in the selected work area, the function will return a 0.
Examples
```
USE test NEW
? test->( FieldPos( "LAST" ) )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FieldGet()` Clipper 5.3, `FieldPut()` Clipper 5.3

`FieldPut()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set the value of a field variable
Syntax
```
FieldPut( <nField>, <expAssign> ) → ValueAssigned
```
Arguments
`nField` The field numeric position
`expAssign` Expression to be assigned to the specified field
Returns
`ValueAssigned` Any expression
Description
This function assigns the value in `expAssing` to the `nField`th field in the current or designated work area. If the operation is successful, the return value of the function will be the same value assigned to the specified field. If the operation is not successful, the function will return a `NIL` data type
Examples
```
USE test NEW
? FieldPut( 1, "Mr. Jones" )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FieldGet()` Clipper 5.3

`FLock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locks a file
Syntax
```
FLock() → lSuccess
```
Returns
`lSuccess` A true (`.T.`) value, if the lock was successful; otherwise false (`.F.`)
Description
This function returns a logical true (`.T.`) if a file lock is attempted and is successfully placed on the current or designated database. This function will also unlock all records locks placed by the same network station.

Examples

```
LOCAL nSum
USE test NEW
IF FLock()
   SUM test->age TO nSum
   ? nSum
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`RLock()` Clipper 5.3

`Found()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine the success of a previous search operation.
Syntax

```
Found() → lSuccess
```

Arguments
(This function has no arguments)
Returns
`lSuccess` A logical true (`.T.`) is successful; otherwise, false (`.F.`)
Description
This function is used to test if the previous `SEEK`, `LOCATE`, `CONTINUE`, or `FIND` operation was successful. Each work area has its own `Found()` flag, so that a `Found()` condition may be tested in unselected work areas by using an alias.
Examples

```
LOCAL nAge := 40
USE test NEW
INDEX ON field->age TO test
SEEK nAge
IF Found()
   ? test->first
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Eof()` Clipper 5.3

Header() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the length of a database file header
Syntax
```
Header() → nBytes
```
Returns
`nBytes` The numeric size of a database file header in bytes
Description

This function returns the number of bytes in the header of the selected database of the database in the designated work area.

If used in conjunction with the `LastRec()`, `RecSize()` and `DiskSpace()` functions, this functions is capable of implementing a backup and restore routine.

Examples
```
USE test NEW
? Header()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`DiskSpace()` Clipper 5.3, `LastRec()` Clipper 5.3, `RecSize()` Clipper 5.3

IndexExt() in Repository in Discussions ^ | ☰ | Improve this doc

Returns the file extension of the index module used in an application
Syntax
```
IndexExt() → cExtension
```
Arguments
None.
Returns
`cExtension` Current driver file extension
Description
This function returns a string that tells what indexes are to be used or will be created in the compiled application. The default value is `.ntx`. This is controlled by the particular database driver that is linked with the application.
Examples
```
IF IndexExt() == ".ntx"
   ? "Current driver being used is DBFNTX"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`IndexKey()` Clipper 5.3, `IndexOrd()` Clipper 5.3

`IndexKey()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Yields the key expression of a specified index file.
Syntax
```
IndexKey( <nOrder> ) → cIndexKey
```
Arguments
`nOrder` Index order number
Returns
`cIndexKey` The index key
Description

This function returns a character string stored in the header of the index file

The index key is displayed for an index file that is designated by `nOrder`, its position in the `USE...INDEX` or `SET INDEX TO` command in the currently selected or designated work area. If there is no corresnponding index key at the specified order position, a `NULL` byte will be returned.

Examples
```
USE test NEW
INDEX ON field->first TO test
? IndexKey( 1 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`IndexOrd()` Clipper 5.3

`IndexOrd()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the numeric position of the controlling index.
Syntax
```
IndexOrd() → nPosition
```
Arguments
None.
Returns

`nPosition` Ordinal position of a controling index
Description
The `IndexOrd()` function returns the numeric position of the current controlling index in the selected or designated work area. A returned value of 0 indicated that no active index is controlling the database, which therefore is in the natural order.
Examples

```
USE test NEW
INDEX ON field->first TO test
IF IndexOrd() > 0
   ? "Current order is", hb_ntos( IndexOrd() )
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`IndexKey()` Clipper 5.3

`LastRec()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the number of records in an active work area or database.
Syntax

```
LastRec() | RecCount()* → nRecords
```

Returns
`nRecords` The number of records
Description
This function returns the number of records present in the database in the selected or designated work area. If no records are present the value of this function will be 0. Additionally, if no database is in use in the selected or designated work area, this function will return a 0 value as well.
Examples

```
USE test NEW
? LastRec(), RecCount()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Eof()` Clipper 5.3

LUpdate() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Yields the date the database was last updated.
Syntax
```
LUpdate() → dModification
```
Arguments
(This function has no arguments)
Returns
`dModification` The date of the last modification.
Description
This function returns the date recorded by the OS when the selected or designated database was last written to disk. This function will only work for those database files in `USE`.
Examples
```
USE test NEW
? LUpdate()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FieldName()` Clipper 5.3, `LastRec()` Clipper 5.3, `RecSize()` Clipper 5.3

NetErr() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Tests the success of a network function
Syntax
```
NetErr( [<lNewError>] ) → lError
```
Arguments
`lNewError` Is a logical Expression.
Returns
`lError` A value based on the success of a network operation or function.
Description
This function return a logical true (`.T.`) is a `USE`, `dbAppend()`, or a `USE...EXCLUSIVE` command is issue and fails in a network environment. In the case of `USE` and `USE...EXCLUSIVE` commands, a `NetErr()` value of `.T.` would be returned if another node of the network has the exclusive use of a file. And the case of the `dbAppend()` command, `NetErr()` will return a logical true (`.T.`) if the file or record is locked by another node or the value of `LastRec()` has been advanced The value of `NetErr()` may be changed via the value of `lNewError`. This allow the run-time error-handling system to control the way certain errors are handled.
Examples
```
USE test NEW
IF ! NetErr()
   INDEX ON field->first TO test
   SET INDEX TO test
   test->first := "Harbour"
```

```
   IF dbSeek( "Harbour" )
      ? test->first
   ENDIF
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`FLock()` Clipper 5.3, `RLock()` Clipper 5.3

`ordBagExt()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the Order Bag extension
Syntax
`ordBagExt() → cBagExt`
Arguments
None
Returns
`cBagExt` The RDD extension name.
Description

This function return th character name of the RDD extension for the order bag. This is determined by the active RDD for the selected work area.

This function replaces the `IndexOrd()` function.

Examples

```
USE test VIA "DBFNTX"
? ordBagExt()  // --> ".ntx"
USE test VIA "DBFCDX"
? ordBagExt()  // --> ".cdx"
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`IndexExt()` Clipper 5.3, `ordBagName()` Clipper 5.3

`ordBagName()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the Order Bag Name.
Syntax
```
ordBagName( <nOrder> | <cOrderName> ) → cOrderBagName
```
Arguments
`nOrder` A numeric value representing the Order bag number.
`cOrderName` The character name of the Order Bag.
Returns
`ordBagName()` returns the Order bag name
Description
This function returns the name of the order bag for the specified work area. If `nOrder` is specidied, it will represent the position in the order list of the target order. If `cOrderName` is specified, it will represent the name of the target order. In essence, it will tell the name of the database (if That RDD is in use) for a given index name or index order number. If `cOrderName` is not specified or `nOrder` is 0, the Current active order will be used.
Examples
```
USE test VIA "DBFCDX" NEW
SET INDEX TO test
? ordBagName( "TeName" )     // --> "Customer"
? ordBagName( "TeLast" )     // --> "Customer"
? ordBagName( "teZip" )      // --> "Customer"
ordSetFocus( "TeName" )
? ordBagName()               // --> "Custumer"
```
Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Alias()` Clipper 5.3, `IndexOrd()` Clipper 5.3, `ordBagExt()` Clipper 5.3

`ordCondSet()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set the Condition and scope for an order
Syntax
```
ordCondSet( [<cForCondition>],
  [<bForCondition>],
  [<lAll>],
  [<bWhileCondition>],
  [<bEval>],
  [<nInterval>],
  [<nStart>],
  [<nNext>],
  [<nRecord>],
  [<lRest>],
  [<lDescend>],
  [<lAdditive>],
  [<lCurrent>],
  [<lCustom>],
  [<lNoOptimize>] )
```
Arguments

`cForCondition` is a string that specifies the `FOR` condition for the order.
`bForCondition` is a code block that defines a `FOR` condition that each record within the scope must meet in order to be processed. If a record does not meet the specified condition, it is ignored and the next record is processed. Duplicate keys values are not added to the index file when a `FOR` condition is Used.
Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database

`ordCreate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create an Order in an Order Bag
Syntax

```
ordCreate( <cOrderBagName>,[<cOrderName>], <cExpKey>,
[<bExpKey>], [<lUnique>] )
```

Arguments
`cOrderBagName` Name of the file that contains one or more Orders.
`cOrderName` Name of the order to be created.
`cExpKey` Key value for order for each record in the current work area
`bExpKey` Code block that evaluates to a key for the order for each record in the work area.
`lUnique` Toggle the unique status of the index.
Description

This function creates an order for the current work area. It is similar to the `dbCreateIndex()` except that this function allows different orders based on the RDD in effect. The name of the file `cOrderBagName` or the name of the order `cOrderName` are technically both considered to be "optional" except that at least one of two must exist in order to create the order.

The parameter `cExpKey` is the index key expression; typically in a .dbf driver, the maximum length of the key is 255 characters.

If `bExpKey` is not specified, then the code block is create by macro expanding the value of `cExpKey`.

If `lUnique` is not specified, then the current internal setting of `SET UNIQUE ON` or `OFF` will be observed.

The active RDD driver determines the capacity in the order for a specific order bag.

If the name `cOrderBagName` is found in the order bag can contain a single order, the the name `cOrderBagName` is erased and a new order is added to the order list in the current or specified work `area.On` the other hand, if it can contain multiples tags and if `cOrderBagName` does not already exist in the order list, then it is added. It is does exist, then the `cOrderBagName` replaces the former name in the order list in the current or specified work area.

Examples

```
USE test VIA "DBFNTX"
? ordCreate( "FNAME",, "field->first" )

USE test VIA "DBFCDX"
? ordCreate( , "LNAME", "field->last" )
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
dbCreateIndex(), ordName(), ordSetFocus()

ordDestroy()Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Remove an Order from an Order Bag
Syntax

```
ordDestroy( <cOrderName> [, <cOrderBagName> ] )
```

Arguments
cOrderName Name of the order to remove
cOrderBagName Name of the order bag from which order id to be removed
Description

This function attempts to remove the order named cOrderName from the file containing the order bag name cOrderBagName. If cOrderBagName is not specified, then the name of the file will be based on the value of the ordName() function. If the extension is not included with the name of the order file, then the extension will be obtained from the default extension of the current and active RDD.

The DBFNTX driver do not support multiple order bags; therefore, there cannot be an order to "destroy" from a bag. This function only works for those drivers with support multiple orders bags (e.g. DBFCDX and RDDADS drivers).

Examples

```
USE test VIA "DBFCDX" NEW
? ordDestroy( "lName", "test" )
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database

See also
ordCreate() Clipper 5.3

ordFor() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the FOR expression of an Order
Syntax
```
ordFor( <xOrder>[, <cOrderBagName>] ) → cForExp
```
Arguments
xOrder It the name of the target order, or the numeric position of the order.
cOrderBagName Name of the order bag.
Returns
ordFor() returns a expression containing the FOR condition for an order.
Description
This function returns a character string that is the expression for the FOR condition for the specified order. The order may be specified if xOrder is the name of the order. However, xOrder may be an numeric which represent the position in the order list of the desired Order.
Examples
```
USE test NEW VIA "DBFCDX"
INDEX ON field->first TO test FOR field->age > 50
? ordFor( "test" )        // --> "field->age > 50"
```
Status
Started
Compliance
This function is CA-Cl*pper compliant with one exception: If the xOrder paramter is not specified or xOrder is 0, the current active order is used.
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
ordCreate() Clipper 5.3, ordKey() Clipper 5.3, ordName(), ordNumber()

ordKey() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the key expression of an Order
Syntax
```
ordKey( <cOrderName> | <nOrder> [, <cOrderBagName>] ) → cExpKey
```
Arguments
xOrder It the name of the target order, or the numeric position of the order.
cOrderBagName Name of the order bag.
Returns
cExpKey Returns a character string, cExpKey.
Examples
```
USE test NEW VIA "DBFCDX"
INDEX ON field->first TO test FOR field->first > "CK"
INDEX ON field->age   TO testage

? ordKey( "test" )  // --> "field->first"
ordSetFocus( 2 )
? ordKey()          // --> "field->age"
```

Status
Started
Compliance
This function is CA-Cl*pper compliant with one exception: If the `xOrder` paramter is not specified or `xOrder` is 0, the current active order is used.
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`ordFor()` Clipper 5.3, `ordKey()` Clipper 5.3, `ordName()`, `ordNumber()`

`RecCount()` → `LastRec()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Counts the number of records in a database.
Syntax
```
RecCount()* | LastRec() → nRecords
```
Arguments
(This function has no arguments)
Returns
`nRecords` The number of records CRIPTION$* This function returns the number of records present in the database in the selected or designated work area. If no records are present the value of this function will be 0. Additionally, if no database is in use in the selected or designated work area, this function will return a 0 value as well.
Examples
```
USE test NEW
? RecCount()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`dbGoBottom()` Clipper 5.3, `Eof()` Clipper 5.3, `LastRec()` Clipper 5.3, `RecNo()` Clipper 5.3

`RecNo()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the current record number or identity.
Syntax
```
RecNo() → Identity
```
Arguments
(This function has no arguments)
Returns

`RecNo()` The record number or identity
Description

This function returns the position of the record pointer in the currently selected of designated work area.

If the database file is empty and if the RDD is the traditional .dbf file, the value of this function will be 1.

Examples

```
USE test NEW
dbGoTop()
? RecNo()      // --> 1
dbGoto( 50 )
? RecNo()      // --> 50
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Bof()` Clipper 5.3, `dbGoBottom()` Clipper 5.3, `dbGoto()` Clipper 5.3, `dbGoTop()` Clipper 5.3, `Eof()` Clipper 5.3, `LastRec()` Clipper 5.3

`RecSize()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the size of a single record in an active database.
Syntax

```
RecSize() → nBytes
```

Arguments
(This function has no arguments)
Returns
`nBytes` The record size.
Description
This function returns the number of bytes used by a single record in the currently selected or designated database file. If no database is in use in this work area, the return value from this function will be 0.
Examples

```
USE test NEW
dbGoTop()
? RecSize()    // --> 1
dbGoto( 50 )
? RecSize()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`DiskSpace()` Clipper 5.3, `FieldName()` Clipper 5.3, `Header()` Clipper 5.3, `LastRec()` Clipper 5.3

`RLock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Lock a record in a work area
Syntax
```
RLock() → lSuccess
```
Arguments
(This function has no arguments)
Returns
`RLock()` True (`.T.`) if record lock is successful; otherwise, it returns false (`.F.`).
Description

This function returns a logical true (`.T.`) if an attempt to lock a specific record in a selected or designated work area is successful. It will yield a false (`.F.`) if either the file or the desired record is currently locked. A record that is locked remains locked until another `RLock()` is issued or until an `UNLOCK` command is executed. On a Network environment the follow command need that the record is locked:

`@...GET`

`DELETE` (single record)

`RECALL` (single record)

`REPLACE` (single record)

Examples
```
LOCAL nAge := 50
USE test NEW
INDEX ON field->age TO test
IF dbSeek( nAge ) .AND. RLock()
   dbDelete()
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also

FLock() Clipper 5.3

Select() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the work area number for a specified alias.
Syntax
```
Select( [<cAlias>] ) → nWorkArea
```
Arguments
`cAlias` is the target work area alias name.
Returns
`Select()` returns the work area number.
Description
This function returns the work area number for the specified alias name `cAlias`. If no parameter is specified, the current work area will be the return value of the function.
Examples
```
LOCAL cOldArea
USE test NEW
USE names NEW
cOldArea := Select( "names" )
SELECT test
LIST
SELECT cOldArea
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
Alias() Clipper 5.3, Used() Clipper 5.3

Used() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks whether a database is in use in a work area
Syntax
```
Used() → lDbfOpen
```
Arguments
(This function has no arguments)
Returns
`lDbfOpen` True is a database is Used;otherwise False
Description
This function returns a logical true (`.T.`) if a database file is in `USE` in the current or designated work area. If no alias is specified along with this function , it will default to the currently selected work area.
Examples
```
USE test NEW
USE names NEW
? Used()             // --> .T.
? test->( Used() )   // --> .T.
```

```
dbCloseArea()
? Used()            // --> .F.
SELECT test
? Used()            // --> .T.
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Database
See also
`Alias()` Clipper 5.3, `Select()` Clipper 5.3

`CDoW()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts a date to the day of week
Syntax
`CDoW( <dDate> ) → cDay`
Arguments
`dDate` Any date expression.
Returns
`cDay` The current day of week.
Description
This function returns a character string of the day of the week, from a date expression `dDate` passed to it. If a `NULL` date is passed to the function, the value of the function will be a `NULL` byte.
Examples

```
? CDoW( Date() )
IF CDoW( Date() + 10 ) == "Sunday"
   ? "This is a sunny day."
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`CMonth()` Clipper 5.3, `Date()` Clipper 5.3, `Day()` Clipper 5.3, `DoW()` Clipper 5.3

`CMonth()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the name of the month.
Syntax
`CMonth( <dDate> ) → cMonth`

Arguments
`dDate` Any date expression.
Returns
`cMonth` The current month name
Description
This function returns the name of the month (January, February, etc.) from a date expression `dDate` passed to it. If a `NULL` date is passed to the function, the value of the function will be a `NULL` byte.
Examples

```
? CMonth( Date() )
IF CMonth( Date() + 10 ) == "March"
   ? "Have you done your system backup?"
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`CDoW()` Clipper 5.3, `Date()` Clipper 5.3, `DoW()` Clipper 5.3, `DToC()` Clipper 5.3, `Month()` Clipper 5.3, `Year()` Clipper 5.3

`CToD()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts a character string to a date expression
Syntax

```
CToD( <cDateString> ) → dDate
```

Arguments
`cDateString` A character date in format "mm/dd/yy"
Returns
`dDate` A date expression
Description
This function converts a date that has been entered as a character expression to a date expression. The character expression will be in the form `MM/DD/YY` (based on the default value in `SET DATE`) or in the appropriate format specified by the `SET DATE TO` command. If an improper character string is passed to the function, an empty date value will be returned.
Examples

```
Set( _SET_DATEFORMAT, "yyyy-mm-dd" )
? CToD( "2000-12-21" )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, Date/Time
See also
Date() Clipper 5.3, DToS() Clipper 5.3, SET DATEClipper 5.3

Date() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Return the Current OS Date
Syntax
```
Date() → dCurDate
```
Arguments
None
Returns
dCurDate Current system date.
Description
This function returns the current system date.
Examples
```
? Date()
? "Today is", hb_ntos( Day( Date() ) ), "of", CMonth( Date() ), "of",
StrZero( Year( Date() ), 4 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
CMonth() Clipper 5.3, CToD() Clipper 5.3, Day() Clipper 5.3, DToC() Clipper 5.3, DToS() Clipper 5.3, Month() Clipper 5.3

Day() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Return the numeric day of the month.
Syntax
```
Day( <cDate> ) → nMonth
```
Arguments
cDate Any valid date expression.
Returns
nMonth Numeric value of the day of month.
Description
This function returns the numeric value of the day of month from a date.
Examples
```
? Day( Date() )
? Day( Date() + 6325 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
CMonth() Clipper 5.3, CToD() Clipper 5.3, Date() Clipper 5.3, DToC() Clipper 5.3, DToS() Clipper 5.3, Month() Clipper 5.3

Days() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert elapsed seconds into days
Syntax
```
Days( <nSecs> ) → nDay
```
Arguments
nSecs The number of seconds
Returns
nDay The number of days
Description
This function converts nSecs seconds to the equivalent number of days; 86399 seconds represents one day, 0 seconds being midnight.
Examples
```
? Days( 2434234 )
? Days( 63251 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
ElapTime(), Seconds() Clipper 5.3, Secs()

DoW() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Value for the day of week.
Syntax
```
DoW( <dDate> ) → nDay
```
Arguments
dDate Any valid date expression
Returns
nDay The current day number
Description
This function returns the number representing the day of the week for the date expressed as dDate. Returned value range is from 1 (Sunday) to 7 (Saturday).
Examples
```
? DoW( Date() )
```

```
? DoW( Date() - 6584 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
CDoW() Clipper 5.3, Date() Clipper 5.3, Day() Clipper 5.3, DToC() Clipper 5.3, DToS() Clipper 5.3

DToC() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Date to character conversion
Syntax
```
DToC( <dDateString> ) → cDate
```
Arguments
`dDateString` Any date
Returns
`dDate` Character representation of date
Description
This function converts any date expression (a field or variable) expressed as `dDateString` to a character expression in the default format `MM/DD/YY`. The date format expressed by this function is controlled in part by the date format specified in the `SET DATE` command
Examples
```
? DToC( Date() )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
Date() Clipper 5.3, DToS() Clipper 5.3, SET DATEClipper 5.3

DToS() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Date to string conversion
Syntax
```
DToS( <dDateString> ) → cDate
```
Arguments
`dDateString` Any date
Returns

`dDate` String notation of the date
Description
This function returns the value of `dDateString` as a character string in the format of `YYYYMMDD`. If the value of `dDateString` is an empty date, this function will return eight blank spaces.
Examples
```
? DToS( Date() )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`Date()` Clipper 5.3, `DToC()` Clipper 5.3, `DToS()` Clipper 5.3

`ElapTime()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Calculates elapsed time.
Syntax
```
ElapTime( <cStartTime>, <cEndTime> ) → cDifference
```
Arguments
`cStartTime` Start in time as a string format `cEndTime` End time as a string format
Returns
`cDifference` Difference between the times
Description
This function returns a string that shows the difference between the starting time represented as `cStartTime` and the ending time as `cEndTime`. If the stating time is greater then the ending time, the function will assume that the date changed once.
Examples
```
STATIC s_cStartTime
INIT PROCEDURE Startup()
   s_cStartTime := Time()
   RETURN
EXIT PROCEDURE StartExit()
   ? "You used this program by", ElapTime( s_cStartTime, Time() )
   RETURN
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also

Day() Clipper 5.3, Seconds() Clipper 5.3, Secs(), Time() Clipper 5.3

hb_DToT() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a tDateTime value from a dDate parameter
Syntax
```
hb_DToT( <dDate> [, <cTime|nSeconds>] ) → <tDateTime>
```
Arguments
dDate Any valid date expression.
Optional: cTime|nSeconds representing a time of the day value. cTime is a string in a valid time format: "hh:mm:ss.nnn". nSeconds is a numeric value in seconds in the range from 0 to 86399.999~ (60 secs * 60 mins * 24 hours - 1 millisecond)
Returns
tDateTime a dateTime value
Description
This function returns a tDateTime value from a dDate value. Optionally, a second parameter with the time of the day value can be provided which can be represented by either of a string time value or a numeric value in seconds.
Examples
```
? hb_DToT( Date() )  // a dateTime with a empty time part
? hb_DToT( Date(), "14:30:00.500" )  // a dateTime with time part 14:30pm
with 500 milliseconds
? hb_DToT( Date(), 3600 )  // a dateTime with time part 1:00am (one hour)
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
Date() Clipper 5.3

hb_Week() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the week number of year.
Syntax
```
hb_Week( <dDate>, [@<nYear>], [@<nDayOfWeek>] ) → nWeekNumber
```
Arguments
dDate Any valid date expression.
nYear Optional parameter to hold the year of the given date.
nDayOfWeek Optional parameter to hold the day number of week.
Returns
nWeekNumber The ordinal week number of the year into which falls
the given dDate.
Description

This function returns the week number of year for the given dDate. The returned value is an ISO 8601 compliant week number. Optionally, can also be obtained the year and/or the day

number of the week of the given `dDate`, if the `nYear` and/or `nDayOfWeek` parameters have been passed by reference. If `dDate` is an empty date expression, the function returns zero(s). Note: new function available after 2017-02-08 19:36 UTC+0100 commit,

not found in earlier versions.

Examples

```
LOCAL nYear, nDayOfWeek
? hb_Week( 0d20170215, @nYear, @nDayOfWeek ), nYear, nDayOfWeek  // --> 7, 2017, 3
? hb_Week( 0d00000000, @nYear, @nDayOfWeek ), nYear, nDayOfWeek  // --> 0, 0, 0
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`Day()` Clipper 5.3, `Month()` Clipper 5.3, `Year()` Clipper 5.3

`Month()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts a date expression to a month value
Syntax

```
Month( <dDate> ) → nMonth
```

Arguments
`dDate` Any valid date expression
Returns
`nMonth` Corresponding number of the month in the year, ranging from
0 to 12
Description
This function returns a number that represents the month of a given date expression `dDate`. If a `NULL` date (`hb_SToD()`) is passed to the function, the value of the function will be 0.
Examples

```
? Month( Date() )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`CDoW()` Clipper 5.3, `CMonth()` Clipper 5.3, `DoW()` Clipper 5.3, `Year()` Clipper 5.3

Seconds() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the number of elapsed seconds past midnight.
Syntax
`Seconds() → nSeconds`
Arguments
None
Returns
`nSeconds` Number of seconds since midnight
Description
This function returns a numeric value representing the number of elapsed seconds based on the current system time. The system time is considered to start at 0 (midnight); it continues up to 86399 seconds. The value of the return expression is displayed in both seconds and hundredths of seconds.
Examples
```
? Seconds()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`Time()` Clipper 5.3

Secs() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the number of seconds from the system date.
Syntax
`Secs( <cTime> ) → nSeconds`
Arguments
`cTime` Character expression in a time string format
Returns
`nSeconds` Number of seconds
Description
This function returns a numeric value that is a number of elapsed seconds from midnight based on a time string given as `cTime`.
Examples
```
? Secs( Time() )
? Secs( Time() - 10 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, Date/Time
See also
ElapTime(), Seconds() Clipper 5.3, Time() Clipper 5.3

Time() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the system time as a string
Syntax
```
Time() → cTime
```
Arguments
None
Returns
cTime Character string representing time
Description
This function returns the system time represented as a character expression in the format of HH:MM:SS
Examples
```
? Time()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
Date() Clipper 5.3, Seconds() Clipper 5.3

Year() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Extracts the year designator of a given date as a numeric value
Syntax
```
Year( <dDate> ) → nYear
```
Arguments
dDate Any valid date expression
Returns
nYear The year portion of the date.
Description
This function returns the numeric value for the year in dDate. The returned value is not affected by the SET CENTURY and SET DATE settings and will always be a four-digit year number, unless the dDate is an empty date expression, in which case it will be zero.
Examples
```
? Year( Date() )
? Year( 0d32510125 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Date/Time
See also
`Day()` Clipper 5.3, `Month()` Clipper 5.3

`__Run()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Run an external program.
Syntax

```
__Run( <cCommand> )
```

Arguments
`cCommand` Command to execute.
Description

This command runs an external program. Ensure that you have enough free memory to be able to run the external program. Do not use it to run 'Terminate and Stay Resident' programs (in case of MS-DOS) since that causes several problems.

Note: This function is what the `RUN` command preprocesses into.

It is considered bad form to use this function directly. Use the `RUN` command instead.

Examples

```
__Run( "edit " + cMyTextFile )    // Runs an external editor
__Run( "command" )                // Gives a OS shell
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`hb_run()`, `RUN` Clipper 5.3

`__SetCentury()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set the Current Century
Syntax

```
__SetCentury([<lFlag> | <cOnOff> ] ) → lPreviousValue
```

Arguments
optional `lFlag` or `cOnOff` (not case sensitive)
`.T.` or `"ON"` to enable the century setting (4-digit years) `.F.` or `"OFF"` to disable the century setting (2-digit years)

Returns
Either the current or previous century setting as a logical value
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment

GetE() → GetEnv() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Obtains a system environmental setting.
Syntax
```
GetE( <cEnviroment> ) → cReturn
```
Arguments
`cEnviroment` Environmental variable to obtain.
Returns
`cReturn` Value of the Environment Variable.
Description

This function yields a string that is the value of the environment variable `cEnviroment`, which is stored at the system-level.

If no environment variable is found, an empty string is returned.

Examples
```
? GetE( "PATH" )
? GetE( "CONFIG" )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`GetEnv()` Clipper 5.3

GetEnv() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Obtains a system environmental setting.
Syntax
```
GetEnv( <cEnviroment> ) → cReturn
```
Arguments
`cEnviroment` Environmental variable to obtain.
Returns

`cReturn` Value of the Environment Variable.
Description

This function yields a string that is the value of the environment variable `cEnviroment`, which is stored at the system-level.

If no environment variable is found, an empty string is returned.

Examples
```
? GetEnv( "PATH" )
? GetEnv( "CONFIG" )
? GetEnv( "HARBOURCMD", "-n -l -es2" )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`GetE()`

`hb_eol()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the newline character(s) to use with the current OS
Syntax
```
hb_eol() → cString
```
Returns
`cString` A character string containing the character or characters required to move the screen cursor or print head to the start of a new line.
Description

Returns a character string containing the character or characters required to move the screen cursor or print head to the start of a new line for the operating system that the program is running on (or thinks it is running on, if an OS emulator is being used).

Under `HB_OS_UNIX` operating system the return value is the Line-Feed character (0x0a, `Chr( 10 )`); with other operating systems (like DOS) the return value is the Carriage-Return plus Line-Feed characters (0x0d 0x0a, `Chr( 13 ) + Chr( 10 )`).

Examples
```
// Get the newline character(s) for the current OS.
OutStd( "Hello World!" + hb_eol() )
? HB_ISSTRING( hb_eol() )
? Len( hb_eol() ) <= 2
```
Status
Ready
Compliance
Harbour specific

Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
OS() Clipper 5.3, OutErr() Clipper 5.3, OutStd() Clipper 5.3

hb_GetEnv() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Obtains a system environmental setting.
Syntax
```
hb_GetEnv( <cEnviroment>, [<cDefaultValue>] ) → cReturn
```
Arguments
`cEnviroment` Environmental variable to obtain.
`cDefaultValue` Optional value to return if `cEnvironment` is not found.
Returns
`cReturn` Value of the environment variable or `cDefaultValue` or an empty string.
Description

This function yields a string that is the value of the environment variable `cEnviroment`, which is stored at the system-level.

If no environment variable can be found, the value of the function will be `cDefaultValue` if it is passed, else an empty string.

Examples
```
? hb_GetEnv( "PATH" )
? hb_GetEnv( "CONFIG" )
? hb_GetEnv( "HARBOURCMD", "-n -l -es2" )
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
GetE(), GetEnv() Clipper 5.3

OS() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the current operating system.
Syntax
```
OS() → cOperatingSystem
```
Returns
`cOperatinSystem` The current operating system.
Description

This function will return the current operating system.
Examples
```
? OS()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
API, Environment

RUNSource code^ | ☰ | Improve this doc

Run an external program.
Syntax
```
RUN <cCommand>
```
Arguments
`cCommand` Command to execute.
Description
This command runs an external program. Please make sure that you have enough free memory to be able to run the external program. Do not use it to run Terminate and Stay Resident programs (in case of DOS) since that causes several problems.
Examples
```
RUN ( "edit " + cMyTextFile )  // Runs an external editor
RUN command                    // Gives a OS shell
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
RUNClipper 5.3

Set()Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Changes or evaluated environmental settings
Syntax
```
Set( <nSet> [, <xNewSetting> [, <xOption> ] ] ) → xPreviousSetting
```
Arguments
`nSet` Set Number
`xNewSetting` Any expression to assign a value to the setting
`xOption` Logical expression
`nSet xNewSetting xOption`
`_SET_ALTERNATE lFlag | cOnOff`

If enabled, QOut() and QQOut() write to the screen and to a file, provided that a file has been opened or created with _SET_ALTFILE. If disabled, which is the default, QOut() and QQOut() only write to the screen (and/or to the PRINTFILE). Defaults to disabled.

_SET_ALTFILE cFileName lAdditive

When set, creates or opens file to write QOut() and QQOut() output to. If lAdditive is TRUE and the file already exists, the file is opened and positioned at end of file. Otherwise, the file is created. If a file is already opened, it is closed before the new file is opened or created (even if it is the same file). The default file extension is .txt. There is no default file name. Call with an empty string to close the file.

_SET_AUTOPEN lFlag | cOnOff

TODO: Document

_SET_AUTORDER lFlag | cOnOff

TODO: Document

_SET_AUTOSHARE lFlag | cOnOff

TODO: Document

_SET_BELL lFlag | cOnOff

When enabled, the bell sounds when the last position of a GET is reached and/or when a GET validation fails. Disabled by default.

_SET_CANCEL lFlag | cOnOff

When enabled, which is the default, pressing Alt+C or Ctrl+Break terminates the program. When disabled, both keystrokes can be read by Inkey(). Note: SET KEY has precedence over SET CANCEL.

_SET_COLOR cColorSet

Sets the current color scheme, using color pairs in the sequence "<standard>, <enhanced>, <border>, <background>, <unselected>". Each color pair uses the format "<foreground>/<background>". The color codes are space or "N" for black, "B" for blue, "G" for green, "BG" for Cyan, "R" for red, "RB" for magenta, "GR" for brown, "W" for white, "N+" for gray, "B+" for bright blue, "G+" for bright green, "BG+" for bright cyan, "R+" for bright red, "RB+" for bright magenta, "GR+" for yellow, and "W+" for bright white. Special codes are "I" for inverse video, "U" for underline on a monochrome monitor (blue on a color monitor), and "X" for blank. The default color is "W/N,N/W,N,N,N/W".

_SET_CONFIRM lFlag | cOnOff

If enabled, an exit key must be pressed to leave a GET. If disabled, which is the default, typing past the end will leave a GET.

_SET_CONSOLE lFlag | cOnOff

If enabled, which is the default, all screen output goes to the screen. When disabled, screen output is suppressed (Note: This setting does not affect OutStd() or OutErr()).

_SET_CURSOR nCursorType

If enabled, which is the default, the cursor is displayed on screen. If disabled, the screen cursor is hidden.

_SET_DATEFORMAT cDateFormat

Sets the default date format for display, date input, and date conversion. Defaults to American mm/dd/yy. Other formats include ANSI yy.mm.dd, British dd/mm/yy, French dd/mm/yy, German dd.mm.yy, Italian dd-mm-yy, Japan yy/mm/dd, and USA mm-dd-yy. SET CENTURY modifies the date format. SET CENTURY ON replaces the ys with YYYY. SET CENTURY OFF replaces the ys with YY.

_SET_DEBUG lStatus

When set to .T., pressing Alt+D activates the debugger. When set to .F., which is the default, Alt+D can be read by Inkey(). (Also affected by AltD(1) and AltD(0))

_SET_DECIMALS nNumberOfDecimals

Sets the number of decimal digits to use when displaying printing numeric values when SET FIXED is ON. Defaults to 2. If SET FIXED is OFF, then SET DECIMALS is only used to determine the number of decimal digits to use after using Exp(), Log(), Sqrt(), or division. Other math operations may adjust the number of decimal digits that the result will display. Note: This never affects the precision of a number. Only the display format is affected.

_SET_DEFAULT cDefaultDirectory

Sets the default directory in which to open, create and check for files. Defaults to current directory (blank).

_SET_DELETED lFlag | cOnOff

If enabled, deleted records will be processed. If disabled, which is the default, deleted records will be ignored.

_SET_DELIMCHARS cDelimiters

Sets the GET delimiter characters. Defaults to "::".

_SET_DELIMITERS lFlag | cOnOff

If enabled, GETs are delimited on screen. If disabled, which is the default, no GET delimiters are used.

_SET_DEVICE cDeviceName

Selects the output device for DevOut(). When set to "PRINTER", all output is sent to the printer device or file set by _SET_PRINTFILE. When set to anything else, all output is sent to the screen. Defaults to "SCREEN".

_SET_EOF lFlag | cOnOff

Defaults to FALSE on *nix, but defaults to TRUE on everything else. If set to FALSE, then Chr(26) does not get written when using COPY TO DELIMITED, COPY TO SDF, or when closing any of the various text files that are created using various SET values. [This is a Harbour extension]

_SET_EPOCH nYear

Determines how to handle the conversion of 2-digit years to 4 digit years. When a 2-digit year is greater than or equal to the year part of the epoch, the century part of the epoch is added to the year. When a 2-digit year is less than the year part of the epoch, the century part of the epoch is incremented and added to the year. The default epoch is 1900, which converts all 2-digit years to 19xx. Example: If the epoch is set to 1950, 2-digit years in the range from 50 to 99 get converted to 19xx and 2-digit years in the range 00 to 49 get converted to 20xx.

_SET_ESCAPE lFlag | cOnOff

When enabled, which is the default, pressing Esc will exit a READ. When disabled, pressing Esc during a READ is ignored, unless the Esc key has been assigned to a function using SET KEY.

_SET_EVENTMASK nEventCodes

Determines which events Inkey() will respond to. INKEY_MOVE allows mouse movement events. INKEY_LDOWN allows the left mouse button down click. INKEY_LUP allows the left mouse button up click. INKEY_RDOWN allows the right mouse button down click. INKEY_RUP allows the right mouse button up clock. INKEY_KEYBOARD allows keyboard keystrokes. INKEY_ALL allows all of the preceding events. Events may be combined (e.g., using INKEY_LDOWN + INKEY_RUP will allow left mouse button down clicks and right mouse button up clicks). The default is INKEY_KEYBOARD.

_SET_EXACT lFlag | cOnOff

When enabled, all string comparisons other than == exclude trailing spaces when checking for equality. When disabled, which is the default, all string comparisons other than == treat two strings as equal if the right hand string is "" or if the right hand string is shorter than or the same length as the left hand string and all of the characters in the right hand string match the corresponding characters in the left hand string.

_SET_EXCLUSIVE lFlag | cOnOff

When enabled, which is the default, all database files are opened in exclusive mode. When disabled, all database files are opened in shared mode. Note: The EXCLUSIVE and SHARED clauses of the USE command can be used to override this setting.

_SET_EXIT lFlag | cOnOff

Toggles the use of Up and Down as READ exit keys. Specifying true (.T.) enables them as exit keys, and false (.F.) disables them. Used internally by the ReadExit() function.

_SET_EXTRA lFlag | cOnOff

QUESTION: What is this for? It does not affect _SET_EXTRAFILE in CA-Cl*pper!

_SET_EXTRAFILE cFileName lAdditive

When set, creates or opens file to write QOut() and QQOut() output to. If lAdditive is TRUE and the file already exists, the file is opened and positioned at end of file. Otherwise, the file is created. If a file is already opened, it is closed before the new file is opened or created (even if it is the same file). The default file extension is .prn. There is no default file name. Call with an empty string to close the file.

_SET_FIXED lFlag | cOnOff

When enabled, all numeric values will be displayed and printed with the number of decimal digits set by SET DECIMALS, unless a PICTURE clause is used. When disabled, which is the default, the number of decimal digits that are displayed depends upon a variety of factors. See _SET_DECIMALS for more.

_SET_INSERT lFlag | cOnOff

When enabled, characters typed in a GET or MemoEdit() are inserted. When disabled, which is the default, characters typed in a GET or MemoEdit() overwrite. Note: This setting can also be toggled between on and off by pressing the Insert key during a GET or MemoEdit().

_SET_INTENSITY lFlag | cOnOff

When enabled, which is the default, GETs and PROMPTs are displayed using the enhanced color setting. When disabled, GETs and PROMPTs are displayed using the standard color setting.

_SET_LANGUAGE cLanguageID

Specifies the language to be used for Harbour messages. [This is a Harbour extension]

_SET_MARGIN nColumns

Sets the left margin for all printed output. The default value is 0. Note: PCol() reflects the printer's column position including the margin (e.g., SET MARGIN TO 5 followed by DevPos(5, 10) makes PCol() return 15).

_SET_MBLOCKSIZE nMemoBlockSize

TODO: Document

_SET_MCENTER lFlag | cOnOff

If enabled, display PROMPTs centered on the MESSAGE row. If disabled, which is the default, display PROMPTS at column position 0 on the MESSAGE row.

_SET_MESSAGE nRow

If set to 0, which is the default, PROMPTs are always suppressed. Otherwise, PROMPTs are displayed on the set row. Note: It is not possible to display prompts on the top-most screen row, because row 0 is reserved for the SCOREBOARD, if enabled.

_SET_MFILEEXT cMemoFileExt

TODO: Document

_SET_OPTIMIZE lFlag | cOnOff

TODO: Document

_SET_PATH cDirectories

Specifies a path of directories to search through to locate a file that cannot be located in the DEFAULT directory. Defaults to no path "". Directories must be separated by a semicolon (e.g., /hb/bin;/hb/tests).

_SET_PRINTER lFlag | cOnOff

If enabled, QOut() and QQOut() write to the screen and to a file, provided that a file has been opened or created with _SET_ALTFILE. If disabled, which is the default, QOut() and QQOut() only write to the screen (and/or to the ALTFILE).

_SET_PRINTFILE cFileName lAdditive

When set, creates or opens file to write QOut(), QQOut() and DevOut() output to. If lAdditive is TRUE and the file already exists, the file is opened and positioned at end of file. Otherwise, the file is created. If a file is already opened, it is closed before the new file is opened or created (even if it is the same file). The default file extension is .prn. The default file name is PRN, which maps to the default printer device. Call with an empty string to close the file.

_SET_SCOREBOARD lFlag | cOnOff

When enabled, which is the default, READ and MemoEdit() display status messages on screen row 0. When disabled, READ and MemoEdit() status messages are suppressed.

_SET_SCROLLBREAK lFlag | cOnOff

QUESTION: What is this flag for?

_SET_SOFTSEEK lFlag | cOnOff

When enabled, a SEEK that fails will position the record pointer to the first key that is higher than the sought after key or to LastRec() + 1 if there is no higher key. When disabled, which is the default, a SEEK that fails will position the record pointer to LastRec() + 1.

_SET_STRICTREAD lFlag | cOnOff

TODO: Document

_SET_TYPEAHEAD nKeyStrokes

Sets the size of the keyboard typeahead buffer. Defaults to 50. The minimum is 16 and the maximum is 4096.

_SET_UNIQUE lFlag | cOnOff

When enabled, indexes are not allowed to have duplicate keys. When disabled, indexes are allowed duplicate keys.

_SET_VIDEOMODE nValue

TODO: Document

_SET_WRAP lFlag | cOnOff

When enabled, lightbar menus can be navigated from the last position to the first and from the first position to the last. When disabled, which is the default, there is a hard stop at the first and last positions.

Returns

Set() The current or previous setting

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Environment

SetMode() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Change the video mode to a specified number of rows and columns

Syntax

```
SetMode( <nRows>, <nCols> ) → lSuccess
```

Arguments

nRows is the number of rows for the video mode to set.

`nCols` is the number of columns for the video mode to set.
Returns
`SetMode()` returns true if the video mode change was successful; otherwise, it returns false.
Description

`SetMode()` is a function that change the video mode depend on the video card and monitor combination, to match the number of rows and columns specified. Note that there are only a real few combination or rows/cols pairs that produce the video mode change. The followings are available for `GTDOS`:

12 rows x 40 columns	12 rows x 80 columns
25 rows x 40 columns	25 rows x 80 columns
28 rows x 40 columns	28 rows x 80 columns
50 rows x 40 columns	43 rows x 80 columns
	50 rows x 80 columns

The follow modes are available to Windows

25 rows x 40 columns	25 rows x 80 columns
50 rows x 40 columns	43 rows x 80 columns
50 rows x 80 columns	

Some modes only are available for color and/or VGA monitors. Any change produced on the screen size is updated in the values returned by `MaxRow()` and `MaxCol()`.

Examples
```
// The first example change to a 12 lines of display mode:
IF SetMode( 12, 40 )
   ? "Hey man are you blind?"
ELSE
   ? "Mom, bring me my glasses!"
ENDIF

// Next example change to a 50 lines mode:
IF SetMode( 50, 80 )
   ? "This wonderful mode was successfully set"
ELSE
   ? "Wait. this monitor is not made of rubber!"
ENDIF
```
Status
Ready
Compliance
Some of these modes are not available in CA-Cl*pper
Platforms
Available on all platforms
Tags
API, Environment
See also
`MaxCol()` Clipper 5.3Clipper Tools 3, `MaxRow()` Clipper 5.3Clipper Tools 3

`SetTypeahead()` Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Sets the typeahead buffer to given size.

Syntax
`SetTypeahead( <nSize> ) → nPreviousSize`
Arguments
`nSize` is a valid typeahead size.
Returns
`nPreviousSize` The previous state of `_SET_TYPEAHEAD`
Description
This function sets the typeahead buffer to a valid given size as is `Set( _SET_TYPEAHEAD )` where used.
Examples
```
// Sets typeahead to 12
SetTypeahead( 12 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`__Accept()`, `__Input()`

`Tone()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sound a tone with a specified frequency and duration.
Syntax
`Tone( <nFrequency>, <nDuration> ) → NIL`
Arguments
`nFrequency` A non-negative numeric value that specifies the frequency of the tone in hertz.
`nDuration` A positive numeric value which specifies the duration of the tone in 1/18 of a second units.
Returns
`Tone()` always returns `NIL`.
Description
`Tone()` is a sound function that could be used to irritate the end user, his or her dog, and the surrounding neighborhood. The frequency is limited to the range 0 to 32767 Hz.
Examples
```
// Good sound
Tone(  500, 1 )
Tone( 4000, 1 )
Tone( 2500, 1 )

// Bad sound
Tone(  300, 1 )
Tone(  499, 5 )
Tone(  700, 5 )

Tone( 800, 1 )       // same as hb_BChar( 7 )
Tone( 32000, 200 )   // any dogs around yet?
Tone( 130.80, 1 )    // musical note - C
Tone( 400, 0 )       // short beep
```

```
Tone( 700 )          // short beep
Tone( 10, 18.2 )     // 1 second delay
Tone( -1 )           // 1/18.2 second delay
Tone()               // 1/18.2 second delay
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`Chr()` Clipper 5.3, `SET BELL` Clipper 5.3

`Version()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the version of Harbour compiler
Syntax

```
Version() → cReturn
```

Arguments
None
Returns
`cReturn` String containing the Harbour version
Description
This function returns the current Harbour version.
Examples

```
? Version()
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Environment
See also
`OS()` Clipper 5.3

`Break()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Exits from a `BEGIN SEQUENCE` block
Syntax

```
Break( <xExp> )
```

Arguments
`xExp` is any valid expression. It is always required. If do not want to pass any argument, just use `NIL`.
Description

This function passes control to the `RECOVER` statement in a `BEGIN SEQUENCE` block.
Examples
```
Break( NIL )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Error
See also
`BEGIN SEQUENCE`

`ErrorSys()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Install default error handler
Syntax
```
ErrorSys() → NIL
```
Arguments
None.
Returns
`ErrorSys()` always return `NIL`.
Description
`ErrorSys()` is called upon startup by Harbour and installs the default error handler. Normally you should not call this function directly, instead use `ErrorBlock()` to install your own error handler.
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Error
See also
`Error class`, `ErrorBlock()`

`__Quit()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Terminates an application.
Syntax
```
__Quit()
```
Arguments
None
Description
This function terminates the current application and returns to the system.
Examples

```
EndApp( .F. )
EndApp( .T. )
STATIC PROCEDURE EndApp( lYesNo )
   IF lYesNo
      __Quit()
   ENDIF
   RETURN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Events
See also
`QUIT`

`__SetFunction()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Assign a character string to a function key
Syntax

```
__SetFunction( <nFunctionKey>, [<cString>] ) → NIL
```

Arguments

`nFunctionKey` is a number in the range 1..40 that represent the function key to be assigned. `cString` is a character string to set. If `cString` is not specified, the function key is going to be set to `NIL` releasing by that any previous `__SetFunction()` or `SetKey()` for that function.
Returns

`__SetFunction()` always return `NIL`.
Description

`__SetFunction()` assign a character string with a function key, when this function key is pressed, the keyboard is stuffed with this character string. `__SetFunction()` has the effect of clearing any `SetKey()` previously set to the same function number and vice versa.

nFunctionKey	Key to be set
1 .. 12	<F1> .. <F12>
13 .. 20	<Shift+F3> .. <Shift+F10>
21 .. 30	<Ctrl+F1> .. <Ctrl+F10>
31 .. 40	<Alt+F1> .. <Alt+F10>

`SET FUNCTION` command is preprocessed into `__SetFunction()` function during compile time.

Examples

```
#include "inkey.ch"
LOCAL cTest, GetList := {}
// Associate <F1> with a string
__SetFunction( 1, "I Am Lazy" + Chr( K_ENTER ) )
CLS
cTest := Space( 20 )
@ 10, 0 SAY "type something or <F1> for lazy mode" GET cTest
```

```
READ
? cTest
```
Status
Ready
Compliance
Harbour use 11 and 12 to represent F11 and F12, while CA-Cl*pper use 11 and 12 to represent Shift+F1 and Shift+F2.
Platforms
Available on all platforms
File
Library is core
Tags
API, Events
See also
`__Keyboard()`, `Inkey()` Clipper 5.3, `SetKey()` Clipper 5.3, `SET KEY` Clipper 5.3

`__Wait()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Stops the application until a key is pressed.
Syntax
```
__Wait( <cMessage> ) → cKey
```
Arguments
`cMessage` is a string.
Returns
Pressed key.
Description
This function stops the application until a key is pressed. The key must be in the range 32..255. Control keys are not processed.
Examples
```
LOCAL cKey
// Wait for a key stroke
__Wait( "Press a key to continue" )

DO WHILE ! cKey == "Q"
   cKey := __Wait( "Press 'Q' to continue" )
ENDDO
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Events
See also
`__Accept()`, `__Input()`

`hb_SetKeyCheck()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Implements common hot-key activation code
Syntax

```
hb_SetKeyCheck( <nKey> [, <p1> ][, <p2> ][, <p3> ] )
```

Arguments

`nKey` is a numeric key value to be tested code-block, if executed

`p1`..`p3` are optional parameters that will be passed to the code-block

Returns

True if there is a hot-key associated with `nKey` and it was executed; otherwise False If there is a hot-key association (before checking any condition): - if there is a condition-block, it is passed one parameter - `nKey` - when the hot-key code-block is called, it is passed 1 to 4 parameters, depending on the parameters passed to `hb_SetKeyCheck()`. Any parameters so passed are directly passed to the code-block, with an additional parameter being `nKey`

Description

`hb_SetKeyCheck()` is intended as a common interface to the `SetKey()` functionality for such functions as `AChoice()`, `dbEdit()`, `MemoEdit()`, `ACCEPT`, `INPUT`, `READ`, and `WAIT`

Examples

```
// FIXME
#include "inkey.ch"
// within ReadModal()
IF hb_SetKeyCheck( K_ALT_X, GetActive() )
   // some other processing ...
ENDIF
// within TBrowse handler
CASE hb_SetKeyCheck( nInkey, oTBrowse )
   RETURN
CASE nInKey == K_ESC
   // some other processing ...
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Events

See also

`hb_SetKeySave()`, `SetKey()` Clipper 5.3

`hb_SetKeyGet()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine a set-key code block and condition-block

Syntax

```
hb_SetKeyGet( <nKey> [, <bConditionByRef> ] )
```

Arguments

`anKey` is an numeric key value

`bConditionByRef` is an optional return-parameter

Returns

Current assigned action-block

Description

The `hb_SetKeyGet()` function returns the current code-block assigned to a key, and optionally assigns the condition-block to the return-parameter

Examples

```
#include "inkey.ch"
```

```
LOCAL bOldF10, bOldF10Cond
bOldF10 := hb_SetKeyGet( K_F10, @bOldF10Cond )
// some other processing ...
SetKey( K_F10, bOldF10, bOldF10Cond )
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Events

See also

`hb_SetKeyCheck()`, `hb_SetKeySave()`, `SetKey()` Clipper 5.3

`hb_SetKeySave()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a copy of internal set-key list, optionally overwriting

Syntax

```
hb_SetKeySave( [ <OldKeys> ] )
```

Arguments

`OldKeys` is an optional set-key list from a previous call to `hb_SetKeySave()`, or `NIL` to clear current set-key list

Returns

Current set-key list

Description

`hb_SetKeySave()` is designed to act like the `Set()` function which returns the current state of an environment setting, and optionally assigning a new value. In this case, the "environment setting" is the internal set-key list, and the optional new value is either a value returned from a previous call to `hb_SetKeySave()` - to restore that list, or the value of `NIL` to clear the current list.

Examples

```
LOCAL aKeys := hb_SetKeySave( NIL )  // removes all current set=keys
// some other processing ...
hb_SetKeySave( aKeys )
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Events

See also

`SetKey()` Clipper 5.3

`SetKey()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Assign an action block to a key
Syntax
```
SetKey( <anKey> [, <bAction> [, <bCondition> ] ] )
```
Arguments
`anKey` is either a numeric key value, or an array of such values
`bAction` is an optional code-block to be assigned
`bCondition` is an optional condition code-block
Returns
Current assigned action-block
Description
The `SetKey()` function returns the current code-block assigned to a key when called with only the key value. If the action block (and optionally the condition block) are passed, the current block is returned, and the new code block and condition block are stored. A group of keys may be assigned the same code block/condition block by using an array of key values in place on the first parameter.
Examples
```
#include "getexit.ch"
#include "inkey.ch"
LOCAL bOldF10 := SetKey( K_F10, {|| Yahoo() } )
LOCAL bBlock
// some other processing ...
SetKey( K_F10, bOldF10 )
// some other processing ...
bBlock := SetKey( K_SPACE )
IF bBlock != NIL  // ...

   // make <F10> exit current get, but only if in a get - ignores other
   // wait states such as menus, achoices, etc...
   SetKey( K_F10, {|| GetActive():State := GE_WRITE }, ;
      {|| GetActive() != NIL } )
ENDIF
```
Status
Ready
Compliance
`SetKey()` is mostly CA-Cl*pper compliant. The only difference is the addition of the condition code-block parameter, allowing set-keys to be conditionally turned off or on. This condition-block cannot be returned once set - see `hb_SetKeyGet()`
Platforms
Available on all platforms
File
Library is core
Tags
API, Events
See also
`hb_SetKeyGet()`, `hb_SetKeySave()`

`dbEval()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Performs a code block operation on the current Database
Syntax
```
dbEval( <bBlock>,
[<bFor>], [<bWhile>],
[<nNext>], [<nRecord>],
[<lRest>] ) → NIL
```

Arguments
`bBlock` Operation that is to be performed
`bFor` Code block for the For condition
`bWhile` Code block for the `WHILE` condition
`nNext` Number of `NEXT` records to process
`nRecord` Record number to work on exactly
`lRest` Toggle to rewind record pointer
Returns
`dbEval()` always returns `NIL`
Description
Performs a code block operation on the current Database
Examples

```
LOCAL nCount

USE test

dbGoto( 4 )
? RecNo()
COUNT TO nCount NEXT 10
? RecNo(), nCount
COUNT TO nCount
? RecNo(), nCount
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is rdd
Tags
API, Execute and Execution
See also
`Eval()` Clipper 5.3

`Eval()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Evaluate a code block
Syntax

```
Eval( <bBlock> [, <xVal> [,...] ] ) → xExpression
```

Arguments
`bBlock` Code block expression to be evaluated
`xVal` Argument to be passed to the code block expression
`xVal...` Argument list to be passed to the code block expression
Returns
`xExpression` The result of the evaluated code block
Description

This function evaluates the code bloc expressed as `bBlock` and returns its evaluated value. If their are multiple expressions within the code block, the last expression will be value of this function.

If the code block requires parameters to be passed to it, they are specified in the parameter list `xVal` and following. Each parameter is separated by a comma within the expression list.

Examples

```
LOCAL bBlock := {|| NIL }
? Eval( 1 )
? Eval( @bBlock )

? Eval( {| p1 | p1 }, "A", "B" )
? Eval( {| p1, p2 | p1 + p2 }, "A", "B" )
? Eval( {| p1, p2, p3 | p1 }, "A", "B" )
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Execute and Execution

See also

`AEval()` Clipper 5.3, `dbEval()` Clipper 5.3

`__Dir()` *Source code in Repository in Discussions ^ | ☰ | Improve this doc

Display listings of files

Syntax

```
__Dir( [<cFileMask>] ) → NIL
```

Arguments

`cFileMask` File mask to include in the function return. It could contain path and standard wildcard characters as supported by your OS (like * and ?). If `cFileMask` contains no path, then `SET DEFAULT` path is used to display files in the mask.

Returns

`__Dir()` always returns `NIL`.

Description

If no `cFileMask` is given, `__Dir()` displays information about all *.dbf in the `SET DEFAULT` path. This information contains: file name, number of records, last update date and the size of each file.

If `cFileMask` is given, `__Dir()` list all files that match the mask with the following details: Name, Extension, Size, Date.

`DIR` command is preprocessed into `__Dir()` function during compile time.

`__Dir()` is a compatibility function, it is superseded by `Directory()` which return all the information in a multidimensional array.

If long file names are available Harbour will use/display the first 15 characters else Harbour will use/display a 8.3 file name consistent with CA-Cl*pper.

Examples

```
__Dir()      // information for all DBF files in current directory

__Dir( "*.dbf" )         // list all DBF file in current directory

// list all PRG files in Harbour Run-Time library
// for MS-DOS compatible operating systems
__Dir( hb_DirSepToOS( "src/rtl/*.prg" ) )

// list all files in the public section on a Unix like machine
__Dir( hb_DirSepToOS( "/pub" ) )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
ADir() Clipper 5.3, Directory(), DIR Clipper 5.3, SET DEFAULT Clipper 5.3

ADir() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Fill pre-defined arrays with file/directory information
Syntax

```
ADir( [<cFileMask>], [<aName>], [<aSize>], [<aDate>],
      [<aTime>], [<aAttr>] ) → nDirEntries
```

Arguments
`cFileMask` File mask to include in the function return. It could contain path and standard wildcard characters as supported by your OS (like * and ?). If you omit `cFileMask` or if `cFileMask` contains no path, then the path from `SET DEFAULT` is used.
`aName` Array to fill with file name of files that meet `cFileMask`. Each element is a Character string and include the file name and extension without the path. The name is the long file name as reported by the OS and not necessarily the 8.3 uppercase name.
`aSize` Array to fill with file size of files that meet `cFileMask`. Each element is a Numeric integer for the file size in Bytes. Directories are always zero in size.
`aDate` Array to fill with file last modification date of files that meet `cFileMask`. Each element is of type Date.
`aTime` Array to fill with file last modification time of files that meet `cFileMask`. Each element is a Character string in the format HH:mm:ss.
`aAttr` Array to fill with attribute of files that meet `cFileMask`. Each element is a Character string, see `Directory()` for information about attribute values. If you pass array to `aAttr`, the function is going to return files with normal, hidden, system and directory attributes. If `aAttr` is not specified or with type other than Array, only files with normal attribute would return.
Returns
`ADir()` return the number of file entries that meet `cFileMask`
Description

`ADir()` return the number of files and/or directories that match a specified skeleton, it also fill a series of given arrays with the name, size, date, time and attribute of those files. The passed

arrays should pre-initialized to the proper size, see example below. In order to include hidden, system or directories `aAttr` must be specified.

`ADir()` is a compatibility function, it is superseded by `Directory()` which returns all the information in a multidimensional array.

Examples

```
LOCAL aName, aSize, aDate, aTime, aAttr, tmp
LOCAL nLen := ADir( "*.dbf" )  // Number of JPG files in this directory
IF nLen > 0
   aName := Array( nLen )  // make room to store the information
   aSize := Array( nLen )
   aDate := Array( nLen )
   aTime := Array( nLen )
   aAttr := Array( nLen )
   ADir( "*.prg", aName, aSize, aDate, aTime, aAttr )
   FOR tmp := 1 TO nLen
      ? ;
         aName[ tmp ], ;
         aSize[ tmp ], ;
         aDate[ tmp ], ;
         aTime[ tmp ], ;
         aAttr[ tmp ]
   NEXT
ELSE
   ? "This directory is clean from smut"
ENDIF
```

Status

Ready

Compliance

`aName` is going to be filled with long file name and not necessarily the 8.3 uppercase name.

Platforms

Available on all platforms

File

Library is core

Tags

API, FileSys

See also

`Array()` Clipper 5.3, `Directory()`, `SET DEFAULT` Clipper 5.3

`CurDir()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the current OS directory name.

Syntax

```
CurDir( [<cDrive>] ) → cPath
```

Arguments

`cDrive` OS drive letter

Returns

`cPath` Name of directory

Description

This function yields the name of the current OS directory on a specified drive. If `cDrive` is not specified, the currently logged drive will be used.

This function should not return the leading and trailing (back)slashes.

If an error has been detected by the function, or the current OS directory is the root, the value of the function will be a `NULL` byte.

Examples
```
? CurDir()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`File()` Clipper 5.3

`DirChange()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Changes the directory
Syntax
```
DirChange( <cDirectory> ) → nError
```
Arguments
`cDirectory` The name of the directory you want do change into.
Returns
`nError` 0 if directory was successfully changed, otherwise the number of the last error.
Description
This function attempt to change the current directory to the one specified in `cDirectory`. If this function fails, it will return the last OS error code number. See `FError()` function for the description of the error.
Examples
```
IF DirChange( hb_DirSepToOS( "./mydir" ) ) == 0
   ? "Change to directory was successful"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.3 function available in builds with `HB_COMPAT_C53` option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`DirRemove()` Clipper 5.3Clipper Tools 3, `DiskChange()` Clipper Tools 3, `DiskName()` Clipper Tools 3, `FError()` Clipper 5.3, `IsDisk()`, `MakeDir()`

`DirRemove()` → `hb_DirDelete()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Attempt to remove an directory
Syntax
`DirRemove( <cDirectory> ) → nError`
Arguments
`cDirectory` The name of the directory you want to remove.
Returns
`nError` 0 if directory was successfully removed, otherwise the number of the last error.
Description
This function attempt to remove the specified directory in `cDirectory` If this function fails, it will return the last OS error code number. See `FError()` function for the description of the error.
Examples

```
LOCAL cDir
IF DirRemove( cDir := hb_DirSepToOS( "./mydir" ) ) == 0
   ? "Removing directory", cDir, "was successful"
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.3 function available in builds with `HB_COMPAT_C53` option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`DirChange()` Clipper 5.3Clipper Tools 3, `DiskChange()` Clipper Tools 3, `DiskName()` Clipper Tools 3, `FError()` Clipper 5.3, `IsDisk()`, `MakeDir()`

`DiskSpace()` Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Get the amount of space available on a disk
Syntax
`DiskSpace( [<nDrive>] ) → nDiskBytes`
Arguments
`nDrive` The number of the drive you are requesting info on where 1 = A, 2 = B, etc. For 0 or no parameter, DiskSpace will operate on the current drive. The default is 0
Returns
`nDiskBytes` The number of bytes on the requested disk that match the requested type.
Description

By default, this function will return the number of bytes of free space on the current drive that is available to the user requesting the information.

If information is requested on a disk that is not available, a runtime error 2018 will be raised.

Examples

```
? "You can use:", hb_ntos( DiskSpace() ), "bytes"

// See tests/diskspac.prg for another example
```

Status

Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Files
Library is core
Header is fileio.ch
Tags
API, FileSys

FClose() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Closes an open file
Syntax
```
FClose( <nHandle> ) → lSuccess
```
Arguments
nHandle File handle
Returns
lSuccess Logical TRUE (.T.) or FALSE (.F.)
Description
This function closes an open file with a file handle of nHandle and writes the associated buffers to the disk. The nHandle value is derived from the FCreate() or FOpen() function.
Examples
```
#include "fileio.ch"
nHandle := FOpen( "test.txt" )
? FSeek( nHandle, 0, FS_END )
FClose( nHandle )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
FCreate() Clipper 5.3, FError() Clipper 5.3, FOpen() Clipper 5.3, FRead() Clipper 5.3, FWrite() Clipper 5.3

FCreate() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Creates a file.
Syntax
```
FCreate( <cFile>, [<nAttribute>] ) → nHandle
```
Arguments
cFile is the name of the file to create.
nAttribute Numeric code for the file attributes.
Returns
nHandle Numeric file handle to be used in other operations.

Description

This function creates a new file with a file name of `cFile`. The default value of `nAttribute` is 0 and is used to set the attribute byte for the file being created by this function. The return value will be a file handle that is associated with the new file. This number will be between zero to 65535, inclusive. If an error occurs, the return value of this function will be `F_ERROR`.

If the file `cFile` already exists, the existing file will be truncated to a file length of 0 bytes.

If specified, the following table shows the value for `nAttribute` and their related meaning to the file `cFile` being created by this function.

<nAttribute>	Meaning
FC_NORMAL	Normal/Default, Read/Write
FC_READONLY	Read-only file attribute is set
FC_HIDDEN	Hidden, Excluded from normal DIR search
FC_SYSTEM	Create, Excluded from normal DIR search

Examples

```
#include "fileio.ch"
IF ( nh := FCreate( "test.txt" ) ) == F_ERROR
    ? "Cannot create file"
ENDIF
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Files

Library is core

Header is `fileio.ch`

Tags

API, FileSys

See also

`FClose()` Clipper 5.3, `FError()` Clipper 5.3, `FOpen()` Clipper 5.3, `FRead()` Clipper 5.3, `FWrite()` Clipper 5.3

`FErase()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Erase a file from disk

Syntax

```
FErase( <cFile> ) → nSuccess
```

Arguments

`cFile` Name of file to erase.

Returns

`nSuccess` 0 if successful, -1 if not

Description

This function deletes the file specified in `cFile` from the disk. No extensions are assumed. The drive and path my be included in `cFile`; neither the `SET DEFAULT` not the `SET PATH` command controls the performance of this function. If the drive or path is not used, the function will look for the file only on the currently selected directory on the logged drive.

If the function is able to successfully delete the file from the disk, the value of the function will be 0; otherwise a -1 will be returned. If not successful, additional information may be obtained by calling the `FError()` function.

Note: Any file to be removed by `FErase()` must still be closed.

Examples

```
#include "fileio.ch"
IF FErase( "test.txt" ) != F_ERROR
   ? "File successfully erased"
ELSE
   ? "File cannot be deleted"
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`FError()` Clipper 5.3, `FRename()` Clipper 5.3

`FError()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Reports the error status of low-level file functions
Syntax

```
FError() → nErrorCode
```

Returns
`nErrorCode` Value of the OS error last encountered by a low-level file function.
`FError()` Return Values

Error	Meaning
0	Successful
2	File not found
3	Path not found
4	Too many files open
5	Access denied
6	Invalid handle
8	Insufficient memory
15	Invalid drive specified
19	Attempted to write to a write-protected disk
21	Drive not ready
23	Data CRC error
29	Write fault
30	Read fault
32	Sharing violation
33	Lock Violation

Description

After every low-level file function, this function will return a value that provides additional information on the status of the last low-level file function's performance. If the `FError()` function returns a 0, no error was detected. Below is a table of possibles values returned by the `FError()` function.
Examples
```
IF ( nHandle := FCreate( "test.txt" ) ) == F_ERROR
   ? "Cannot create file, OS error", FError()
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`FClose()` Clipper 5.3, `FErase()` Clipper 5.3, `FOpen()` Clipper 5.3, `FWrite()` Clipper 5.3

`File()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Tests for the existence of file(s)
Syntax
```
File( <cFileSpec> ) → lExists
```
Arguments
`cFileSpec` File name skeleton or file name to find.
Returns
`lExists` a logical true (`.T.`) if the file exists or logical false (`.F.`).
Description

This function return a logical true (`.T.`) if the given file name `cFileSpec` exist.

File name skeletons symbols may be used in the file name in `cFileSpec`, as may the drive and/or path name. If a path is not explicitly specified, `File()` will look for the file in the `SET DEFAULT` path, then in each `SET PATH path`, until the file is found or there are no more paths to search. The `PATH` environment variable is never searched and the current drive/directory is only searched if `SET DEFAULT` is blank.

Examples
```
? File( hb_DirSepToOS( "/hb/doc/pp.txt" ) )
? File( "*.txt" )
```
Status
S (wild card support is missing)
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags

API, FileSys
See also
Set() Clipper 5.3, SET DEFAULT Clipper 5.3, SET PATH Clipper 5.3

FOpen() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Open a file.
Syntax
```
FOpen( <cFile>, [<nMode>] ) → nHandle
```
Arguments
`cFile` Name of file to open.
`nMode` File open mode.
Returns
`nHandle` A file handle.
Description

This function opens a file expressed as `cFile` and returns a file handle to be used with other low-level file functions. The value of `nMode` represents the status of the file to be opened; the default value is 0. The file open modes are as follows:

nMode	Meaning
FO_READ	Read only
FO_WRITE	Write only
FO_READWRITE	Read/write
FO_EXCLUSIVE	Exclusive read only
FO_DENYWRITE	Prevent others from writing
FO_DENYREAD	Deny read only
FO_DENYNONE	Not deny, Let to others Read / Write
FO_SHARED	same as FO_DENYNONE

If there is an error in opening a file, a `F_ERROR` will be returned by the function. Files handles may be in the range of 0 to 65535. The status of the `SET DEFAULT TO` and `SET PATH TO` commands has no effect on this function. Directory names and paths must be specified along with the file that is to be opened.

If an error has occurred, see the returns values from `FError()` for possible reasons for the error.

Examples
```
#include "fileio.ch"
IF ( nH := FOpen( "test.txt", FO_READWRITE + FO_DENYNONE ) ) == F_ERROR
   ? "File cannot be opened"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Files
Library is core
Header is `fileio.ch`

Tags
API, FileSys
See also
FClose() Clipper 5.3, FCreate() Clipper 5.3, FError() Clipper 5.3

FRead() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Reads a specified number of bytes from a file.
Syntax
```
FRead( <nHandle>, @<cBuffer>, <nBytes> ) → nBytes
```
Arguments
`nHandle` File handle
`cBuffer` Character expression passed by reference.
`nBytes` Number of bytes to read.
Returns
`nBytes` the number of bytes successfully read from the file. `nHandle`
Description

This function reads the characters from a file whose file handle is `nHandle` into a character memory variable expressed as `cBuffer`. The function returns the number of bytes successfully read into `cBuffer`.

The value of `nHandle` is obtained from either a call to the FOpen() or the FCreate() function.

The `cBuffer` expression is passed by reference and must be defined before this function is called. It also must be at least the same length as `nBytes`.

`nBytes` is the number of bytes to read, starting at the current file pointer position. If this function is successful in reading the characters from the file, the length of `cBuffer` or the number of bytes specified in `nBytes` will be the value returned. The current file pointer advances the number of bytes read with each successive read. The return value is the number of bytes successfully read from the file. If a 0 is returned, or if the number of bytes read matches neither the length of `cBuffer` nor the specified value in `nBytes` an end-of-file condition has been reached.

Examples
```
#include "fileio.ch"
cBuffer := Space( 500 )
IF ( nH := FOpen( "test.txt" ) ) == F_ERROR
   FRead( nH, @cBuffer, 500 )
   ? cbuffer
ENDIF
FClose( nH )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, FileSys
See also
Bin2I() Clipper 5.3, Bin2L() Clipper 5.3, Bin2W() Clipper 5.3, FError() Clipper 5.3, FWrite() Clipper 5.3

FReadStr() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Reads a string from a file.
Syntax
```
FReadStr( <nHandle>, <nBytes> ) → cString
```
Arguments
nHandle File handle number.
nBytes Number of bytes to read.
Returns
cString an character expression
Description

This function returns a character string of nBytes bytes from a file whose file handle is nHandle.

The value of the file handle nHandle is obtained from either the FOpen() or FCreate() functions.

The value of nBytes is the number of bytes to read from the file. The returned string will be the number of characters specified in nBytes or the number of bytes read before a zero byte is found.

Examples
```
#include "fileio.ch"
LOCAL cStr
IF ( nH := FOpen( "test.txt" ) ) != F_ERROR
   cStr := FReadStr( nH, 100 )
   ? cStr
   FClose( nH )
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
Bin2I() Clipper 5.3, Bin2L() Clipper 5.3, Bin2W() Clipper 5.3, FError() Clipper 5.3, FRead() Clipper 5.3, FSeek() Clipper 5.3

FRename() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Renames a file
Syntax
`FRename( <cOldFile>, <cNewFile> ) → nSuccess`
Arguments
`cOldFile` Old file name to be changed
`cNewFile` New file name
Returns
`nSuccess` If successful, a 0 will be returned otherwise, a -1 will be returned.
Description

This function renames the specified file `cOldFile` to `cNewFile`. A file name and/or directory name may be specified for either para- meter. However, if a path is supplied as part of `cNewFile` and this path is different from either the path specified in `cOldFile` or (if none is used) the current drive and directory, the function will not execute successfully.

Neither parameter is subject to the control of the `SET PATH TO` or `SET DEFAULT TO` commands. In attempting to locate the file to be renamed, this function will search the default drive and directory or the drive and path specified in `cOldFile`. It will not search directories named by the `SET PATH TO` and `SET DEFAULT TO` commands or by the `PATH` environment variable.

If the file specified in `cNewFile` exists or the file is open, the function will be unable to rename the file. If the function is unable to complete its operation, it will return a value of -1. If it is able to rename the file, the return value for the function will be 0. A call to `FError()` function will give additional information about any error found.

Examples
```
#include "fileio.ch"
nResult := FRename( "test.txt", "test1.txt" )
IF nResult != 0
   ? "File could not be renamed"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`ERASE`Clipper 5.3, `FErase()` Clipper 5.3, `FError()` Clipper 5.3, `File()` Clipper 5.3, `RENAME`Clipper 5.3

`FSeek()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Positions the file pointer in a file.
Syntax
`FSeek( <nHandle>, <nOffset>, [<nOrigin>] ) → nPosition`
Arguments

`nHandle` File handle.
`nOffset` The number of bytes to move.
`nOrigin` The relative position in the file.
Returns
`nPosition` the current position relative to begin-of-file
Description

This function sets the file pointer in the file whose file handle is `nHandle` and moves the file pointer by `expN2` bytes from the file position designated by `nOrigin`. The returned value is the relative position of the file pointer to the beginning-of-file marker once the operation has been completed.

`nHandle` is the file handle number. It is obtained from the `FOpen()` or `FCreate()` function.

The value of `nOffSet` is the number of bytes to move the file pointer from the position determined by `nOrigin`. The value of `nOffset` may be a negative number, suggesting backward movement.

The value of `nOrigin` designates the starting point from which the file pointer should he moved, as shown in the following table:

<nOrigin>	File position
FS_SET	Beginning of file
FS_RELATIVE	Current file pointer position
FS_END	End of file

If a value is not provided for `nOrigin`, it defaults to 0 and moves the file pointer from the beginning of the file.

Examples

```
// here is a function that read one text line from an open file

// nH = file handle obtained from FOpen()
// cB = a string buffer passed-by-reference to hold the result
// nMaxLine = maximum number of bytes to read

STATIC FUNCTION FReadLn( nH, cB, nMaxLine )
   LOCAL cLine, nSavePos, nEol, nNumRead
   cLine := Space( nMaxLine )
   cB := ""
   nSavePos := FSeek( nH, 0, FS_RELATIVE )
   nNumRead := FRead( nH, @cLine, nMaxLine )
   IF ( nEol := hb_BAt( hb_eol(), hb_BLeft( cLine, nNumRead ) ) ) == 0
      cB := cLine
   ELSE
      cB := hb_BLeft( cLine, nEol - 1 )
      FSeek( nH, nSavePos + nEol + 1, FS_SET )
   ENDIF
   RETURN nNumRead != 0
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
Files
Library is core
Header is `fileio.ch`
Tags
API, FileSys
See also
`FCreate()` Clipper 5.3, `FError()` Clipper 5.3, `FOpen()` Clipper 5.3, `FRead()` Clipper 5.3, `FReadStr()` Clipper 5.3, `FWrite()` Clipper 5.3

`FWrite()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Writes characters to a file.
Syntax
```
FWrite( <nHandle>, <cBuffer>, [<nBytes>] ) → nBytesWritten
```
Arguments
`nHandle` File handle number.
`cBuffer` Character expression to be written.
`nBytes` The number of bytes to write.
Returns
`nBytesWritten` the number of bytes successfully written.
Description

This function writes the contents of `cBuffer` to the file designated by its file handle `nHandle`. If used, `nBytes` is the number of bytes in `cBuffer` to write.

The returned value is the number of bytes successfully written to the file. If the returned value is 0, an error has occurred (unless this is intended). A successful write occurs when the number returned by `FWrite()` is equal to either `Len( cBuffer )` or `nBytes`.

The value of `cBuffer` is the string or variable to be written to the open file `nHandle`.

The value of `nBytes` is the number of bytes to write out to the file. The disk write begins with the current file position in `nHandle`. If this variable is not used, the entire contents of `cBuffer` is written to the file. To truncate a file, a call of `FWrite( nHandle, "", 0 )` is needed.

Examples
```
nHandle := FCreate( "test.txt" )
FOR x := 1 TO 10
   FWrite( nHandle, hb_ntos( x ) )
NEXT
FClose( nHandle )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys

See also
FClose() Clipper 5.3, FCreate() Clipper 5.3, FError() Clipper 5.3, FOpen() Clipper 5.3, I2Bin() Clipper 5.3, L2Bin() Clipper 5.3

hb_DiskSpace() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the amount of space available on a disk
Syntax
```
hb_DiskSpace( [<cDrive>] [, <nType>] ) → nDiskBytes
```
Arguments
`cDrive` The drive letter you are requesting info on. The default is A:
`nType` The type of space being requested. The default is `HB_DISK_AVAIL`.
Returns
`nDiskBytes` The number of bytes on the requested disk that match the requested type.
Description

By default, this function will return the number of bytes of free space on the current drive that is available to the user requesting the information.

There are 4 types of information available:

`HB_DISK_AVAIL` The amount of space available to the user making the

request. This value could be less than `HB_FS_FREE` if disk quotas are supported by the O/S in use at runtime, and disk quotas are in effect. Otherwise, the value will be equal to that returned for `HB_FS_FREE`.

`HB_DISK_FREE` The actual amount of free disk space on the drive.

`HB_DISK_USED` The number of bytes in use on the disk.

`HB_DISK_TOTAL` The total amount of space allocated for the user if

disk quotas are in effect, otherwise, the actual size of the drive.

If information is requested on a disk that is not available, a runtime error 2018 will be raised.

Examples
```
#include "fileio.ch"

? "You can use:", hb_ntos( hb_DiskSpace() ), "bytes"
? "Out of a total of:", hb_ntos( hb_DiskSpace( , HB_DISK_TOTAL ) )

// See tests/diskspac.prg for another example
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Files
Library is core

Header is fileio.ch
Tags
API, FileSys

hb_FEof() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Check for end-of-file.
Syntax
```
hb_FEof( <nHandle> ) → lIsEof
```
Arguments
nHandle The handle of an open file.
Returns
lIsEof .T. if the file handle is at end-of-file, otherwise .F.
Description

This function checks an open file handle to see if it is at EOF.

If the file handle is missing, not numeric, or not open, then this function returns .T. and sets the value returned by FError() to -1 (F_ERROR) or a C-compiler dependent errno value (EBADF or EINVAL).

Examples
```
LOCAL nH := FOpen( "test.txt" )
? FReadStr( nH, 80 )
IF hb_FEof( nH )
   ? "End-of-file reached"
ELSE
   ? FReadStr( nH, 80 )
ENDIF
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
FError() Clipper 5.3

hb_FLock() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locks part or all of any file
Syntax
```
hb_FLock( <nHandle>, <nOffset>, <nBytes> [, <nType ] )
          → lSuccess
```
Arguments
nHandle OS file handle
set> Offset of the first byte of the region to be locked.
nBytes Number of bytes to be locked.

e> The type (read or write) of lock requested.
Returns
`lSuccess .T.` if the lock was obtained, else `.F.`
Description

This function attempts to lock a region of the file whose file handle is `nHandle`. This is a low-level file function. To lock Harbour data files use either the `FLock()` or `RLock()` function.

The value of `nHandle` is obtained from either a call to the `FOpen()` or the `FCreate()` function.

`nOffset` is the offset (from the beginning of the file) to the first of the region to be locked. (Offsets from the current position nd of file are not currently supported.)

`nBytes` is the length of the region to be locked in bytes.

`nType` is the type of lock requested. There are two types of locks: exclusive write locks (`nType` = 0x0000) - the default, and shared read locks (`nType` = 0x0100). Additionally you can specify a blocking version of this function (that is it won't return until either an error has occurred or the lock has been obtained) by adding 0x0200 to the above values.

Examples
```
// refer to tests/tflock.prg
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`FClose()` Clipper 5.3, `FCreate()` Clipper 5.3, `FError()` Clipper 5.3, `FOpen()` Clipper 5.3, `hb_FUnlock()`

`hb_FUnlock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Unlocks part or all of any file
Syntax
```
hb_FUnlock( <nHandle>, <nOffset>, <nBytes> ) → lSuccess
```
Arguments
`nHandle` OS file handle
set> Offset of the first byte of the region to be locked.
`nBytes` Number of bytes to be locked.
Returns
`lSuccess .T.` if the lock was removed, else `.F.`
Description

This function attempts to unlock a region of the file whose file handle is `nHandle`. This is a low-level file function. To unlock Harbour data files use the `dbUnlock()` function.

The value of `nHandle` is obtained from either a call to the `FOpen()` or the `FCreate()` function.

`nOffset` is the offset (from the beginning of the file) to the first of the region to be unlocked. (Offsets from the current position nd of file are not currently supported.)

`nBytes` is the length of the region to be unlocked in bytes.

Examples
```
// refer to tests/tflock.prg
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
`FClose()` Clipper 5.3, `FCreate()` Clipper 5.3, `FError()` Clipper 5.3, `FOpen()` Clipper 5.3, `hb_FLock()`

`IsDisk()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Verify if a drive is ready
Syntax
```
IsDisk( <cDrive> ) → lSuccess
```
Arguments
`cDrive` An valid drive letter
Returns
`lSuccess` `.T.` is the drive is ready, otherwise `.F.`
Description
This function attempts to access a drive. If the access to the drive was successful, it will return true (`.T.`), otherwise false (`.F.`). This function is useful for backup function, so you can determine if the drive that will receive the backup data is ready or not.
Examples
```
IF IsDisk( "A" )
   ? "Drive is ready"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.3 function available in builds with `HB_COMPAT_C53` option enabled (default)
Platforms
Available on all platforms
File
Library is core

Tags
API, FileSys
See also
DirChange() Clipper 5.3Clipper Tools 3, DirRemove() Clipper 5.3Clipper Tools 3, DiskChange() Clipper Tools 3, DiskName() Clipper Tools 3, MakeDir()

MakeDir() → hb_DirCreate() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a new directory
Syntax
```
MakeDir( <cDirectory> ) → nError
```
Arguments
cDirectory The name of the directory you want to create.
Returns
nError 0 if directory was successfully created, otherwise the number of the last error.
Description
This function attempt to create a new directory with the name contained in cDirectory. If this function fails, it will return the last OS error code number. See FError() function for the description of the error
Examples
```
LOCAL cDir
IF MakeDir( cDir := hb_DirSepToOS( "./mydir" ) ) == 0
   ? "Directory", cDir, "successfully created"
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.3 function available in builds with HB_COMPAT_C53 option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, FileSys
See also
DirChange() Clipper 5.3Clipper Tools 3, DirRemove() Clipper 5.3Clipper Tools 3, DiskChange() Clipper Tools 3, DiskName() Clipper Tools 3, FError() Clipper 5.3, IsDisk()

hb_gcAll() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Scans the memory and releases all garbage memory blocks.
Syntax
```
hb_gcAll()
```
Arguments
None
Description
This function releases all memory blocks that are considered as the garbage.
Status
Ready
Compliance
Harbour specific

Platforms
Available on all platforms
Tags
API, Garbage Collector
See also
`hb_gcCollectAll()`

`hb_gcAlloc()` in Repository in Discussions ^ | ☰ | Improve this doc

Allocates memory that will be collected by the garbage collector.
Syntax

```
#include "hbapi.h"
void * hb_gcAlloc( HB_SIZE nSize,
                   PHB_GARBAGE_FUNC pCleanupFunc );
```

Arguments
`ulSize` Requested size of memory block
`pCleanupFunc` Pointer to `HB_GARBAGE_FUNC` function that will be called directly before releasing the garbage memory block or `NULL`. This function should release all other memory allocated and stored inside the memory block. For example, it releases all items stored inside the array. The functions receives a single parameter: the pointer to memory allocated by `hb_gcAlloc()`.
Returns
The pointer to allocated memory or it generates an internal unrecoverable error.
Description
`hb_gcAlloc()` is used to allocate the memory that will be tracked by the garbage collector. It allows to properly release memory in case of self-referencing or cross-referencing Harbour level variables. Memory allocated with this function should be released with `hb_gcFree()` function or it will be automatically deallocated by the GC if it is not locked or if it is not referenced by some Harbour level variable.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Garbage Collector
See also
`hb_gcFree()`

`hb_gcCollectAll()` in Repository in Discussions ^ | ☰ | Improve this doc

Scans all memory blocks and releases the garbage memory.
Syntax

```
void hb_gcCollectAll( void );
```

Arguments
None.
Returns
Nothing.
Description

This function scans the eval stack, the memvars table, the array of static variables and table of created classes for referenced memory blocks. After scanning all unused memory blocks and blocks that are not locked are released.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Garbage Collector
See also
`hb_gcAlloc()`, `hb_gcFree()`

`hb_gcFree()` in Repository in Discussions ^ | ☰ | Improve this doc

Releases the memory that was allocated with hb_gcAlloc().
Syntax
```
void hb_gcFree( void * pMemoryPtr );
```
Arguments
`pMemoryPtr` The pointer to memory for release. This memory pointer have to be allocated with `hb_gcAlloc()` function.
Returns
Nothing.
Description
`hb_gcFree()` is used to deallocate the memory that was allocated with the `hb_gcAlloc()` function.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Garbage Collector
See also
`hb_gcAlloc()`

`hb_gcItemRef()` in Repository in Discussions ^ | ☰ | Improve this doc

Marks the memory to prevent deallocation by the garbage collector.
Syntax
```
void hb_gcItemRef( PHB_ITEM pItem );
```
Arguments
`pItem` The pointer to item structure that will be scanned. The passed item can be of any datatype although arrays, objects and codeblocks are scanned only. Other datatypes don't require locking so they are simply ignored.
Returns
Nothing.
Description

The garbage collector uses `hb_gcItemRef()` function during scanning of referenced memory pointers. This function checks the type of passed item and scans recursively all other memory blocks referenced by this item if it is an array, an object or a codeblock.

NOTE: This function is reserved for the garbage collector only. It

cannot be called from the user code - calling it can cause unpredicted results (memory blocks referenced by the passed item can be released prematurely during the closest garbage collection).

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Garbage Collector
See also
`hb_gcAlloc()`, `hb_gcFree()`

`hb_Hash()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a hash table
Syntax

```
hb_Hash( [ <Key1>, <Value1> ], [ <KeyN>, <ValueN> ], ... ) -> hTable
```

Arguments
`Key1` entry key; can be of type: number, date, datetime, string, pointer
`Value1` entry value; can be of type: block, string, numeric, date/datetime, logical, nil, pointer, array, hash table
Equivalent to:

```
hTable := { => }
hTable := { <Key1> => <Value1>, <Key2> => <Value2>, <KeyN> => <ValueN> }
```

Returns
A hash table built from the initial key/value pairs
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HAllocate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Preallocates a hash table
Syntax

```
hb_HAllocate( <hTable>, <nItems> )
```

Arguments
`hTable` a hash table

`nItems` number of items to preallocate in the hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HAutoAdd()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sets the 'auto add' flag for the hash table
Syntax
```
hb_HAutoAdd( <hTable>, [<lFlag>] ) -> <lPreviousFlag>
```
Arguments
`hTable` a hash table
`lFlag` a logical value indicating to turn on or off the 'auto add' flag of the hash table
Returns
The previous value of the 'auto add' flag
Description
This function is equivalent to `hb_HAutoAdd()` but it returns the old flag value rather than the hash table
Examples
```
LOCAL hsTable, lFlag
hsTable := { "one" => 1, "two" => 2 }
// turn 'auto add' on for a new hash table, storing old flag
lFlag := hb_HAutoAdd( hsTable, .T. )
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HBinary()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sets the 'binary' flag for the hash table
Syntax
```
hb_HBinary( <hTable>, [<lFlag>] ) -> <lPreviousFlag>
```
Arguments
`hTable` a hash table
`lFlag` a logical value indicating to turn on or off the 'binary' flag of the hash table
Returns
The previous value of the 'binary' flag
Description
This function is equivalent to `hb_HBinary()` but it returns the old flag value rather than the hash table
Examples
```
LOCAL hsTable, lFlag
```

```
hsTable := { "one" => 1, "two" => 2 }
// turn 'binary' on for a new hash table, storing old flag
lFlag := hb_HBinary( hsTable, .T. )
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HCaseMatch() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sets the 'case match' flag for the hash table
Syntax
```
hb_HCaseMatch( <hTable>, [<lFlag>] ) -> <lPreviousFlag>
```
Arguments
`hTable` a hash table
`lFlag` a logical value indicating to turn on or off the 'case match' flag of the hash table
Returns
The previous value of the 'case match' flag
Description
This function returns the old flag value
Examples
```
LOCAL hsTable, lFlag
hsTable := { "one" => 1, "two" => 2 }
// turn 'case match' on for a new hash table, storing old flag
lFlag := hb_HCaseMatch( hsTable, .T. )
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HClone() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Creates a copy of a hash table
Syntax
```
hb_HClone( <hTable> ) -> <hsDestination>
```
Arguments
`hTable` a hash table
Returns
A cloned copy of the hash table
Status
Ready
Compliance
Harbour specific
Platforms

Available on all platforms
Tags
API, Hash table

hb_HCopy() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Adds entries from the source hash table to the destination hash table
Syntax
```
hb_HCopy( <hsDestination>, <hsSource>, [<nStart>], [<nCount>] ) ->
<hsDestination>
```
Arguments
`hsDestination` a destination hash table
`hsSource` a source hash table
`nStart` starting index, defaults to 1 if omitted
`nCount` counter, defaults to (length) - `nStart` is omitted
Returns
The destination hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HDefault() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns/sets a default value for a hash table.
Syntax
```
hb_HDefault( <hTable>, <DefaultValue> ) -> <OldDefaultValue>
```
Arguments
`hTable` a hash table
`DefaultValue`
Returns
The previous default value assigned to the hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HDel() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Removes a key/value pair from a hash table
Syntax
```
hb_HDel( <hTable>, <Key> ) -> <hTable>
```
Arguments
`hTable` a hash table

`Key` key to be removed from the hash table; can be of type: number, date, datetime, string, pointer
Returns
The hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HDelAt()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Removes an entry from a hash table based on its index position
Syntax
```
hb_HDelAt( <hTable>, <nPosition> ) -> <hTable>
```
Arguments
`hTable` a hash table
`nPosition` the position of an entry within the hash table that will be deleted
Returns
The hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HEval()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Evaluate a code block across the contents of a hash table
Syntax
```
hb_HEval( <hTable>, <bBlock>, [<nStart>], [<nCount>] ) -> <hTable>
```
Arguments
`hTable` a hash table
`bBlock` code block to be evaluated
`nStart` starting index, defaults to 1 if omitted
`nCount` counter, defaults to (length) - `nStart` is omitted
Returns
The hash table
Description

The code block is evaluated for every hash table entry starting at `nStart` for `nCount` items.

The code block is passed the entry key, value, and numeric position

Status

Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HFill() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Fills a hash table with a value
Syntax
```
hb_HFill( <hTable>, <Value> ) -> <hTable>
```
Arguments
`hTable` a hash table
`Value` fill value; can be of type: block, string, numeric, date/datetime, logical, nil, pointer, array, hash table
Returns
The hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HGet() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a hash value
Syntax
```
hb_HGet( <hTable>, <Key> ) -> <Value>
```
Arguments
`hTable` a hash table
`Key` key to be retrieve from the hash table; can be of type: number, date, datetime, string, pointer
Returns
Either the value within the hash table for the given key.
An array access error occurs of the key is not found
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HGetDef() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a hash value, or a default value if the key is not present
Syntax
`hb_HGetDef( <hTable>, <Key>, [<DefaultValue>] ) -> <Value>`
Arguments
`hTable` a hash table
`Key` key to be retrieve from the hash table; can be of type: number, date, datetime, string, pointer
`DefaultValue` a default value to be returned if the hash table does not contain the key
Returns
Either the value within the hash table for the given key, or the default value.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HHasKey()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determines whether a hash table has an entry with a give key
Syntax
`hb_HHasKey( <hTable>, <Key> ) -> lExists`
Arguments
`hTable` a hash table
`Key` a key value to be queried for; can be of type: number, date, datetime, string, pointer
Returns
A logical value indicating whether the key exists within the hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HKeyAt()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Gets a hash table key at a given position
Syntax
`hb_HKeyAt( <hTable>, <nPosition> ) -> <Key>`
Arguments
`hTable` a hash table
`nPosition` the position of an entry within the hash table that will be returned
Returns
The key at the given position of the hash table; the type will be one: number, date, datetime, string, pointer
Status
Ready

Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HKeys() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns an array of the keys of a hash table
Syntax
```
hb_HKeys( <hTable> ) -> <aKeys>
```
Arguments
`hTable` a hash table
Returns
An array of all the hash table keys
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HMerge() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Merges a source hash table into a destination hash table
Syntax
```
hb_HMerge( <hsDestination>, <hsSource>, <bBlock>|<nPosition> ) ->
<hsDestination>
```
Arguments
`hsDestination` a destination hash table
`hsSource` a source hash table
`bBlock` a code block that will be evaluated for each entry within the source hash table; the code block will be passed the entry key, value and position; if the code block returns a true value, the entry will be added to the destination hash table
`nPosition` the position of an entry within the source hash table that will be appended to the destination hash table
TODO: the source code passes either a number or `HB_HASH_UNION`; research this
Returns
The destination hash table with the contents of the source hash table merged
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HPairAt() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a two-dimensional array of a hash table entry key/value pair
Syntax
`hb_HPairAt( <hTable>, <nPosition> ) -> <aKeyValue>`
Arguments
`hTable` a hash table
`nPosition` the position of an entry within the hash table that will be returned
Returns
A two-dimensional array of the key/value pair entry of the hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HPos() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locates the index of a key within a hash table
Syntax
`hb_HPos( <hTable>, <Key> ) -> nPosition`
Arguments
`hTable` a hash table
`Key` key for which its position is to be determined; can be of type: number, date, datetime, string, pointer
Returns
A integer number being the index position of the key within the hash table.
TODO: what is the return value if the key does not exist? zero (0)? RTE?
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HScan() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Scans a hash table
Syntax
`hb_HScan( <hTable>, <Value>, [<nStart>], [<nCount>, [<lExact>] ) -> nPosition`
Arguments
`hTable` a hash table
`Value` to be located within the hash table
`nStart` starting index, defaults to 1 if omitted
`nCount` counter, defaults to (length) - `nStart` is omitted

`lExact` logical value indicating whether the comparison is to be be exact or not
Returns
The position of the located value within the hash table, or zero (0) if not found.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HSet()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sets a hash value
Syntax
```
hb_HSet( <hTable>, <Key>, <Value> ) -> <hTable>
```
Arguments
`hTable` a hash table
`Key` the key of the entry to be set; can be of type: number, date, datetime, string, pointer
`Value` the entry value
Returns
The hash table
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

`hb_HSort()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Reorganizes the internal list of the hash table to be sorted
Syntax
```
hb_HSort( <hTable> ) -> <hsSortedTable>
```
Arguments
`hTable` a hash table
Returns
The hash table sorted
TODO: is the original table altered?
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HValueAt() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Gets/sets a hash value at a given position
Syntax
```
hb_HValueAt( <hTable>, <nPosition>, [<NewValue>] ) -> <Value>
```
Arguments
`hTable` a hash table
`nPosition` the position of an entry within the hash table that will be returned
`NewValue` a new value to be assigned to the hash table at the given position
Returns
The existing value, or the new value if it is given
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_HValues() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns an array of the values of a hash table
Syntax
```
hb_HValues( <hTable> ) -> <aValues>
```
Arguments
`hTable` a hash table
Returns
An array of all the hash values
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Hash table

hb_idleAdd() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Adds the background task.
Syntax
```
hb_idleAdd( <bAction> ) → nHandle
```
Arguments
`bAction` is a codeblock that will be executed during idle states. There are no arguments passed to this codeblock during evaluation.
Returns
`nHandle` The handle (an integer value) that identifies the task. This handle can be used for deleting the task.
Description

`hb_idleAdd()` adds a passed codeblock to the list of background tasks that will be evaluated during the idle states. There is no limit for the number of tasks.
Examples
```
nTask := hb_idleAdd( {|| SayTime() } )
```
Status
Ready
Compliance
Harbour extension similar to `ft_OnTick()` function available in NanForum library.
Platforms
Available on all platforms
Tags
API, Idle states
See also
`hb_idleDel()`, `hb_idleState()`

`hb_idleDel()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Removes the background task from the list of tasks.
Syntax
```
hb_idleDel( <nHandle> ) → bAction
```
Arguments
`nHandle` is the identifier of the task returned by the `hb_idleAdd()` function.
Returns
`bAction` `NIL` if invalid handle is passed or a codeblock that was passed to `hb_idleAdd()` function
Description

`hb_idleDel()` removes the action associated with passed identifier from the list of background tasks. The identifier should be the value returned by the previous call of `hb_idleAdd()` function.

If specified task is defined then the codeblock is returned otherwise the `NIL` value is returned.

Examples
```
nTask := hb_idleAdd( {|| SayTime() } )
Inkey( 10 )
bAction := hb_idleDel( nTask )
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Idle states
See also
`hb_idleAdd()`, `hb_idleState()`

`hb_idleState()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Evaluates a single background task and calls the garbage collector.
Syntax

`hb_idleState()`
Arguments
None
Description

`hb_idleState()` requests the garbage collection and executes a single background task defined by the codeblock passed with `hb_idleAdd()` function. Every call to this function evaluates a different task in the order of task creation. There are no arguments passed during a codeblock evaluation.

This function can be safely called even if there are no background tasks defined.

Examples

```
nTask1 := hb_idleAdd( {|| SayTime() } )
nTask2 := hb_idleAdd( {|| SaveScreen() } )
DO WHILE ! bFinished
   bFinished := DoSomethingVeryImportant()
   hb_idleState()
ENDDO
cbAction := hb_idleDel( nTask1 )
hb_idleDel( nTask2 )
```

Status
Ready
Compliance
Harbour extension similar to `ft_IAmIdle()` function available in NanForum library.
Platforms
Available on all platforms
Tags
API, Idle states
See also
`hb_idleAdd()`, `hb_idleDel()`

`__mvDbgInfo()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function returns the information about the variables for debugger
Syntax

```
__mvDbgInfo( <nScope> [, <nPosition> [, @<cVarName>] ] )
```

Arguments
`nScope` = the scope of variables for which an information is asked
Supported values (defined in `hbmemvar.ch`)
HB_MV_PUBLIC
HB_MV_PRIVATE (or any other value)
`nPosition` = the position of asked variable on the list of variables with specified scope - it should start from position 1 `cVarName` = the value is filled with a variable name if passed by reference and `nPosition` is specified
Returns
The return value depends on the number of arguments passed
Description

This function retrieves the information about memvar variables. It returns either the number of variables with given scope (when the first argument is passed only) or a value of variable identified by its position in the variables' list (when second argument is passed). It also returns the name of a variable if optional third argument is passed by reference.

If requested variable doesn't exist (requested position is greater then the number of defined variables) then `NIL` value is returned and variable name is set to `?`.

The dynamic symbols table is used to find a `PUBLIC` variable then the `PUBLIC` variables are always sorted alphabetically. The `PRIVATE` variables are sorted in the creation order.

Note:

Due to dynamic nature of memvar variables there is no guarantee that successive calls to retrieve the value of Nth `PUBLIC` variable will return the value of the same variable.

Examples

```
#include "hbmemvar.ch"

MEMVAR cPublic
MEMVAR cPrivate
MEMVAR ccPublic
MEMVAR ccPrivate

PROCEDURE Main()

   LOCAL nCount, tmp, xValue, cName

   nCount := __mvDbgInfo( HB_MV_PUBLIC )
   FOR tmp := 1 TO nCount
      xValue := __mvDbgInfo( HB_MV_PUBLIC, tmp, @cName )
      ? tmp, cName, xValue
   NEXT

   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )

   PUBLIC cPublic := "cPublic in MAIN"

   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )

   PRIVATE cPrivate := "cPrivate in MAIN"

   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )

   CountMemvars()

   ? "Back in Main"
   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )

   RETURN

STATIC PROCEDURE CountMemvars()

   LOCAL nCount, tmp, xValue, cName

   PUBLIC ccPublic := "ccPublic"
   PRIVATE ccPrivate := "ccPrivate"

   ? "In CountMemvars"
   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )
```

```
   PRIVATE cPublic := "cPublic"

   ? "PUBLIC=", __mvDbgInfo( HB_MV_PUBLIC )
   ? "PRIVATE=", __mvDbgInfo( HB_MV_PRIVATE )

   nCount := __mvDbgInfo( HB_MV_PRIVATE ) + 1
   FOR tmp := 1 TO nCount
      xValue := __mvDbgInfo( HB_MV_PRIVATE, tmp, @cName )
      ? tmp, "=", cName, xValue
   NEXT

   nCount := __mvDbgInfo( HB_MV_PUBLIC ) + 1
   FOR tmp := 1 TO nCount
      xValue := __mvDbgInfo( HB_MV_PUBLIC, tmp, @cName )
      ? tmp, "=", cName, xValue
   NEXT

   RETURN
```

Status
Ready
Compliance
This function should be called from the debugger only.
Platforms
Available on all platforms
File
Library is core
Tags
API, Internal

__SetHelpK() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set F1 as the default help key
Syntax
`__SetHelpK()`
Arguments
None.
Description
Set F1 to execute a function called `HELP` if such a function is linked into the program.
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Internal
See also
`__XHelp()`, `SetKey()` Clipper 5.3, `SET KEY` Clipper 5.3

__TextRestore() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Restore console output settings as saved by __TextSave()
Syntax
`__TextRestore()`
Arguments
none.
Description
`__TextRestore()` is used in the preprocessing of the `TEXT TO` command to restore console output settings that were previously saved by `__TextSave()`.
Status
Ready
Compliance
Undocumented CA-Cl*pper v5.x function available in builds with `HB_CLP_UNDOC` option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, Internal
See also
`__TextSave()`, `Set()` Clipper 5.3, `SET ALTERNATE` Clipper 5.3, `SET PRINTER` Clipper 5.3, `TEXT`

`__TextSave()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Redirect console output to printer or file and save old settings
Syntax
`__TextSave( <cFile> )`
Arguments
`cFile` is either "PRINTER" (note the uppercase) in which console output is `SET` to `PRINTER`, or a name of a text file with a default `.txt` extension, that is used to redirect console output.
Description
`__TextSave()` is used in the preprocessing of the `TEXT TO` command to redirect the console output while saving old settings that can be restored later by `__TextRestore()`.
Status
Ready
Compliance
Undocumented CA-Cl*pper v5.x function available in builds with `HB_CLP_UNDOC` option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, Internal
See also
`__TextRestore()`, `Set()` Clipper 5.3, `SET ALTERNATE` Clipper 5.3, `SET PRINTER` Clipper 5.3, `TEXT`

`__XHelp()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determines whether a HELP() user defined function exists.
Syntax
`__XHelp() → xValue`
Arguments
None
Returns
This function returns aleatory values.
Description
This is an internal undocumented CA-Cl*pper function, which will try to call the user defined function `HELP()` if it is defined in the current application. This is the default `SetKey()` handler for the F1 key.
Status
Ready
Compliance
Undocumented CA-Cl*pper v5.x function available in builds with `HB_CLP_UNDOC` option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, Internal

`CLIPINIT()` in Repository in Discussions ^ | ☰ | Improve this doc

Initialize various Harbour sub-systems
Syntax
`CLIPINIT() → NIL`
Arguments
none.
Returns
`CLIPINIT()` always return `NIL`.
Description
`CLIPINIT()` is one of the pre-defined `INIT PROCEDURE` and is executed at program startup. It declare an empty `MEMVAR PUBLIC` array called GetList that is going to be used by the Get system. It activates the default error handler, and (at least for the moment) calls the function that sets the default help key.
Status
Ready
Compliance
It is said that `CLIPINIT()` should not call the function that sets the default help key since CA-Cl*pper does it in some other place.
Platforms
Available on all platforms
Tags
API, Internal
See also
`INIT PROCEDURE`

`hb_inetAccept()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Wait until a socket is ready
Syntax
`hb_inetAccept( <socket> ) → <SOCKET>`
Arguments
An INET socket
Returns
`socket` a socket previously created / opened
Description

Waits until a connection is available on a socket created with `hb_inetServer()`, returns a socket that can be used to communicate with the incoming client.

On error, `NIL` is returned and error code sets in the passed `SOCKET`.

This error can be accessed using `hb_inetErrorCode()` function.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetAddress()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get a remote server address
Syntax
`hb_inetAddress( <socket> ) → cResult`
Arguments
`socket` a socket previously created / opened
Returns
Server address
Description

Returns a string representing the remote server address in quad dot notation, e.g. "127.0.0.1", or the local server address if the socket is server side.

TODO: have a version that returns a vector of 4 numbers.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetCleanup()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Terminate Harbour INET support
Syntax
`hb_inetCleanup()`
Arguments
(This function has no arguments)
Description
Closes inet support; mainly used for Windows. Put it at the end of any program using Inet functions, just before the program exits.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetClearError()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Clear the socket error value
Syntax
`hb_inetClearError( <socket> )`
Arguments
`socket` a socket previously created / opened
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetClearPeriodCallback()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Clear the periodic callback value of a socket
Syntax
`hb_inetClearPeriodCallback( <socket> )`
Arguments
`socket` a socket previously created / opened
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetClearTimeLimit() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Clear the time limit value of a socket
Syntax
```
hb_inetClearTimeLimit( <socket> )
```
Arguments
`socket` a socket previously created / opened
Description
Clears the default time limit of the given socket.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetClearTimeout() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Clear the timeout value of a socket
Syntax
```
hb_inetClearTimeout( <socket> )
```
Arguments
`socket` a socket previously created / opened
Description
Clears the default timeout of the given socket. Default timeout is used in all blocking operations.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetClose() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Close an INET socket
Syntax
```
hb_inetClose( <socket> ) → nResult
```
Arguments
`socket` a socket previously created / opened
Returns
Returns 0 on success or -1 on error; on error, the error code is set; (actually, on success the socket error code is set to 1 — socket closed)
Description

Closes the socket, notifying both ends of the communication pipe that the connection is over.

If you have threads waiting for data to be read from this socket, this method will make them stop waiting and return an error (socket closed) to their callers.

The method does not destroy the socket, which can be used by subordinate threads to check that the socket is closed, and so they should stop as soon as they can. Don't destroy the socket unless you are sure that no other thread is using it.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetConnect() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Connect a socket to a remote server by IP address or name
Syntax

```
hb_inetConnect( <cAddress>, <nPort> ) → <SOCKET>

hb_inetConnect( <cAddress>, <nPort>, <socket> ) → NIL
```

Arguments

```
cAddress
nPort
socket
```

Returns
(First form) INET socket
(Second form has no return value)
Description

Connects to a remote server described by `cAddress`, that can be in quad dot notation (e.g. "127.0.0.1") or in DNS name (e.g. "example.org"), using the desired port.

`hb_inetConnect()` uses "gethostbyname" C system call to find the network address of the specified server; this means that this call is an hybrid function doing both a DNS scan and a `TCP/IP` connection. `hb_inetConnect()` is not thread safe, and the program must take care that two `hb_inetConnect()` functions are never called at the same moment from two different threads (or that `hb_inetGetHosts()` is not called in the same moment as an `hb_inetConnect()`).

The second version of this function accepts a pre-built socket as a parameter. This allows to kill asynchronously a thread waiting for `hb_inetConnect()` to connect, and then cleaning up the leftover socket data. Also, it is possible to give timeout to the given `SOCKET`, but this timeout will be used only in the connection phase, after that the network address resolution is completed. Use `hb_inetGetHosts()` and `hb_inetConnectIP()` for a finer timeout control. On error, the error of the returned socket is set. The error could be due to unavailable name resolving service, host name not valid, host address not reachable and host reachable but port not open.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetConnectIP() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Connect to a remote server by IP address
Syntax

```
hb_inetConnectIP( <cAddress>, <nPort> ) → <SOCKET>

hb_inetConnectIP( <cAddress>, <nPort>, <socket> ) → NIL
```

Arguments

```
cAddress
nPort
socket
```

Returns
(First form) INET socket
(Second form has no return value)
Description

Connects to a remote server described by `cAddress`, that can be specified only in quad dot IPv4 notation (e.g. "127.0.0.1"), using the desired port. This version of `hb_inetConnect()` does not use `gethostbyname()`, and thus is thread safe and can be used in combination with `hb_inetGetHosts()` to have a finer timeout control while connecting to a server, and a finer error control.

The second version of this function accepts a pre-built socket as a parameter. This allows to kill asynchronously a thread waiting for `hb_inetConnectIP()` to connect, and then cleaning up the leftover socket data. Also, it is possible to give timeout at the given `SOCKET`.

On error, the error of the returned socket is set.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetCount() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the number of bytes last read or sent
Syntax

```
hb_inetCount( <socket> ) → nResult
```

Arguments

`socket` a socket previously created / opened
Returns
Last socket operation character count
Description
Returns the amount of characters read or written in the latest socket operation.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetCreate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create an INET socket
Syntax
```
hb_inetCreate( [ <nTimeout> ] ) → <SOCKET>
```
Arguments
`nTimeout` Socket timeout (optional) TODO: what is the scale (seconds, milliseconds?)
Returns
An INET socket
Description
Creates the raw data of the socket, that can be passed to a asynchronous connection function (`hb_inetConnect()` or `hb_inetConnectIP()`). This will prevent the connection function from allocating some data that could be never used in certain cases, i.e. an asynchronously detected timeout.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetCRLF()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get a CRLF sequence for internet protocols
Syntax
```
hb_inetCRLF() → cResult
```
Arguments
(This function has no arguments)
Returns
Internet `CRLF` sequence
Description
Returns a `CRLF` sequence used in many internet protocols.
Status
Ready
Compliance

Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetDataReady() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get whether there is data ready in a socket
Syntax
```
hb_inetDataReady( <socket>, [ <nMillisec> ] ) → nResult
```
Arguments
`socket` a socket previously created / opened
`nMillisec`
Returns
If there is data available 1 (one) is returned, 0 (zero) if there is no data and -1 if there is an error.
Description

Verifies if some data is available to be read in the socket without blocking execution of the caller.

If `nMillisecs` is not given, the function returns immediately 1 if there is some data to be read, 0 if there isn't any data and -1 in case of error.

If `nMillisecs` is given, the function will wait up to that amount of milliseconds for data to be available; if some data arrives in the meanwhile, the wait is immediately interrupted.

The next `hb_inetRecv()` function will read all the available data (up to the required length) without blocking.

On error, `hb_inetErrorCode()` and `hb_inetErrorDesc()` can be use to determine what kind of error happened.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetDGram() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a datagram socket
Syntax
```
hb_inetDGram( [<lBroadcast>] ) → <SOCKET>
```
Arguments
`lBroadcast`
Returns

An INET socket
Description

Creates a datagram-oriented socket that will be able to send data and eventually receive data. Since the socket is not bound, the program cannot retrieve the address at which this socket appears to be, but a second socket receiving a message sent from this one would be able to reply correctly with a datagram that can be read from a non-bound socket.

If `lBroadcast` is set to `.T.`, the routine creates a broadcast capable socket: it will be able to receive and send broadcast messages. On most systems this requires special user privileges.

Returns the socket, and if an error occurs, the socket error message and code are set.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetDGramBind()` Source code □ in Repository in Discussions ^ | ☰ | Improve this doc

Create a bound datagram socket
Syntax

```
hb_inetDGramBind( <nPort>, [<cAddress> [, <lBroadcast>] ] ) → <SOCKET>
```

Arguments

`nPort`
`cAddress`
`bBroadcast`

Returns
An INET socket
Description

Creates a datagram-oriented socket and binds it to a particular port, and eventually to a certain interface if `cAddress` is given and not `NIL`.

If `lBroadcast` is set to `.T.`, the routine creates a broadcast capable socket: it will be able to receive and send broadcast messages. On most systems this requires special user privileges.

Returns the socket

If an error occurs, the socket error message and code are set.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags

API, INET

hb_inetDGramRecv() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get data from a datagram socket
Syntax
`hb_inetDGramRecv( <socket>, @<cBuffer> [, <nSize> ] ) → nBytesRead`
Arguments
`socket` a socket previously created / opened
`cBuffer` is the target buffer and must be passed by reference
`nSize`
Returns
Returns number of bytes read, or -1 on error
Description

Reads at maximum `nSize` bytes incoming from a UDP socket, if `nSize` is given, or reads at maximum `cBuffer` length if `nSize` is not given.

There isn't any guarantee that all the data required to be read is really sent from the kernel to the application: the kernel should just return the last complete datagram that has been received, up to `nSize` bytes.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetDGramSend() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Send data to a datagram socket
Syntax
`hb_inetDGramSend( <socket>, <cAddress>, <nPort>, <cBuffer> [, <nSize> ] ) → nBytesSent`
Arguments
`socket` a socket previously created / opened
`cAddress`
`nPort`
`cBuffer`
`nSize`
Returns
Returns number of bytes sent, or -1 on error
Description

Sends a datagram (a fixed length data) to a determined IP address (`cAddress`, to be specified in quad-dot notation) and port.

If `nSize` is not specified, all the data in `cBuffer` will be sent; if `nSize` is specified, only the first `nSize` bytes of `cBuffer` will be sent.

There isn't any guarantee that all the data required to be written is really sent to the socket: the calling program should check for the numeric return and send iteratively the unsent data to complete the message.

Anyway, the raw datagram is sent and received as once, and any data less than the system datagram size will be sent and received as a single item.

If the socket is created in broadcast mode, the `cAddress` element can be a broadcast address.

Returns -1 on error, or the number of bytes actually sent on success.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetErrorCode()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the last INET error code
Syntax

```
hb_inetErrorCode( <socket> ) → nResult
```

Arguments
`socket` a socket previously created / opened
Returns
Last error code
Description

Returns the last error code that has been provoked by a network operation, or 0 if none.

Error codes are the ones used for winsock or UnixSockets (they are the same); 1 is reserved for "connection closed" message.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetErrorDesc()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the last INET error code description
Syntax

```
hb_inetErrorDesc( <socket> ) → cResult
```

Arguments

`socket` a socket previously created / opened
Returns
System-dependent error string
Description
Returns a string describing the last error that occurred in the socket; the string is system dependent, and should be used only for debugging purposes.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetFD()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

?
Syntax
```
hb_inetFD( <socket> [, <lNoSocket> ] ) → nResult
```
Arguments
`socket` a socket previously created / opened
`lNoSocket`
Returns
?
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetGetAlias()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get an array of aliases of a server
Syntax
```
hb_inetGetAlias( <cName> ) → aHosts
```
Arguments
`cName`
Returns
Array of server aliases
Description

Returns an array containing the aliases (`CNAME DNS` records) by which the server is currently known.

Whether this function is able to have the complete list of aliases or not depends on the verbosity of the DNS server.

Status

Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetGetHosts() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get an array of IP addresses of a host
Syntax
```
hb_inetGetHosts( <cName> ) → aHosts
```
Arguments
`cName`
Returns
An array of IP addresses
Description

Returns an array containing all the IP addresses associated with a given host name. The IP addresses returned by this function are strings in quad dot notations, e.g. "127.0.0.1", and can be directly used into `hb_inetConnectIP()`.

`cName` can be any string: valid DNS names (e.g. "example.org"), locally available names (e.g. "localhost" or windows Network Neighborhood names), or even IP addresses in quad dot notation.

NOTE: This function is not thread safe (by design), and programmers must be sure not to use it at the same time in two different threads, or not to use it together with a `hb_inetConnect()`. If this kind of situation should ever arise, you are advised to use a thread `MUTEX`.

On error, and if the server cannot be found, the function returns `NIL`.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetGetRcvBufSize() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the socket receive buffer size
Syntax
```
hb_inetGetRcvBufSize( <socket> ) → nResult
```
Arguments
`socket` a socket previously created / opened

Returns
Returns the socket receive buffer size or -1 if the socket is closed or an error occurs
Description
Returns the socket receive buffer size or -1 if the socket is closed or an error occurs
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetGetSndBufSize()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the socket send buffer size
Syntax
```
hb_inetGetSndBufSize( <socket> ) → nResult
```
Arguments
`socket` a socket previously created / opened
Returns
Returns the socket send buffer size or -1 if the socket is closed or an error occurs
Description
Returns the socket send buffer size or -1 if the socket is closed or an error occurs
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetInit()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Activate Harbour INET support
Syntax
```
hb_inetInit() → lResult
```
Arguments
(This function has no arguments)
Returns
Returns `.T.` or `.F.` whether the internal INET system was successfully initialized
Description
Activates inet support; mainly used for winsock start up at the moment, but could be used in the future for many other purpose. Put it at the beginning of every program using INET functions.
Status
Ready
Compliance
Harbour specific

Platforms
Available on all platforms
Tags
API, INET

hb_inetIsSocket() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get whether a variable is a socket
Syntax
`hb_inetIsSocket( <socket> ) → lResult`
Arguments
`socket` a socket previously created / opened
Returns
Returns whether the passed parameter is a socket
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetPeriodCallback() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get or change the periodic callback value of a socket
Syntax
`hb_inetPeriodCallback( <socket> [, <xCallback> ] ) → xPreviousCallback`
Arguments
`socket` a socket previously created / opened
`xCallback` a new periodic callback
Returns
The previous periodic callback value
Description

Sets or returns the socket periodic callback value

`xCallback` can be one of: a codeblock, an array of (...), or a (symbol) TODO: describe these better

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetPort() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the port a socket is bound to.
Syntax
`hb_inetPort( <socket> ) → cResult`
Arguments
`socket` a socket previously created / opened
Returns
Port name the socket is bound to.
Description
Returns the port to which this socket is bound, or the remote port if this socket is connected with a remote host or client
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetRecv()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Read from a socket
Syntax
`hb_inetRecv( <socket>, @<cResult>, [ <nAmount> ] ) → nResult`
Arguments
`socket` a socket previously created / opened
`cResult` is the target buffer and must be passed by reference
`nAmount` is the upper limit of characters to be read from the socket. If not passed this defaults to the length of `cResult`
Returns
The number of the characters read from the socket.
Description

Reads from the socket into a buffer.

The parameter `cString` must be preallocated so that it has enough space to receive the data. The routine will block the thread until some bytes are read from the socket, the socket is closed (either from the receiver or the sender side) or a network error occurs, whichever comes first. In the latter cases, an error is set, and only the characters received until premature end of communications are returned.

Notice that there is no guarantee that all the available bytes will be read before the function returns, in fact, `hb_inetRecv()` returns as soon it is able to fill `cString` with one or more bytes. To block the current process until the whole `cString` is filled (or `nAmount` bytes are read), use the `hb_inetRecvAll()`.

Status
Ready
Compliance
Harbour specific
Platforms

Available on all platforms
Tags
API, INET

hb_inetRecvAll() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Read from a socket without blocking
Syntax
```
hb_inetRecvAll( <socket>, @<cResult>, [ <nAmount> ] ) → nResult
```
Arguments
`socket` a socket previously created / opened
`cResult` is the target buffer and must be passed by reference
`nAmount` is the upper limit of characters to be read from the socket. If not passed this defaults to the length of `cResult`
Returns
The number of the characters read from the socket. Might be less than `nAmount` on premature socket closing or on network error.
Description
This function works exactly as `hb_inetRecv()` except that it blocks until `nAmount` bytes are read, if `nAmount` is given, or `cString` is filled for its whole length.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetRecvEndblock() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Read a block from a socket
Syntax
```
hb_inetRecvEndblock( <socket> [, <cBlock >[, @<nBytesRead> [, <nMaxLength>
[, <nBufSize> ]]]] ) → cResult
```
Arguments
`socket` a socket previously created / opened
`cBlock`
`nBytesRead`
`nMaxLength`
`nBufSize`
Returns
Block read
Description
This function operates exactly the same way as `hb_inetRecvLine()`, but the "record termination" is customizable through the `cBlock` parameter. If not given, this parameter defaults to the `CRLF` sequence.
Status
Ready
Compliance
Harbour specific

Platforms
Available on all platforms
Tags
API, INET

hb_inetRecvLine() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Read a line from a socket
Syntax

```
hb_inetRecvLine( <socket> [, @<nBytesRead>, [, <nMaxLength> [, <nBufSize>
]]] ) → cResult
```

Arguments

`socket` a socket previously created / opened
`nBytesRead` must be passed by reference
`nMaxLength`
`nBufSize`
Returns
Line read
Description

Blocks the calling thread until a sequence `CRLF` is read from the socket. Incremental allocation and end-of-line checking are done in an efficient way.

If an error occurs, or if the stream is closed before a `CRLF` is read, the function returns nothing and sets the socket error.

The returned string does not contain the trailing `CRLF` sequence, so an empty line is effectively returned as an empty string.

If the `nBytesRead` parameter is given, it will contain the number of bytes read from the socket, including the `CRLF` sequence, so that in normal conditions, `nResult` will report a count equal to the length of the returned string plus 2. `nBytesRead` will be 0 if stream is closed before a `CRLF` sequence is read, and will be -1 on error.

An optional `nMaxLength` parameter can be given to allow a maximum character count before the data is returned anyway. If this number is reached before a `CRLF` sequence is encountered, `nBytesRead` will contain the value one.

Finally, a `nBufSize` parameter can be given. If not, memory allocation will be increased by discrete amounts of 80 bytes. The programmer can provide here a different allocation strategy (e.g. setting `nBufSize` equal to `nMaxLength`, memory for reading the line will be allocated only once, at the beginning of the function).

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetstatus() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the status of a socket
Syntax
```
hb_inetstatus( <socket> ) → nResult
```
Arguments
`socket` a socket previously created / opened
Returns
Returns 1 (one) if the socket exists, -1 if it does not
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetSend() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Sent data through a socket
Syntax
```
hb_inetSend( <socket>, <cBuffer> [, <nLength> ] ) → nResult
```
Arguments
`socket` a socket previously created / opened
`cBuffer`
`nLength`
Returns
The amount of data written, 0 (zero) if the socket is closed, or -1 on an error
Description

Send data being stored in a string over the socket.

The `nLength` parameter can be given to allow writing only a part of the string.

There is no guarantee that all of `cBuffer` will be sent, as this is a decision that is up to the OS; this function does not take care to ensure that the data is really sent; check the returned number and send the part that has not been sent.

To ensure that all the data is sent before the function returns, use the `hb_inetSendAll()` function.

On error, the error in the socket is set.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags

API, INET

hb_inetSendAll() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Send data through a socket with blocking
Syntax
`hb_inetSendAll( <socket>, <cBuffer> [, <nLength> ] ) → nResult`
Arguments
`socket` a socket previously created / opened
`cBuffer`
`nLength`
Returns
The amount of data written, 0 (zero) if the socket is closed, or -1 on an error
Description
This function works exactly as `hb_inetSend()` but it ensures that all the data to be sent is written before returning.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetServer() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a socket bound to a port
Syntax
`hb_inetServer( <port> [, <cBindAddr> [, <nListenLimit> ]]  ) → <SOCKET>`
Arguments
`port`
`cBindAddr`
`nListenLimit` is an internal parameter and rarely needs to be passed, defaults to 10
Returns
An INET socket
Description

Creates a server that can accept connections from client on a certain port.

If the computer on which `hb_inetServer()` is called has more than one logical interface (e.g. one network card, one loopback and one PPP address), `cBindAddr` can be specified to select only one of these interfaces to accept connections for this process. This is useful when a server is present on two networks, and the service is to be available only in one of them. Also, the same port on other addresses is left free to be used, so you can have different server programs running for different networks but managing the same service. For example, an FTP server available to the internal network could be radically different from an FTP server available for the internet.

`nListenLimit` is the number of incoming connections accepted by kernel before the kernel has the chance to report them to the application program. If the sockets receive `nListenLimit`

connections before accepting them all, the `nListenLimit + 1` connection will be notified to be "busy" by the kernel. The default value of 10 is enough for even a heavy duty server.

On error, sets error description in the newly returned socket.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetSetRcvBufSize()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set the receive buffer size of a socket
Syntax
```
hb_inetSetRcvBufSize( <socket>, <nSize> ) → nSize
```
Arguments
`socket` a socket previously created / opened
`nSize`
Returns
Returns the passed `nSize` or -1 on error
Description
Sets the receive buffer size of a socket
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

`hb_inetSetSndBufSize()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set the send buffer size of a socket
Syntax
```
hb_inetSetSndBufSize( <socket>, <nSize> ) → nSize
```
Arguments
`socket` a socket previously created / opened
`nSize`
Returns
Returns the passed `nSize` or -1 on error
Description
Sets the send buffer size of a socket
Status
Ready

Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetTimeLimit() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get or change the time limit value of a socket
Syntax
```
hb_inetTimeLimit( <socket> [, <nTimeLimit> ) → NIL
```
Arguments
`socket` a socket previously created / opened
`nTimeLimit`
Returns
Returns the previous time limit value of the socket
Description
Sets or changes the time limit value of the socket.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_inetTimeout() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get or change the timeout value of a socket
Syntax
```
hb_inetTimeout( <socket> [, <nTimeout> ] ) → nPreviousTimeout
```
Arguments
`socket` a socket previously created / opened
`nTimeout` is the new socket timeout value
Returns
Returns the previous timeout value of the socket
Description
Sets or changes the timeout value of the socket.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, INET

hb_cdpSelect() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Select the active code page by language ID

Syntax

`hb_cdpSelect( [<cNewLang>] ) → cOldLang`

Arguments

`cNewLang` The optional ID of the language module. Possible values for `cNewLang` are below as defined in the Codepage library, sorted by language.

Language	Codepage	<cNewLang>
Bulgarian	866	BG866
Bulgarian	ISO-8859-5	BGISO
Bulgarian	MIK	BGMIK
Bulgarian	Windows-1251	BGWIN
Croatian	437	HR437
Croatian	852	HR852
Croatian	Windows-1250	HRWIN
Czech	852	CS852
Czech	ISO-8859-2	CSISO
Czech	KAM	CSKAM
Czech	Windoes-1250	CSWIN
English	437	EN
French	850	FR850
German	850	DE850
German	ISO-8859-1	DEWIN
Greek	737	EL737
Greek	Windows-1253	ELWIN
Hungarian (ntxhu852)	852	HU852C
Hungarian (sixhu852)	852	HU852
Hungarian (sixhu852)	CWI-2	HUCWI
Hungarian	ISO-8859-2	HUISO
Hungarian	Windows-1250	HUWIN
Italian	437	IT437
Italian	850	IT850
Italian	ISO-8859-1b	ITISB
Italian	ISO-8859-1	ITISO
Lithuanian	Windows-1257	LTWIN
Polish	852	PL852
Polish	ISO-8859-2	PLISO
Polish	Mazowia	PLMAZ
Polish	Windows-1250	PLWIN
Portuguese	850	PT850
Portuguese	ISO-8859-1	PTISO
Russian	866	RU866
Russian	KOI-8	RUKOI8
Russian	Windows-1251	RU1251
Serbian	Windows-1251	SRWIN
Slovak	852	SK852
Slovak	ISO-8859-2	SKISO
Slovak	Kamenicky	SKKAM
Slovak	Windows-1250	SKWIN
Slovenian	437	SL437
Slovenian	852	SL852
Slovenian	ISO-8859-2	SLISO

Slovenian	Windows-1250	SLWIN
Spanish	850	ES850
Spanish	ISO-8859-1	ESWIN
Spanish Modern	ISO-8859-1	ESMWIN
Swedish	850	SV850
Swedish (Clipper)	437	SV437C
Swedish	ISO-8859-1	SVWIN
Turkish	857	TR857
Turkish	Windows-1254	TRWIN
Ukrainian	866	UA866
Ukrainian	KOI-8U	UAKOI8
Ukrainian	Windows-1251	UA1251

Returns

`cOldLang` The old language identifier

Description

`hb_cdpSelect()` set the active code page use by Harbour for sorting and comparing strings. The default code page use ASCII order (cLang == "EN").

NOTE: You must `REQUEST` every code page module you intend to use. For example: to use the French 850 code page you must add the following to your program: `REQUEST HB_CODEPAGE_FR850`

Examples

```
REQUEST HB_CODEPAGE_FR850
hb_cdpSelect( "EN" )
? hb_cdpSelect()
? hb_UTF8ToStr( "É < G is" ), hb_BChar( 144 ) < "G"  // --> É < G is .F.
hb_cdpSelect( "FR850" )
? hb_cdpSelect()
? hb_UTF8ToStr( "É < G is" ), hb_BChar( 144 ) < "G"  // --> É < G is .T.
```

Status

Ready

Compliance

This function is a Harbour Extension.

Platforms

Available on all platforms

Tags

API, Language and Nation

See also

`hb_langName()`, `hb_langSelect()`, `hb_Translate()`, `NationMsg()`, `REQUEST`

`hb_langErrMsg()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Description of an error code using current language

Syntax

```
hb_langErrMsg( <nErrorCode> ) → cErrorMessage
```

Arguments

`nErrorCode` is one of the generic error codes (`EG_...`) defined in `error.ch`

Returns

`hb_langErrMsg()` return the error message string represented by the code `nErrorCode`.

Description

This function return the error message associated with an error code using the current language selected.
Examples
```
#include "error.ch"
REQUEST HB_LANG_ES
? "English:", hb_langErrMsg( EG_ARG )  // --> English: Argument error
hb_langSelect( "es" )
? "Spanish:", hb_langErrMsg( EG_ARG )  // --> Spanish: Error de argumento
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Header is error.ch
Tags
API, Language and Nation
See also
hb_langSelect(), NationMsg()

hb_langMessage() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns international strings messages and errors
Syntax
```
hb_langMessage( <nMsg>[, <cLangID>] ) → cMessage
```
Arguments
nMsg is the message number to get. cLangID is an optional language module ID. Uses the currently
selected language module, if not specified.
Returns
hb_langMessage() return the text associated with the code nMsg.
Description

hb_langMessage() is similar to NationMsg() but give access to the whole list of language messages: Day and month names, generic error messages, internal errors, and others...

Use the header file hblang.ch for a list of base values for nMsg.

Examples
```
#include "hblang.ch"
REQUEST HB_LANG_ES
? "English:", hb_langMessage( HB_LANG_ITEM_BASE_DAY + 1 )  // --> English:
Monday
hb_langSelect( "es" )
? "Spanish:", hb_langMessage( HB_LANG_ITEM_BASE_DAY + 1 )  // --> Spanish:
Lunes
? "English:", hb_langMessage( HB_LANG_ITEM_BASE_DAY + 1, "en" )
```
Status
Ready
Compliance
Harbour specific
Platforms

Available on all platforms
File
Header is hblang.ch
Tags
API, Language and Nation
See also
hb_langSelect(), NationMsg(), REQUEST

hb_langName() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the name of the language module
Syntax
```
hb_langName( [<cLangID>] ) → cLangName
```
Arguments
cLangID is an optional language module ID. Uses the currently selected language module, if not specified.
Returns
cLangName Name of the language module
Description
This function return the name of the language module in use or specified.
Examples
```
REQUEST HB_LANG_PT
REQUEST HB_LANG_RO
REQUEST HB_LANG_ES
REQUEST HB_LANG_PL
? hb_langName( "pl" )
? hb_langName( "<non-existent>" )
hb_langSelect( "pt" )       // Default language is now Portuguese
? CDoW( Date() )            // --> Segunda-feira
? "Current language is", hb_langName()              // --> Portuguese
? "Old language id selected is", hb_langSelect()    // --> pt
hb_langSelect( "ro" )       // Default language is now Romanian
? CMonth( Date() )          // --> Mai
? "Old language id selected is", hb_langSelect()    // --> ro
hb_langSelect( "es" )       // Default language is now Spanish
? "Current language is", hb_langName()              // --> Spanish
? CMonth( Date() )          // --> Mayo
? CDoW( Date() )            // --> Lunes
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Language and Nation
See also
hb_langSelect(), NationMsg()

hb_langSelect() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Select a specific nation message module
Syntax
```
hb_langSelect( [<cNewLang>][, <cCodepage>] ) → cOldLang
```

Arguments

`cNewLang` The optional ID of the language module. Possible values for `cNewLang` are below as defined in the Lang library, sorted by language. `cCodepage` Optional codepage ID into which the language module strings are automatically converted by Harbour.

Language	<cNewLang>
Basque	eu
Belorussian	be
Bulgarian	bg
Catalan	ca
Chinese Traditional	zh
Chinese Simplified	zh_sim
Croatian	hr
Czech	cs
Dutch	nl
Esperanto	eo
French	fr
Galician	gl
German	de
Greek	el
Hebrew	he
Hungarian	hu
Icelandic	is
Indonesian	id
Italian	it
Korean	ko
Lithuanian	lt
Polish	pl
Portuguese	pt
Romanian	ro
Russian	ru
Serbian (cyrillic)	sr_cyr
Serbian (latin)	sr_lat
Slovak	sk
Slovenian	sl
Spanish	es
Swedish	sv
Turkish	tr
Ukrainian	uk

Returns

`cOldLang` The old language identifier

Description

This function set a default language module for date/month names, internal warnings, NatMsg messages and internal errors. When a Lang ID is selected all messages will be output with the current language selected until another one is selected or the program ends. The default language is English (cLang == "EN").

NOTE: You must `REQUEST` every language module you intend to use. For example: to use the Portuguese language you must add the following to your program: `REQUEST HB_LANG_PT`

Examples

```
REQUEST HB_LANG_PT
REQUEST HB_LANG_RO
REQUEST HB_LANG_ES
hb_langSelect( "pt" )       // Default language is now Portuguese
? CDoW( Date() )            // --> Segunda-feira
? "Old language id selected is", hb_langSelect()  // --> pt
hb_langSelect( "ro" )       // Default language is now Romanian
? CMonth( Date() )          // --> Mai
? "Old language id selected is", hb_langSelect()  // --> ro
hb_langSelect( "es" )       // Default language is now Spanish
? CMonth( Date() )          // --> Mayo
? CDoW( Date() )            // --> Lunes
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
API, Language and Nation
See also
`hb_cdpSelect()`, `hb_langName()`, `NationMsg()`, `REQUEST`

`hb_Translate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Translate a string from one code page to the other
Syntax
`hb_Translate( <cSrcText>, [<cPageFrom>], [<cPageTo>] ) → cDstText`
Arguments
`cSrcText` Is the source string to translate.
`cPageFrom` Is the optional character code page ID of the source string. If not specified, the default code page is used.
`cPageTo` Is the optional character code page ID of the destination string. If not specified, the default code page is used.
Returns
`hb_Translate()` return destination string converted from the source string.
Description

`hb_Translate()` try to convert a source string from one code page into the other. If a code page ID is not recognized, or not linked in, the default code page is used. `hb_Translate()` is used usually to convert between the MS-DOS and the Windows code pages of the same language.

NOTE: If the source code page and target code page does not have the same number of characters, a translation cannot be done and the destination string is a copy of the source string.

NOTE: You must `REQUEST` every code page module you intend to use. For example: to use the CP-850 code page you must add the following to your program: `#include "hbextcdp.ch"`

Examples

```
#include "hbextcdp.ch"
LOCAL cTxt := "A" + hb_BChar( 142 ) + "BC"
```

```
? "CP-850 text:", hb_StrToHex( cTxt )
? "UTF-8 text:", hb_Translate( cTxt, "cp850", "UTF8" )
```

Status

Ready

Compliance

This function is a Harbour Extension.

Platforms

Available on all platforms

Tags

API, Language and Nation

See also

hb_cdpSelect(), hb_langSelect(), NationMsg(), REQUEST

IsAffirm() → __natIsAffirm() Source code in Repository in Discussions

^ | ☰ | Improve this doc

Checks if passed char is an affirmation char

Syntax

```
IsAffirm( <cChar> ) → lTrueOrFalse
```

Arguments

cChar is a char or string of chars

Returns

lTrueOrFalse True if passed char is an affirmation char, otherwise false

Description

This function is used to check if a user's input is true or not according to the msgxxx module used.

Examples

```
// Wait until user enters Y
DO WHILE ! IsAffirm( cYesNo )
   ACCEPT "Sure: " TO cYesNo
ENDDO
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Language and Nation

See also

IsNegative(), NationMsg()

IsNegative() → __natIsNegative() Source code in Repository in Discussions

^ | ☰ | Improve this doc

Checks if passed char is a negation char.

Syntax

```
IsNegative( <cChar> ) → lTrueOrFalse
```

Arguments

cChar is a char or string of chars

Returns
`lTrueOrFalse` True if passed char is a negation char, otherwise false.
Description
This function is used to check if a user's input is true or not according to the msgxxx module used.
Examples
```
// Wait until user enters N
DO WHILE ! IsNegative( cYesNo )
   ACCEPT "Sure: " TO cYesNo
ENDDO
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Language and Nation
See also
`IsAffirm()`, `NationMsg()`

`NationMsg()` → `__natMsg()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns international strings messages.
Syntax
`NationMsg( <nMsg> ) → cMessage`
Arguments
`nMsg` is the message number you want to get.
Returns
`cMessage` if `nMsg` is a valid message selector, returns the message. If `nMsg` is `NIL`, it returns "Invalid Argument", and if `nMsg` is any other type it returns an empty string.
Description
`NationMsg()` returns international message descriptions.
Examples
```
// Displays "Sure Y/N: "  and waits until user enters Y
// Y/N is the string for NationMsg( 12 ) with default natmsg module.
DO WHILE ! IsAffirm( cYesNo )
   ACCEPT "Sure " + NationMsg( 12 ) + ": " TO cYesNo
ENDDO
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Language and Nation
See also

IsAffirm(), IsNegative()

hb_SetMacro() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Enable/disable the macro compiler runtime features.
Syntax
hb_SetMacro(<nOption>, [<lOnOff>]) → lOldSetting
Arguments
nOption One of the HB_SM_* constants defined in set.ch.
lOnOff .T. to enable or .F. to disable a feature
Returns
hb_SetMacro() return the old state of requested feature.
Description

This function enables or disables some features of the macro compiler. The Harbour is extending the macro features compared to an original set available in CA-Cl*pper. Enabling/disabling some of them allows to keep strict CA-Cl*pper compatibility.

Available features are:

HB_SM_HARBOUR - enables Harbour extensions:

operators: ++, --, +=, -=, *=, /=, ^= objects: assignments to an instance variable

HB_SM_XBASE - enables other Xbase++ dialects extensions:

expanding of expressions lists

HB_SM_SHORTCUTS - enables optimized evaluation of

logical operators (.AND., .OR.)

HB_SM_PREPROC - enables preprocessing of commands

This is meaningful if Harbour is compiled with HB_MACRO_STATEMENTS flag

Examples
```
#include "hbmacro.ch"
INIT PROCEDURE IWANTCLIPPER()
   ? hb_SetMacro( HB_SM_HARBOUR, .F. )
   ? hb_SetMacro( HB_SM_XBASE, .F. )
   RETURN
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Files
Header file is set.ch
Library is core
Tags

API, Macro
See also
Macro compiler

Abs() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the absolute value of a number.
Syntax
```
Abs( <nNumber> ) → nAbsNumber
```
Arguments
`nNumber` Any number.
Returns
`nAbsNumber` The absolute numeric value.
Description
This function yields the absolute value of the numeric value or expression `nNumber`.
Examples
```
LOCAL nNumber := 50
LOCAL nNumber1 := 27

? nNumber - nNumber1
? nNumber1 - nNumber
? Abs( nNumber - nNumber1 )
? Abs( nNumber1 - nNumber )
? Abs( -1 * 345 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Math

Exp() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Calculates the value of e raised to the passed power.
Syntax
```
Exp( <nNumber> ) → nValue
```
Arguments
`nNumber` Any real number.
Returns
`nValue` The anti-logarithm of `nNumber`
Description
This function returns the value of e raised to the power of `nNumber`. It is the inverse of `Log()`.
Examples
```
? Exp( 45 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
File
Library is core
Tags
API, Math
See also
`Log()` Clipper 5.3

`hb_matherBlock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set/Get math error handling codeblock
Syntax
```
hb_matherBlock( [<bNewBlock>] ) → bOldBlock
```
Arguments
`bNewBlock`
Returns
`bOldBlock` is the current error handler codeblock
Status
Ready
Platforms
Available on all platforms
File
Library is core
Tags
API, Math

`hb_matherMode()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set/Get math error handling mode
Syntax
```
hb_matherMode( [<nNewMode>] ) → nOldMode
```
Arguments
[`nNumber`] new math error handling mode, one of the following
constants, defined in `hbmath.ch`:
HB_MATH_ERRMODE_DEFAULT
HB_MATH_ERRMODE_CDEFAULT
HB_MATH_ERRMODE_USER
HB_MATH_ERRMODE_USERDEFAULT
HB_MATH_ERRMODE_USERCDEFAULT
Returns
`nOldMode` old math error handling mode
Status
Ready
Platforms
Available on all platforms
Files
Header file is `hbmath.ch`
Library is core
Tags
API, Math

Int() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the integer port of a numeric value.
Syntax
```
Int( <nNumber> ) → nIntNumber
```
Arguments
`nNumber` Any numeric value.
Returns
`nIntNumber` The integer portion of the numeric value.
Description
This function converts a numeric expression to an integer. All decimal digits are truncated. This function does not round a value upward or downward; it merely truncates a number at the decimal point.
Examples
```
Set( _SET_DECIMALS, 5 )
? Int( 632512.62541 )
? Int( 845414111.91440 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Math
See also
`Round()` Clipper 5.3, `StrZero()`

Log() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the natural logarithm of a number.
Syntax
```
Log( <nNumber> ) → nLog
```
Arguments
`nNumber` Any numeric expression.
Returns
`nExponent` The natural logarithm of `nNumber`.
Description
This function returns the natural logarithm of the number `nNumber`. If `nNumber` is 0 or less than 0, a numeric overflow occurs, which is depicted on the display device as a series of asterisks. This function is the inverse of `Exp()`.
Examples
```
? Log( 632512 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File

Library is core
Tags
API, Math
See also
`Exp()` Clipper 5.3

`Max()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the maximum of two numbers or dates.
Syntax
```
Max( <xValue>, <xValue1> ) → xMax
```
Arguments
`xValue` Any date or numeric value.
`xValue1` Any date or numeric value (same type as `xValue`).
Returns
`xMax` The larger numeric (or later date) value.
Description
This function returns the larger of the two passed expressions. If `xValue` and `xValue1` are numeric data types, the value returned by this function will be a numeric data type as well and will be the larger of the two numbers passed to it. If `xValue` and `xValue1` are date data types, the return value will be a date data type as well. It will be the later of the two dates passed to it.
Examples
```
? Max( 214514214, 6251242142 )
? Max( 0d20001111, 0d20140621 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Math
See also
`Min()` Clipper 5.3

`Min()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determines the minimum of two numbers or dates.
Syntax
```
Min( <xValue>, <xValue1> ) → xMin
```
Arguments
`xValue` Any date or numeric value.
`xValue1` Any date or numeric value.
Returns
`xMin` The smaller numeric (or earlier date) value.
Description

This function returns the smaller of the two passed expressions. `xValue` and `xValue1` must be the same data type. If numeric, the smaller number is returned. If dates, the earlier date is returned.

Examples

```
? Min( 214514214, 6251242142 )
? Min( 0d20001111, 0d20140621 )
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Math

See also

`Max()` Clipper 5.3

`Mod()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the modulus of two numbers.

Syntax

```
Mod( <nNumber>, <nNumber1> ) → <nRemainder>
```

Arguments

`nNumber` Numerator in a divisional expression.

`nNumber1` Denominator in a divisional expression.

Returns

`nRemainder` The remainder after the division operation.

Description

This function returns the remainder of one number divided by another.

Examples

```
? Mod( 12, 8.521 )
? Mod( 12, 0 )
? Mod( 62412.5142, 4522114.12014 )
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Math

See also

`%`

`Round()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Rounds off a numeric expression.

Syntax

```
Round( <nNumber>, <nPlace> ) → nResult
```
Arguments
`nNumber` Any numeric value.
`nPlace` The number of places to round to.
Returns
`nResult` The rounded number.
Description
This function rounds off the value of `nNumber` to the number of decimal places specified by `nPlace`. If the value of `nPlace` is a negative number, the function will attempt to round `nNumber` in whole numbers. Numbers from 5 through 9 will be rounded up, all others will be rounded down.
Examples
```
? Round( 632512.62541, 5 )
? Round( 845414111.91440, 3 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Math
See also
`Int()` Clipper 5.3, `SET FIXED` Clipper 5.3, `Str()` Clipper 5.3, `Val()` Clipper 5.3

`Sqrt()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Calculates the square root of a number.
Syntax
```
Sqrt( <nNumber> ) → nSqrt
```
Arguments
`nNumber` Any numeric value.
Returns
`nSqrt` The square root of `number`.
Description
This function returns the square root of `nNumber`. The precision of this evaluation is based solely on the setting of `_SET_DECIMALS`. Any negative number passed as `nNumber` will always return a 0.
Examples
```
Set( _SET_DECIMALS, 5 )
? Sqrt( 632512.62541 )
? Sqrt( 845414111.91440 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, Math
See also
`Round()` Clipper 5.3

`__objAddData()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Add a VAR to an already existing class
Syntax
```
__objAddData( <oObject>, <cDataName> ) → oObject
```
Arguments
`oObject` is the object to work on.
`cDataName` is the symbol name of the new `VAR` to add.
Returns
`__objAddData()` return a reference to `oObject`.
Description
`__objAddData()` is a low-level class support function that add a new `VAR` to an object.
`oObject` is unchanged if a symbol with the name `cDataName` already exist in `oObject`.
Examples
```
// create a new THappy class and add a lHappy VAR
LOCAL oHappy := HBClass():New( "THappy" )
__objAddData( oHappy, "lHappy" )
oHappy:lHappy := .T.
IF oHappy:lHappy
   ? "Happy, Happy, Joy, Joy !!!"
ELSE
   ? ":(..."
ENDIF
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
`__objAddInline()`, `__objAddMethod()`, `__objDelData()`, `__objGetMsgList()`, `__objGetValueList()`, `__objHasData()`, `__objSetValueList()`

`__objAddInline()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Add an INLINE to an already existing class
Syntax
```
__objAddInline( <oObject>, <cInlineName>, <bInline> ) → oObject
```
Arguments
`oObject` is the object to work on.
`cInlineName` is the symbol name of the new `INLINE` to add.
`bInline` is a code block to associate with the `INLINE` method.
Returns
`__objAddInline()` return a reference to `oObject`.

Description

`__objAddInline()` is a low-level class support function that add a new `INLINE` method to an object. `oObject` is unchanged if a symbol with the name `cInlineName` already exist in `oObject`.

Examples

```
// create a new THappy class and add a Smile INLINE method
LOCAL oHappy := HBClass():New( "THappy" )
LOCAL bInline := {| Self, nType | HB_SYMBOL_UNUSED( Self ), ;
   { ":)", ";)", "*SMILE*" }[ nType ] }
__objAddInline( oHappy, "Smile", bInline )
? oHappy:Smile( 1 )       // --> :)
? oHappy:Smile( 2 )       // --> ;)
? oHappy:Smile( 3 )       // --> *SMILE*
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Objects

See also

`__objAddData()`, `__objAddMethod()`, `__objDelInline()`, `__objGetMethodList()`, `__objGetMsgList()`, `__objHasMethod()`, `__objModInline()`

`__objAddMethod()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Add a METHOD to an already existing class

Syntax

```
__objAddMethod( <oObject>, <cMethodName>, <nFuncPtr> ) → oObject
```

Arguments

`oObject` is the object to work on.

`cMethodName` is the symbol name of the new `METHOD` to add.

`nFuncPtr` is a pointer to a function to associate with the method.

Returns

`__objAddMethod()` return a reference to `oObject`.

Description

`__objAddMethod()` is a low-level class support function that add a new `METHOD` to an object. `oObject` is unchanged if a symbol with the name `cMethodName` already exist in `oObject`.

Note that `nFuncPtr` is a special pointer to a function that was created using the @ operator, see example below.

Examples

```
// create a new THappy class and add a Smile method
LOCAL oHappy := HBClass():New( "THappy" )
__objAddMethod( oHappy, "Smile", @MySmile() )
? oHappy:Smile( 1 )       // --> :)
? oHappy:Smile( 2 )       // --> ;)
? oHappy:Smile( 3 )       // --> *SMILE*
```

```
STATIC FUNCTION MySmile( nType )
   IF HB_ISNUMERIC( nType )
      SWITCH nType
      CASE 1 ; RETURN ":)"
      CASE 2 ; RETURN ";)"
      CASE 3 ; RETURN "*SMILE*"
      ENDSWITCH
   ENDIF
   RETURN NIL
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
__objAddData(), __objAddInline(), __objDelMethod(), __objGetMethodList(), __objGetMsgList(), __objHasMethod(), __objModMethod()

__objDelData() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Delete a VAR (instance variable) from class
Syntax
`__objDelMethod( <oObject>, <cDataName> ) → oObject`
Arguments
`oObject` is the object to work on.
`cDataName` is the symbol name of `VAR` to be deleted (removed) from the object.
Returns
`__objDelData()` return a reference to `oObject`.
Description
`__objDelData()` is a low-level class support function that delete (remove) a `VAR` from an object. `oObject` is unchanged if a symbol with the name `cDataName` does not exist in `oObject`.
Examples

```
// create a new THappy class and add a lHappy VAR
LOCAL oHappy := HBClass():New( "THappy" )
__objAddData( oHappy, "lHappy" )
? __objHasData( oHappy, "lHappy" )  // --> .T.
// remove lHappy VAR
__objDelData( oHappy, "lHappy" )
? __objHasData( oHappy, "lHappy" )  // --> .F.
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags

API, Objects
See also
__objAddData(), __objGetMsgList(), __objGetValueList(), __objHasData(), __objSetValueList()

__objDelInline() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Delete a METHOD INLINE from class
Syntax
__objDelInline(<oObject>, <cSymbol>) → oObject
Arguments
oObject is the object to work on.
cSymbol is the symbol name of METHOD or INLINE method to be deleted (removed) from the object.
Returns
__objDelInline() return a reference to oObject.
Description
__objDelInline() is a low-level class support function that delete (remove) a METHOD or an INLINE method from an object. oObject is unchanged if a symbol with the name cSymbol does not exist in oObject.
Examples

```
// create a new THappy class and add a Smile method
LOCAL oHappy := HBClass():New( "THappy" )
__objAddMethod( oHappy, "Smile", @MySmile() )
? __objHasMethod( oHappy, "Smile" )  // --> .T.
// remove Smile method
__objDelInline( oHappy, "Smile" )
? __objHasMethod( oHappy, "Smile" )  // --> .F.

STATIC FUNCTION MySmile( nType )
   DO CASE
   CASE nType == 1
      RETURN ":)"
   CASE nType == 2
      RETURN ";)"
   ENDCASE
   RETURN NIL
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
__objAddInline(), __objAddMethod(), __objGetMethodList(), __objGetMsgList(), __objHasMethod(), __objModInline(), __objModMethod()

__objDelMethod() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Delete a METHOD from class

Syntax
`__objDelMethod( <oObject>, <cSymbol> ) → oObject`
Arguments
`oObject` is the object to work on.
`cSymbol` is the symbol name of `METHOD` or `INLINE` method to be deleted (removed) from the object.
Returns
`__objDelMethod()` return a reference to `oObject`.
Description

`__objDelMethod()` is a low-level class support function that deletes (removes) a `METHOD` or an `INLINE` method from an object. `oObject` is unchanged if a symbol with the name `cSymbol` does not exist in `oObject`.

`__objDelInline()` is exactly the same as `__objDelMethod()`.

Examples

```
// create a new THappy class and add a Smile method
LOCAL oHappy := HBClass():New( "THappy" )
__objAddMethod( oHappy, "Smile", @MySmile() )
? __objHasMethod( oHappy, "Smile" )  // --> .T.
// remove Smile method
__objDelMethod( oHappy, "Smile" )
? __objHasMethod( oHappy, "Smile" )  // --> .F.

STATIC FUNCTION MySmile( nType )
   DO CASE
   CASE nType == 1
      RETURN ":)"
   CASE nType == 2
      RETURN ";)"
   ENDCASE
   RETURN NIL
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
`__objAddInline()`, `__objAddMethod()`, `__objGetMethodList()`, `__objGetMsgList()`, `__objHasMethod()`, `__objModInline()`, `__objModMethod()`

`__objDerivedFrom()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine whether a class is derived from another class
Syntax
`__objDerivedFrom( <oObject>, <xSuper> ) → lIsParent`
Arguments
`oObject` is the object to check.

`xSuper` is the object that may be a parent. `xSuper` can be either an Object or a Character string with the class name.

Returns

`__objDerivedFrom()` return a logical TRUE (`.T.`) if `oObject` is derived from `xSuper`.

Description

`__objDerivedFrom()` is a low-level class support function that check is one class is a super class of the other, or in other words, does class `oObject` a child or descendant of `xSuper`.

Examples

```
// Create three classes and check their relations

#include "hbclass.ch"
PROCEDURE Main()
   LOCAL oSuper  := TMood():New()
   LOCAL oObject := THappy():New()
   LOCAL oDress  := TShirt():New()
   ? __objDerivedFrom( oObject, oSuper )  // --> .T.
   ? __objDerivedFrom( oSuper, oObject )  // --> .F.
   ? __objDerivedFrom( oObject, oDress )  // --> .F.
   RETURN

CREATE CLASS TMood
   METHOD New() INLINE Self
ENDCLASS

CREATE CLASS THappy INHERIT TMood
   METHOD Smile() INLINE QOut( "*smile*" )
ENDCLASS

CREATE CLASS TShirt
   VAR Color
   VAR Size
   METHOD New() INLINE Self
ENDCLASS
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Objects

See also

`__objHasData()`, `__objHasMethod()`

`__objGetMethodList()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return names of all METHOD for a given object

Syntax

```
__objGetMethodList( <oObject> ) → aMethodNames
```

Arguments

`oObject` is an object to scan.

Returns

__objGetMethodList() return an array of character stings with all METHOD names for a given object. __objGetMethodList() would return an empty array {} if the given object does not contain any METHOD.
Description
__objGetMethodList() is a low-level class support function that let you find all class functions names for a given object. It is equivalent to __objGetMsgList(oObject, .F.).
Examples

```
// show information about TBrowse class
LOCAL oB := TBrowseNew( 0, 0, 24, 79 ), tmp
FOR EACH tmp IN __objGetMethodList( oB )
   ? "METHOD name:", tmp
NEXT
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
__objGetMsgList(), __objGetValueList(), __objHasData(), __objHasMethod()

__objGetMsgList() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return names of all VAR or METHOD for a given object
Syntax

```
__objGetMsgList( <oObject>, [<lData>], [nClassType] ) → aNames
```

Arguments
oObject is an object to scan.
lData is an optional logical value that specifies the information to return. A value of .T. instruct the function to return list of all VAR names, .F. return list of all METHOD names. Default value is .T.
nClassType is on optional numeric code for selecting which class type to return. Default value is HB_MSGLISTALL, returning the whole list.
Returns
__objGetMsgList() return an array of character stings with all VAR names or all METHOD names for a given object. __objGetMsgList() would return an empty array {} if the given object does not contain the requested information.
Description

__objGetMsgList() is a low-level class support function that let you find all instance variable or method names for a given object.

If specified, the following table shows the values for nClassType that allow you to distinguish between VAR and CLASS VAR:

hboo.ch	Meaning
HB_MSGLISTALL	All types
HB_MSGLISTCLASS	CLASS VAR only

HB_MSGLISTPURE VAR only

VAR are instance variable usable within each object from a class, where each object has its own VARS.

CLASS VAR are shared by all objects from a Class, so the changed value within Object1 will be reflected when accessing the CLASS VAR from Object2.

Examples

```
#include "hboo.ch"
// show information about TBrowse class
LOCAL oB := TBrowseNew( 0, 0, 24, 79 ), tmp
FOR EACH tmp IN __objGetMsgList( oB, .T. )
   ? "VAR name:", tmp
NEXT
FOR EACH tmp IN __objGetMsgList( oB, .T., HB_MSGLISTCLASS )
   ? "CLASS VAR name:", tmp
NEXT
FOR EACH tmp IN __objGetMsgList( oB, .F. )
   ? "METHOD name:", tmp
NEXT
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Files
Header file is hboo.ch
Library is core
Tags
API, Objects
See also
__objGetMethodList(), __objGetValueList(), __objHasData(), __objHasMethod()

__objGetValueList() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return an array of VAR names and values for a given object
Syntax

```
__objGetValueList( <oObject>, [<aExcept>] ) → aData
```

Arguments
oObject is an object to scan.
aExcept is an optional array with VAR names you want to exclude from the scan.
Returns
__objGetValueList() return a 2D array that contain pairs of a VAR symbol name and the value of VAR. __objGetValueList() would return an empty array {} if the given object does not contain the requested information.
Description

__objGetValueList() is a low-level class support function that return an array with VAR names and value, each array element is a pair of: aData[i][HB_OO_DATA_SYMBOL] contain the symbol name

`aData[ i ][ HB_OO_DATA_VALUE ]` contain the value of `VAR`

Examples

```
// FIXME
// show information about TBrowse class
#include "hboo.ch"
LOCAL oB := TBrowseNew( 0, 0, 24, 79 ), tmp
FOR EACH tmp IN __objGetValueList( oB )
   ? ;
     "VAR name:", tmp[ HB_OO_DATA_SYMBOL ], ;
     "   value=", tmp[ HB_OO_DATA_VALUE ]
NEXT
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

Files

Header file is `hboo.ch`

Library is core

Tags

API, Objects

See also

`__objGetMethodList()`, `__objGetMsgList()`, `__objHasData()`, `__objHasMethod()`, `__objSetValueList()`

`__objHasData()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine whether a symbol exist in object as VAR

Syntax

```
__objHasData( <oObject>, <cSymbol> ) → lExist
```

Arguments

`oObject` is an object to scan.

`cSymbol` is the name of the symbol to look for.

Returns

`__objHasData()` return `.T.` if the given `cSymbol` exist as `VAR` (instance variable) in object `oObject), .F. if it does not exist.

Description

`__objHasData()` is a low-level class support function that let you find out if a symbol is an instance variable in a given object.

Examples

```
LOCAL oB := TBrowseNew( 0, 0, 24, 79 )
? __objHasData( oB, "nLeft" )      // --> .T.
? __objHasData( oB, "lBugFree" )   // --> .F.
? __objHasData( oB, "Left" )       // --> .F. since this is a METHOD
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags
API, Objects
See also
__objGetMethodList(), __objGetMsgList(), __objHasMethod()

__objHasMethod() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine whether a symbol exist in object as METHOD
Syntax
```
__objHasMethod( <oObject>, <cSymbol> ) → lExist
```
Arguments
oObject is an object to scan.
cSymbol is the name of the symbol to look for.
Returns
__objHasMethod() return .T. if the given cSymbol exist as METHOD (class function) in object `oObject), .F. if it does not exist.
Description
__objHasMethod() is a low-level class support function that let you find out if a symbol is a class function in a given object.
Examples
```
LOCAL oB := TBrowseNew( 0, 0, 24, 79 )
? __objHasMethod( oB, "nLeft" )     // --> .F. since this is a VAR
? __objHasMethod( oB, "FixBugs" )   // --> .F.
? __objHasMethod( oB, "Left" )      // --> .T.
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
__objGetMethodList(), __objGetMsgList(), __objHasData()

__objModInline() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Modify (replace) an INLINE method in an already existing class
Syntax
```
__objModInline( <oObject>, <cInlineName>, <bInline> ) → oObject
```
Arguments
oObject is the object to work on.
cInlineName is the symbol name of the INLINE method to modify.
bInline is a new code block to associate with the INLINE method.
Returns
__objModInline() return a reference to oObject.
Description
__objModInline() is a low-level class support function that modify an INLINE method in an object and replace it with a new code block. oObject is unchanged if a symbol with the name

`cInlineName` does not exist in `oObject`. `__objModInline()` is used in inheritance mechanism.

Examples

```
// create a new THappy class and add a Smile INLINE method
LOCAL oHappy := HBClass():New( "THappy" )
LOCAL bMyInline   := {| Self, nType | HB_SYMBOL_UNUSED( Self ), ;
   { ":)", ";)" }[ nType ] }
LOCAL bYourInline := {| Self, nType | HB_SYMBOL_UNUSED( Self ), ;
   { "*SMILE*", "*WINK*" }[ nType ] }
__objAddInline( oHappy, "Smile", bMyInline )
? oHappy:Smile( 1 )       // --> :)
? oHappy:Smile( 2 )       // --> ;)
// replace Smile inline method with a new code block
__objModInline( oHappy, "Smile", bYourInline )
? oHappy:Smile( 1 )       // --> *SMILE*
? oHappy:Smile( 2 )       // --> *WINK*
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Objects

See also

`__objAddInline()`, `__objDelInline()`, `__objGetMethodList()`, `__objGetMsgList()`, `__objHasMethod()`

`__objModMethod()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Modify (replace) a METHOD in an already existing class

Syntax

```
__objModMethod( <oObject>, <cMethodName>, <nFuncPtr> ) → oObject
```

Arguments

`oObject` is the object to work on.

`cMethodName` is the symbol name of the `METHOD` to modify.

`nFuncPtr` is a pointer to a new function to associate with the method.

Returns

`__objModMethod()` return a reference to `oObject`.

Description

`__objModMethod()` is a low-level class support function that modify a `METHOD` in an object and replace it with a new function. `oObject` is unchanged if a symbol with the name `cMethodName` does not exist in `oObject`. `__objModMethod()` is used in inheritance mechanism.

Note that `nFuncPtr` is a special pointer to a function that was created using the @ operator, see example below.

Examples

```
// create a new THappy class and add a Smile method
LOCAL oHappy := HBClass():New( "THappy" )
```

```
__objAddMethod( oHappy, "Smile", @MySmile() )
? oHappy:Smile( 1 )       // --> :)
? oHappy:Smile( 2 )       // --> ;)
// replace Smile method with a new function
__objAddMethod( oHappy, "Smile", @YourSmile() )
? oHappy:Smile( 1 )       // --> *SMILE*
? oHappy:Smile( 2 )       // --> *WINK*

STATIC FUNCTION MySmile( nType )
   DO CASE
   CASE nType == 1
      RETURN ":)"
   CASE nType == 2
      RETURN ";)"
   ENDCASE
   RETURN NIL

STATIC FUNCTION YourSmile( nType )
   DO CASE
   CASE nType == 1
      RETURN "*SMILE*"
   CASE nType == 2
      RETURN "*WINK*"
   ENDCASE
   RETURN NIL
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Objects
See also
__objAddMethod(), __objDelMethod(), __objGetMethodList(), __objGetMsgList(), __objHasMethod()

__objSetValueList() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Set object with an array of VAR names and values
Syntax
`__objSetValueList( <oObject>, <aData> ) → oObject`
Arguments
`oObject` is an object to set.
`aData` is a 2D array with a pair of instance variables and values for setting those variable.
Returns
`__objSetValueList()` return a reference to `oObject`.
Description
`__objSetValueList()` is a low-level class support function that let you set a group of instance variables with values. each array element in `aData` is a pair of: aData[i][`HB_OO_DATA_SYMBOL`] which contain the variable name to set aData[i][`HB_OO_DATA_VALUE`] contain the new variable value.
Examples

```
// set some TBrowse instance variable
```

```
#include "hboo.ch"
LOCAL oB := TBrowse():New()
LOCAL aData := Array( 4, 2 )
aData[ 1 ][ HB_OO_DATA_SYMBOL ] := "nTop"
aData[ 1 ][ HB_OO_DATA_VALUE  ] := 1
aData[ 2 ][ HB_OO_DATA_SYMBOL ] := "nLeft"
aData[ 2 ][ HB_OO_DATA_VALUE  ] := 10
aData[ 3 ][ HB_OO_DATA_SYMBOL ] := "nBottom"
aData[ 3 ][ HB_OO_DATA_VALUE  ] := 20
aData[ 4 ][ HB_OO_DATA_SYMBOL ] := "nRight"
aData[ 4 ][ HB_OO_DATA_VALUE  ] := 70
__objSetValueList( oB, aData )
? oB:nTop     // --> 1
? oB:nLeft    // --> 10
? oB:nBottom  // --> 20
? oB:nRight   // --> 70
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Files
Header file is `hboo.ch`
Library is core
Tags
API, Objects
See also
`__objGetValueList()`

`FieldBlock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return a code block that sets/gets a value for a given field
Syntax
`FieldBlock( <cFieldName> ) → bFieldBlock`
Arguments
`cFieldName` is a string that contain the field name.
Returns
`FieldBlock()` return a code block that when evaluate could retrieve a field value or assigning a new value to the field. If `cFieldName` is not specified or from type other than character, `FieldBlock()` return `NIL`.
Description

`FieldBlock()` return a code block that sets/gets the value of field. When this code block is evaluated without any parameters passed then it returns the current value of the given field. If the code block is evaluated with a parameter, than its value is used to set a new value to the field, this value is also return by the block. If the block is evaluate and there is no field with the name `cFieldName` in the current work area, the code block return `NIL`.

Note that `FieldBlock()` works on the current work area, if you need a specific work area code block use `FieldWBlock()` instead.

Examples
```
// open a file named test.dbf that have a field named "first"
```

```
LOCAL bField
USE test
bField := FieldBlock( "first" )
? "Original value of field 'first':", Eval( bField )
Eval( bField, "Mr X new name" )
? "New value for the field 'first':", Eval( bField )
```

Status

Ready

Compliance

If the block is evaluate and there is no field with the name `cFieldName` in the current work area, the code block return `NIL`.

CA-Cl*pper would raise `BASE/1003` error if the field does not exist.

Platforms

Available on all platforms

File

Library is core

Tags

API, RDD

See also

`Eval()` Clipper 5.3, `FieldWBlock()` Clipper 5.3, `MemVarBlock()` Clipper 5.3

`FieldWBlock()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

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

Syntax

```
FieldWBlock( <cFieldName>, <nWorkArea> ) → bFieldBlock
```

Arguments

`cFieldName` is a string that contain the field name.

`nWorkArea` is the work area number in which `cFieldName` exist.

Returns

`FieldWBlock()` return a code block that when evaluate could retrieve field value or assigning a new value for a field in a given work area. If `cFieldName` is not specified or from type other than character, or if `nWorkArea` is not specified or is not numeric `FieldWBlock()` return `NIL`.

Description

`FieldWBlock()` return a code block that sets/gets the value of field from a given work area. When this code block is evaluated without any parameters passed then it returns the current value of the given field. If the code block is evaluated with a parameter, than its value is used to set a new value to the field, this value is also return by the block. If the block is evaluate and there is no field with the name `cFieldName` in work area number `nWorkArea`, the code block return `NIL`.

Examples

```
LOCAL bField
// open a file named 'one' in work area 1 that has a field named "first"
SELECT 1
USE one
// open a file named 'two' in work area 2 that also has a field named
"first"
SELECT 2
USE two
SELECT 1
// this block works on the field "first" that exists on work area 2
bField := FieldWBlock( "first", 2 )
? "Original 'first' values:", one->first, two->first
? "'first' value for file 'two':", Eval( bField )
Eval( bField, "'two' has updated 'first'" )
```

```
? "and now:", one->first, two->first
```
Status
Ready
Compliance
If the block is evaluate and there is no field with the name `cFieldName` in the given work area, the code block return `NIL`.
CA-Cl*pper would raise `BASE/1003` error if the field does not exist.
Platforms
Available on all platforms
File
Library is core
Tags
API, RDD
See also
`Eval()` Clipper 5.3, `FieldBlock()` Clipper 5.3, `MemVarBlock()` Clipper 5.3

`AllTrim()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Removes leading and trailing blank spaces from a string
Syntax
```
AllTrim( <cString> ) → cExpression
```
Arguments
`cString` Any character string
Returns
`cExpression` An string will all blank spaces removed from `cString`
Description
This function returns the string `cExpression` will all leading and trailing blank spaces removed.
Examples
```
? AllTrim( "Hello Harbour" )
? AllTrim( "     Hello Harbour" )
? AllTrim( "Hello Harbour     " )
? AllTrim( "     hello Harbour     " )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`LTrim()` Clipper 5.3, `RTrim()` Clipper 5.3, `Trim()` Clipper 5.3

`Asc()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the ASCII value of a character
Syntax
```
Asc( <cCharacter> ) → nAscNumber
```
Arguments

`cCharacter` Any character expression
Returns
`nAscNumber` ASCII value
Description
This function return the ASCII value of the leftmost character of any character expression passed as `cCharacter`.
Examples

```
? Asc( "A" )
? Asc( "a" )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`Chr()` Clipper 5.3

`At()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locates the position of a substring in a main string.
Syntax

```
At( <cSearch>, <cString> ) → nPos
```

Arguments
`cSearch` Substring to search for
`cString` Main string
Returns
`At()` return the starting position of the first occurrence of the substring in the main string
Description
This function searches the string `cString` for the characters in the first string `cSearch`. If the substring is not contained within the second expression, the function will return 0.
Examples

```
? At( "cde", "abcdefgfedcba" )  // --> 3
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`RAt()` Clipper 5.3

`Chr()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts an ASCII value to it character value
Syntax
```
Chr( <nAsciiNum> ) → cReturn
```
Arguments
`nAsciiNum` Any ASCII character code.
Returns
`cReturn` Character expression of that ASCII value
Description

This function returns the ASCII character code for `nAsciiNum`. The number expressed must be an integer value within the range of 0 to 255 inclusive. The `Chr()` function will send the character returned to whatever device is presently set.

The `Chr()` function may be used for printing special codes as well as normal and graphics character codes.

Examples
```
? Chr( 32 )
? Chr( 65 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`Asc()` Clipper 5.3, `Inkey()` Clipper 5.3

`hb_At()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locates the position of a substring in a main string.
Syntax
```
hb_At( <cSearch>, <cString>, [<nStart>], [<nEnd>] ) → nPos
```
Arguments
`cSearch` Substring to search for
`cString` Main string
`nStart` First position to search in `cString`, by default 1
`nEnd` End position to search, by default `cString` length
Returns
`hb_At()` return the starting position of the first occurrence of the substring in the main string
Description
This function searches the string `cString` for the characters in the first string `cSearch`. If the substring is not contained within the second expression, the function will return 0. The third and fourth parameters lets you indicate a starting and end offset to search in.
Examples
```
? hb_At( "cde", "abcdefgfedcba" )      // --> 3
? hb_At( "cde", "abcdefgfedcba", 4 )   // --> 0
```
Status

Ready
Compliance
This function is sensitive to `HB_CLP_STRICT` settings during build.
`nStart` and `nEnd` are Harbour extensions and do not exist if `HB_CLP_STRICT` is defined. In that case, the whole string is searched.
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`hb_RAt()`

`hb_AtI()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Locates the position of a substring in a main string.
Syntax
```
hb_AtI( <cSearch>, <cString>, [<nStart>], [<nEnd>] ) → nPos
```
Arguments
`cSearch` the sub-string to search for
`cString` The main string to search into, for `cSearch`
`nStart` Beginning search position into `cString`, default: 1
`nEnd` Ending search position, default: Length of `cString` (i.e. entire `cString`)
Returns
`hb_AtI()` returns the position (if any), into main string, where first time the substring appears.
Description

This function has same functionality as `hb_At()` with the significant difference that it's case Insensitive.

Optionally, with `nStart` can be defined the position into main string from where the search of `cSearch` must begin and with `nEnd` the position where it must stop. If neither of them is defined, `nStart` is 1st position and `nEnd` the ending of `cString`.

Examples
```
? hb_At( "AS", "as simple as possible", 5 )   // --> 0
? hb_AtI( "AS", "as simple as possible", 5 )  // --> 11
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`hb_At()`

hb_ATokens() Source code in Repository in Discussions ^ | ☰ | Improve this doc

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

Syntax

```
hb_ATokens( <cString>, [<cDelim>|<lEOL>], [<lSkipStrings>], ;
            [<lDoubleQuoteOnly>] ) → aTokens
```

Arguments

`cString` Complex string to be parsed.

`cDelim`|`lEOL Character(s) used as delimiter of separate tokens.

If `lEOL` flag defined instead of `cDelim`, then end of line marker(s) will be used as delimiter.

`lSkipStrings` Boolean flag indicating if quoted substrings will be tokenized or not.

`lDoubleQuoteOnly` Boolean flag indicating that only double-quoted substrings will be tokenized.

Returns

`aTokens` A character array, filled with the individual tokens found.

Description

This function analyses the complex string `cString` and splits it into separate sub-strings (tokens) that are delimited by `cDelim` or by space character, if no `cDelim` delimiter is passed, or by EOL marker if `lEOL` instead of `cDelim` is specified.

The located tokens, are stored in an array which is returned by the function.

If `lSkipStrings` is `.T.` (default: `.F.`), the quoted sub-strings (if any) within `cString` are not tokenized. If `lDoubleQuoteOnly` is `.T.` only the double quote `"` is recognized as a quote sign. This argument is meaningful only when `lSkipStrings` is `.T.`

Notes

1) the tokenization process is case sensitive, in the (rare) case

where `cDelim` is an alphabetic character.

2) The delimiters are removed (trimmed) from tokens.

Examples

```
LOCAL cString := "Harbour is proven to be stable, robust and efficient."
LOCAL aTokens := hb_ATokens( cString )
AEval( aTokens, {| token, n | QOut( hb_ntos(n), token ) } )
?
aTokens := hb_ATokens( cString, "," )
AEval( aTokens, {| token, n | QOut( hb_ntos(n), token ) } )
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags
API, Strings
See also
SubStr() Clipper 5.3

hb_LeftEq() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if a sub-string matches to leftmost part of a string.
Syntax
```
hb_LeftEq( <cString>, <cSubString> ) → lMatch
```
Arguments
`cString` Main string of comparison.
`cSubString` Sub-string compared to leftmost part of `cString`
Returns
`lMatch` Boolean flag indicating if the matching was successful
Description

This function checks if all characters (one by one and with the given order) of `cSubString` match to leftmost (same length) part of `cString` or in other words, checks if `cString` starts with `cSubString`, in which case returns `.T.`, otherwise `.F.`

Basically it's equivalent to the expression: `Left( <cString>, Len( <cSubString> ) ) == <cSubString>` but faster and shorter.

NOTE: Case sensitive!

Examples
```
? hb_LeftEq( "Harbour", "Ha" )  // --> .T.
? hb_LeftEq( "Harbour", "ha" )  // --> .F.
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
At() Clipper 5.3, hb_LeftEqI(), Left() Clipper 5.3

hb_LeftEqI() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if a sub-string matches to leftmost part of a string.
Syntax
```
hb_LeftEqI( <cString>, <cSubString> ) → lMatch
```
Arguments
`cString` Main string of comparison.
`cSubString` Sub-string compared to leftmost part of `cString`.
Returns

lMatch Boolean flag indicating if the matching was successful.
Description
This function is identical to hb_LeftEq() (see above for details) but it is case *insensitive*!
Examples
```
? hb_LeftEqI( "Harbour", "HA" )  // --> .T.
? hb_LeftEqI( "Harbour", "ha" )  // --> .T.
? hb_LeftEq( "Harbour", "HA" )   // --> .F.
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
At() Clipper 5.3, hb_LeftEqI(), Left() Clipper 5.3

hb_MemoRead() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the text file's contents as a character string
Syntax
```
hb_MemoRead( <cFileName> ) → cString
```
Arguments
cFileName is the file name to read from disk.
It must include the file extension. If file to be read lives in another directory, you must include the path.
Returns
Returns the contents of a text file as a character string.
If cFileName cannot be found or read hb_MemoRead() returns an empty string ("").
Description

hb_MemoRead() is a function that reads the content of a text file (till now) from disk (floppy, HDD, CD-ROM, etc.) into a memory string. In that way you can manipulate as any character string or assigned to a memo field to be saved in a database.

hb_MemoRead() function is used together with MemoEdit() and hb_MemoWrit() to get from disk text from several sources that would be edited, searched, replaced, displayed, etc.

It is used to import data from other sources to our database.

Note: hb_MemoRead() does not use the settings SET DEFAULT or SET PATH to search for cFileName. It searches for cFileName in the current directory.

Over a network, hb_MemoRead() attempts to open cFileName in read-only mode and shared. If the file is used in mode exclusive by another process, the function will returns a null string ("").

`hb_MemoRead()` vs `MemoRead()`: `hb_MemoRead()` is identical to `MemoRead()` except it won't truncate the last byte (on non-Unix compatible systems) if it's a EOF char.

Examples

```
// This example uses hb_MemoRead() to assign the contents of a text
// file to a character variable for later search

LOCAL cFile := "account.prg"
LOCAL cString := hb_MemoRead( cFile )
LOCAL cCopyright := "Melina"

IF ! cCopyright $ cString                          // check for copyright
   ? hb_MemoWrit( cFile, cCopyright + cString )  // if not, add it!
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`hb_MemoWrit()`, `MemoEdit()`, `MemoRead()` Clipper 5.3, `REPLACE`

`hb_MemoWrit()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Write a memo field or character string to a text file on disk
Syntax

```
hb_MemoWrit( <cFileName>, <cString> ) → lSuccess
```

Arguments
`cFileName` is the file name to be written to disk.
It must include the file extension. If file to be read lives in another directory, you must include the path.
`cString` Is the memo field or character string, to be write to `cFile`.
Returns
Function returns true (`.T.`) if the writing operation was successful; otherwise, it returns false (`.F.`).
Description

This a function that writes a memo field or character string to a text file on disk (floppy, HDD, CD-ROM, etc.) If you not specified a path, `hb_MemoWrit()` writes `cFileName` to the current directory. If `cFileName` exists, it is overwritten.

`hb_MemoWrit()` function is used together with `hb_MemoRead()` and `MemoEdit()` to save to disk text from several sources that was edited, searched, replaced, displayed, etc.

Note that `hb_MemoWrit()` do not use the directory settings `SET DEFAULT`.

`hb_MemoWrit()` vs `MemoWrit()`: `hb_MemoWrit()` never writes the obsolete EOF char at the end of the file.

Examples

```
//  This example uses hb_MemoWrit() to write the contents of a character
//  variable to a text file.

LOCAL cFile := "account.prg"
LOCAL cString := hb_MemoRead( cFile )
LOCAL cCopyright := "Melina"

IF ! cCopyright $ cString                          // check for copyright
   ? hb_MemoWrit( cFile, cCopyright + cString )  // if not, add it!
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`hb_MemoWrit()`, `MemoEdit()`, `MemoRead()` Clipper 5.3

`hb_ntoc()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts a numeric value to string
Syntax

```
hb_ntoc( <nValue>, [<nDecs>] ) → cValue
```

Arguments
`nValue` is the numeric value to convert.
`nDecs` decimal digits to retain (if any).
Returns
`cValue` A string representation of `nValue`
Description

This function converts the given `nValue` numeric value to a string value, while (trying to) keep all or at least `nDecs` significant digits in double numbers, unless `<nDecs>` is lesser than actual decimal digits of `nValue`, in which case the result will be rounded.

`SET DECIMAL` setting has no effect on the returned value (ignored), which means that, unlike e.g. `Str()`, all non-significant digits (e.g.: trailing decimal zeros) will be removed. Likewise, all leading empty spaces will be trimmed.

Returns stringified value of `<nValue>`, preserving all (or at least `<nDecs>`) significant digits, if any. Interestingly, if `<nValue>` is `NIL` or not numeric, this function will return a null string and, unlike `Str()`, will NOT cause an RTE. NOTE: new function, available after 2016-06-20 21:59 UTC+0200 commit,

(it is not available in earlier versions).

Examples

```
LOCAL n := ( 5 / 2 ) + 0.009

? hb_ntoc( n )      // --> 2.509
? Str( n )          // -->          2.51
? hb_ntoc( n, 2 )   // --> 2.51
? Str( n, 5, 2 )    // -->  2.51
? hb_ntos( n )      // --> 2.51
? "--- decimals set to 7 ----"
SET DECIMALS TO 7
? Str( n )          // -->          2.51
? hb_ntoc( n )      // --> 2.509
? Str( n, 10, 7 )   // -->  2.5090000
? hb_ntoc( n, 7 )   // --> 2.509
? "--- pass non numeric / NIL value ----"
? Str( "42" )       // --> RTE
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags

API, Strings

See also

`hb_ntos()`, `Str()` Clipper 5.3

`hb_ntos()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts a numeric value to string.

Syntax

```
hb_ntos( <nValue> ) → cValue
```

Arguments

`nValue` is the numeric value to convert.

Returns

`cValue` A string representation of `nValue`

Description

This function converts any numeric value to a string, trimming all the leading empty spaces. If `<nValue>` is `NIL` or not numeric, this function will return a null string.

Essentially, `hb_ntos()` function is equivalent to `LTrim( Str( <nValue> ) )` but quite simpler and faster.

Examples

```
LOCAL n := ( 5 / 2 ) + 0.009

? Str( n )       // -->          2.51
? hb_ntos( n )   // --> 2.51
```

Status

Ready

Compliance

Harbour specific

Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
hb_ntoc(), LTrim() Clipper 5.3, Str() Clipper 5.3

hb_RAt() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Searches for last occurrence a substring of a string.
Syntax
```
hb_RAt( <cSearch>, <cString>, [<nStart>], [<nEnd>] ) → nPos
```
Arguments
cSearch Substring to search for
cString Main string
nStart First position to search in cString, by default 1.
nEnd End position to search, by default cString length
Returns
hb_RAt() return the location of beginning position of last occurrence a substring of a string.
Description
This function searches for last occurrence a cSearch in cString. If the function is unable to find any occurrence of cSearch in cString, the return value is 0. 3rd and 4th parameters define inclusive range for 2nd parameter on which operation is performed. If 3rd and 4th parameters is not specified, then hb_RAt() is equal to RAt().
Examples
```
LOCAL cString := "acdefcdeedcb"
LOCAL cSearch := "cde"
LOCAL i, y, r
LOCAL nLen := Len( cString )

FOR y := 1 TO nLen
   FOR i := 1 TO nLen
      IF ( r := hb_RAt( cSearch, cString, y, i ) ) > 0
         ? 'hb_RAt( "' + cSearch + '", "' + cString + '",', hb_ntos( y ) +
",", hb_ntos( i ), ") =", ;
            hb_ntos( r )
      ENDIF
   NEXT
NEXT

? hb_RAt( cSearch, "abcdefgfedcba" )  // --> 3
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also

hb_At(), RAt() Clipper 5.3, Right() Clipper 5.3, SubStr() Clipper 5.3

hb_ValToStr() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts any scalar type to a string.
Syntax
```
hb_ValToStr( <xValue> ) → cString
```
Arguments
`xValue` is any scalar argument.
Returns
`cString` A string representation of `xValue` using default conversions.
Description
`hb_ValToStr()` can be used to convert any scalar value to a string.
Examples
```
Set( _SET_DATEFORMAT, "yyyy-mm-dd" )
? hb_ValToStr( 4 ) == "          4"
? hb_ValToStr( 4.0 / 2 ) == "          2.00"
? hb_ValToStr( "String" ) == "String"
? hb_ValToStr( 0d20010101 ) == "2001-01-01"
? hb_ValToStr( NIL ) == "NIL"
? hb_ValToStr( .F. ) == ".F."
? hb_ValToStr( .T. ) == ".T."
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
Str() Clipper 5.3

HardCR() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Replace all soft carriage returns with hard carriages returns.
Syntax
```
HardCR( <cString> ) → cConvertedString
```
Arguments
`cString` is a string of chars to convert.
Returns
`cConvertedString` Transformed string.
Description
Returns a string/memo with soft carriage return chars converted to hard carriage return chars.
Examples
```
// FIXME
? HardCR( data->CNOTES )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`MemoTran()` Clipper 5.3, `StrTran()` Clipper 5.3

`IsAlpha()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if leftmost character in a string is an alphabetic character
Syntax
```
IsAlpha( <cString> ) → lAlpha
```
Arguments
`cString` Any character string
Returns
`lAlpha` Logical true (`.T.`) or false (`.F.`).
Description
This function return a logical true (`.T.`) if the first character in `cString` is an alphabetic character. If not, the function will return a logical false (`.F.`).
Examples
```
? IsAlpha( "hello" )
? IsAlpha( "12345" )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`IsDigit()` Clipper 5.3, `IsLower()` Clipper 5.3, `IsUpper()` Clipper 5.3, `Lower()` Clipper 5.3, `Upper()` Clipper 5.3

`IsDigit()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if leftmost character is a digit character
Syntax
```
IsDigit( <cString> ) → lDigit
```
Arguments
`cString` Any character string
Returns
`lDigit` Logical true (`.T.`) or false (`.F.`).
Description
This function takes the character string `cString` and checks to see if the leftmost character is a digit, from 1 to 9. If so, the function will return a logical true (`.T.`); otherwise, it will return a logical false (`.F.`).

Examples
```
? IsDigit( "12345" )  // --> .T.
? IsDigit( "abcde" )  // --> .F.
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
IsAlpha() Clipper 5.3, IsLower() Clipper 5.3, IsUpper() Clipper 5.3, Lower() Clipper 5.3, Upper() Clipper 5.3

IsLower() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if leftmost character is an lowercased letter.
Syntax
```
IsLower( <cString> ) → lLower
```
Arguments
`cString` Any character string
Returns
`lLower` Logical true (`.T.`) or false (`.F.`).
Description
This function takes the character string `cString` and checks to see if the leftmost character is a lowercased letter. If so, the function will return a logical true (`.T.`); otherwise, it will return a logical false (`.F.`).
Examples
```
? IsLower( "ABCde" )  // --> .F.
? IsLower( "aBCde" )  // --> .T.
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
IsAlpha() Clipper 5.3, IsDigit() Clipper 5.3, IsUpper() Clipper 5.3, Lower() Clipper 5.3, Upper() Clipper 5.3

IsUpper() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if leftmost character is an uppercased letter.
Syntax
```
IsUpper( <cString> ) → lUpper
```

Arguments
`cString` Any character string
Returns
`lUpper` Logical true (`.T.`) or false (`.F.`).
Description
This function checks to see if the leftmost character if `cString` is a uppercased letter. If so, the function will return a logical true (`.T.`); otherwise, it will return a logical false (`.F.`).
Examples
```
? IsUpper( "Abcde" )  // --> .T.
? IsUpper( "abcde" )  // --> .F.
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`IsAlpha()` Clipper 5.3, `IsDigit()` Clipper 5.3, `IsLower()` Clipper 5.3, `Lower()` Clipper 5.3, `Upper()` Clipper 5.3

`Left()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Extract the leftmost substring of a character expression
Syntax
```
Left( <cString>, <nLen> ) → cReturn
```
Arguments
`cString` Main character to be parsed
`nLen` Number of bytes to return beginning at the leftmost position
Returns
`cReturn` Substring of evaluation
Description

This functions returns the leftmost `nLen` characters of `cString`. It is equivalent to the following expression:

```
SubStr( <cString>, 1, <nLen> )
```
Examples
```
? Left( "Hello Harbour", 5 )  // --> "Hello"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings

See also
`At()` Clipper 5.3, `RAt()` Clipper 5.3, `Right()` Clipper 5.3, `SubStr()` Clipper 5.3

`Lower()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Universally lowercases a character string expression.
Syntax
```
Lower( <cString> ) → cLowerString
```
Arguments
`cString` Any character expression.
Returns
`cLowerString` Lowercased value of `cString`
Description
This function converts any character expression passes as `cString` to its lowercased representation. Any non alphabetic character withing `cString` will remain unchanged.
Examples
```
? Lower( "HARBOUR" )     // --> "harbour"
? Lower( "Hello All" )   // --> "hello all"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`IsLower()` Clipper 5.3, `IsUpper()` Clipper 5.3, `Upper()` Clipper 5.3

`LTrim()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Removes leading spaces from a string
Syntax
```
LTrim( <cString> ) → cReturn
```
Arguments
`cString` Character expression with leading spaces
Returns
`LTrim()` returns a copy of the original string with leading spaces removed.
Description
This function trims the leading space blank
Examples
```
? "|" + LTrim( "Hello      " ) + "|"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, Strings
See also
AllTrim() Clipper 5.3, RTrim() Clipper 5.3, Trim() Clipper 5.3

MemoRead() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return the text file's contents as a character string
Syntax
```
MemoRead( <cFileName> ) → cString
```
Arguments
cFileName is the file name to read from disk.
It must include the file extension. If file to be read lives in another directory, you must include the path.
Returns
Returns the contents of a text file as a character string.
If cFileName cannot be found or read MemoRead() returns an empty string ("").
Description

MemoRead() is a function that reads the content of a text file (till now) from disk (floppy, HDD, CD-ROM, etc.) into a memory string. In that way you can manipulate as any character string or assigned to a memo field to be saved in a database.

MemoRead() function is used together with MemoEdit() and MemoWrit() to get from disk text from several sources that would be edited, searched, replaced, displayed, etc.

It is used to import data from other sources to our database.

Note: MemoRead() does not use the settings SET DEFAULT or SET PATH to search for cFileName. It searches for cFileName in the current directory.

Over a network, MemoRead() attempts to open cFileName in read-only mode and shared. If the file is used in mode exclusive by another process, the function will returns a null string ("").

Examples
```
// This example uses MemoRead() to assign the contents of a text
// file to a character variable for later search

LOCAL cFile := "account.prg"
LOCAL cString := MemoRead( cFile )
LOCAL cCopyright := "Melina"

IF ! cCopyright $ cString                      // check for copyright
   ? MemoWrit( cFile, cCopyright + cString )   // if not, add it!
ENDIF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File

Library is core
Tags
API, Strings
See also
hb_MemoRead(), MemoEdit(), MemoWrit() Clipper 5.3, REPLACE

MemoTran() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Converts hard and soft carriage returns within strings.
Syntax
```
MemoTran( <cString>, <cHard>, <cSoft> ) → cConvertedString
```
Arguments
cString is a string of chars to convert.
cHard is the character to replace hard returns with. If not specified defaults to semicolon.
cSoft is the character to replace soft returns with. If not specified defaults to single space.
Returns
cConvertedString Transformed string.
Description
Returns a string/memo with carriage return chars converted to specified chars.
Examples
```
// FIXME
? MemoTran( data->CNOTES )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
HardCR() Clipper 5.3, StrTran() Clipper 5.3

MemoWrit() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Write a memo field or character string to a text file on disk
Syntax
```
MemoWrit( <cFileName>, <cString> ) → lSuccess
```
Arguments
cFileName is the file name to be written to disk.
It must include the file extension. If file to be read lives in another directory, you must include the path.
cString Is the memo field or character string, to be write to
cFile.
Returns
Function returns true (.T.) if the writing operation was successful; otherwise, it returns false (.F.).
Description

This a function that writes a memo field or character string to a text file on disk (floppy, HDD, CD-ROM, etc.) If you not specified a path, `MemoWrit()` writes `cFileName` to the current directory. If `cFileName` exists, it is overwritten.

`MemoWrit()` function is used together with `MemoRead()` and `MemoEdit()` to save to disk text from several sources that was edited, searched, replaced, displayed, etc.

Note that `MemoWrit()` do not use the directory settings `SET DEFAULT`.

Examples

```
//  This example uses MemoWrit() to write the contents of a character
// variable to a text file.

LOCAL cFile := "account.prg"
LOCAL cString := MemoRead( cFile )
LOCAL cCopyright := "Melina"

IF ! cCopyright $ cString                    // check for copyright
   ? MemoWrit( cFile, cCopyright + cString )  // if not, add it!
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`hb_MemoWrit()`, `MemoEdit()`, `MemoRead()` Clipper 5.3

`PadC()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Centers an expression for a given width
Syntax

```
PadC( <xVal>, <nWidth>, <cFill> ) → cString
```

Arguments
`xVal` A Number, Character or Date value to pad
`nWidth` Width of output string
`cFill` Character to fill in the string
Returns
`cString` The Center string of `xVal`
Description

This function takes an date, number or character expression `xVal` and attempt to center the expression within a string of a given width expressed as `nWidth`. The default character used to pad either side of `xVal` will be a blank space. This character may be explicitly specified the value of `cFill`.

If the length of `xVal` is longer then `nWidth`, this function will truncate the string `xVal` from the leftmost side to the length of `nWidth`.

Examples

```
? PadC( "Harbour", 20 )
? PadC( 34.5142, 20 )
? PadC( Date(), 35 )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`AllTrim()` Clipper 5.3, `PadL()`, `PadR()`

`PadL()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Left-justifies an expression for a given width
Syntax

```
PadL( <xVal>, <nWidth>, <cFill> ) → cString
```

Arguments
`xVal` An number, Character or date to pad
`nWidth` Width of output string
`cFill` Character to fill in the string
Returns
`cString` The left-justifies string of `xVal`
Description

This function takes an date, number, or character expression `xVal` and attempt to left-justify it within a string of a given width expressed as `nWidth`. The default character used to pad left side of `xVal` will be an blank space; however, this character may be explicitly specified the value of `cFill`.

If the length of `xVal` is longer then `nWidth`, this function will truncate the string `xVal` from the leftmost side to the length of `nWidth`.

Examples

```
? PadL( "Harbour", 20 )
? PadL( 34.5142, 20 )
? PadL( Date(), 35 )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings

See also
AllTrim() Clipper 5.3, PadC(), PadR()

PadR() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Right-justifies an expression for a given width
Syntax
```
PadR( <xVal>, <nWidth>, <cFill> ) → cString
```
Arguments
`xVal` A Number, Character or Date value to pad
`nWidth` Width of output string
`cFill` Character to fill in the string
Returns
`cString` The right-justifies string of `xVal`
Description

This function takes an date, number, or character expression `xVal` and attempt to right-justify it within a string of a given width expressed as `nWidth`. The default character used to pad right side of `xVal` will be an blank space; however, this character may be explicitly specified the value of `cFill`.

If the length of `xVal` is longer then `nWidth`, this function will truncate the string `xVal` from the leftmost side to the length of `nWidth`.

Examples
```
? PadR( "Harbour", 20 )
? PadR( 34.5142, 20 )
? PadR( Date(), 35 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
AllTrim() Clipper 5.3, PadC(), PadL()

RAt() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Searches for last occurrence a substring of a string.
Syntax
```
RAt( <cSearch>, <cString> ) → nPos
```
Arguments
`cSearch` Substring to search for
`cString` Main string
Returns
`RAt()` return the location of beginning position of last occurrence a substring of a string.

Description
This function searches for last occurrence a `cSearch` in `cString`. If the function is unable to find any occurrence of `cSearch` in `cString`, the return value is 0.
Examples
```
? RAt( "cde", "abcdefgfcdeedcba" )  // --> 9
? RAt( "cdr", "abcdefgfedcba" )     // --> 0
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`At()` Clipper 5.3, `hb_RAt()`, `Right()` Clipper 5.3, `SubStr()` Clipper 5.3

`Replicate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Repeats a single character expression
Syntax
```
Replicate( <cString>, <nSize> ) → cReplicateString
```
Arguments
`cString` Character string to be replicated
`nSize` Number of times to replicate `cString`
Returns
`cReplicateString` A character expression contain the `cString` fill character.
Description

This function returns a string composed of `nSize` repetitions of `cString`. The length of the character string returned by this function is limited to the memory available.

A value of 0 for `nSize` will return a null string.

Examples
```
? Replicate( "a", 10 )      // --> "aaaaaaaaaa"
? Replicate( "b", 100000 )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`PadC()`, `PadL()`, `PadR()`, `Space()` Clipper 5.3

Right() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Extract the rightmost substring of a character expression
Syntax
```
Right( <cString>, <nLen> ) → cReturn
```
Arguments
`cString` Character expression to be parsed
`nLen` Number of bytes to return beginning at the rightmost position
Returns
`cReturn` Substring of evaluation
Description

This functions returns the rightmost `nLen` characters of `cString`. It is equivalent to the following expressions:

```
SubStr( <cString>, -<nLen> )
SubStr( <cString>, Len( <cString> ) - <nLen> + 1, <nLen> )
```
Examples
```
? Right( "Hello Harbour", 5 )  // --> "rbour"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`At()` Clipper 5.3, `Left()` Clipper 5.3, `RAt()` Clipper 5.3, `SubStr()` Clipper 5.3

RTrim() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Remove trailing spaces from a string.
Syntax
```
RTrim( <cExpression> ) → cString
```
Arguments
`cExpression` Any character expression
Returns
`cString` A formatted string with out any blank spaced.
Description

This function returns the value of `cString` with any trailing blank removed.

This function is identical to `RTrim()` and the opposite of `LTrim()`. Together with `LTrim()`, this function equated to the `AllTrim()` function.

Examples
```
? RTrim( "Hello" )    // --> "Hello"
? RTrim( "" )         // --> ""
? RTrim( "UA    " )   // --> "UA"
```

```
? RTrim( "   UA" )   // --> "   UA"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`AllTrim()` Clipper 5.3, `LTrim()` Clipper 5.3, `Trim()` Clipper 5.3

`Space()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a string of blank spaces
Syntax
`Space( <nSize> ) → cString`
Arguments
`nSize` The length of the string
Returns
`cString` A string containing blank spaces
Description

This function returns a string consisting of `nSize` blank spaces. If the value of `nSize` is 0, a null string ("") will be returned.

This function is useful to declare the length of a character memory variable.

Examples

```
#include "dbstruct.ch"

PROCEDURE Main()

   LOCAL cBigString
   LOCAL cFirst
   LOCAL cString := Space( 20 )    // Create an character memory variable
                                   // with length 20
   ? Len( cString )  // --> 20
   cBigString := Space( 100000 )  // Create a memory variable with 100000
                                   // blank spaces
   ? Len( cBigString )
   USE test NEW
   cFirst := MakeEmpty( 1 )
   ? Len( cFirst )

   RETURN

STATIC FUNCTION MakeEmpty( xField )

   LOCAL nRecord
   LOCAL xRetValue

   IF ! Alias() == ""
      nRecord := RecNo()
```

```
      dbGoto( 0 )
      IF HB_ISSTRING( xField )
         xField := AScan( dbStruct(), {| aFields | aFields[ DBS_NAME ] ==
Upper( xField ) } )
      ELSE
         hb_default( @xField, 0 )
         IF xField < 1 .OR. xField > FCount()
            xField := 0
         ENDIF
      ENDIF
      IF xField != 0
         xRetValue := FieldGet( xField )
      ENDIF
      dbGoto( nRecord )
   ENDIF

   RETURN xRetValue
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`PadC()`, `PadL()`, `PadR()`, `Replicate()` Clipper 5.3

`Str()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert a numeric expression to a character string.
Syntax
`Str( <nNumber>, [<nLength>], [<nDecimals>] ) → cNumber`
Arguments
`nNumber` is the numeric expression to be converted to a character string.
`nLength` is the length of the character string to return, including decimal digits, decimal point, and sign.
`nDecimals` is the number of decimal places to return.
Returns
`Str()` returns `nNumber` formatted as a character string. If the optional length and decimal arguments are not specified, `Str()` returns the character string according to the following rules:
Results of `Str()` with No Optional Arguments

Expression	Return Value Length
Field Variable	Field length plus decimals
Expressions/constants	Minimum of 10 digits plus decimals
Val()	Minimum of 3 digits
Month()/Day()	3 digits
Year()	5 digits
RecNo()	7 digits

Description

`Str()` is a numeric conversion function that converts numeric values to character strings. It is commonly used to concatenate numeric values to character strings. `Str()` has applications displaying numbers, creating codes such as part numbers from numeric values, and creating index keys that combine numeric and character data.

`Str()` is like `Transform()`, which formats numeric values as character strings using a mask instead of length and decimal specifications.

The inverse of `Str()` is `Val()`, which converts character numbers to numerics.

* If `nLength` is less than the number of whole number digits in

`nNumber`, `Str()` returns asterisks instead of the number.

* If `nLength` is less than the number of decimal digits

required for the decimal portion of the returned string, Harbour rounds the number to the available number of decimal places.

* If `nLength` is specified but `nDecimals` is omitted (no

decimal places), the return value is rounded to an integer.

Examples

```
? Str( 10, 6, 2 )    // --> " 10.00"
? Str( -10, 8, 2 )   // --> "  -10.00"
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`StrZero()`, `Transform()` Clipper 5.3, `Val()` Clipper 5.3

`StrTran()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Translate substring value with a main string
Syntax

```
StrTran( <cString>, <cLocString>, [<cRepString>], [<nPos>],
         [<nOccurrences>] ) → cReturn
```

Arguments
`cString` The main string to search
`cLocString` The string to locate in the main string
`cRepString` The string to replace the `cLocString`
`nPos` The first occurrence to be replaced
`nOccurrences` Number of occurrence to replace

Returns
`cReturn` Formatted string
Description

This function searches for any occurrence of `cLocString` in `cString` and replaces it with `cRepString`. If `cRepString` is not specified, a `NULL` byte will replace `cLocString`.

If `nPos` is used, its value defines the first occurrence to be replaced. The default value is 1. Additionally, if used, the value of `nOccurrences` tell the function how many occurrences of `cLocString` in `cString` are to the replaced. The default of `nOccurrences` is all occurrences.

Examples

```
? StrTran( "Harbour  Power", "  ", " " )  // --> "Harbour Power"
? StrTran( "Harbour  Power  The Future  of  xBase", "  ", " " , , 2 )
  // --> "Harbour Power The future  of  xBase"
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Libraty is rtl
Tags
API, Strings
See also
`At()` Clipper 5.3, `SubStr()` Clipper 5.3

`StrZero()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert a numeric expression to a character string, zero padded.
Syntax

```
StrZero( <nNumber>, [<nLength>], [<nDecimals>] ) → cNumber
```

Arguments
`nNumber` is the numeric expression to be converted to a character string.
`nLength` is the length of the character string to return, including decimal digits, decimal point, and sign.
`nDecimals` is the number of decimal places to return.
Returns
`StrZero()` returns `nNumber` formatted as a character string. If the optional length and decimal arguments are not specified, `StrZero()` returns the character string according to the following rules:
Results of `StrZero()` with No Optional Arguments

Expression	Return Value Length
Field Variable	Field length plus decimals
Expressions/constants	Minimum of 10 digits plus decimals
Val()	Minimum of 3 digits
Month()/Day()	3 digits
Year()	5 digits
RecNo()	7 digits

Description

`StrZero()` is a numeric conversion function that converts numeric values to character strings. It is commonly used to concatenate numeric values to character strings. `StrZero()` has applications displaying numbers, creating codes such as part numbers from numeric values, and creating index keys that combine numeric and character data.

`StrZero()` is like `Transform()`, which formats numeric values as character strings using a mask instead of length and decimal specifications.

The inverse of `StrZero()` is `Val()`, which converts character numbers to numerics.

* If `nLength` is less than the number of whole number digits in

`nNumber`, `Str()` returns asterisks instead of the number.

* If `nLength` is less than the number of decimal digits

required for the decimal portion of the returned string, Harbour rounds the number to the available number of decimal places.

* If `nLength` is specified but `nDecimals` is omitted (no

decimal places), the return value is rounded to an integer.

The `StrZero()` function was part of the CA-Cl*pper samples.

Examples

```
? StrZero( 10, 6, 2 )    // --> "010.00"
? StrZero( -10, 8, 2 )   // --> "-0010.00"
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`Str()` Clipper 5.3

`SubStr()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a substring from a main string
Syntax

```
SubStr( <cString>, <nStart>, [<nLen>] ) → cReturn
```

Arguments
`cString` Character expression to be parsed
`nStart` Start position
`nLen` Number of characters to return
Returns

`cReturn` Substring of evaluation
Description

This functions returns a character string formed from `cString`, starting at the position of `nStart` and continuing on for a length of `nLen` characters. If `nLen` is not specified, the value will be all remaining characters from the position of `nStart`.

The value of `nStart` may be negative. If it is, the direction of operation is reversed from a default of left-to-right to right-to-left for the number of characters specified in `nStart`. If the number of characters from `nStart` to the end of the string is less than `nLen` the rest are ignored.

Examples

```
? SubStr( "Hello Harbour",  7, 4 )  // --> "Harb"
? SubStr( "Hello Harbour", -3, 3 )  // --> "our"
? SubStr( "Hello Harbour",  7    )  // --> "Harbour"
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
`At()` Clipper 5.3, `Left()` Clipper 5.3, `Right()` Clipper 5.3

`Transform()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Formats a value based on a specific picture template.
Syntax

```
Transform( <xExpression>, <cTemplate> ) → cFormatted
```

Arguments
`xExpression` Any expression to be formatted.
`cTemplate` Character string with picture template
Returns
`cFormatted` Formatted expression in character format
Description

This function returns `xExpression` in the format of the picture expression passed to the function as `cTemplate`.

Their are two components that can make up `cTemplate` : a function string and a template string. Function strings are those functions that globally tell what the format of `xExpression` should be. These functions are represented by a single character precede by the @ symbol.

There are a couple of rules to follow when using function strings and template strings:

- First, a single space must fall between the function template

and the template string if they are used in conjunction with one another.

- Second, if both components make up the value of `cTemplate`, the

function string must precede the template string. Otherwise, the function string may appear with out the template string and vice versa.

The table below shows the possible function strings available with the `Transform()` function.

@B	Left justify the string within the format.
@C	Issue a CR after format is numbers are positive.
@D	Put dates in SET DATE format.
@E	Put dates in BRITISH format.
@L	Make a zero padded string out of the number.
@R	Insert non template characters.
@X	Issue a DB after format is numbers are negative.
@Z	Display any zero as blank spaces.
@(	Quotes around negative numbers
@!	Convert alpha characters to uppercased format.

The second part of `cTemplate` consists of the format string. Each character in the string may be formatted based on using the follow characters as template markers for the string.

A,N,X,9,#	Any data type
L	Shows logical as "T" or "F"
Y	Shows logical as "Y" or "N"
!	Convert to uppercase
$	Dollar sing in place of leading spaces in numeric expression
*	Asterisks in place of leading spaces in numeric expression
,	Commas position
.	Decimal point position

Examples

```
LOCAL cString := "This is harbour"
LOCAL nNumber := 9923.34
LOCAL nNumber1 := -95842.00
LOCAL lValue := .T.
LOCAL dDate := Date()
? "working with String"
? "Current String is", cString
? "All uppercased", Transform( cString, "@!" )
? "Date is", dDate
? "Date is", Transform( dDate, "@D" )
? Transform( nNumber, "@L 99999999" )  // --> "009923.34"
? Transform( 0      , "@L 9999"     )  // --> "0000"
```

Status

Ready

Compliance

The @L function template is a FoxPro/Xbase++ Extension

Platforms

Available on all platforms

File

Library is core

Tags

API, Strings
See also
@...SAYClipper 5.3, DevOutPict()Clipper 5.3

Trim()→ RTrim()Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Remove trailing spaces from a string.
Syntax
```
Trim( <cExpression> ) → cString
```
Arguments
cExpression Any character expression
Returns
cString A formatted string with out any blank spaced.
Description

This function returns the value of cString with any trailing blank removed.

This function is identical to RTrim() and the opposite of LTrim(). Together with LTrim(), this function equated to the AllTrim() function.

Examples
```
? Trim( "Hello" )     // --> "Hello"
? Trim( "" )          // --> ""
? Trim( "UA    " )     // --> "UA"
? Trim( "    UA" )     // --> "    UA"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
AllTrim()Clipper 5.3, LTrim()Clipper 5.3, RTrim()Clipper 5.3

Upper()Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Converts a character expression to uppercase format
Syntax
```
Upper( <cString> ) → cUpperString
```
Arguments
cString Any character expression.
Returns
cUpperString Uppercased value of cString
Description
This function converts all alpha characters in cString to upper case values and returns that formatted character expression.
Examples
```
? Upper( "harbour" )  // --> "HARBOUR"
```

```
? Upper( "Harbour" )  // --> "HARBOUR"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
IsLower() Clipper 5.3, IsUpper() Clipper 5.3, Lower() Clipper 5.3

Val() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Convert a number from a character type to numeric
Syntax
```
Val( <cNumber> ) → nNumber
```
Arguments
`cNumber` Any valid character string of numbers.
Returns
`nNumber` The numeric value of `cNumber`
Description

This function converts any number previously defined as an character expression `cNumber` into a numeric expression.

This functions is the oppose of the `Str()` function.

Examples
```
? Val( "31421" )  // --> 31421
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Strings
See also
Str() Clipper 5.3, Transform() Clipper 5.3

__TypeFile() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Show the content of a file on the console and/or printer
Syntax
```
__TypeFile( <cFile>, [<lPrint>] ) → NIL
```
Arguments

`cFile` is a name of the file to display. If the file have an extension, it must be specified (there is no default value).
`lPrint` is an optional logical value that specifies whether the output should go only to the screen (`.F.`) or to both the screen and printer (`.T.`), the default is (`.F.`).
Returns
`__TypeFile()` always return `NIL`.
Description

`__TypeFile()` function type the content of a text file on the screen with an option to send this information also to the printer. The file is displayed as is without any headings or formatting.

If `cFile` contain no path, `__TypeFile()` try to find the file first in the `SET DEFAULT` directory and then in search all of the `SET PATH` directories. If `cFile` cannot be found a run-time error occur.

Use `SET CONSOLE OFF` to suppress screen output. You can pause the output using Ctrl-S, press any key to resume.

`__TypeFile()` function is used in the preprocessing of the `TYPE` command.

Examples

```
// The following examples assume a file name `test.txt` exist in all
// specified paths, a run-time error would displayed if it does not

// display test.txt file on screen
__TypeFile( "test.txt" )

// display test.txt file on screen and printer
__TypeFile( "test.txt", .T. )

// display test.txt file on printer only
SET CONSOLE OFF
__TypeFile( "test.txt", .T. )
SET CONSOLE ON
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`COPY FILE`Clipper 5.3, `SET DEFAULT`Clipper 5.3, `SET PATH`Clipper 5.3, `SET PRINTER`Clipper 5.3, `TYPE`Clipper 5.3

`Col()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the current screen column position
Syntax

```
Col() → nPosition
```

Arguments
None.
Returns
`nPosition` Current column position
Description
This function returns the current cursor column position. The value for this function can range between 0 and `MaxCol()`.
Examples
```
? Col()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`MaxCol()` Clipper 5.3Clipper Tools 3, `MaxRow()` Clipper 5.3Clipper Tools 3, `Row()` Clipper 5.3

`DevOutPict()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Displays a value to a device using a picture template
Syntax
```
DevOutPict( <xExp>, <cPicture>, [<cColorString>] )
```
Arguments
`xExp` is any valid expression.
`cPicture` is any picture transformation that `Transform()` can use.
`cColorString` is an optional string that specifies a screen color to use in place of the default color when the output goes to the screen.
Description
Outputs any expression using a picture transformation instead of using the default transformation for the type of expression.
Examples
```
// Output a negative dollar amount using debit notation.
DevOutPict( -1.25, "@D$ 99,999.99 )  // --> $(     1.25)
```
Status
Ready
Compliance
`DevOutPict()` is mostly CA-Cl*pper compliant. Any differences are due to enhancements in the Harbour `Transform()` over CA-Cl*pper.
Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`DevOut()`, `Transform()` Clipper 5.3

hb_ColorIndex() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Extract one color from a full color-spec string.
Syntax
```
hb_ColorIndex( <cColorSpec>, <nIndex> ) → cColor
```
Arguments
`cColorSpec` is a color list
`nIndex` is the position of the color item to be extracted, the first position is the zero.
Returns
The selected color string, or if anything goes wrong, an empty string.
Description

CA-Cl*pper has a color spec string, which has more than one color in it, separated with commas.

This function will extract a given item from this list. You may use the manifest constants defined in `color.ch` to identify and extract common colors.

Examples
```
#include "color.ch"
? hb_ColorIndex( "W/N, N/W", CLR_ENHANCED )  // --> "N/W"
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`ColorSelect()`

MaxCol() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the maximum number of columns in the current video mode
Syntax
```
MaxCol() → nPosition
```
Arguments
None.
Returns
`nPosition` The maximum number of columns possible in current video mode
Description
This function returns the current cursor column position. The value for this function can range between 0 and `MaxCol()`.
Examples
```
? MaxCol()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible

Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`Col()` Clipper 5.3, `MaxRow()` Clipper 5.3Clipper Tools 3, `Row()` Clipper 5.3

`MaxRow()` Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Returns the current screen row position
Syntax
```
MaxRow() → nPosition
```
Arguments
None.
Returns
`nPosition` The maximum number of rows possible in current video mode
Description
This function returns the current cursor row location. The value for this function can range between 0 and `MaxCol()`.
Examples
```
? MaxRow()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Terminal
See also
`Col()` Clipper 5.3, `MaxCol()` Clipper 5.3Clipper Tools 3, `Row()` Clipper 5.3

`RESTORE SCREEN`Source code^ | ☰ | Improve this doc

Restore screen image and coordinate from an internal buffer
Syntax
```
RESTORE SCREEN
```
Arguments
none.
Description

`RESTORE SCREEN` restores saved image of the whole screen from an internal buffer that was saved by `SAVE SCREEN`, it also restore cursor position. After a call to 'RESTORE SCREEN` the internal buffer is cleared.

`RESTORE SCREEN` command is preprocessed into `__XRestScreen()` function during compile time. Note that `RESTORE SCREEN FROM` is preprocessed into `RestScreen()` function.

Examples

```
// save the screen, display list of files than restore the screen
SAVE SCREEN
DIR *.*
RESTORE SCREEN
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

API, Terminal

See also

__XRestScreen(), __XSaveScreen(), SAVE SCREENClipper 5.3

Row() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Returns the current screen row position

Syntax

```
Row() → nPosition
```

Arguments

None.

Returns

nPosition Current screen row position

Description

This function returns the current cursor row location. The value for this function can range between 0 and MaxCol().

Examples

```
? Row()
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

Library is core

Tags

API, Terminal

See also

Col() Clipper 5.3, MaxCol() Clipper 5.3Clipper Tools 3, MaxRow() Clipper 5.3Clipper Tools 3

SAVE SCREENSource code^ | ☰ | Improve this doc

Save whole screen image and coordinate to an internal buffer

Syntax

```
SAVE SCREEN
```

Arguments

none.

Description

SAVE SCREEN saves the image of the whole screen into an internal buffer, it also save current cursor position. The information could later be restored by RESTORE SCREEN. Each call to SAVE SCREEN overwrite the internal buffer.

SAVE SCREEN command is preprocessed into __XSaveScreen() function during compile time. Note that SAVE SCREEN TO is preprocessed into SaveScreen() function.

Examples

```
// save the screen, display list of files than restore the screen
SAVE SCREEN
DIR *.*
RESTORE SCREEN
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

API, Terminal

See also

__XRestScreen(), __XSaveScreen(), RESTORE SCREENClipper 5.3

__AtPrompt() Source code□in Repositoryin Discussions ^ | ☰ | Improve this doc

Display a menu item on screen and define a message

Syntax

```
__AtPrompt( <nRow>, <nCol>, <cPrompt>, [<xMsg>] ) → .F.
```

Arguments

nRow is the row number to display the menu cPrompt. Value could range from zero to MaxRow().

nCol is the column number to display the menu cPrompt. Value could range from zero to MaxCol().

cPrompt is the menu item character string to display.

xMsg define a message to display each time this menu item is highlighted. xMsg could be a character string or code block that is evaluated to a character string. If xMsg is not specified or of the wrong type, an empty string ("") would be used.

Returns

__AtPrompt() always return .F.

Description

With __AtPrompt() you define and display a menu item, each call to __AtPrompt() add another item to the menu, to start the menu itself you should call the __MenuTo() function (MENU TO command). You can define any row and column combination and they will be displayed at the order of definition. After each call to __AtPrompt(), the cursor is placed one column to the right of the last text displayed, and Row() and Col() are updated.

@...PROMPT command is preprocessed into __AtPrompt() function during compile time.

Examples

```
LOCAL nChoice
// display a two line menu with status line at the bottom
```

```
// let the user select favorite day
SET MESSAGE TO 24 CENTER
@ 10, 2 PROMPT "Sunday" MESSAGE "This is the 1st item"
@ 11, 2 PROMPT "Monday" MESSAGE "Now we're on the 2nd item"
MENU TO nChoice
DO CASE
CASE nChoice == 0           // user press <Esc> key
   QUIT
CASE nChoice == 1           // user select 1st menu item
   ? "Guess you don't like Mondays"
CASE nChoice == 2           // user select 2nd menu item
   ? "Just another day for some"
ENDCASE
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
__MenuTo(), AChoice() Clipper 5.3, MENU TO Clipper 5.3, SET INTENSITY Clipper 5.3, SET MESSAGE Clipper 5.3, SET WRAP Clipper 5.3

__Input() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Stops application
Syntax
```
__Input( <cMessage> ) → cString
```
Arguments
`cMessage` is any valid expression.
Returns
`cString` input value macroed
Description
This function waits for a console input and returns macroed expression entered.
Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
__Accept(), __Wait()

__Keyboard() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Use hb_keyPut() instead
Syntax
```
KEYBOARD <cString>
CLEAR TYPEAHEAD
```
Arguments
`cString` is the optional string to stuff into the Harbour keyboard buffer after clearing it first. Note: The character `;` is converted to `Chr( 13 )` (this is an undocumented CA-Cl*pper feature).
Description
Clears the Harbour keyboard type-ahead buffer and then inserts an optional string into it.
Examples
```
// Stuff an Enter key into the keyboard buffer
KEYBOARD Chr( 13 )
// Clear the keyboard buffer
CLEAR TYPEAHEAD

KEYBOARD Chr( 13 ); ? Inkey()  // --> 13
KEYBOARD ";"; ? Inkey()  // --> 13
KEYBOARD "Hello"; CLEAR TYPEAHEAD; ? Inkey()  // --> 0
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
`CLEAR TYPEAHEAD`, `KEYBOARD`Clipper 5.3

`__MenuTo()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Invoked a menu defined by set of `@...PROMPT`
Syntax
```
__MenuTo( <bBlock>, <cVariable> ) → nChoice
```
Arguments
`bBlock` is a set/get code block for variable named `cVariable`.
`cVariable` is a character string that contain the name of the variable to hold the menu choices, if this variable does not exist a `PRIVATE` variable with the name `cVariable` would be created to hold the result.
Returns
`__MenuTo()` return the number of select menu item, or 0 if there was no item to select from or if the user pressed the Esc key.
Description

`__MenuTo()` invoked the menu define by previous `__AtPrompt()` call and display a highlight bar that the user can move to select an option from the menu. If `cVariable` does not exist or not visible, a `PRIVATE` variable named `cVariable` is created and hold the current menu selection. If there is a variable named `cVariable`, its value is used to select the first highlighted item.

Menu prompts and messages are displayed in current Standard color, highlighted bar is displayed using current Enhanced color.

Pressing the arrow keys move the highlighted bar. When a menu item is highlighted the message associated with it is displayed on the line specified with `SET MESSAGE`. If `SET WRAP` is `ON` and the user press up arrow while on the first selection the last menu item is highlighted, if the user press Down arrow while on the last item, the first item is highlighted.

Following are active keys that handled by `__MenuTo()`:

key	Meaning
Up	Move to previous item
Down	Move to next item
Left	Move to previous item
Right	Move to next item
Home	Move to the first item
End	Move to the last item
PgUp	Select menu item, return position
PgDn	Select menu item, return position
Enter	Select menu item, return position
Esc	Abort selection, return 0
First letter	Select next menu with the same first letter, return this item position.

upon exit the cursor is placed at `MaxRow()` - 1, 0 `__MenuTo()` can be nested without loosing the previous prompts.

`MENU TO` command is preprocessed into `__MenuTo()` function during compile time.

Examples

```
LOCAL nChoice
// display menu item on each screen corner and let user select one
CLS
SET MESSAGE TO MaxRow() / 2 CENTER
SET WRAP ON
@ 0           , 0             PROMPT "1. Upper left"   MESSAGE " One "
@ 0           , MaxCol() - 16 PROMPT "2. Upper right"  MESSAGE " Two "
@ MaxRow() - 1, MaxCol() - 16 PROMPT "3. Bottom right" MESSAGE "Three"
@ MaxRow() - 1, 0             PROMPT "4. Bottom left"  MESSAGE "Four "
MENU TO nChoice
SetPos( MaxRow() / 2, MaxCol() / 2 - 10 )
IF nChoice == 0
   ?? "<Esc> was pressed"
ELSE
   ?? "Selected option is", nChoice
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core

Tags
API, User interface
See also
__AtPrompt(), @...PROMPTClipper 5.3, AChoice()Clipper 5.3, SET INTENSITYClipper 5.3, SET MESSAGEClipper 5.3, SET WRAPClipper 5.3

__NoNoAlert() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Override //NOALERT command-line switch
Syntax
```
__NoNoAlert()
```
Arguments
This function takes no arguments.
Description
The //NOALERT command-line switch cause Cl*pper to ignore calls to the Alert() function, this function override this behavior and always display Alert() dialog box.
Examples
```
// make sure alert are been displayed
__NoNoAlert()
```
Status
Ready
Compliance
Undocumented CA-Cl*pper v5.x function available in builds with HB_CLP_UNDOC option enabled (default)
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface

__XRestScreen() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Restore screen image and coordinate from an internal buffer
Syntax
```
__XRestScreen()
```
Arguments
none.
Description

__XRestScreen() restores saved image of the whole screen from an internal buffer that was saved by __XSaveScreen(), it also restore cursor position. After a call to __XRestScreen() the internal buffer is cleared.

RESTORE SCREEN command is preprocessed into __XRestScreen() function during compile time. Note that RESTORE SCREEN FROM is preprocessed into RestScreen() function.

__XRestScreen() is a compatibility function, it is superseded by RestScreen() which allow you to restore the screen from a variable.

Examples

```
// save the screen, display list of files than restore the screen
SAVE SCREEN
DIR *.*
RESTORE SCREEN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
__XRestScreen(), __XSaveScreen(), SAVE SCREENClipper 5.3

__XSaveScreen() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Save whole screen image and coordinate to an internal buffer
Syntax
`__XSaveScreen()`
Arguments
none.
Description

`__XSaveScreen()` saves the image of the whole screen into an internal buffer, it also save current cursor position. The information could later be restored by `__XRestScreen()`. Each call to `__XSaveScreen()` overwrite the internal buffer.

`SAVE SCREEN` command is preprocessed into `__XSaveScreen()` function during compile time. Note that `SAVE SCREEN TO` is preprocessed into `SaveScreen()` function.

`__XSaveScreen()` is a compatibility function, it is superseded by `SaveScreen()` which allow you to save part or all the screen into a variable.

Examples
```
// save the screen, display list of files than restore the screen
SAVE SCREEN
DIR *.*
RESTORE SCREEN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also

RestScreen(), RESTORE SCREENClipper 5.3, SaveScreen()

AChoice() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Allows selection of an element from an array
Syntax
```
AChoice( <nTop>, <nLeft>, <nBottom>, <nRight>, <acMenuItems>,
[<alSelableItems> | <lSelableItems>], [<cUserFunction> | <bUserBlock>],
[<nInitialItem>], [<nWindowRow>] ) → nPosition
```
Arguments
nTop - topmost row used to display array (default 0)
nLeft - leftmost row used to display array (default 0)
nBottom - bottommost row used to display array (default MaxRow())
nRight - rightmost row used to display array (default MaxCol())
acMenuItems - the character array of items from which to select
alSelableItems - an array of items, either logical or character,
which is used to determine if a particular item may be selected. If the type of a given item is character, it is macro evaluated, and the result is expected to be a logical. A value of .T. means that the item may be selected, .F. that it may not. (See next argument: lSelectableItems)
lSelableItems - a logical value which is used to apply to all
items in acMenuItems. If .T., all items may be selected; if .F., none may be selected. (See previous argument: alSelectableItems) Default .T.
cUserFunction - the name of a function to be called which may
affect special processing of keystrokes. It is specified without parentheses or parameters. When it is called, it will be supplied with the parameters: nMode, nCurElement, and nRowPos. Default NIL.
bUserBlock - a codeblock to be called which may
affect special processing of keystrokes. It should be specified in the form {| nMode, nCurElemenet, nRowPos | MyFunc(nMode, nCurElemenet, nRowPos) }. Default NIL.
nInitialItem - the number of the element to be highlighted as
the current item when the array is initially displayed. 1 origin. Default 1.
nWindowRow - the number of the window row on which the initial
item is to be displayed. 0 origin. Default 0.
Returns
nPosition - the number of the item to be selected, or 0 if the selection was aborted.
Description
Allows selection of an element from an array. Please see standard CA-Cl*pper documentation for AChoice() for additional detail.
Examples
```
LOCAL aItems := { "One", "Two", "Three" }
LOCAL nChoice := AChoice( 10, 10, 20, 20, aItems )
IF nChoice == 0
   ? "You did not choose an item"
ELSE
   ? ;
      "You chose element", hb_ntos( nChoice ), ;
      "which has a value of", aItems[ nChoice ]
ENDIF
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms

File
Library is core
Tags
API, User interface
See also
MENU TOClipper 5.3

Alert() Source code in Repositoryin Discussions ^ | ☰ | Improve this doc

Display a dialog box with a message
Syntax
```
Alert( <xMessage>, [<aOptions>], [<cColorNorm>], [<nDelay>] ) → nChoice or NIL
```
Arguments
`xMessage` Message to display in the dialog box. `xMessage` can be of any Harbour type. If `xMessage` is an array of Character strings, each element would be displayed in a new line. If `xMessage` is a Character string, you could split the message to several lines by placing a semicolon `;` in the desired places.
`aOptions` Array with available response. Each element should be Character string. If omitted, default is `{ "Ok" }`.
`cColorNorm` Color string to paint the dialog box with. If omitted, default color is `"W+/R"`.
`nDelay` Number of seconds to wait to user response before abort. Default value is 0, that wait forever.
Returns
`Alert()` return Numeric value representing option number chosen.
If Esc was pressed, return value is zero.
The return value is `NIL` if `Alert()` is called with no parameters, or if `xMessage` type is not Character and `HB_CLP_STRICT` option was used. If `nDelay` seconds had passed without user response, the return value is 1.
Description

`Alert()` display simple dialog box on screen and let the user select one option. The user can move the highlight bar using arrow keys or Tab key. To select an option the user can press Enter, Space or the first letter of the option.

If the program is executed with the `//NOALERT` command-line switch, nothing is displayed and it simply returns `NIL`. This switch could be overridden with `__NoNoAlert()`.

If the GT system is linked in, `Alert()` display the message using the full screen I/O system, if not, the information is printed to the standard output using `OutStd()`.

Examples
```
LOCAL cMessage, aOptions, nChoice

// harmless message
cMessage := "Major Database Corruption Detected!;" + ;
            "(deadline in few hours);;" + ;
            "where DO you want to go today?"

// define response option
aOptions := { "Ok", "www.example.org", "Oops" }

// show message and let end user select panic level
```

```
nChoice := Alert( cMessage, aOptions )
DO CASE
CASE nChoice == 0
   // do nothing, blame it on some one else
CASE nChoice == 1
   ? "Please call home and tell them you're going to be late"
CASE nChoice == 2
   // make sure your resume is up to date
CASE nChoice == 3
   ? "Oops mode is not working in this version"
ENDCASE
```

Status

Ready

Compliance

This function is sensitive to `HB_CLP_STRICT` settings during the compilation of `src/rtl/alert.prg`

defined: `xMessage` accept Character values only and return `NIL` if other types are passed.

undefined: `xMessage` could be any type, and internally converted to Character string. If type is Array, multi-line message is displayed.

defined: Only the first four valid `aOptions` are taken.

undefined: `aOptions` could contain as many as needed options.

If `HB_COMPAT_C53` was define during compilation of `src/rtl/alert.prg` the Left-Mouse button could be used to select an option.

The interpretation of the `//NOALERT` command-line switch is done only if `HB_CLP_UNDOC` was define during compilation of `src/rtl/alert.prg`

`cColorNorm` is a Harbour extension, or at least undocumented in Clipper 5.2 NG.

`nDelay` is a Harbour extension.

Platforms

Available on all platforms

File

Library is core

Tags

API, User interface

See also

`__NoNoAlert()`, `@...PROMPT`Clipper 5.3, `MENU TO`Clipper 5.3, `OutStd()` Clipper 5.3

`Browse()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Browse a database file

Syntax

`Browse( [<nTop>, <nLeft>, <nBottom>, <nRight>] ) → lOk`

Arguments

`nTop` coordinate for top row display.

`nLeft` coordinate for left column display.

`nBottom` coordinate for bottom row display.

`nRight` coordinate for right column display.

Returns

`Browse()` return `.F.` if there is no database open in this work area, else it return `.T.`

Description

`Browse()` is a general purpose database browser, without any thinking you can browse a file using the following keys:

Key	Meaning
Left	Move one column to the left (previous field)
Right	Move one column to the right (next field)
Up	Move up one row (previous record)
Down	Move down one row (next record)
PgUp	Move to the previous screen
PgDn	Move to the next screen
Ctrl+PgUp	Move to the top of the file
Ctrl+PgDn	Move to the end of the file
Home	Move to the leftmost visible column
End	Move to the rightmost visible column
Ctrl+Left	Pan one column to the left
Ctrl+Right	Pan one column to the right
Ctrl+Home	Move to the leftmost column
Ctrl+End	Move to the rightmost column
Esc	Terminate Browse()

On top of the screen you see a status line with the following indication:

Record ###/###	Current record number / Total number of records.
<none>	There are no records, the file is empty.
<new>	You are in append mode at the bottom of file.
<Deleted>	Current record is deleted.
<bof>	You are at the top of file.

You should pass whole four valid coordinate, if less than four parameters are passed to `Browse()` the coordinate are default to: 1, 0, `MaxRow()`, `MaxCol()`.

Examples

```
// this one shows you how to browse around
USE test
Browse()
```

Status
Started
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
`dbEdit()`*Clipper 5.3, `TBrowse class`

`dbEdit()`*Source code in Repository in Discussions ^ | ☰ | Improve this doc

Browse records in a table
Syntax

```
dbEdit( [<nTop>], [<nLeft>], [<nBottom>], [<nRight>], [<acColumns>],
[<xUserFunc>], [<xColumnSayPictures>], [<xColumnHeaders>],
```

[<xHeadingSeparators>], [<xColumnSeparators>], [<xFootingSeparators>], [<xColumnFootings>]) → lOk

Arguments

nTop coordinate for top row display. nTop could range from 0 to MaxRow(), default is 0.

nLeft coordinate for left column display. nLeft could range from 0 to MaxCol(), default is 0.

nBottom coordinate for bottom row display. nBottom could range from 0 to MaxRow(), default is MaxRow().

nRight coordinate for right column display. nRight could range from 0 to MaxCol(), default is MaxCol().

acColumns is an array of character expressions that contain database fields names or expressions to display in each column. If not specified, the default is to display all fields from the database in the current work area.

xUserFunc is a name of a user defined function or a code block that would be called every time unrecognized key is been pressed or when there are no keys waiting to be processed and dbEdit() goes into idle mode. If xUserFunc is a character string, it must contain root name of a valid user define function without parentheses. Both the user define function or the code block should accept two parameters: nMode, nCurrentColumn. Both should return a numeric value that correspond to one of the expected return codes (see table below for a list of nMode and return codes).

xColumnSayPictures is an optional picture. If xColumnSayPictures is a character string, all columns would used this value as a picture string. If xColumnSayPictures is an array, each element should be a character string that correspond to a picture string for the column with the same index. Look at the help for @...SAY to get more information about picture values.

xColumnHeaders contain the header titles for each column, if this is a character string, all columns would have that same header, if this is an array, each element is a character string that contain the header title for one column. Header may be split to more than one line by placing semicolon ; in places where you want to break line. If omitted, the default value for each column header is taken from acColumns or field name if acColumns was not specified.

xHeadingSeparators is an array that contain characters that draw the lines separating the headers and the fields data. Instead of an array you can use a character string that would be used to display the same line for all fields. Default value is a double line.

xColumnSeparators is an array that contain characters that draw the lines separating displayed columns. Instead of an array you can use a character string that would be used to display the same line for all fields. Default value is a single line.

xFootingSeparators is an array that contain characters that draw the lines separating the fields data area and the footing area. Instead of an array you can use a character string that would be used to display the same line for all footers. Default is to have to no footing separators.

xColumnFootings contain the footing to be displayed at the bottom of each column, if this is a character string, all columns would have that same footer, if this is an array, each element is a character string that contain the footer for one column. Footer may be split to more than one line by placing semicolon ; in places where you want to break line. If omitted, no footer are displayed.

Returns

dbEdit() return .F. if there is no database in use or if the number of columns to display is zero, else dbEdit() return .T.

Description

dbEdit() display and edit records from one or more work areas in a grid on screen. Each column is defined by element from acColumns and is the equivalent of one field. Each row is equivalent of one database record.

Following are active keys that handled by `dbEdit()`:

Key	Meaning
Left	Move one column to the left (previous field)
Right	Move one column to the right (next field)
Up	Move up one row (previous record)
Down	Move down one row (next record)
PgUp	Move to the previous screen
PgDn	Move to the next screen
Ctrl+PgUp	Move to the top of the file
Ctrl+PgDn	Move to the end of the file
Home	Move to the leftmost visible column
End	Move to the rightmost visible column
Ctrl+Left	Pan one column to the left
Ctrl+Right	Pan one column to the right
Ctrl+Home	Move to the leftmost column
Ctrl+End	Move to the rightmost column

When `xUserFunc` is omitted, two more keys are active:

Key	Meaning
Esc	Terminate Browse()
Enter	Terminate Browse()

When `dbEdit()` execute `xUserFunc` it pass the following arguments: `nMode` and the index of current record in `acColumns`. If `acColumns` is omitted, the index number is the `FieldName()` number of the open database structure.

`dbEdit() nMode` could be one of the following:

dbedit.ch	Meaning
DE_IDLE	dbEdit() is idle, all movement keys have been handled.
DE_HITTOP	Attempt to cursor past top of file.
DE_HITBOTTOM	Attempt to cursor past bottom of file.
DE_EMPTY	No records in work area, database is empty.
DE_EXCEPT	Key exception.

The user define function or code block must return a value that tell `dbEdit()` what to do next.

User function return codes:

dbedit.ch	Meaning
DE_ABORT	Abort dbEdit().
DE_CONT	Continue dbEdit() as is.
DE_REFRESH	Force reread/redisplay of all data rows.

The user function is called once in each of the following cases: - The database is empty. - The user try to move past top of file or past bottom file. - Key exception, the uses had pressed a key that is not handled by `dbEdit()`. - The keyboard buffer is empty or a screen refresh had just occurred `dbEdit()` is a compatibility function, it is superseded by the TBrowse class and there for not recommended for new applications.

Examples

```
// Browse a file using default values
USE test
dbEdit()
```

Status

Started

Compliance

`xUserFunc` can take a code block value, this is a Harbour extension.

CA-Cl*pper will throw an error if there's no database open, Harbour would return `.F.`

CA-Cl*pper is buggy and will throw an error if the number of columns is zero, Harbour would return `.F.`

The CA-Cl*pper 5.2 NG state that the return value is `NIL`, this is wrong and should be read logical.

There is an undocumented result code (3) from the user defined function in CA-Cl*pper (both 87 and 5.x). This is an Append Mode which: "split the screen to allow data to be appended in windowed area". This mode is not supported by Harbour.

Platforms

Available on all platforms

Files

Header files are `dbedit.ch`, `inkey.ch`

Library is core

Tags

API, User interface

See also

`@...SAY`Clipper 5.3, `Browse()`Clipper 5.3, `TBrowse class`, `Transform()`Clipper 5.3

`hb_keyPut()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Put an inkey code to the keyboard buffer.

Syntax

```
hb_keyPut( <nInkeyCode> )
```

Arguments

`nInkeyCode` is the inkey code, which should be inserted into the keyboard buffer.

Returns

There is no return value.

Description

Inserts an inkey code to the string buffer. The buffer is **not** cleared in this operation. This function allows to insert such inkey codes which are not in the range of 0 to 255. To insert more than one code, call the function repeatedly. The zero code cannot be inserted.

Examples

```
#include "inkey.ch"
// Stuff an <Alt+PgDn> key into the keyboard buffer
hb_keyPut( K_ALT_PGDN ); ? Inkey()  // --> 417
hb_keyPut( K_F11 ); ? Inkey()  // --> -40
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

File

Library is core

Tags
API, User interface
See also
CLEAR TYPEAHEAD, Inkey() Clipper 5.3, KEYBOARDClipper 5.3

Inkey() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Extracts the next key code from the Harbour keyboard buffer.
Syntax
```
Inkey( [<nTimeout>] [, <nEvents>] ) → nKey
```
Arguments
nTimeout is an optional timeout value in seconds, with a granularity of 1/10th of a second. If omitted, Inkey() returns immediately. If set to 0, Inkey() waits until an input event occurs. If set to any other value, Inkey() will return either when an input event occurs or when the timeout period has elapsed. If only this parameter is specified and it is not numeric, it will be treated as if it were 0. But if both parameters are specified and this parameter is not numeric, it will be treated as if it were not present.
nEvents is an optional mask of input events that are to be enabled. If omitted, defaults to hb_set.HB_SET_EVENTMASK. Valid input masks are in inkey.ch and are explained below. It is recommended that the mask names be used rather than their numeric values, in case the numeric values change in future releases of Harbour. To allow more than one type of input event, simply add the various mask names together.

inkey.ch	Meaning
INKEY_MOVE	Mouse motion events are allowed
INKEY_LDOWN	The mouse left click down event is allowed
INKEY_LUP	The mouse left click up event is allowed
INKEY_RDOWN	The mouse right click down event is allowed
INKEY_RUP	The mouse right click up event is allowed
INKEY_KEYBOARD	All keyboard events are allowed
INKEY_ALL	All mouse and keyboard events are allowed
HB_INKEY_EXTENDED	Extended keyboard codes are used.

If the parameter is not numeric, it will be treated as if it were set to hb_set.HB_SET_EVENTMASK.
Returns
0 in case of timeout with no input event, otherwise returns a value in the range -47 to 386 for keyboard events or the range 1001 to 1007 for mouse events. Mouse events and non-printable keyboard events are represented by the K_<event> values listed in inkey.ch. Keyboard event return codes in the range 32 through 127 are equivalent to the printable ASCII character set. Keyboard event return codes in the range 128 through 255 are assumed to be printable, but results may vary based on hardware and nationality. If HB_INKEY_EXTENDED mode is used, then the return value for keyboard events ranges from 1 through 767 and 1077 through 1491, although not all codes are used.
Extended key codes consist of the PC keyboard scan code and one or more offset values. If no keyboard modifier was used, then HB_INKEY_NONE is added. The Alt key adds HB_INKEY_ALT, the Ctrl key adds HB_INKEY_CTRL, the Shift key adds HB_INKEY_SHIFT, and enhanced keys (KeyPad+/ and CursorPad keys) add HB_INKEY_ENHANCED. For example, F1 is scan code 59, so if you just press F1, you get key code 315, but Alt+F1 gives 443, Ctrl+F1 gives 571, and Shift+F1 gives 699. And NumPad+/ gives 1077, 1205, 1333, and 1461. At this time, the only value that can combine with other values is HB_INKEY_ENHANCED (i.e., there are no Alt+Ctrl combinations, etc.)

Note: The extended key code set is larger than the normal key code set. As a result, if you switch between the normal and extended modes, you need to be aware that some codes get translated into a zero in normal mode (because there is no corresponding code in normal mode) and that these codes get removed from the keyboard input buffer in normal mode and you won't be able to go back and fetch them later in extended mode.

Description

`Inkey()` can be used to detect input events, such as key-press, mouse movement, or mouse key clicks (up and/or down).

Examples

```
#include "inkey.ch"
// Wait for the user to press the <Esc> key
? "Please press the <Esc> key."
DO WHILE Inkey( 0.1 ) != K_ESC
ENDDO

KEYBOARD "AB"; ? Inkey(), Inkey()  // -->   65   66
```

Status

Started

Compliance

`Inkey()` is compliant with the CA-Cl*pper 5.3 `Inkey()` function with one exception: The Harbour `Inkey()` function will raise an argument error if the first parameter is less than or equal to 0 and the second parameter (or the default mask) is not valid, because otherwise `Inkey()` would never return, because it was, in effect, asked to wait forever for no events (Note: In CA-Cl*pper, this also blocks `SET KEY` events).

Platforms

Available on all platforms

Files

Library is core

Header is `inkey.ch`

Tags

API, User interface

`LastKey()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the last key extracted from the keyboard buffer.

Syntax

```
LastKey( [<nInputMask>] ) → nKey
```

Arguments

`nInputMask` is an optional integer value composed of one or more `INKEY_` or `HB_INKEY_` constants. The sole purpose of this argument is to allow switching between using `HB_INKEY_EXTENDED` key codes and using the normal CA-Cl*pper-compatible key codes

Returns

`nKey` The last key extracted from the keyboard buffer.

Description

Returns the value of the last key extracted from the Harbour keyboard buffer

Examples

```
#include "inkey.ch"
// Continue looping unless the <Esc> key was pressed in MainFunc()
DO WHILE .T.
   MainFunc()
   IF LastKey() == K_ESC
      EXIT
   ENDIF
ENDDO
```

```
KEYBOARD "AB"; ? Inkey(), LastKey()  // -->   65   65

STATIC PROCEDURE MainFunc()
   Inkey( 0 )
   RETURN
```

Status

Ready

Compliance

`LastKey()` is compliant with CA-Cl*pper 5.3, but has been extended for Harbour.

Platforms

Available on all platforms

File

Library is core

Tags

API, User interface

See also

`Inkey()` Clipper 5.3, `LastKey()` Clipper 5.3

`MCol()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the mouse cursor column position.

Syntax

`MCol() → nMouseColumn`

Arguments

None

Returns

`nMouseColumn` The mouse cursor column position.

Description

This function returns the column position of the mouse cursor. On graphical systems the value represents pixels. On character-based systems the value represents character columns as in CA-Cl*pper.

Examples

```
IF MCol() < 1
   ? "Mouse is on left edge!"
ENDIF
```

Status

Ready

Compliance

`MCol()` is compliant with CA-Cl*pper 5.3, but has been extended to work on graphical systems as well as character-based systems.

Platforms

Available on all platforms

File

Library is core

Tags

API, User interface

See also

`MRow()` Clipper 5.3

`MENU TO` Source code ^ | ☰ | Improve this doc

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

Syntax

`MENU TO <cVariable>`

Arguments

`cVariable` is a character string that contain the name of the variable to hold the menu choices, if this variable does not exist a `PRIVATE` variable with the name `cVariable` would be created to hold the result.

Description

`MENU TO` invoked the menu define by previous `__AtPrompt()` call and display a highlight bar that the user can move to select an option from the menu. If `cVariable` does not exist or not visible, a `PRIVATE` variable named `cVariable` is created and hold the current menu selection. If there is a variable named `cVariable`, its value is used to select the first highlighted item.

Menu prompts and messages are displayed in current Standard color, highlighted bar is displayed using current Enhanced color.

Pressing the arrow keys move the highlighted bar. When a menu item is highlighted the message associated with it is displayed on the line specified with `SET MESSAGE`. If `SET WRAP` is `ON` and the user press Up arrow while on the first selection the last menu item is highlighted, if the user press Down arrow while on the last item, the first item is highlighted.

Following are active keys that handled by `MENU TO`:

key	Meaning
Up	Move to previous item
Down	Move to next item
Left	Move to previous item
Right	Move to next item
Home	Move to the first item
End	Move to the last item
PgUp	Select menu item, return position
PgDn	Select menu item, return position
Enter	Select menu item, return position
Esc	Abort selection, return 0
First letter	Select next menu with the same first letter, return this item position.

upon exit the cursor is placed at `MaxRow()` - 1, 0 `MENU TO` can be nested without loosing the previous prompts.

`MENU TO` command is preprocessed into `__MenuTo()` function during compile time.

Examples

```
LOCAL nChoice
// display menu item on each screen corner and let user select one
CLS
SET MESSAGE TO MaxRow() / 2 CENTER
SET WRAP ON
@ 0           , 0             PROMPT "1. Upper left"   MESSAGE " One "
@ 0           , MaxCol() - 16 PROMPT "2. Upper right"  MESSAGE " Two "
@ MaxRow() - 1, MaxCol() - 16 PROMPT "3. Bottom right" MESSAGE "Three"
@ MaxRow() - 1, 0             PROMPT "4. Bottom left"  MESSAGE "Four "
MENU TO nChoice
```

```
SetPos( MaxRow() / 2, MaxCol() / 2 - 10 )
IF nChoice == 0
   ?? "<Esc> was pressed"
ELSE
   ?? "Selected option is", nChoice
ENDIF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
API, User interface
See also
__AtPrompt(), @...PROMPTClipper 5.3, AChoice() Clipper 5.3, SET INTENSITYClipper 5.3, SET MESSAGEClipper 5.3, SET WRAPClipper 5.3

MRow() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns the mouse cursor row position.
Syntax
`MRow() → nMouseRow`
Arguments
None
Returns
`nMouseRow` The mouse cursor row position.
Description
This function returns the current mouse row cursor position. On graphical systems the value represents pixel rows. On character-based systems the value represents character rows as in CA-Cl*pper.
Examples

```
IF MRow() < 1
   ? "Mouse is on top row!"
ENDIF
```

Status
Ready
Compliance
`MRow()` is compliant with CA-Cl*pper 5.3, but has been extended to work on graphical systems as well as character-based systems.
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
MCol() Clipper 5.3

NextKey() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Get the next key code in the buffer without extracting it.
Syntax

NextKey([<nInputMask>]) → nKey

Arguments

nInputMask is an optional integer value composed of one or more INKEY_ or HB_INKEY_ constants. The sole purpose of this argument is to allow switching between using HB_INKEY_EXTENDED key codes and using the normal CA-Cl*pper-compatible key codes

Returns

nKey The value of the next key in the Harbour keyboard buffer.

Description

Returns the value of the next key in the Harbour keyboard buffer without extracting it.

Examples

```
#include "inkey.ch"
// Use NextKey() with Inkey() to change display characters, or by
// itself to exit the loop, so that the caller can detect the <Esc>.
LOCAL nKey, cChar := "+"
DO WHILE .T.
   ?? cChar
   nKey := NextKey()
   DO CASE
   CASE nKey == K_ESC
      EXIT
   CASE nKey != 0
      cChar := hb_keyChar( nKey )
   ENDCASE
ENDDO

KEYBOARD "AB"; ? NextKey(), NextKey()  // -->   65   65
```

Status

Ready

Compliance

NextKey() is compliant with CA-Cl*pper 5.3, but has been extended for Harbour.

Platforms

Available on all platforms

File

Library is core

Tags

API, User interface

See also

Inkey() Clipper 5.3, LastKey() Clipper 5.3

OutErr() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Write a list of values to the standard error device

Syntax

OutErr(<xExp,...>)

Arguments

xExp,... is a list of expressions to display. Expressions are any mixture of Harbour data types.

Description

OutErr() write one or more values into the standard error device. Character and Memo values are printed as is, Dates are printed according to the SET DATE FORMAT, Numeric values are converted to strings, Logical values are printed as .T. or .F., NIL are printed as NIL, values of any other kind are printed as empty string. There is one space separating each two

values. Note that Numeric value can take varying length when converted into string depending on its source (see Str() for detail).

There is an undocumented CA-Cl*pper command-line switch //STDERR which can set the file handle to write output from OutErr(). If not specified the default STDERR is used, //STDERR or //STDERR:0 set OutErr() to output to the same file handle as OutStd(), //STDERR:n set output to file handle n. Like other undocumented features this switch is available only if src/rtl/console.c was compiled with the HB_CLP_UNDOC flag.

Examples

```
// write error log information
OutErr( hb_DateTime(), "Core meltdown detected" )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
?|??, DevOut(), DevOutPict() Clipper 5.3, DispOut(), DispOutAt(), OutStd() Clipper 5.3, QOut(), QQOut(), Str() Clipper 5.3

OutStd() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Write a list of values to the standard output device
Syntax

```
OutStd( <xExp,...> )
```

Arguments
xExp,... is a list of expressions to display. Expressions are any mixture of Harbour data types.
Description

OutStd() write one or more values into the standard output device. Character and Memo values are printed as is, Dates are printed according to the SET DATE FORMAT, Numeric values are converted to strings, Logical values are printed as .T. or .F., NIL are printed as NIL, values of any other kind are printed as empty string. There is one space separating each two values. Note that Numeric value can take varying length when converted into string depending on its source (see Str() for detail).

OutStd() is similar to QQOut() with the different that QQOut() send its output to the Harbour console stream, which can or cannot be redirected according with the screen driver, and OutStd() send its output to the standard output device (STDOUT) and can be redirected.

Examples

```
OutStd( "Hello" )           // --> Hello

OutStd( 1, .T., NIL, "A" )
OutStd( "B" )               // -->          1 .T. NIL AB
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
?|??, DevOut(), DevOutPict() Clipper 5.3, DispOut(), DispOutAt(), OutErr() Clipper 5.3, QOut(), QQOut(), Str() Clipper 5.3

ReadKey() *Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine which key terminated a READ.
Syntax
```
ReadKey() → nKeyCode
```
Arguments
None.
Returns
ReadKey() returns a numeric code representing the key that caused READ to terminate.
Description

ReadKey() is used after a READ was terminated to determine the exit key pressed. If the GET buffer was updated during READ, 256 is added to the return code.

Exit Key	Return code (not updated)	Return code (updated)
Up	4	260
Down	5	261
Page-Up	6	262
Page-Down	7	263
Ctrl Page-Up	34	290
Ctrl Page-Down	35	291
Esc	12	268
Ctrl End	14	270
Enter	15	271
Key >= 32	15	271
otherwise	0	0

ReadKey() is a compatibility function so try not to use it. ReadKey() is superseded by LastKey() which returns the Inkey() code for that key. Updated() could be used to find if the GET buffer was changed during the READ.

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
File
Library is core
Tags
API, User interface
See also
`@...GET`Clipper 5.3, `Inkey()` Clipper 5.3, `LastKey()` Clipper 5.3, `ReadExit()`, `READ`, `Updated()`

`ReadVar()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Return variable name of current GET or MENU
Syntax
`ReadVar( [<cVarName>] ) → cOldVarName`
Arguments
`cVarName` is a new variable name to set.
Returns
`ReadVar()` return the old variable name. If no variable previously was set, `ReadVar()` return "".
Description
`ReadVar()` is set inside a `READ` or `MENU TO` command to hold the uppercase name of the `GET` / `MENU TO` variable, and re-set back to old value when those commands finished. You should not normally set a variable name but rather use it to retrieve the name of a `GET` variable when executing a `VALID` or `WHEN` clause, or during `SET KEY` execution and you are inside a `READ` or `MENU TO`.
Examples

```
#include "inkey.ch"
LOCAL What_Is_Bug
// display a menu, press <F1> to view the MENU TO variable name
CLS
@ 1, 10 PROMPT "blood sucking insect that infect beds   "
@ 2, 10 PROMPT "germ; virus infection                   "
@ 3, 10 PROMPT "defect; snag; (source of) malfunctioning"
@ 4, 10 PROMPT "small hidden microphone                 "
@ 6, 10 SAY "(Press <F1> for a hint)"
SetKey( K_F1, {|| ShowVar() } )
MENU TO What_Is_Bug

STATIC PROCEDURE ShowVar()
   Alert( ReadVar() )  // --> "WHAT_IS_BUG"
   RETURN
```

Status
Ready
Compliance
`ReadVar()` works exactly like CA-Cl*pper's `ReadKey()`.
Note however, that the `cVarName` parameter is not documented and used internally by CA-Cl*pper.
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also

__AtPrompt(), __MenuTo(), @...GETClipper 5.3, @...PROMPTClipper 5.3, MENU TOClipper 5.3, READ, SetKey() Clipper 5.3

TBrowseDB() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a new TBrowse object to be used with database file
Syntax
```
TBrowseDB( [<nTop>], [<nLeft>], [<nBottom>], [<nRight>] ) → oBrowse
```
Arguments
nTop coordinate for top row display.
nLeft coordinate for left column display.
nBottom coordinate for bottom row display.
nRight coordinate for right column display.
Returns
TBrowseDB() return new TBrowse object with the specified coordinate and a default :SkipBlock, :GoTopBlock and :GoBottomBlock to browse a database file.
Description
TBrowseDB() is a quick way to create a TBrowse() object along with the minimal support needed to browse a database. Note that the returned TBrowse() object contain no TBColumn() objects and you need to add column for each field by your self.
Examples
```
// For a good example, look at the source code for Browse() function
// at src/rtl/browse.prg
```
Status
Started
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, User interface
See also
Browse() Clipper 5.3, TBColumn class, TBrowse class, TBrowseNew()

__mvClear() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function releases all PRIVATE and PUBLIC variables
Syntax
```
__mvClear()
```
Arguments
None
Returns
Nothing
Description
This function releases all PRIVATE and PUBLIC variables. It is used to implement CLEAR MEMORY statement. The memory occupied by all visible variables are released - any attempt to access the variable will result in a runtime error. You have to reuse PRIVATE or PUBLIC statement to create again the variable that was cleared by this function.
Status

Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
__mvPublic()

__mvExist() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine if a given name is a PUBLIC or PRIVATE memory variable
Syntax
```
__mvExist( <cVarName> ) → lVariableExist
```
Arguments
cVarName - string that specifies the name of variable to check
Returns
__mvExist() return TRUE (.T.) if a MEMVAR named cVarName exist.
Description
This function determine if a PUBLIC or PRIVATE variable with the name cVarName exist or not.
Examples
```
LOCAL   TheLocal
STATIC  TheStatic
PUBLIC  ThePublic
PRIVATE ThePrivate
? __mvExist( "NotExist"   )  // --> .F.
? __mvExist( "TheLocal"   )  // --> .F.
? __mvExist( "TheStatic"  )  // --> .F.
? __mvExist( "ThePublic"  )  // --> .T.
? __mvExist( "ThePrivate" )  // --> .T.
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
MEMVARClipper 5.3, PRIVATE, PUBLIC

__mvGet() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function returns value of memory variable
Syntax
```
__mvGet( <cVarName> ) → xVar
```
Arguments

`cVarName` - string that specifies the name of variable
Returns
`xVar` The value of variable
Description
This function returns the value of `PRIVATE` or `PUBLIC` variable if this variable exists otherwise it generates a runtime error. The variable is specified by its name passed as the function parameter.
Examples

```
? ValType( MemVarBlock( "myvar" ) )
STATIC FUNCTION MemVarBlock( cMemvar )
   RETURN {| x | ;
      iif( PCount() == 0, ;
         __mvGet( cMemvar ), ;
         __mvPut( cMemvar, x ) ) }
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
`__mvPut()`

`__mvPrivate()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function creates a PRIVATE variable
Syntax

```
__mvPrivate( <variable_name> )
```

Arguments
`variable_name` = either a string that contains the variable's name or an one-dimensional array of strings with variable names No skeleton are allowed here.
Returns
Nothing
Description
This function can be called either by the Harbour compiler or by user. The compiler always passes the item of `HB_IT_SYMBOL` type that stores the name of variable. If a variable with the same name exists already then the value of old variable is hidden until the new variable is released. The new variable is always initialized to `NIL` value.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management

__mvPublic() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function creates a PUBLIC variable
Syntax
```
__mvPublic( <variable_name> )
```
Arguments
`variable_name` = either a string that contains the variable's name or an one-dimensional array of strings with variable names No skeleton are allowed here.
Returns
Nothing
Description
This function can be called either by the Harbour compiler or by user. The compiler always passes the item of `HB_IT_SYMBOL` type that stores the name of variable. If a variable with the same name exists already then the new variable is not created - the previous value remains unchanged. If it is first variable with this name then the variable is initialized with `.T.` value.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management

__mvPut() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function set the value of memory variable
Syntax
```
__mvPut( <cVarName> [, <xValue>] ) → xValue
```
Arguments
`cVarName` - string that specifies the name of variable `xValue` - a value of any type that will be set - if it is not specified then `NIL` is assumed
Returns
`xValue` A value assigned to the given variable.
Description
This function sets the value of `PRIVATE` or `PUBLIC` variable if this variable exists otherwise it generates a runtime error. The variable is specified by its name passed as the function parameter. If a value is not specified then the `NIL` is assumed
Examples
```
? ValType( MemVarBlock( "myvar" ) )
STATIC FUNCTION MemVarBlock( cMemvar )
   RETURN {| x | ;
      iif( PCount() == 0, ;
         __mvGet( cMemvar ), ;
         __mvPut( cMemvar, x ) ) }
```
Status
Ready
Compliance
Harbour specific
Platforms

Available on all platforms
File
Library is core
Tags
API, Variable management
See also
__mvPut()

__mvRelease() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function releases PRIVATE variables
Syntax
```
__mvRelease( <skeleton>, <include_exclude_flag> )
```
Arguments
`skeleton` = string that contains the wildcard mask for variables' names that will be released. Supported wildcards: '*' and '?' `include_exclude_flag` = logical value that specifies if variables that match passed skeleton should be either included in deletion (if `.T.`) or excluded from deletion (if `.F.`)
Returns
Nothing
Description
This function releases values stored in memory variables. It shouldn't be called directly, it should be placed into `RELEASE ALL` command. If the released variable is a `PRIVATE` variable then previously hidden variable with the same name becomes visible after exit from the procedure where released variable was created. If you access the released variable in the same function/procedure where it was created the the `NIL` value is returned. You can however assign a new value to released variable without any side effects. `PUBLIC` variables are not changed by this function.
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management

__mvScope() Source code in Repository in Discussions ^ | ☰ | Improve this doc

If variable exists then returns its scope.
Syntax
```
__mvScope( <cVarName> )
```
Arguments
`cVarName` = a string with a variable name to check
Returns
The symbolic values are defined in hbmemvar.ch

HB_MV_NOT_FOUND	variable is not declared (not found in symbol table)
HB_MV_UNKNOWN	if variable doesn't exist (but found in symbol table)
HB_MV_ERROR	if information cannot be obtained (memory error or argument error)

HB_MV_PUBLIC for public variables
HB_MV_PRIVATE_GLOBAL for private variables declared outside of current function/procedure
HB_MV_PRIVATE_LOCAL for private variables declared in current function/procedure

Examples

```
#include "hbmemvar.ch"

MEMVAR pPublic
MEMVAR mPrivateGlobal
MEMVAR mPrivateLocal

PROCEDURE Main()

   PUBLIC pPublic
   PRIVATE mPrivateGlobal

   CallProc()
   ? __mvScope( "mPrivateLocal" )    // --> HB_MV_UNKNOWN

   RETURN

STATIC PROCEDURE CallProc()

   PRIVATE mPrivateLocal

   ? __mvScope( "pPublic" )          // --> HB_MV_PUBLIC
   ? __mvScope( "mPrivateGlobal" )   // --> HB_MV_PRIVATE_GLOBAL
   ? __mvScope( "mPrivateLocal" )    // --> HB_MV_PRIVATE_LOCAL
   ? __mvScope( "mFindMe" )          // --> HB_MV_NOT_FOUND

   IF __mvScope( "pPublic" ) > HB_MV_ERROR
      ? "Variable exists"
   ELSE
      ? "Variable not created yet"
   ENDIF

   RETURN
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management

__mvXRelease() Source code in Repository in Discussions ^ | ☰ | Improve this doc

This function releases value stored in PRIVATE or PUBLIC variable

Syntax

```
__mvXRelease( <variable_name> )
```

Arguments

`variable_name` = either a string that contains the variable's name or an one-dimensional array of strings with variable names No skeleton are allowed here.

Returns

Nothing
Description

This function releases values stored in memory variable. It shouldn't be called directly, rather it should be placed into `RELEASE` command. If the released variable is a `PRIVATE` variable then previously hidden variable with the same name becomes visible after exit from the procedure where released variable was created. If you access the released variable in the same function/procedure where it was created the the `NIL` value is returned. You can however assign a new value to released variable without any side effects.

It releases variable even if this variable was created in different procedure.

Examples

```
MEMVAR mPrivate

PROCEDURE Main()

   PRIVATE mPrivate := "PRIVATE from Main()"

   ? mPrivate     // --> PRIVATE from Main()
   Test()
   ? mPrivate     // --> PRIVATE from Main()

   RETURN

STATIC PROCEDURE Test()

   PRIVATE mPrivate := "PRIVATE from Test()"

   ? mPrivate           // --> PRIVATE from Test()
   RELEASE mPrivate
   ? mPrivate           // --> NIL
   mPrivate := "Again in Main()"

   RETURN
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management

`Empty()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Checks if the passed argument is empty.
Syntax

```
Empty( <xExp> ) → lIsEmpty
```

Arguments
`xExp` is any valid expression.
Returns
A logical value. It is true (`.T.`) if the passed argument is empty otherwise it is false (`.F.`).

Description
This function checks if an expression has empty value and returns a logical indicating whether it the expression is empty or not.
Examples
```
? Empty( "I'm not empty" )  // --> .F.
? Empty( NIL )              // --> .T.
? Empty( 0 )                // --> .T.
? Empty( .F. )              // --> .T.
? Empty( "" )               // --> .T.
? Empty( "  " )             // --> .T.
? Empty( 1 )                // --> .F.
? Empty( .T. )              // --> .F.
? Empty( "smile" )          // --> .F.
? Empty( Date() )           // --> .F.
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
`Len()` Clipper 5.3

`hb_PIsByRef()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Determine if a parameter is passed by reference.
Syntax
```
hb_PIsByRef( nParam ) → lParamIsByRef
```
Arguments
`nParam` is the parameter number to test.
Returns
`lVarIsByRef` a logical value indicating if the parameter is passed by reference to actual function or procedure.
Description

This function return a logical value indicating if the parameter is passed by reference to actual function or procedure.

This function is based on the form that Harbour manages to the variables for reference. When a variable is passed by reference, what receives the function or procedure is, a pointer to the previous variable, be this the container variable of the data or a pointer to another variable. The function observes if the variable passed points to a common variable or to a variable passed by reference.

Examples
```
PROCEDURE Main()

   LOCAL cVar := "Test local"

   MEMVAR m_nVar
```

```
   PRIVATE m_nVar := 0

   Test( @cVar, @m_nVar, cVar, m_nVar )

   RETURN

STATIC PROCEDURE Test( Arg1, Arg2, Arg3, Arg4 )
   ? hb_PIsByRef( 1 )  // --> .T.
   ? hb_PIsByRef( 2 )  // --> .T.
   ? hb_PIsByRef( 3 )  // --> .F.
   ? hb_PIsByRef( 4 )  // --> .F.
   RETURN
```

Status
Started
Compliance
Harbour specific
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
`ValType()` Clipper 5.3

`Len()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns size of a string or size of an array.
Syntax

```
Len( <cString> | <aArray> ) → nLength
```

Arguments
`acString` is a character string or the array to check.
Returns
The length of the string or the number of elements that contains an array.
Description
This function returns the string length or the size of an array or the size of a hash table. If it is used with a multidimensional array it returns the size of the first dimension.
Examples

```
LOCAL cName

? Len( "Harbour" )          // --> 7
? Len( { "One", "Two" } )   // --> 2

cName := ""
ACCEPT "Enter your name: " TO cName
? Len( cName )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags

API, Variable management
See also
AAdd() Clipper 5.3, ASize() Clipper 5.3, Empty() Clipper 5.3, LTrim() Clipper 5.3, RTrim() Clipper 5.3

MemVarBlock() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Returns a codeblock that sets/gets a value of memvar variable
Syntax
```
MemVarBlock( <cMemvarName> ) → bBlock
```
Arguments
cMemvarName - a string that contains the name of variable
Returns
bBlock a codeblock that sets/get the value of variable
Description
This function returns a codeblock that sets/gets the value of PRIVATE or PUBLIC variable. When this codeblock is evaluated without any parameters passed then it returns the current value of given variable. If the second parameter is passed for the codeblock evaluation then its value is used to set the new value of given variable - the passed value is also returned as a value of the codeblock evaluation.
Examples
```
LOCAL cbSetGet
PUBLIC xPublic

cbSetGet := MemVarBlock( "xPublic" )
Eval( cbSetGet, "new value" )
? "Value of xPublic variable", Eval( cbSetGet )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
__mvGet(), __mvPut()

Type() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Retrieves the type of an expression
Syntax
```
Type( <cExp> ) → cRetType
```
Arguments
cExp must be a character expression.
Returns
cRetType a string indicating the type of the passed expression.

<cRetType>	Meaning
"A"	Array
"B"	Block

"C" Character (string)
"D" Date
"L" Logical
"M" Memo
"N" Numeric
"O" Object
"P" Pointer
"S" Symbol
"U" NIL, local or static variable, or not linked-in function
"UE" syntax error in the expression or invalid arguments
"UI" function with non-reserved name was requested

Description

This function returns a string which represents the data type of the argument. The argument can be any valid Harbour expression. If there is a syntax error in passed expression then "UE" is returned. If there is a call for any non-reserved Harbour function then "UI" is returned (in other words there is no call for passed UDF function during a data type determination - this is CA-Cl*pper compatible behavior). Additionally if requested user defined function is not linked into executable then "U" is returned.

The data type of expression is checked by invoking a macro compiler and by evaluation of generated code (if there is no syntax errors). This causes that `Type()` cannot determine a type of local or static variables - only symbols visible at runtime can be checked.

Notice the subtle difference between `Type()` and `ValType()` functions. `ValType()` function doesn't call a macro compiler - it simply checks the type of passed argument of any type. `Type()` requires a string argument with a valid Harbour expression - the data type of this expression is returned.

Examples

```
LOCAL c, cFilter
MEMVAR a, b

? Type( "{ 1, 2 }" )                                    // --> "A"
? Type( 'iif( .T., SubStr( "TYPE", 2, 1 ), .F. )' )     // --> "C"
? Type( 'At( "OK", MyUDF() ) > 0' )                     // --> "UI"
? Type( "{ 1, 2 }[ 5 ]" )                               // --> "UE"

PRIVATE a := "A", b := "B"
? Type( "a + b + c" )     // --> "U" ('c' variable is a local one)

cFilter := Space( 60 )
ACCEPT "Enter filter expression:" TO cFilter
IF Type( cFilter ) $ "CDLMN"
   // this is a valid expression
   SET FILTER TO &cFilter
ENDIF
```

Status

Ready

Compliance

- Incompatibility with CA-Cl*pper:

In the following code:

```
PRIVATE lCond := 0
? Type( "iof( lCond, 'true', MyUDF() )" )
```

CA-Cl*pper will print "UE" - in Harbour the output will be "UI"

- If "UI" is returned then the syntax of the expression is
correct. However invalid arguments can be passed to function/procedure that will cause runtime errors during evaluation of expression.
- Harbour supports two new types (Pointer and Symbol) which does
not exists in CA-Cl*pper.

Platforms

Available on all platforms

File

Library is core

Tags

API, Variable management

See also

`ValType()` Clipper 5.3

`ValType()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Retrieves the data type of an expression

Syntax

```
ValType( <xExp> ) → cRetType
```

Arguments

`xExp` is any valid expression.

Returns

`cRetType` a character indicating the type of the passed expression.

<cRetType>	Meaning
"A"	Array
"B"	Block
"C"	Character (string)
"D"	Date
"L"	Logical
"M"	Memo
"N"	Numeric
"O"	Object
"P"	Pointer
"S"	Symbol
"U"	NIL

Description

This function returns one character which represents the data type of the argument.

Examples

```
? ValType( Array( 1 ) )     // --> "A"
? ValType( {|| 1 + 1 } )    // --> "B"
? ValType( "Harbour" )      // --> "C"
? ValType( Date() )         // --> "D"
? ValType( .T. )            // --> "L"
? ValType( 1 )              // --> "N"
? ValType( TBrowse() )      // --> "O"
? ValType( hb_idleAdd() )   // --> "P" Harbour extension
? ValType( @QOut() )        // --> "S" Harbour extension
? ValType( NIL )            // --> "U"
```

Status

Ready

Compliance

`ValType()` is CA-Cl*pper compliant, with the addition of the new Harbour types: Pointer and Symbol.

Platforms
Available on all platforms
File
Library is core
Tags
API, Variable management
See also
`Type()` Clipper 5.3

`hb_setListenerAdd()` in Repository in Discussions ^ | ☰ | Improve this doc

Syntax
C Prototype

```
#include "hbset.h"
hb_setListenerAdd( PHB_SET_LISTENER_CALLBACK callback ) → int
```

Arguments
`callback` A pointer to a function taking two enum parameters and returning no value. The first parameter identifies the `SET` parameter that is to be changed and the second parameter identifies whether the call is from before or after the value is changed. The callback function will be called twice whenever a `SET` parameter is changed using the Harbour `SET` function. The first call takes place before the `SET` value is changed and the second one is after the `SET` parameter has been changed.
Returns
An integer value representing the callback handle, in case the caller needs to deactivate the callback function.
Description
This function allows a subsystem that needs to track the status of some `SET` parameters to be notified whenever a `SET` parameter gets changed.
Examples

```
void callback_function( HB_set_enum set, HB_set_listener_enum when )
{
   printf("\nCalled for SET parameter %d %s changing.",
      set, (when ? "after" : "before"));
}
int handle = hb_setListenerAdd( callback_function );
```

Status
Ready
Compliance
Not applicable
Platforms
Available on all platforms
File
Library is core
Tags
C level API, Environment
See also
`hb_setListenerRemove()`

`hb_setListenerNotify()` in Repository in Discussions ^ | ☰ | Improve this doc

Syntax
C Prototype

```
#include "hbset.h"
hb_setListenerNotify( HB_set_enum set, HB_set_listener_enum
when ) → int
```

Arguments

`set` The number of the `SET` parameter that is to be or was changed.

`when` Set to `HB_SET_LISTENER_BEFORE` when called before the `SET` parameter is to be changed and set to `HB_SET_LISTENER_AFTER` when called after the `SET` parameter has been changed.

Returns

`int`

Description

This function notifies all `SET` listener callback functions. It must be called any time you change the value of a `SET` parameter directly instead of using the Harbour `SET` function. Both before and after the change.

Examples

```
hb_setListenerNotify( HB_SET_DECIMALS, HB_SET_LISTENER_BEFORE );
hb_set.HB_SET_DECIMALS = 3;
hb_setListenerNotify( HB_SET_DECIMALS, HB_SET_LISTENER_AFTER );
```

Status

Ready

Compliance

Not applicable

Platforms

Available on all platforms

File

Library is core

Tags

C level API, Environment

See also

`hb_setListenerAdd()`

`hb_setListenerRemove()` in Repository in Discussions ^ | ☰ | Improve this doc

Syntax

`C Prototype`

```
#include "hbset.h"
hb_setListenerRemove( int handle ) → int
```

Arguments

`handle` The handle for the `SET` listener callback function to be removed.

Returns

The handle if the callback function could not be located or the negative value of the handle if the callback function was removed.

Description

This function removes a `SET` listener callback function.

Examples

```
int handle = hb_setListenerAdd( callback_function );
/* ... */
hb_setListenerRemove( handle );
```

Status

Ready

Compliance

Not applicable

Platforms
Available on all platforms
File
Library is core
Tags
C level API, Environment
See also
`hb_setListenerAdd()`

`hb_idleState()` Source code in Repository in Discussions ^ | ☰ | Improve this doc

Evaluates a single background task and calls the garbage collector.
Syntax

```
void hb_idleState( void );
```

Arguments
None
Description

`hb_idleState()` is a C function that requests garbage collection and executes a single background task defined by the codeblock passed with `hb_idleAdd()` function. It also releases the CPU time slices for platforms that require it.

Every call for this function evaluates different task in the order of task creation. There are no arguments passed during codeblock evaluation.

This function can be safely called even if there are no background tasks defined.

This function is automatically called from the `Inkey()` function.

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
C level API, Idle states
See also
`hb_idleAdd()`, `hb_idleDel()`, `hb_idleState()`

`hb_mathGetErrMode()` in Repository in Discussions ^ | ☰ | Improve this doc

get math error handling mode
Syntax
C Prototype

```
#include "hbmath.h"
hb_mathGetErrMode( void ) → imode
```

Returns
imode math error handling mode
Status
Ready

Compliance
Not applicable
Platforms
Available on all platforms
Files
Header file is hbmath.h
Library is core
Tags
C level API, Math
See also
hb_mathSetErrMode()

hb_mathGetHandler() □in Repositoryin Discussions ^ | ☰ | Improve this doc

get current Harbour math error handler
Syntax
C Prototype

```
#include "hbmath.h"
hb_mathGetHandler( void ) → HB_MATH_HANDLERPROC handlerproc
```

Arguments
handlerproc custom math handler
typedef int (* HB_MATH_HANDLERPROC)(HB_MATH_EXCEPTION * err)
Returns
handerproc
Status
Ready
Compliance
Not applicable
Platforms
Available on all platforms
Files
Header file is hbmath.h
Library is core
Tags
C level API, Math

hb_mathGetLastError() □in Repositoryin Discussions ^ | ☰ | Improve this doc

get the last math lib error
Syntax
C Prototype

```
#include "hbmath.h"
hb_mathGetLastError( HB_MATH_EXCEPTION * phb_exc )
                   → int iMathErrorType
```

Arguments
phb_exc pointer to HB_MATH_EXCEPTION structure, if not NULL,
the structure will be filled with information about the last math error:

```
typedef struct _HB_MATH_EXCEPTION {
  int type;        // Math error type, is one of the constants
                   // HB_MATH_ERR_xxx defined in hbmath.ch
  char *funcname;  // Pointer to name of the math C RTL routine
                   // that caused the error.
```

```
  char *error;     // Pointer to error description.
  double arg1;     // First and
  double arg2;     // Second double argument to the math routine.
  double retval;   // Corrected return value for the math routine.
  int retvalwidth; // Width and
  int retvaldec;   // Decimals of the corrected return value,
                   // both default to -1
  int handled;     // 1, if the math error is already corrected,
                   // 0 otherwise.
} HB_MATH_EXCEPTION;
```

Returns

`iMathErrorType`

Status

Ready

Compliance

Not applicable

Platforms

Available on all platforms

Files

Header file is `hbmath.h`

Library is core

Tags

C level API, Math

hb_mathIsMathErr() in Repository in Discussions ^ | ☰ | Improve this doc

Check if Harbour math error handling is available

Syntax

C Prototype

```
#include "hbmath.h"
hb_mathIsMathErr( void ) → int iIsMathHandler
```

Returns

`iIsMathHandler`

Status

Ready

Compliance

Not applicable

Platforms

Available on all platforms

Files

Header file is `hbmath.h`

Library is core

Tags

C level API, Math

hb_mathResetError() in Repository in Discussions ^ | ☰ | Improve this doc

Reset the internal math error information structure

Syntax

C Prototype

```
#include "hbmath.h"
hb_mathResetError( void )
```

Status
Ready
Compliance
Not applicable
Platforms
Available on all platforms
Files
Header file is hbmath.h
Library is core
Tags
C level API, Math

hb_mathSetErrMode() □in Repositoryin Discussions ^ | ☰ | Improve this doc

set math error handling mode
Syntax
C Prototype

```
#include "hbmath.h"
hb_mathSetErrMode( int imode ) → int ioldmode
```

Arguments
imode math error handling mode, one of the following
constants, defined in hbmath.ch:
HB_MATH_ERRMODE_DEFAULT
HB_MATH_ERRMODE_CDEFAULT
HB_MATH_ERRMODE_USER
HB_MATH_ERRMODE_USERDEFAULT
HB_MATH_ERRMODE_USERCDEFAULT
Returns
ioldmode old math error handling mode
Status
Ready
Compliance
Not applicable
Platforms
Available on all platforms
Files
Header file is hbmath.h
Library is core
Tags
C level API, Math
See also
hb_mathGetErrMode()

hb_mathSetHandler() □in Repositoryin Discussions ^ | ☰ | Improve this doc

set the Harbour math handler
Syntax
C Prototype

```
#include "hbmath.h"
hb_mathSetHandler( HB_MATH_HANDLERPROC handlerproc )
                  → HB_MATH_HANDLERPROC previous_handerproc
```

Arguments
handlerproc custom math handler
typedef int (* HB_MATH_HANDLERPROC)(HB_MATH_EXCEPTION * err)
Returns
previous_handlerproc previous math handler
typedef int (* HB_MATH_HANDLERPROC)(HB_MATH_EXCEPTION * err)
Status
Ready
Compliance
Not applicable
Platforms
Available on all platforms
Files
Header file is hbmath.h
Library is core
Tags
C level API, Math

CLASS VARSource code^ | ☰ | Improve this doc

Define a CLASS VAR variable for a class (NOT for an Object!)
Syntax
```
CLASS VAR <DataName1> [, <DataNameN>] [ AS <type> ] [ INIT <uValue> ]
```
Arguments
DataName1 Name of the VAR
type Optional data type specification from the following:
Character, Numeric, Date, Logical, Codeblock, NIL
uValue Optional initial value at program startup
Description

CLASS VAR variables can also be thought of as the "properties" of an entire class. Each CLASS VAR exists only once, no matter how many objects are created. A common usage is for a counter that is incremented whenever an object is created and decremented when one is destroyed, thus monitoring the number of objects in existence for this class.

You can use the AS <type> clause to enforce that the CLASS VAR is maintained as a certain type. Otherwise it will take on the type of whatever value is first assigned to it. Use the INIT <uValue> clause to initialize that VAR to uValue whenever the class is first used.

Examples
```
#include "hbclass.ch"
CREATE CLASS TWindow
   VAR   hWnd, nOldProc
   CLASS VAR lRegistered AS LOGICAL
ENDCLASS
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags

Class, Data
See also
CLASS, METHOD, VAR

VARSource code^ | ☰ | Improve this doc

Alternate syntax for VAR: instance variable for the objects.
Syntax

```
VAR <DataName1> [, <DataNameN>] [ AS <type> ] [ INIT <uValue> ]
[[EXPORTED | VISIBLE] | [PROTECTED] | [HIDDEN]] [READONLY | RO]
```

Arguments
DataName1 Name of the VAR
type Optional data type specification from the following:
Character, Numeric, Date, Logical, Codeblock, NIL.
uValue Optional initial value when creating a new object.
EXPORTED Specifies that this VAR is accessible to functions and
methods outside of the class. VISIBLE is a synonym for EXPORTED.
PROTECTED Specifies that this VAR is only accessible to functions and methods within this class and its subclasses.
HIDDEN Specifies that this VAR is only accessible to the
class where it was defined, and is not inherited by the subclasses.
READONLY Restricts assignment to the variable. If specified with
the EXPORTED clause, assignment is only permitted from the current class and its subclasses. If specified with the PROTECTED clause, assignment is only permitted from the current class. RO is a synonym for READONLY.
Description

VAR elements can also be thought of as the "properties" of an object. They can be of any data type, including codeblock. Once an object has been created, the VAR elements are referenced with the colon : as in `MyObject:Heading := "Last name"`. Usually a class also defines methods to manipulate the VAR.

You can use the AS <type> clause to enforce that the VAR is maintained as a certain type. Otherwise it will take on the type of whatever value is first assigned to it.

Use the INIT <uValue> clause to initialize that VAR to uValue whenever a new object is created.

VAR can be a synonym for VAR, or it can use a slightly different syntax for compatibility with other dialects.

Examples

```
#include "hbclass.ch"

CREATE CLASS TBColumn

   VAR Block      // Code block to retrieve data for the column
   VAR Cargo      // User-definable variable
   VAR ColorBlock // Code block that determines color of data items
   VAR ColSep     // Column separator character
   VAR DefColor   // Array of numeric indexes into the color table
   VAR Footing    // Column footing
   VAR FootSep    // Footing separator character
```

```
   VAR Heading    // Column heading
   VAR HeadSep    // Heading separator character
   VAR Width      // Column display width
   VAR ColPos     // Temporary column position on screen

   METHOD New()   // Constructor

ENDCLASS

METHOD New() CLASS TBColumn
   RETURN Self
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Data
See also
CLASS, CLASS VAR, METHOD, VAR

CLASSSource code^ | ☰ | Improve this doc

Define a Class for Object Oriented Programming
Syntax
`[CREATE] CLASS <ClassName> [ <FROM, INHERIT> <SuperClass1> [, <SuperClassN>] ] [STATIC]`
Arguments
`ClassName` Name of the class to define. By tradition, Harbour classes start with "T" to avoid collisions with user- created classes.
`SuperClass1...n` The Parent class(es) to use for inheritance. Harbour supports Multiple Inheritance.
`STATIC` This clause causes the class function to be declared as a static function. It will therefore not be available outside the current module.
Description

`CLASS` creates a class from which you can create objects. The `CLASS` command begins the class specification, in which the `VAR` elements (also known as instance variables) and `METHODS` of the class are named. The following scoping commands may also appear. They control the default scope of `VAR` and `METHOD` commands that follow them.

```
   EXPORTED:
   VISIBLE:
   HIDDEN:
   PROTECTED:
```

The class specification ends with the `END CLASS` command.

Classes can inherit from multiple `SuperClasses`, and the chain of inheritance can extend to many levels.

A program uses a Class by calling the Class Constructor, usually the `:New()` method, to create an object. That object is usually assigned to a variable, which is used to access the `VAR` elements and methods.

Harbour's OOP syntax and implementation supports Scoping (Protect, Hidden and Readonly) and Delegating, and is largely compatible with Class(y)(tm), TopClass(tm) and Visual Objects(tm).

Examples

```
#include "hbclass.ch"

CREATE CLASS TBColumn

   VAR Block      // Code block to retrieve data for the column
   VAR Cargo      // User-definable variable
   VAR ColorBlock // Code block that determines color of data items
   VAR ColSep     // Column separator character
   VAR DefColor   // Array of numeric indexes into the color table
   VAR Footing    // Column footing
   VAR FootSep    // Footing separator character
   VAR Heading    // Column heading
   VAR HeadSep    // Heading separator character
   VAR Width      // Column display width
   VAR ColPos     // Temporary column position on screen

   METHOD New()    // Constructor

ENDCLASS

METHOD New() CLASS TBColumn
   RETURN Self
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Definition
See also
`HBClass()`, `METHOD`, `VAR`

ENDCLASSSource code^ | ☰ | Improve this doc

End the declaration of a class.
Syntax

```
ENDCLASS
```

Arguments
(This statement has no arguments)
Description
`ENDCLASS` marks the end of a class declaration. It is usually followed by the class methods that are not `INLINE`.
Examples

```
#include "hbclass.ch"
CREATE CLASS TWindow
   VAR hWnd, nOldProc
```

```
ENDCLASS
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Definition
See also
CLASS, METHOD, VAR

ERROR HANDLERSource code^ | ☰ | Improve this doc

Designate a method as an error handler for the class
Syntax
```
ERROR HANDLER <MethodName>( [<params,...>] )
```
Arguments
`MethodName` Name of the method to define
`params,...` Optional parameter list
Description
`ERROR HANDLER` names the method that should handle errors for the class being defined.
Examples
```
#include "hbclass.ch"

CREATE CLASS TWindow
   ERROR HANDLER MyErrHandler()
ENDCLASS

METHOD MyErrHandler() CLASS TWindow
   RETURN Self
```
Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Method
See also
CLASS, METHOD, ON ERROR, VAR

MESSAGESource code^ | ☰ | Improve this doc

Route a method call to another Method
Syntax
```
MESSAGE <MessageName>   METHOD <MethodName>( [<params,...>] )
MESSAGE <MessageName>() METHOD <MethodName>( [<params,...>] )
```
Arguments
`MessageName` The pseudo-method name to define
`MethodName` The method to create and call when `MessageName` is invoked.
`params,...` Optional parameter list for the method

Description

The `MESSAGE` command is a seldom-used feature that lets you re-route a call to a method with a different name. This can be necessary if a method name conflicts with a public function that needs to be called from within the class methods.

For example, your app may have a public function called `BeginPaint()` that is used in painting windows. It would also be natural to have a Window class method called `:BeginPaint()` that the application can call. But within the class method you would not be able to call the public function because internally methods are based on static functions (which hide public functions of the same name).

The `MESSAGE` command lets you create the true method with a different name (`::xBeginPaint()`), yet still allow the `::BeginPaint()` syntax to call `::xBeginPaint()`. This is then free to call the public function `BeginPaint()`.

Examples

```
// FIXME
#include "hbclass.ch"
CREATE CLASS TWindow
   VAR    hWnd, nOldProc
   METHOD New() CONSTRUCTOR
   MESSAGE BeginPaint METHOD xBeginPaint()
ENDCLASS
```

Status

Ready

Compliance

Harbour specific

Platforms

Available on all platforms

Tags

Class, Method

See also

`CLASS`, `METHOD`, `VAR`

METHOD Source code^ | ☰ | Improve this doc

Declare a METHOD for a class in the class header

Syntax

```
METHOD <MethodName>( [<params,...>] ) [CONSTRUCTOR]
METHOD <MethodName>( [<params,...>] ) INLINE <Code,...>
METHOD <MethodName>( [<params,...>] ) BLOCK  <CodeBlock>
METHOD <MethodName>( [<params,...>] ) EXTERN <NAME>( [<args,...>] )
METHOD <MethodName>( [<params,...>] ) SETGET
METHOD <MethodName>( [<params,...>] ) VIRTUAL
METHOD <MethodName>( [<param>] )      OPERATOR <op>
METHOD <MethodName>( [<params,...>] ) CLASS <ClassName>
```

Arguments

`MethodName` Name of the method to define

`params,...` Optional parameter list

Description

Methods are "class functions" which do the work of the class. All methods must be defined in the class header between the CLASS and ENDCLASS commands. If the body of a method is not fully defined here, the full body is written below the ENDCLASS command using this syntax:

METHOD MethodName([params,...]) CLASS ClassName

Methods can reference the current object with the keyword Self: or its shorthand version ::.

CLAUSES:

CONSTRUCTOR Defines a special method Class Constructor method,

used to create objects. This is usually the New() method. Constructors always return the new object.

INLINE Fast and easy to code, INLINE lets you define the

code for the method immediately within the definition of the Class. Any methods not declared INLINE or BLOCK must be fully defined after the ENDCLASS command. The Code, ... following INLINE receives a parameter of Self. If you need to receive more parameters, use the BLOCK clause instead.

BLOCK Use this clause when you want to declare fast 'inline'

methods that need parameters. The first parameter to CodeBlock must be Self, as in:

METHOD <MethodName> BLOCK {| Self, <arg1>, <arg2>, ..., <argN> | ... }

EXTERN If an external function does what the method needs,

use this clause to make an optimized call to that function directly.

SETGET For calculated Data. The name of the method can be

manipulated like a Data element to Set or Get a value.

VIRTUAL Methods that do nothing. Useful for Base classes where

the child class will define the method's behavior, or when you are first creating and testing a Class.

OPERATOR Operator Overloading for classes.

See example tests/testop.prg for details.

CLASS ClassName

Use this syntax only for defining a full method after the ENDCLASS command.

Examples

```
// FIXME
```

```
#include "hbclass.ch"
CREATE CLASS TWindow
   VAR    hWnd, nOldProc
   METHOD New() CONSTRUCTOR
   METHOD Capture() INLINE  SetCapture( ::hWnd )
   METHOD End() BLOCK  {| Self, lEnd | iif( lEnd := ::lValid(), ;
      ::PostMsg( "close" ), ), lEnd }
   METHOD EraseBkGnd( hDC )
   METHOD cTitle( cNewTitle ) SETGET
   METHOD Close() VIRTUAL
ENDCLASS

METHOD New() CLASS TWindow
   LOCAL nVar, cStr
   // ... <code> ...
   // ... <code> ...
   RETURN Self
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Method
See also
`CLASS`, `HBClass()`, `VAR`

ON ERRORSource code^ | ☰ | Improve this doc

Designate a method as an error handler for the class
Syntax

```
ON ERROR <MethodName>( [<params,...>] )
```

Arguments
`MethodName` Name of the method to define
`params,...` Optional parameter list
Description
`ON ERROR` is a synonym for `ERROR HANDLER`. It names the method that should handle errors for the class being defined.
Examples

```
#include "hbclass.ch"

CREATE CLASS TWindow
   ON ERROR MyErrHandler()
ENDCLASS

METHOD MyErrHandler() CLASS TWindow
   RETURN Self
```

Status
Ready
Compliance
Harbour specific
Platforms
Available on all platforms
Tags
Class, Method

See also
CLASS, ERROR HANDLER, METHOD, VAR

COPY STRUCTURESource code^ | ☰ | Improve this doc

Create a new database based on current database structure
Syntax
```
COPY STRUCTURE TO <xcFileName> [FIELDS <field,...>]
```
Arguments
`TO xcFileName` is the name of the new database file to create. `.dbf` is the default extension if none is given. It can be specified as a literal file name or as a character expression enclosed in parentheses.
`FIELDS` field,...` is an optional list of field names to copy from the currently open database in the specified order, the default is all fields. Names could be specified as uppercase or lowercase.
Description

`COPY STRUCTURE` create a new empty database file with a structure that is based on the currently open database in this work-area.

`COPY STRUCTURE` can be use to create a sub-set of the currently open database, based on a given field list.

`COPY STRUCTURE` command is preprocessed into `__dbCopyStruct()` function during compile time.

Examples
```
// Create a new file that contains the same structure
USE test
COPY STRUCTURE TO MyCopy

// Create a new file that contains part of the original structure
USE test
COPY STRUCTURE TO SomePart FIELDS name, address
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Database
See also
`__dbCopyStruct()`, `__dbCopyXStruct()`, `__dbCreate()`, `__dbStructFilter()`, COPY STRUCTURE EXTENDEDClipper 5.3, `dbCreate()` Clipper 5.3, `dbStruct()` Clipper 5.3

COPY STRUCTURE EXTENDEDSource code^ | ☰ | Improve this doc

Copy current database structure into a definition file
Syntax
```
COPY STRUCTURE EXTENDED TO <xcFileName>
```
Arguments

`xcFileName` The name of the target definition file to create. `.dbf` is the default extension if none is given. It can be specified as a literal file name or as a character expression enclosed in parentheses.

Description

`COPY STRUCTURE EXTENDED` create a new database named `cFileName` with a pre-defined structure (also called "structure extended file"):

Field name	Type	Length	Decimals
FIELD_NAME	C	10	0
FIELD_TYPE	C	1	0
FIELD_LEN	N	3	0
FIELD_DEC	N	3	0

Each record in the new file contains information about one field in the original file. `CREATE FROM` could be used to create a database from the structure extended file.

For prehistoric compatibility reasons, Character fields which are longer than 255 characters are treated in a special way by writing part of the length in the `FIELD_DEC` according to the following formula (this is done internally):

```
FIELD->FIELD_DEC := Int( nLength / 256 )
FIELD->FIELD_LEN :=      nLength % 256
```

Later if you want to calculate the length of a field you can use the following formula:

```
nLength := iif( FIELD->FIELD_TYPE == "C", ;
                FIELD->FIELD_DEC * 256 + FIELD->FIELD_LEN, ;
                FIELD->FIELD_LEN )
```

`COPY STRUCTURE EXTENDED` command is preprocessed into `__dbCopyXStruct()` function during compile time.

Examples

```
// Open a database, then copy its structure to a new file,
// Open the new file and list all its records
USE test
COPY STRUCTURE EXTENDED TO teststru
USE teststru
LIST
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

Command, Database

See also

`__dbCopyStruct()`, `__dbCopyXStruct()`, `__dbCreate()`, `COPY STRUCTURE`Clipper 5.3, `CREATE`Clipper 5.3, `CREATE FROM`Clipper 5.3, `dbCreate()` Clipper 5.3, `dbStruct()` Clipper 5.3

CREATESource code^ | ☰ | Improve this doc

Create empty structure extended file
Syntax
```
CREATE <xcFileName> [VIA <xcRDDName>] [ALIAS <xcAlias>]
```
Arguments
`xcFileName` is the target file name to create and then open. `.dbf` is the default extension if none is given. It can be specified as literal file name or as a character expression enclosed in parentheses.
`VIA xcRDDName` is RDD name to create target with. If omitted, the default RDD is used. It can be specified as literal name or as a character expression enclosed in parentheses.
`ALIAS xcAlias` is an optional alias to `USE` the target file with. If not specified, alias is based on the root name of `xcFileName`.
Description

`CREATE` a new empty structure extended file with the name `cFileName` and then open it in the current work-area. The new file has the following structure:

Field name	Type	Length	Decimals
FIELD_NAME	C	10	0
FIELD_TYPE	C	1	0
FIELD_LEN	N	3	0
FIELD_DEC	N	3	0

`CREATE` command is preprocessed into `__dbCopyStruct()` function during compile time and use this mode.

Examples
```
// CREATE a new structure extended file, append some records and
// then CREATE FROM this file a new database file

CREATE template
dbAppend()
FIELD->FIELD_NAME := "CHANNEL"
FIELD->FIELD_TYPE := "N"
FIELD->FIELD_LEN  := 2
FIELD->FIELD_DEC  := 0
dbAppend()
FIELD->FIELD_NAME := "PROGRAM"
FIELD->FIELD_TYPE := "C"
FIELD->FIELD_LEN  := 20
FIELD->FIELD_DEC  := 0
dbAppend()
FIELD->FIELD_NAME := "REVIEW"
FIELD->FIELD_TYPE := "C"      // this field is 1000 char long
FIELD->FIELD_LEN  := 232      // 1000 % 256 = 232
FIELD->FIELD_DEC  := 3        // 1000 / 256 = 3
dbCloseArea()
CREATE TV_Guide FROM template
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms

Tags
Command, Database
See also
__dbCopyStruct(), __dbCopyXStruct(), __dbCreate(), COPY STRUCTURE Clipper 5.3, COPY STRUCTURE EXTENDED Clipper 5.3, CREATE FROM Clipper 5.3, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

CREATE FROM Source code^ | ☰ | Improve this doc

Create new database file from a structure extended file
Syntax

```
CREATE <xcFileName> FROM <xcFileFrom> [VIA <xcRDDName>] [NEW] [ALIAS
<xcAlias>]
```

Arguments
`xcFileName` is the target file name to create and then open. `.dbf` is the default extension if none is given. It can be specified as literal file name or as a character expression enclosed in parentheses.
`FROM xcFileFrom` is a structure extended file name from which the target file `xcFileName` is going to be built. It can be specified as literal file name or as a character expression enclosed in parentheses.
`VIA xcRDDName` is RDD name to create target with. If omitted, the default RDD is used. It can be specified as literal name or as a character expression enclosed in parentheses.
`NEW` open the target file name `xcFileName` in the next available unused work-area and making it the current work-area. If omitted open the target file in current work-area.
`ALIAS xcAlias` is an optional alias to `USE` the target file with. If not specified, alias is based on the root name of `xcFileName`.
Description

`CREATE FROM` open a structure extended file `xcFileFrom` where each record contain at least the following fields (in no particular order): `FIELD_NAME`, `FIELD_TYPE`, `FIELD_LEN` and `FIELD_DEC`. Any other field is ignored. From this information the file `xcFileName` is then created and opened in the current or new work-area (according to the `NEW` clause), if this is a new work-area it becomes the current.

For prehistoric compatibility reasons, structure extended file Character fields which are longer than 255 characters should be treated in a special way by writing part of the length in the `FIELD_DEC` according to the following formula:

```
FIELD->FIELD_DEC := Int( nLength / 256 )
FIELD->FIELD_LEN :=      nLength % 256
```

`CREATE FROM` command is preprocessed into `__dbCopyStruct()` function during compile time and uses this mode.

Examples

```
// See example in the CREATE command
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms

Available on all platforms
Tags
Command, Database
See also
__dbCopyStruct(), __dbCopyXStruct(), __dbCreate(), COPY STRUCTURE Clipper 5.3, COPY STRUCTURE EXTENDED Clipper 5.3, CREATE Clipper 5.3, dbCreate() Clipper 5.3, dbStruct() Clipper 5.3

PACK Source code^ | ☰ | Improve this doc

Remove records marked for deletion from a database
Syntax
```
PACK
```
Arguments
(This command has no arguments)
Description

This command removes records that were marked for deletion from the currently selected database. This command does not pack the contents of a memo field; those files must be packed via low-level functions.

All open index files will be automatically reindexed once PACK command has completed its operation. On completion, the record pointer is placed on the first record in the database.

Examples
```
USE test
dbGoto( 10 )
DELETE NEXT 10
? LastRec()
PACK
? LastRec()
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Database
See also
dbEval() Clipper 5.3, Deleted() Clipper 5.3, DELETE, RECALL, ZAP Clipper 5.3

ZAP Source code^ | ☰ | Improve this doc

Remove all records from the current database file
Syntax
```
ZAP
```
Arguments
(This command has no arguments)
Description
This command removes all of the records from the database in the current work area. This operation also updates any index file in use at the time of this operation. In addition, this

command removes all items within an associated memo file. In a network environment, any file that is about to be ZAPped must be used exclusively.
Examples
```
USE test
? LastRec()  // --> 500
ZAP
? LastRec()  // --> 0
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Database
See also
DELETE, PACKClipper 5.3, USE

SET ALTERNATESource code^ | ☰ | Improve this doc

Toggle and echos output to an alternate file
Syntax
```
SET ALTERNATE TO <cFile> [ADDITIVE]
SET ALTERNATE on | OFF | ( <lAlter> )
```
Arguments
`cFile` Name of alternate file.
`lAlter` Logical expression for toggle
Description
This command toggles and output console information to the alternate file `cFile`, provided that the command is toggled on or the condition `lAlter` is set to a logical true (`.T.`). If `cFile` does not has a file extension, .txt will be assumed. The file name may optionally have a drive letter and/or directory path. If none is specified, the current drive and directory will be used. If the `ALTERNATE` file is created but no `ALTERNATE ON` command is issued, nothing will be echoed to the file. If `ADDITIVE` clause is used, then the information will be appended to the existing alternate file. Otherwise, a new file will be created with the specified name (or an existing one will be overwritten) and the information will be appended to the file. The default is to create a new file. A `SET ALTERNATE TO` command will close the alternate file
Examples
```
SET ALTERNATE TO test.txt
SET ALTERNATE ON
? "Harbour"
? "is"
? "Power"
SET ALTERNATE TO
SET ALTERNATE OFF
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment

See also
CLOSE, Set() Clipper 5.3, SET CONSOLE Clipper 5.3, SET PRINTER Clipper 5.3

SET BELL Source code^ | ☰ | Improve this doc

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

```
SET BELL on | OFF | ( <lBell> )
```

Arguments
lBell Logical expression for toggle command
Description

This command toggles the bell to sound whenever a character is entered into the last character position of a GET, or if an invalid data type is entered into a GET.

If lBell is a logical true (.T.), the bell will be turned ON; otherwise, the bell will be turned off.

Examples

```
LOCAL cDummy, GetList := {}
SET BELL ON
cDummy := Space( 20 )
@ 3, 2 GET cDummy
READ
SET BELL OFF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
Set() Clipper 5.3

SET CENTURY Source code^ | ☰ | Improve this doc

Toggle the century digits in all dates display
Syntax

```
SET CENTURY on | OFF | ( <lCent> )
```

Arguments
lCent Logical expression for toggle
Description
This command allows the input and display of dates with the century prefix. It will be in the standard MM/DD/YYYY format unless specified by the SET DATE command or Set() function. If lCent is a logical true (.T.), the command will be set on; otherwise, the command will be set off
Examples

```
SET CENTURY ON
? Date()
SET CENTURY OFF
```

Status

Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
CToD() Clipper 5.3, Date() Clipper 5.3, DToC() Clipper 5.3, Set() Clipper 5.3, SET DATEClipper 5.3, SET EPOCHClipper 5.3

SET CONSOLESource code^ | ☰ | Improve this doc

Toggle the console display
Syntax

```
SET CONSOLE ON | off | ( <lConsole> )
```

Arguments
lConsole Logical expression for toggle command
Description

This command turns the screen display either off or on for all screens display other then direct output via the @...SAY commands or the - DevOut() function.

If lConsole is a logical true (.T.), the console will be turned ON; otherwise, the console will be turned off.

Examples

```
SET CONSOLE ON
? Date()
SET CONSOLE OFF
? Date()
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
Set() Clipper 5.3, SET DEVICEClipper 5.3

SET DATESource code^ | ☰ | Improve this doc

Assigns a date format or chooses a predefined date data set.
Syntax

```
SET DATE FORMAT [TO] <cFormat>

SET DATE [TO] [ANSI / BRITISH / FRENCH / GERMAN / ITALIAN / JAPAN / USA /
AMERICAN]
```

Arguments
cFormat Keyword for date format

Description

This command sets the date format for function display purposes. If specified, `cFormat` may be a customized date format in which the letters d, m and y may be used to design a date format. The default is an `AMERICAN` date format; specifying no parameters will set the date format to `AMERICAN`. Below is a table of the various predefined dates formats.

Syntax	Date Format
ANSI	yy.mm.dd
BRITISH	dd/mm/yy
FRENCH	dd/mm/yy
GERMAN	dd.mm.yy
ITALIAN	dd-mm-yy
JAPAN	yy.mm.dd
USA	mm-dd-yy
AMERICAN	mm/dd/yy

Examples

```
SET DATE JAPAN
? Date()
SET DATE GERMAN
? Date()
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

Command, Environment

See also

`CToD()` Clipper 5.3, `Date()` Clipper 5.3, `DToC()` Clipper 5.3, `Set()` Clipper 5.3, `SET DATE` Clipper 5.3, `SET EPOCH` Clipper 5.3

`SET DECIMALS` Source code^ | ☰ | Improve this doc

Toggle the console display

Syntax

```
SET DECIMALS TO [<nDecimal>]
```

Arguments

`nDecimal` Number of decimals places

Description

This command establishes the number of decimal places that Harbour will display in mathematical calculations, functions, memory variables, and fields. Issuing no parameter with this command will the default number of decimals to 0. For decimals to be seen, the `SET FIXED ON` command must be activated.

Examples

```
SET FIXED ON
? 25141251 / 362
SET DECIMALS TO 10
? 214514.214 / 6325
```

Status

Ready

Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
Set() Clipper 5.3, SET FIXEDClipper 5.3

SET DEFAULTSource code^ | ☰ | Improve this doc

Establishes the Harbour search drive and directory.
Syntax

```
SET DEFAULT TO [<cPath>]
```

Arguments
cPath Drive and/or path.
Description
This command changes the drive and directory used for reading and writing database, index, memory, and alternate files. Specifying no parameters with this command will default the operation to the current logged drive and directory.
Examples

```
SET DEFAULT TO /hb/tests
Set( _SET_DEFAULT, hb_DirSepToOS( "/hb/tests" ) )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
CurDir() Clipper 5.3, Set() Clipper 5.3, SET PATHClipper 5.3

SET DEVICESource code^ | ☰ | Improve this doc

Directs all @...SAY output to a device.
Syntax

```
SET DEVICE TO [printer | SCREEN ]
```

Arguments
None.
Description

This command determines whether the output from the @...SAY command and the DevPos() and DevOut() function will be displayed on the printer.

When the device is set to the PRINTER, the SET MARGIN value adjusts the position of the column values accordingly. Also, an automatic page eject will be issued when the current printer head position is less than the last printed row. Finally, if used in conjunction with the @...GET commands, the values for the GETs will all be ignored.

Examples

```
SET DEVICE TO SCREEN
? 25141251 / 362
SET DEVICE TO PRINTER
SET PRINTER TO LPT1
? 214514.214 / 6325
SET PRINTER OFF
SET DEVICE TO SCREEN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
`@...SAY`Clipper 5.3, `Set()` Clipper 5.3, `SetPRC()`, `SET PRINTER`Clipper 5.3

`SET EPOCH`Source code^ | ☰ | Improve this doc

Specify a base year for interpreting dates
Syntax

```
SET EPOCH TO <nEpoch>
```

Arguments
`nEpoch` Base Century.
Description
This command sets the base year value for dates that have only two digits. The default setting is 1900. Dates between 0100-01-01 and 2999-12-31 are fully supported.
Examples

```
SET EPOCH TO 2000
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
`CToD()` Clipper 5.3, `Date()` Clipper 5.3, `DToC()` Clipper 5.3, `Set()` Clipper 5.3, `SET CENTURY`Clipper 5.3, `SET DATE`Clipper 5.3

`SET FIXED`Source code^ | ☰ | Improve this doc

Set the number of decimal position to be displayed
Syntax

```
SET FIXED on | OFF | ( <lFixed> )
```

Arguments
`lFixed` Logical expression for toggle
Description

This command activates a system wide fixed placement of decimals places shown for all numeric outputs. If the value of `lFixed` is a logical true (`.T.`), `FIXED` will be turned `ON`; otherwise it will be turned `OFF`.

When `SET DECIMALS OFF` is used, the following rules apply to the number of decimal placed displayed.

Addition	Same as operand with the greatest number of decimal digits
Subtraction	Same as operand with the greatest number of decimal digits
Multiplication	Sum of operand decimal digits
Division	Determined by SET DECIMALS TO
Exponential	Determined by SET DECIMALS TO
Log()	Determined by SET DECIMALS TO
Exp()	Determined by SET DECIMALS TO
Sqrt()	Determined by SET DECIMALS TO
Val()	Determined by SET DECIMALS TO

Examples

```
SET FIXED ON
? 25141251 / 362
SET FIXED OFF
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
`Exp()` Clipper 5.3, `Log()` Clipper 5.3, `Set()` Clipper 5.3, `SET DECIMALS` Clipper 5.3, `Sqrt()` Clipper 5.3, `Val()` Clipper 5.3

`SET FUNCTION` Source code^ | ☰ | Improve this doc

Assign a character string to a function key
Syntax

```
SET FUNCTION <nFunctionKey> TO [<cString>]
```

Arguments
`nFunctionKey` is a number in the range 1..40 that represent the function key to be assigned. `cString` is a character string to set. If `cString` is not specified, the function key is going to be set to `NIL` releasing by that any previous Set Function or `SetKey()` for that function.
Description

Set Function assign a character string with a function key, when this function key is pressed, the keyboard is stuffed with this character string. Set Function has the effect of clearing any `SetKey()` previously set to the same function number and vice versa.

nFunctionKey	Key to be set
1 .. 12	<F1> .. <F12>
13 .. 20	<Shift+F3> .. <Shift+F10>
21 .. 30	<Ctrl+F1> .. <Ctrl+F10>
31 .. 40	<Alt+F1> .. <Alt+F10>

`SET FUNCTION` command is preprocessed into `__SetFunction()` function during compile time.

Examples

```
#include "inkey.ch"
LOCAL cTest, GetList := {}
// Associate <F1> with a string
SET FUNCTION 1 TO "I Am Lazy" + Chr( K_ENTER )
CLS
cTest := Space( 20 )
@ 10, 0 SAY "type something or <F1> for lazy mode" GET cTest
READ
? cTest
```

Status

Ready

Compliance

Harbour use 11 and 12 to represent F11 and F12, while CA-Cl*pper use 11 and 12 to represent Shift+F1 and Shift+F2.

Platforms

Available on all platforms

Tags

Command, Environment

See also

`__Keyboard()`, `Inkey()` Clipper 5.3, `SetKey()` Clipper 5.3

SET INTENSITY Source code^ | ☰ | Improve this doc

Toggles the enhanced display of PROMPTs and GETs.

Syntax

```
SET INTENSITY ON | off | ( <lInte> )
```

Arguments

`lInte` Logical expression for toggle command

Description

This command set the field input color and `@...PROMPT` menu color to either highlighted (inverse video) or normal color. The default condition is `ON` (highlighted).

Examples

```
SET INTENSITY ON
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

Command, Environment

See also

`@...GET` Clipper 5.3, `@...PROMPT` Clipper 5.3, `@...SAY` Clipper 5.3, `Set()` Clipper 5.3

SET KEY Source code^ | ☰ | Improve this doc

Assign an action block to a key

Syntax

```
SET KEY <anKey> TO <bAction>] [WHEN <bCondition>]
```

Arguments
`anKey` is either a numeric key value, or an array of such values
`bAction` is an optional code-block to be assigned
`bCondition` is an optional condition code-block
Description
The `SET KEY` Command function is translated to the `SetKey()` function witch returns the current code-block assigned to a key when called with only the key value. If the action block (and optionally the condition block) are passed, the current block is returned, and the new code block and condition block are stored. A group of keys may be assigned the same code block/condition block by using an array of key values in place on the first parameter.
Examples

```
#include "getexit.ch"
#include "inkey.ch"
LOCAL bOldF10 := SetKey( K_F10, {|| Yahoo() } )
LOCAL bBlock
// some other processing ...
SET KEY K_F10 TO bOldF10
// some other processing ...
bBlock := SetKey( K_SPACE )
IF bBlock != NIL  // ...

   // make <F10> exit current get, but only if in a get - ignores other
   // wait-states such as menus, achoices, etc...
   SetKey( K_F10, {|| GetActive():State := GE_WRITE }, ;
      {|| GetActive() != NIL } )
ENDIF
```

Status
Ready
Compliance
`SET KEY` is mostly CA-Cl*pper compliant. The only difference is the addition of the condition code-block parameter, allowing set-keys to be conditionally turned off or on. This condition-block cannot be returned once set - see `hb_SetKeyGet()`
Platforms
Available on all platforms
Tags
Command, Environment
See also
`hb_SetKeyGet()`, `hb_SetKeySave()`

SET MESSAGESource code^ | ☰ | Improve this doc

Establishes a message row for `@...PROMPT` command
Syntax

```
SET MESSAGE TO [<nRow> [CENTER]]
```

Arguments
`nRow` Row number to display the message
Description

This command is designed to work in conjunction with the `MENU TO` and `@...PROMPT` commands. With this command, a row number between 0 and `MaxRow()` may be specified in `nRow`. This establishes the row on witch any message associated with an `@...PROMPT` command will appear.

If the value of nRow is 0, all messages will be suppressed. All messaged will be left-justifies unless the CENTER clause is used. In this case, the individual messages in each @...PROMPT command will be centered at the designated row (unless nRow is 0). All messages are independent; therefore, the screen area is cleared out by the centered message will vary based on the length of each individual message.

Specifying no parameters with this command set the row value to 0, witch suppresses all messages output. The British spelling of CENTRE is also supported.

Examples
See tests/menuto.prg
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
@...PROMPTClipper 5.3, MENU TOClipper 5.3, Set()Clipper 5.3, SET WRAPClipper 5.3

SET PATHSource code^ | ☰ | Improve this doc

Specifies a search path for opening files
Syntax
```
SET PATH TO [<cPath>]
```
Arguments
cPath Search path for files
Description
This command specifies the search path for files required by most commands and functions not found in the current drive and directory. This pertains primarily, but not exclusively, to databases, indexes, and memo files, as well as to memory, labels and reports files. The search hierarchy is: 1 Current drive and directory, 2 The SET DEFAULT path; 3 The SET PATH path.
Examples
```
SET PATH TO /hb/tests
Set( _SET_PATH, hb_DirSepToOS( "/hb/tests" ) )
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
CurDir()Clipper 5.3, Set()Clipper 5.3, SET DEFAULTClipper 5.3

SET PRINTERSource code^ | ☰ | Improve this doc

Toggles the printer and controls the printer device

Syntax

```
SET PRINTER on | OFF
SET PRINTER ( <lPrinter> )
SET PRINTER TO [<cPrinter>] [ADDITIVE]
```

Arguments

`lPrinter` Logical condition by which to toggle the printer

`cPrinter` A device name or an alternate name

Description

This command can direct all output that is not controlled by the `@...SAY` command and the `DevPos()` and `DevOut()` functions to the printer. If specified, the condition `lPrinter` toggles the printer `ON` if a logical true (`.T.`) and `OFF` if a logical false (`.F.`). If no argument is specified in the command, the alternate file (if one is open) is closed, or the device is reselected and the `PRINTER` option is turned `OFF`.

If a device is specified in `cPrinter`, the output will be directed to that device instead of to the `PRINTER`. A specified device may be a literal string or a variable, as long as the variable is enclosed in parentheses. For a network, do not use a trailing colon when redirecting to a device.

If an alternate file is specified, `cPrinter` becomes the name of a file that will contain the output. If no file extension is specified an extension of .prn will be defaulted to.

If the `ADDITIVE` clause is specified, the information will be appended to the end of the specified output file. Otherwise, a new file will be created with the specified name (or an existing file will first be cleared) and the information will then be appended to the file. The default is to create a new file.

Examples

```
SET PRINTER ON
SET PRINTER TO LPT1
? 25141251 / 362
SET PRINTER ( .F. )
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

Command, Environment

See also

`DevOut()`, `Set()` Clipper 5.3, `SET CONSOLE` Clipper 5.3, `SET DEVICE` Clipper 5.3

SET WRAP Source code^ | ☰ | Improve this doc

Toggle wrapping the PROMPTs in a menu.

Syntax

```
SET WRAP on | OFF | ( <lWrap> )
```

Arguments

`lWrap` Logical expression for toggle

Description

This command toggles the highlighted bars in a `@...PROMPT` command to wrap around in a bottom-to-top and top-to-bottom manner. If the value of the logical expression `lWrap` is a logical false (`.F.`), the wrapping mode is set `OFF`; otherwise, it is set `ON`.
Examples
```
// See tests/menuto.prg
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Environment
See also
`@...PROMPT`Clipper 5.3, `MENU TO`Clipper 5.3

`COPY FILE`Source code^ | ☰ | Improve this doc

Copies a file.
Syntax
```
COPY FILE <cFile> TO <cFile1>
```
Arguments
`cFile` File name of source file `cFile1` File name of target file
Description
This command makes an exact copy of `cFile` and names it `cFile1`. Both files must have the file extension included; the drive and the directory names must also be specified if they are different from the default drive and/or director. `cFile1` also can refer to a OS device (e.g. `LPT1`). This command does not observe the `SET PATH TO` or `SET DEFAULT TO` settings.
Examples
```
COPY FILE test.dbf TO adir.prg
COPY FILE test.txt TO LPT1
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, FileSys
See also
`ERASE`Clipper 5.3, `FErase()` Clipper 5.3, `FRename()` Clipper 5.3, `RENAME`Clipper 5.3

`DELETE FILE`Source code^ | ☰ | Improve this doc

Remove a file from disk
Syntax
```
DELETE FILE <xcFile>
```
Arguments
`xcFile` Name of file to remove
Description

This command removes a file from the disk. The use of a drive, directory, and wild-card skeleton operator is allowed for the root of the file name. The file extension is required. The SET DEFAULT and SET PATH commands do not affect this command.

The file must be considered closed by the operating system before it may be deleted.

Examples
```
DELETE FILE test.txt
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, FileSys
See also
CurDir() Clipper 5.3, ERASEClipper 5.3, FErase() Clipper 5.3, File() Clipper 5.3

DIRSource code^ | ☰ | Improve this doc

Display listings of files
Syntax
```
DIR [<cFileMask>]
```
Arguments
cFileMask File mask to include in the function return. It could contain path and standard wildcard characters as supported by your OS (like * and ?). If cFileMask contains no path, then SET DEFAULT path is used to display files in the mask.
Description

If no cFileMask is given, __Dir() display information about all *.dbf in the SET DEFAULT path, this information contain: file name, number of records, last update date and the size of each file.

If cFileMask is given, __Dir() list all files that match the mask with the following details: Name, Extension, Size, Date.

DIR command is preprocessed into __Dir() function during compile time.

__Dir() is a compatibility function, it is superseded by Directory() which returns all the information in a multidimensional array.

If long file names are available Harbour will use/display the first 15 characters else Harbour will use/display a 8.3 file name consistent with CA-Cl*pper.

Examples
```
DIR          // information for all DBF files in current directory

DIR "*.dbf"          // list all DBF file in current directory

// list all PRG files in Harbour Run-Time library
// for MS-DOS compatible operating systems
```

```
DIR "src/rtl/*.prg"

// list all files in the public section on a Unix like machine
DIR "/pub"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, FileSys
See also
__Dir()*, ADir() Clipper 5.3, Directory(), SET DEFAULTClipper 5.3

ERASESource code^ | ☰ | Improve this doc

Remove a file from disk
Syntax
```
ERASE <xcFile>
```
Arguments
xcFile Name of file to remove
Description

This command removes a file from the disk. The use of a drive, directory, and wild-card skeleton operator is allowed for the root of the file name. The file extension is required. The SET DEFAULT and SET PATH commands do not affect this command.

The file must be considered closed by the operating system before it may be deleted.

Examples
```
ERASE test.txt
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, FileSys
See also
CurDir() Clipper 5.3, DELETE FILEClipper 5.3, FErase() Clipper 5.3, File() Clipper 5.3

RENAMESource code^ | ☰ | Improve this doc

Changes the name of a specified file
Syntax
```
RENAME <cOldFile> TO <cNewFile>
```
Arguments
cOldFile Old file name
cNewFile New File name
Description

This command changes the name of cOldFile to cNewFile. Both cOldFile and cNewFile must include a file extension. This command if not affected by the SET PATH TO or SET DEFAULT TO commands; drive and directory designates must be specified if either file is in a directory other then the default drive and directory.

If cNewFile id currently open or if it previously exists, this command will not perform the desired operation.

Examples
```
RENAME test.txt TO test.old
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
Command, FileSys
See also
CurDir() Clipper 5.3, ERASEClipper 5.3, FErase() Clipper 5.3, File() Clipper 5.3, FRename() Clipper 5.3

TYPESource code^ | ☰ | Improve this doc

Show the content of a file on the console, printer or file
Syntax
```
TYPE <xcFile> [TO PRINTER] [TO FILE <xcDestFile>]
```
Arguments
xcFile is a name of the file to display. If the file have an extension, it must be specified (there is no default value). It can be specified as literal file name or as a character expression enclosed in parentheses.
TO PRINTER is an optional keyword that specifies that the output should go to both the screen and printer.
TO FILE xcDestFile copy the source xcFile also to a file. If no extension is given .txt is added to the output file name. xcDestFile can be specified as literal file name or as a character expression enclosed in parentheses.
Description

TYPE command type the content of a text file on the screen with an option to send this information also to the printer or to an alternate file. The file is displayed as is without any headings or formatting.

If xcFile contain no path, TYPE try to find the file first in the SET DEFAULT directory and then in search all of the SET PATH directories. If xcFile cannot be found a run-time error occur.

If xcDestFile contain no path it is created in the SET DEFAULT directory.

Use SET CONSOLE OFF to suppress screen output. You can pause the output using Ctrl-S, press any key to resume.

Examples

```
// The following examples assume a file name `test.txt` exist in all
// specified paths, a run-time error would displayed if it does not

// display test.txt file on screen
TYPE test.txt

// display test.txt file on screen and printer
TYPE test.txt TO PRINTER

// display test.txt file on printer only
SET CONSOLE OFF
TYPE test.txt TO PRINTER
SET CONSOLE ON

// display test.txt file on screen and into a file myreport.txt
TYPE test.txt TO FILE myreport
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

Tags

Command, FileSys

See also

__TypeFile(), COPY FILEClipper 5.3, SET DEFAULTClipper 5.3, SET PATHClipper 5.3, SET PRINTERClipper 5.3

LABEL FORMSource code^ | ☰ | Improve this doc

Displays labels to the screen or an alternate device

Syntax

```
LABEL FORM <cLabelName> [TO PRINTER] [TO FILE <cFile>] [<cScope>]
           [WHILE <bWhile> ] [FOR <bFor> ] [SAMPLE] [NOCONSOLE]
```

Arguments

cLabelName Name of label file cFile Name of an alternate file cScope Expression of a scoping condition bWhile WHILE condition bFor FOR condition

Description

This command allows labels to be printed based on the format outlined in LBL file specified as cLabelName. By default, output will go to the screen however this output may be rerouted with either the TO PRINTER or the TO FILE clause.

If the TO FILE clause is specified, the name of the ASCII text file containing the generated labels will be cFile.

If no file extension is specified a .txt extension is added. cScope is the scope condition for this command. Valid scopes include NEXT <expN> (number of records to be displayed, where expN is the number of records), RECORD <expN> (a specific record to be printed), REST (all records starting from the current record position, and ALL (all records). The default is ALL.

Both logical expression may work ill conjunction with one another where bFor is the logical expression for the FOR condition (for records to be displayed within a given value range) and

bWhile for the WHILE condition (for records to be displayed until they fail to meet the condition).

If the SAMPLE clause is specified, test labels will be generated.

If the NOCONSOLE clause is specified, the console will be turned off while this command is being executed.

This command follows the search criteria outlined in the SET PATH TO command. The path may be specified, along, with (the drive letter, in cLabelName

Examples

```
USE test NEW
LABEL FORM test.lbl
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
Command, Legacy
See also
REPORT FORMClipper 5.3

REPORT FORMSource code^ | ☰ | Improve this doc

Display a report
Syntax

```
REPORT FORM <cReportName> [TO PRINTER] [TO FILE <cFile>] [<cScope>]
            [WHILE <bWhile> ] [FOR <bFor> ]
            [PLAIN |HEADING <cHeading>] [NOEJECT] [SUMMARY]
            [NOCONSOLE]
```

Arguments
cReportName Name of report file
cFile Name of alternate file
cScope Scope.
bWhile Logical expression of WHILE condition .
bFor Logical expression of FOR condition.
cHeading Report heading
Description

This command prints out the report named cReportName, which is a standard FRM file. The file extension is not required because .frm will be assumed. The SET PATH TO and SET DEFAULT TO commands affect the search for the file cReportName; unless a drive and path are specified in cReportName, REPORT will search the path specified in the SET PATH command if it cannot find the report form in the current directory.

The output of the report will be offset based on the setting of the SET MARGIN TO value.

By default, output will go to the console; however, it may be controlled via either the `TO PRINTER` or `TO FILE` clause. If the output is to go to the file, the name of the alternate file is specified in `cFile`. Unless specified in `cFile`, the default file extension will be `.txt`.

`cScope` is the scope for this command. Valid scopes include `NEXT <expN>` (where `expN` is the number of records), `RECORD <expN>` (a specific record to be displayed), `REST` (all records from the current record position), and `ALL` (all records). The default is `ALL`.

Both logical expressions may work in conjunction with one another, where `bFor` is the logical expression for the `FOR` condition (for records to be displayed within a given range) and `bWhile` for the `WHILE` condition (for records to be displayed until the condition fails).

If the `PLAIN` clause is specified, date and page numbers are suppressed. In addition, there is no automatic page breaking, and the report title and column headings appear only once at the top of the form.

If the `HEADING` clause is used, `cHeading` is displayed on the first title of each report page. The value of `cHeading` is evaluated only once before executing the report; varying the values of `cHeading` is not allowed. The `PLAIN` clause will take precedence over the `HEADING` clause if both are included.

If the `NOEJECT` clause is used, the initial page eject on the report will not be issued when the output clause `TO PRINTER` is specified. Otherwise, this clause has no effect.

If the `SUMMARY` Clause is specified, the report will contain only groups, subgroups, and grand total information. The detailed title item information will be ignored.

If the `NOCONSOLE` clause is specified, output to the console will be turned off while this command is being executed.

Examples

```
USE test NEW
REPORT FORM test.frm
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
Library is core
Tags
Command, Legacy
See also
`LABEL FORM`Clipper 5.3

EJECT^ | ☰ | Improve this doc

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

```
EJECT
```

Arguments
None
Description

This command issue an form-feed command to the printer. If the printer is not properly hooked up to the computer, an error will not be generated and the command will be ignored.

Once completed, the values of `PRow()` and `PCol()`, the row and column indicators to the printer, will be set to 0. Their values, however, may be manipulated before or after issuing an `EJECT` by using the `DevPos()` function.

On compile time this command is translated into `__Eject()` function.

Examples

```
LOCAL Curpos
USE test NEW
SET DEVICE TO PRINTER
Curpos := 0
DO WHILE ! Eof()
   ? test->first, test->last
   Curpos++
   IF Curpos > 59
      Curpos := 0
      EJECT
   ENDIF
ENDDO
SET DEVICE TO SCREEN
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, Printer
See also
`DevPos()`, `PCol()`, `PRow()`, `SET PRINTER`Clipper 5.3

`@...GET`Source code^ | ☰ | Improve this doc

Creates a GET object and displays it to the screen
Syntax

```
@ <nRow>, <nCol> [SAY <cSay> [PICTURE <cSayPict>] COLOR <cSayColor> ]
GET <xVar> [PICTURE <cGetPict>] [WHEN <lWhen>] [COLOR <cGetColor>]
[VALID <lValid> / RANGE <xStart>, <xEnd>]
```

Arguments
`nRow` The row coordinate.
`nCol` The column coordinate.
`cSay` Message to display.
`cSayPict` Character expression of `PICTURE` displayed.
`cSayColor` Color to be Used for the `SAY` expression.
`xVar` An variable/field name.
`cGetPict` Character expression of `PICTURE` to get.
`lWhen` Logical expression to allow `GET`.

lValid Logical expression to validate GET input.
xStart Lower RANGE value.
xEnd Upper RANGE value.
cGetColor Color string to be used for the GET expression.
Description

This command adds a GET object to the reserved array variable named GetList and displays it to the screen. The field or variable to be added to the GET object is specified in xVar and is displayed at row, column coordinate nRow, nCol.

If the SAY clause is used cSay will be displayed starting at nRow, nCol, with the field variable xVar displayed at Row(), Col() + 1. If cSayPicr, the picture template for the SAY expression cSay, is used, all formatting rules contained will apply See the TRANSFORM I function for further information.

If cGetPict is specified, the PICTURE clause of xVar will be used for the GET object and all formatting rules will apply. See the table below for GET formatting rules.

If the WHEN clause is specified, when lWhen evaluates to a logical true (.T.) condition, the GET object will he activated otherwise the GET object will be skipped and no information will be obtained via the screen. The name of a user-defined function returning a logical true (.T.) or false (F.) or a code block may be, specified in lWhen This clause not activated until a READ command or ReadModal() function call is issued.

If the VALID clause is specified and lValid evaluates to it logical true (.T.) condition the current GET will be considered valid and the get operation will continue onto the next active GET object. If not, the cursor will remain on this GET object until aborted or until the condition in lValid evaluates to true (.T.). The name of a user-defined function returning a logical true (.T.) or false (.F.) or it code block may be specified in lValid. This clause is not activated until a READ command or ReadModal() function call is issued.

If the RANGE clause is specified instead of the VALID clause, the two inclusive range values for xVar must be specified in xStart and xEnd. Id xVar is a date data type, xStart and xEnd must also be date data types; if xVar is a numeric data type xStart and xEnd must also be numeric data types. If a value fails the RANGE test, the message "OUT OF RANGE" will appear in the SCOREBOARD area (row = 0, col = 60). The RANGE message may be turned off it the SET SCOREBOARD command or Set() function appropriately toggled.

Notes

GET functions/formatting rules:

@A Allows only alphabetic characters.
@B Numbers will be left justified
@C All positive numbers will be followed by CR.
@D All dates will be in the SET DATE format.
@E Dates will be in British formal: numbers in European format.
@K Allows a suggested value to be seen within the GET area but clears It if any non cursor key is pressed when the cursor is in the first Position in the GET area.

@R Non template characters will be inserted.
@S<nSize> Allows horizontal scrolling of a field or variable that
is <nSize> characters wide.
@X All negative numbers will be followed by DB
@Z Displays zero values as blanks.
@! Forces uppercase lettering
@(Displays negative numbers in parentheses with leading spaces.
@) Displays negative numbers in parentheses without leading spaces.

`GET` templates/formatting rules:

A Only alphabetic characters allowed.
N Only alphabetic and numeric characters allowed
X Any character allowed.
L Only T or F allowed For logical data.
Y Only Y or N allowed for logical data.
9 Only digits, including signs, will be allowed.
Only digits, signs. and spaces will he allowed.
! Alphabetic characters are converted to Uppercase.
$ Dollar will be displayed in place of leading
spaces for numeric data types.
* Asterisk,, will Be displayed in place of leading spaces
for numeric data types.
. Position of decimal point.
, Position of comma.

Format `PICTURE` functions may he grouped together as well as used in conjunction with a `PICTURE` templates; however, a blank space must be included in the `PICTURE` string if there are both functions and templates.

Examples

```
LOCAL cVar := Space( 50 ), nId := 0
LOCAL GetList := {}
CLS
@ 3, 1 SAY "Name" GET cVar PICTURE "@!S 30"
@ 4, 1 SAY "Id"   GET nId  PICTURE "999.999"
READ
? "The name you entered is", cVar
? "The id you entered is", nId
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, User interface
See also
`@...SAY`Clipper 5.3, `READ`, `Transform()` Clipper 5.3

`@...PROMPT`Source code^ | ☰ | Improve this doc

Display a menu item on screen and define a message
Syntax
```
@ <nRow>, <nCol> PROMPT <cPrompt> [MESSAGE <xMsg>]
```
Arguments
`nRow` is the row number to display the menu `cPrompt`. Value could range from zero to `MaxRow()`.
`nCol` is the column number to display the menu `cPrompt`. Value could range from zero to `MaxCol()`.
`cPrompt` is the menu item character string to display.
`xMsg` define a message to display each time this menu item is highlighted. `xMsg` could be a character string or code block that is evaluated to a character string. If `xMsg` is not specified or of the wrong type, an empty string ("") would be used.
Description

With `@...PROMPT` you define and display a menu item, each call to `@...PROMPT` add another item to the menu, to start the menu itself you should call the `__MenuTo()` function (`MENU TO` command). You can define any row and column combination and they will be displayed at the order of definition. After each call to `@...PROMPT`, the cursor is placed one column to the right of the last text displayed, and `Row()` and `Col()` are updated.

`@...PROMPT` command is preprocessed into `__AtPrompt()` function during compile time.

Examples
```
LOCAL nChoice
// display a two line menu with status line at the bottom
// let the user select favorite day
SET MESSAGE TO 24 CENTER
@ 10, 2 PROMPT "Sunday" MESSAGE "This is the 1st item"
@ 11, 2 PROMPT "Monday" MESSAGE "Now we're on the 2nd item"
MENU TO nChoice
DO CASE
CASE nChoice == 0           // user press <Esc> key
   QUIT
CASE nChoice == 1           // user select 1st menu item
   ? "Guess you don't like Mondays"
CASE nChoice == 2           // user select 2nd menu item
   ? "Just another day for some"
ENDCASE
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, User interface
See also
`__MenuTo()`, `AChoice()` Clipper 5.3, `MENU TO` Clipper 5.3, `SET INTENSITY` Clipper 5.3, `SET MESSAGE` Clipper 5.3, `SET WRAP` Clipper 5.3

`@...SAY` Source code^ | ☰ | Improve this doc

Displays data at specified coordinates of the current device.
Syntax

```
@ <nRow>, <nCol> SAY <xValue> [ PICTURE <cPict> ] [COLOR <cColor>]
```
Arguments
`nRow` Row coordinate
`nCol` Column coordinate
`xValue` Value to display
`cPict` `PICTURE` format
`cColor` Color string
Description

This command displays the contents of `xValue` at row column coordinates `nRow`, `nCol`. A `PICTURE` clause may be specified in `cPict`. If the current device is set to the printer, the output will go to the printer; the default is for all output to go to the screen.

For a complete list of `PICTURES` templates and functions, see the `@...GET` command.

Examples
```
CLS
@ 2, 1 SAY "Harbour"
@ 3, 1 SAY "is" COLOR "b/r+"
@ 4, 1 SAY "Power" PICTURE "@!"
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, User interface
See also
`@...GET`Clipper 5.3, `SET DEVICE`Clipper 5.3, `Transform()` Clipper 5.3

KEYBOARDSource code^ | ☰ | Improve this doc

Stuffs the keyboard with a string.
Syntax
```
KEYBOARD <cString>
```
Arguments
`cString` String to be processed, one character at a time, by the Harbour keyboard processor
Description

This command stuffs the input buffer with `cString`.

The number of characters that can be stuffed into the keyboard buffer is controlled by the `SET TYPEAHEAD` command and may range from 0 to 32622, with each character being in the ASCII range of 0 to 255.

None of the extended keys may be stuffed into the keyboard buffer.

Issuing a `KEYBOARD " "` will clear the keyboard buffer.

Examples
```
// Stuff an Enter key into the keyboard buffer
```

```
KEYBOARD Chr( 13 )
// Clear the keyboard buffer
CLEAR TYPEAHEAD

KEYBOARD Chr( 13 ); ? Inkey()  // --> 13
KEYBOARD "Hello"; CLEAR TYPEAHEAD; ? Inkey()  // --> 0
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tags
Command, User interface
See also
__Keyboard(), CLEAR TYPEAHEAD

RTE_BASE/1003^ | ☰ | Improve this doc

Attempt to access non-existing or hidden variable
Description

The specified variable was not found.

If it is a database field ensure that the required database is open.

If it is a private or public variable then it must be first created using PRIVATE or PUBLIC statement.

Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1068^ | ☰ | Improve this doc

Bound error in array access
Description
The attempt to retrieve data from non-array value.
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1069^ | ☰ | Improve this doc

Bound error in array access
Description

The attempt to set data to non-array value.
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1070^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared arguments do not match.
Examples
```
? "a" == 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1072^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared arguments do not match.
Examples
```
? "a" != 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1073^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared argument do not match.
Examples
```
? "a" < 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1074^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared arguments do not match.
Examples
```
? "a" <= 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1075^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared arguments do not match.
Examples
```
? "a" > 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1076^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of compared arguments do not match.
Examples
```
? "a" >= 0
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1077^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
Operation is not allowed for passed argument. The argument is not a logical value.
Examples
```
? ! "a"
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag

Runtime errors

RTE_BASE/1078^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of one or both arguments is not a logical.
Examples
```
// FIXME
.AND.
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1079^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The type of one or both arguments is not a logical.
Examples
```
// FIXME
.OR.
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1081^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The plus operation is not allowed for used arguments.
Examples
```
+
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1082^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The minus operation is not allowed for used arguments.
Examples

```
? "a" - 1
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1085^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Mod()
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1086^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The value of argument cannot be incremented.
Examples
```
LOCAL cVar := "a"
cVar++
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1089^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Abs( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1090^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Int( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1092^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Min( "a", "b" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1093^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Max( "a", "b" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1094^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Round( "a", 0 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag

Runtime errors

RTE_BASE/1095^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Log( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1096^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Exp( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1097^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not an numeric value
Examples
```
Sqrt( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1098^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not a string value
Examples
```
Val( 1 )
```

Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1099^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is not a numeric value
Examples
```
Str( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1100^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The specified argument is not a string.
Examples
```
RTrim( 1 )
Trim( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1101^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The specified argument is not a string.
Examples
```
LTrim( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1102^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The first argument passed to a function is not a string.
Examples
```
Upper( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1103^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The first argument passed to a function is not a string.
Examples
```
Lower( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1104^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The specified argument is not a numeric value.
Examples
```
Chr( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE BASE/1105^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The arguments passed to a function are of incorrect type.
Examples
```
Space( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag

Runtime errors

RTE_BASE/1106^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The arguments passed to a function are of incorrect type.
Examples
```
Replicate( 1, "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1107^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The specified argument is not a string.
Examples
```
Asc( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1108^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The specified argument is not a string.
Examples
```
At( 1, "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1109^ | ☰ | **Improve this doc**

Invalid type of arguments
Description
The arguments of '$' operator are not a strings.
Examples
```
? 1 $ "a"
```

Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1110^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The first argument passed to a function is not a string.
Examples
```
SubStr( 1, 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1111^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The passed argument is neither a string nor an array.
Examples
```
Len( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1112^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function are of incorrect type
Examples
```
Year( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1113^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function are of incorrect type
Examples
```
Month( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1114^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function are of incorrect type
Examples
```
Day( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1115^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function are of incorrect type
Examples
```
DoW( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1116^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function are of incorrect type
Examples
```
CMonth( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag

Runtime errors

RTE_BASE/1117^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is of incorrect type
Examples
```
CDoW( "a" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1120^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The argument (or arguments) passed to a function is of incorrect type
Examples
```
DToS( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1122^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The argument (or arguments) passed to a function is of incorrect type
Examples
```
Transform( "a", 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1124^ | ☰ | **Improve this doc**

Incorrect type of argument
Description
The first argument is not a string.
Examples
```
Left( 1, 1 )
```

Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1126^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The first arguments passed to a function is not a string.
Examples
```
StrTran( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1132^ | ☰ | **Improve this doc**

Bound error in array access
Description
The specified index into an array was greater then the number of elements in the array.
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/1133^ | ☰ | **Improve this doc**

Bound error in array element assignment
Description
The specified index into an array was greater then the number of elements in the array.
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/2010^ | ☰ | **Improve this doc**

Incorrect arguments type
Description
Passed Runtime errors was not strings with file names to copy.
Examples

```
__CopyFile( 1, 2 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/2012^ | ☰ | **Improve this doc**

File error
Description
An error has occurred during the attempt to open, create or write during copy operation
Examples
```
__CopyFile( "<nothere>", "test.txt" )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/2017^ | ☰ | **Improve this doc**

Invalid argument passed to a function
Description
The first argument is not an array or/and the second argument is not a code block
Examples
```
AEval( 1 )
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/2020^ | ☰ | **Improve this doc**

Invalid argument passed to function
Description
The passed value is negative. Only values > 0 are allowed.
Examples
```
// SET DECIMALS
// SET EPOCH
// SET MARGIN
// SET MESSAGE
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3001^ | ☰ | **Improve this doc**

Incorrect argument type
Description
The passed argument is not an object. Only data of type `OBJECT` can be cloned by this function
Examples
```
__objClone( 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3002^ | ☰ | **Improve this doc**

Super class does not return an object
Description
Passed argument is not a name of defined class or specified class doesn't have a super class
Examples
```
// FIXME
__clsInstSuper()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3003^ | ☰ | **Improve this doc**

Cannot find super class
Description
Passed argument is not a name of defined class
Examples
```
__clsInstSuper()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3004^ | ☰ | **Improve this doc**

Cannot modify a DATA item in a class
Description
The attempt to modify a data member of a class was made. Only `INLINE` and `METHOD` can be modified
Examples

```
// FIXME
__clsModMsg()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3005^ | ☰ | **Improve this doc**

Incorrect arguments type
Description
Either the first argument was not an object or the second argument wasn't a string.
Examples
```
__clsAssocType( 1, 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3007^ | ☰ | **Improve this doc**

Invalid type of argument
Description
The passed arguments are causing conflict in handling of the request. There is no point in waiting forever for no input events!
Examples
```
// FIXME
Inkey()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3008^ | ☰ | **Improve this doc**

Invalid type of argument
Description
The passed argument(s) is not a string. It should be a string with a variable name or an one-dimensional array of strings.
Examples
```
__mvPrivate( 1 )
__mvPublic( 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms

Tag
Runtime errors

RTE_BASE/3009^ | ☰ | **Improve this doc**

Incorrect argument passed to __mvGet() function
Description
`__mvGet()` function expects only one argument: a string with a name of variable. The value of this variable will be returned.
Examples
```
__mvGet( 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3010^ | ☰ | **Improve this doc**

Incorrect argument passed to __mvPut() function
Description
`__mvPut()` function expects at least one argument: a string with a name of variable. The value of this variable will be set.
Examples
```
__mvPut( 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3012^ | ☰ | **Improve this doc**

Invalid argument passed to a function
Description
The first argument is not a string with function/procedure name that should be called.
Examples
```
Do( 1 )
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3101^ | ☰ | **Improve this doc**

Invalid argument passed to an object/class function
Description

One passed argument is not of the required type.
Examples
```
// __obj*()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3102^ | ☰ | **Improve this doc**

A symbol should be modified or deleted from a class, but the symbol doesn't exist.
Description
A symbol should be modified or deleted from a class, but the symbol doesn't exist.
Examples
```
// __obj*()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_BASE/3103^ | ☰ | **Improve this doc**

A symbol should be added to a class, but the symbol already exists.
Description
A symbol should be added to a class, but the symbol already exists.
Examples
```
// __obj*()
```
Compliance
Harbour specific
Platforms
Available on all platforms
Tag
Runtime errors

RTE_TERM/2013^ | ☰ | **Improve this doc**

Create error
Description
The specified file cannot be created due some OS error.
Examples
```
// Set()
// SET ALTERNATE TO
```
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
Tag
Runtime errors

FIELD^ | ☰ | Improve this doc

Declares a list of database field names.
Syntax
```
FIELD <xField> [, <xFieldn...> [IN <cDatabase>]
```
Arguments
`xField` A valid field name
`xFieldn` Additional field name
`cDatabase` An valid alias name
Description
This command declares the names of fields `xField` (and `xFieldn` and following) with an optional alias identifier as `cDatabase` for each. This command allow Harbour to resolve any reference to a field specified in the field list by viewing it as a field when it is not referenced by an alias. If a field is not listed in this list and it is not explicitly tagged with an alias identifier, it may be viewed as a memory variable, which may cause run-time errors. This command has no effect on memory variables or on field reference buried within a macro expression.
Examples
```
FIELD first
FIELD age
USE test NEW
first := "FirstName"
age := 25
```
Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
None.
Tags
Statement, RDD
See also
MEMVARClipper 5.3, PRIVATE, PUBLIC, STATIC

LOCAL^ | ☰ | Improve this doc

Initializes a local memory variable or array
Syntax
```
LOCAL <xVar> [:= <xInit> ]
```
Arguments
`xVar` Name of a memory variable or array.
`xInit` Value to be assigned to a variable or array
Description

This command created a `LOCAL` memory variable or array. The name of either is specified in `xVar`. If more then one variable is being initialized with the `LOCAL` command, separate each entry with a comma. If a variable or an array is to be assigned a start-up value, that expression may be specified in `xInit` and following. Is Strong type compile mode is used, the Compiler will check if the value received matches the type specified in `xType`.

LOCAL variables are symbols generated at run time and are resolved at compile time. The visibility and life span of a LOCAL variable or array is limited to the function or procedure in which it is defined.

No macro expansions are allowed in the LOCAL declaration statement.

No Harbour command other then FUNCTION, PROCEDURE, PUBLIC, PRIVATE, PARAMETERS, MEMVAR, STATIC and FIELD, may precede the LOCAL command.

LOCAL array reference may not be initialized (i.e., assigned values) on the same command-line as the LOCAL command statement. This can be done later in the program.

LOCAL variables and arrays are not affected by the RELEASE command.

Examples

```
LOCAL n, lVar := .T.

n := iif( lVar, "A", 3 )
n := 2
n := "a"
n := Seconds() + 2
n := Int( Seconds() + 2 )
```

Status
Ready
Compliance
CA-Cl*pper v5.x compatible
Platforms
Available on all platforms
File
None
Tags
Statement, Variable management
See also
FIELDClipper 5.3, MEMVARClipper 5.3, PRIVATE, PUBLIC, STATIC

MEMVAR^ | ☰ | Improve this doc

Declares private and public variables and arrays.
Syntax

```
MEMVAR <xVar>
```

Arguments
xVar Memory variable Name
Description

This command tells the compiler to resolve any reference to a memory variable designated within this list s if it possessed an explicit memory variable alias with either the M-> or MEMVAR-> prefix. Only those memory variables that do not contain any such explicit are affected by this command. Those memory variables within macro expansions are not affected by this command.

The MEMVAR declaration must appear before any executable commands; it is similar to the LOCAL, STATIC, FIELD, PARAMETERS, FUNCTION, and PROCEDURE commands statements.

Examples

```
MEMVAR y AS NUMERIC

LOCAL n, lVar := .T.

n := iif( lVar, "A", 3 )
n := 2
n := "a"
n := Seconds() + 2
n := Int( Seconds() + 2 )
y := n

? y
```

Status

Ready

Compliance

CA-Cl*pper v5.x compatible

Platforms

Available on all platforms

File

None.

Tags

Statement, Variable management

See also

FIELDClipper 5.3, LOCALClipper 5.3, PRIVATE, PUBLIC, STATIC

TBrowseNew() Source code in Repository in Discussions ^ | ☰ | Improve this doc

Create a Browse Object

Syntax

```
TBrowseNew( <nTop>, <nLeft>, <nBottom>, <nRight> ) → oBrowse
```

Arguments

`nTop` Top Row

`nLeft` Top Left Column

`nBottom` Bottom Row

`nRight` Bottom Right Column

Returns

`oBrowse` An new Browse Object

Description

This function set up a browsing window at top-left coordinates of `nTop`, `nLeft` to bottom-right coordinates of `nBottom`, `nRight`. To browse Database files use `TBrowseDB()` function instead.

Data no link

`:aColumns` Array to hold all browse columns

`:autoLite` Logical value to control highlighting

`:cargo` User-definable variable

`:colorSpec` Color table for the TBrowse display

`:colPos` Current cursor column position

`:colSep` Column separator character

`:footSep` Footing separator character

`:freeze` Number of columns to freeze

`:goBottomBlock` Code block executed by `TBrowse:goBottom()`

`:goTopBlock` Code block executed by `TBrowse:goTop()`

:headSep Heading separator character
:hitBottom Indicates the end of available data
:hitTop Indicates the beginning of available data
:leftVisible Indicates position of leftmost unfrozen column in display
:nBottom Bottom row number for the TBrowse display
:nLeft Leftmost column for the TBrowse display
:nRight Rightmost column for the TBrowse display
:nTop Top row number for the TBrowse display
:rightVisible Indicates position of rightmost unfrozen column in display
:rowCount Number of visible data rows in the TBrowse display
:rowPos Current cursor row position
:skipBlock Code block used to reposition data source
:stable Indicates if the TBrowse object is stable
:aRedraw Array of logical items indicating, is appropriate row need to be redraw
:RelativePos Indicates record position relatively position of first record on the screen
:lHeaders Internal variable which indicates whether there are column footers to paint
:lFooters Internal variable which indicates whether there are column footers to paint
:aRect The rectangle specified with :ColorRect()
:aRectColor The color positions to use in the rectangle specified with :ColorRect()
:aKeys Holds the Default movement keys
Methods link
:AddColumn() Adds an new TBColumn object to the current Browse
:Applykey() Perform the Browse Key movement
:SetKey() Add an New key to the Keyboard dictionary
Methods no link
:New(nTop, nLeft, nBottom, nRight) Create an new Browse class and set the default values
:Down() Moves the cursor down one row
:End() Moves the cursor to the rightmost visible data column
:GoBottom() Repositions the data source to the bottom of file
:GoTop() Repositions the data source to the top of file
:Home() Moves the cursor to the leftmost visible data column
:Left() Moves the cursor left one column
:PageDown() Repositions the data source downward
:PageUp() Repositions the data source upward
:PanEnd() Moves the cursor to the rightmost data column
:PanHome() Moves the cursor to the leftmost visible data column
:PanLeft() Pans left without changing the cursor position
:PanRight() Pans right without changing the cursor position
:Right() Moves the cursor right one column
:Up() Moves the cursor up one row
:ColCount() Return the Current number of Columns
:ColorRect() Alters the color of a rectangular group of cells
:ColWidth(nColumn) Returns the display width of a particular column
:Configure(nMode) Reconfigures the internal settings of the TBrowse() object nMode is an undocumented parameter in CA-Cl*pper
:LeftDetermine() Determine leftmost unfrozen column in display
:DeHilite() Dehighlights the current cell
:DelColumn(nPos) Delete a column object from a browse
:ForceStable() Performs a full stabilization

`:GetColumn( nColumn )` Gets a specific `TBColumn()` object
`:Hilite()` Highlights the current cell
`:InsColumn( nPos, oCol )` Insert a column object in a browse
`:Invalidate()` Forces entire redraw during next stabilization
`:RefreshAll()` Causes all data to be recalculated during the next stabilize
`:RefreshCurrent()` Causes the current row to be refilled and repainted on next stabilize
`:SetColumn( nColumn, oCol )` Replaces one `TBColumn()` object with another
`:Stabilize()` Performs incremental stabilization
`:DispCell( nColumn, cColor )` Displays a single cell
Examples
`// See tests/tbrowse.prg`
Status
Started
Compliance
This functions is Compatible with CA-Cl*pper 5.2. The `:applykey()` and `:SetKey()` methods are only visible if `HB_COMPAT_C53` is defined.
Platforms
Available on all platforms
File
Library is core
Tag
TBrowse class
See also
`TBColumnNew()`, `TBrowseNew()`

`TBrowse():AddColumn()` in Repository in Discussions ^ | ☰ | Improve this doc

Add an New Column to an TBrowse Object
Syntax
`:AddColumn( <oCol> ) → Self`
Arguments
`oCol` Is an TbColumn object
Returns
`Self` The Current object
Description
This method add an new column object specified as `oCol` to the assigned browsing object.
Platforms
Available on all platforms
Tag
TBrowse Method

`TBrowse():ApplyKey()` in Repository in Discussions ^ | ☰ | Improve this doc

Evaluates an code block associated with an specific key
Syntax
`:ApplyKey( <nKey> ) → nResult`
Arguments
`nKey` An valid Inkey code
Returns
`nResult` Value returned from the evaluated Code Block See Table Below
Value	Meaning

-1 User request for the browse lost input focus
0 Code block associated with <nkey> was evaluated
1 Unable to locate <nKey> in the dictionary, Key was not processed

Description

This method evaluate an code block associated with `nkey` that is contained in the TBrowse:SetKey() dictionary.

Examples

```
LOCAL oTb := TBrowseNew( 0, 0, 10, 20 )
DO WHILE .T.
   oTb:forceStable()
   IF oTb:applykey( Inkey( 0 ) ) == -1
      EXIT
   ENDIF
ENDDO
```

Platforms

Available on all platforms

Tag

TBrowse Method

`TBrowse():SetKey()` in Repository in Discussions ^ | ☰ | Improve this doc

Get an optionally Set an new Code block associated to a inkey value

Syntax

```
:SetKey( <nKey>[, <bBlock>] ) → bOldBlock
```

Arguments

`nKey` An valid inkey Code

`bBlock` An optional action to associate to the inkey value.

Returns

`bOldBlock` If a key-press has it code block changes, it will return the previous one; otherwise, it will return the current one

Description

This method Get an optionally set an code block that is associated to an inkey value. The table below show the default key-press/Code Block definitions

Inkey Value	Code Block
K_DOWN	{| oB, nKey | oB:Down(), 0 }
K_END	{| oB, nKey | oB:End(), 0 }
K_CTRL_PGDN	{| oB, nKey | oB:GoBottom(), 0 }
K_CTRL_PGUP	{| oB, nKey | oB:GoTop(), 0 }
K_HOME	{| oB, nKey | oB:Home(), 0 }
K_LEFT	{| oB, nKey | oB:Left(), 0 }
K_PGDN	{| oB, nKey | oB:PageDown(), 0 }
K_PGUP	{| oB, nKey | oB:PageUp(), 0 }
K_CTRL_END	{| oB, nKey | oB:PanEnd(), 0 }
K_CTRL_HOME	{| oB, nKey | oB:PanHome(), 0 }
K_CTRL_LEFT	{| oB, nKey | oB:PanLeft(), 0 }
K_CTRL_RIGHT	{| oB, nKey | oB:PanRight(), 0 }
K_RIGHT	{| oB, nKey | oB:Right(), 0 }
K_UP	{| oB, nKey | oB:Up(), 0 }
K_ESC	{| oB, nKey | -1 }

The keys handlers can be queried, added and replace an removed from the internal keyboard dictionary. See the example.

```
oTb:SetKey( K_TAB, {| oTb, nKey | -1 } )
```

An default key handler can be declared by specifying a value of 0 for `nKey`. It associate code block will be evaluated each time TBrowse:Applykey() is called with an key value that is not contained in the dictionary. For example

```
oTb:SetKey( 0, {| oTb, nKey | DefKeyHandler( oTb, nkey } )
```

This call the a function named `DefKeyHandler()` when `nKey` is not contained in the dictionary.

To remove an key-press/code block definition, specify `NIL` for `bBlock`

```
oTb:SetKey( K_ESC, NIL )
```

Examples

```
#include "inkey.ch"
LOCAL oTb := TBrowseNew( 0, 0, 10, 20 )
oTb:SetKey( K_F10, {| oTb, nkey | ShowListByname( oTb ) } )
```

Platforms

Available on all platforms

Tag

TBrowse Method

Generated by hbdoc on 2017-10-28 10:02 UTC
Based on revision f6a35f2

Content processing, layout & design © 2017 vszakats