******* INTNALS.LIB *******

First release uploaded to NANFORUM on August 08, 1990.

Written by : Dao Dang Trieu Duong [CIS 72357,3365].
This is the first release and it is free (FREEWARE). Anyone can copy or use this library.
UNDER NO CIRCUMTANCES MAY ANY FEE BE CHARGED TO DO SO.

You can use this library in the development of commercial softwares but AT YOUR OWN RISK.
UNDER NO CIRCUMTANCES, CAN I BE HELD RESPONSIBLE OF LOST OF DATA, LOST OF PROFITS OR INCONVENIENTS ETC.

The functions are written using Microsoft C. Most of them don't required the large model library LLIBCA to be linked with your programs. They has been tested but since I don't have a lot of time to spent on this project, I am not sure that they are free of bugs.

I welcome any feedback, bug reports, comments or criticisms. You can reach me with the CIS shown above.

The functions use CLIPPER (Summer 87) internals variables and data structures. Since I don't receive CLIPPER 5.0 yet, I can't guaranted that they will work under this version. Upon the release of version 5.0, I will upload a revised set of functions as soon as possible.

Included in INTNALS.LIB, is EXTOR.OBJ written by Richard McConnell. I use the functions included in it to return the error codes (our_error) to the calling modules. I hope that Richard will excuse me for this liberty since the library is free for those nice folks of NANFORUM.

One last word, a great thank to those folks who upload useful ideas and programs to NANFORUM.

CONVENTION FOR THE FOLLOWING DOCUMENT.

[]	Optional parameter
<>	Requested parameter

Function NUMERIC alt_handle() [SET ... TO]

Purpose Get/Set the file handle of the ALTERNATE file.

Syntax alt_handle([expN])

Parameters expN = new file handle

Returns a numeric that is the handle of the ALTERNATE file declared within SET ALTERNATE TO command.

Usage A file must be open using Clipper low-level function FOPEN() before using alt_handle() with a parameter.

If an ALTERNATE file is currently open, close it after calling alt_to() with a parameter to preserve its data.

Example
```
M->handle = FOPEN('myfile.dat',2)
M->old_handle = alt_handle(M->handle)
FCLOSE(M->old_handle)
```

Library INTNALS.LIB

See also alt_to()

Function LOGICAL alt_to() [SET ... TO]

Purpose Get/Set the state of SET ALTERNATE TO.

Syntax alt_to([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical.

.T. if there's an ALTERNATE file ready to receive direct outputs from commands other than @ SAY ...GET;
.F. otherwise.

Usage If used with a parameter, this function must be followed by alt_handle().

Example
```
M->alt_state = alt_to(.T.)
M->handle = FOPEN('myfile.dat',2)
M->old_handle = alt_to(M->handle)
```

Library INTNALS.LIB

See also alt_handle()

Function NUMERIC date_to() [SET ... TO]

Purpose Get/Set the state of SET DATE TO command.

Syntax date_to([expN])

Parameters expN = a numeric which has one of the following values:

Country	ExpN	Format of the date
AMERICAN	1	mm/dd/yy
ANSI	2	yy.mm.dd
BRITISH	3	dd/mm/yy
FRENCH	4	dd/mm/yy
GERMAN	5	dd.mm.yy
ITALIAN	6	dd-mm-yy

Returns a numeric indicating the current format of the date.
(same code as preceding.)

Usage

Example
```
SET DATE TO AMERICAN
M->old_datfrm = date_to(4) ===> set date to FRENCH
? old_datfrm               ===> old date format is AMERICAN
```

Library INTNALS.LIB

See also

Function NUMERIC decim_to() [SET ... TO]

Purpose Get/Set the state of SET DECIMALS TO command.

Syntax decim_to([expN])

Parameters expN = number of decimals to be set as default.

Returns a numeric indicating the number of decimals currently set by SET DECIMALS TO command.

Usage

Example
```
SET DECIMALS TO 2
M->old_decim = decim_to(4) ===> SET DECIMALS TO 4
? old_decim                ===> 2
```

Library INTNALS.LIB

See also

Function CHARACTER default_to() [SET ... TO]

Purpose Get/Set the state of SET DEFAULT TO command.

Syntax default_to([expC])

Parameters expC = drive and path of the DEFAULT directory
(anull-terminated string of 64 chars maximum).

Returns a string that is the DEFAULT directory defined by
SET DEFAULT TO command.

Usage

Example

```
SET DEFAULT TO clipper\data
M->old_defo = default_to("\clipper\data1")
                        ===> SET DEVICE TO \clipper\data1
? old_defo              ===> 'clipper\data'
```

Library INTNALS.LIB

See also path_to()

Function NUMERIC device_to() [SET ... TO]

Purpose Get/Set the state of SET DEVICE TO command.

Syntax device_to([expN])

Parameters expN = a number indicating the device to be used.

0 = Screen
1 = Printer

Returns a numeric indicating the device in use.
(same as preceding.)

Usage

Example
```
SET DEVICE TO PRINT
M->old_devic = device_to(0) ===> SET DEVICE TO SCREEN
? old_devic                 ===> 1 (Printer)
```

Library INTNALS.LIB

See also

Function CHARACTER ldelim_to() [SET ... TO]

Purpose Get/Set a field left delimiter.
(SET DELIMITERS TO)

Syntax ldelim_to([expC])

Parameters expC = a character to be used as a field left delimiter.

Returns a character which is the current field left delimiter.

Usage

Example
```
SET DELIMITERS TO "[]"
M->ldelim = ldelim_to("{") ===> Set left delimiter to {
? M->ldelim                ===> [
```

Library INTNALS.LIB

See also rdelim_to(), is_delim()

Function CHARACTER rdelim_to() [SET ... TO]

Purpose Get/Set a field right delimiter.
(SET DELIMITERS TO)

Syntax rdelim_to([expC])

Parameters expC = a character to be used as a field right delimiter.

Returns a character which is the current field right delimiter.

Usage

Example
```
SET DELIMITERS TO "[]"
M->rdelim = rdelim_to("}") ===> Set left delimiter to }
? M->rdelim                ===> ]
```

Library INTNALS.LIB

See also ldelim_to(), is_delim()

Function NUMERIC margin_to() [SET ... TO]

Purpose Get/Set the value of SET MARGIN TO.

Syntax margin_to([expN])

Parameters expN = the column of the left margin.

Returns a numeric indicating the current value of the left margin as defined by the SET MARGIN TO command.

Usage

Example
```
SET MARGIN TO 75
M->lmarg = margin_to(70) ===> Set left margin to col 70
? M->lmarg               ===> 75
```

Library INTNALS.LIB

See also

Function CHARACTER path_to() [SET ... TO]

Purpose Get/Set the state of SET PATH TO.

Syntax path_to([expC])

Parameters expC = drive and path of the PATH directory
(anull-terminated string of 64 chars maximum).

Returns a string that is the PATH directory defined by
SET PATH TO command.

Usage

Example
```
SET PATH TO \clipper\data
M->path = path_to("\clipper\data1")
? M->path            ===> \clipper\data
```

Library INTNALS.LIB

See also default_to()

Function NUMERIC printer_to() [SET ... TO]

Purpose Get/Set the handle of SET PRINTER TO device/file.

Syntax printer_to([expN])

Parameters expN = device/file handle.

Returns a numeric that is the device/file handle to which printer outputs will be redirected.

Usage A file must be open using Clipper low-level function FOPEN() before using printer_to() with a file handle as parameter.

Example
```
M->handle = FOPEN('output.dat',2)
printer_to(M->handle)          ===> redirect printer output
                                    to file output.dat
```

Library INTNALS.LIB

See also is_print()

Function LOGICAL is_alt() [SET ... ON]

Purpose Get/Set the state of ALTERNATE (ON/OFF)

Syntax is_alt([expL])

Parameters expL = a logical :

.T. to set it ON;
.F. to set it OFF

Returns a logical (same as preceding).

Usage

Example

```
SET ALTERNATE ON
M->altern = is_alt(.F.)   ===> SET ALTERNATE OFF
? M->altern               ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_bell() [SET ... ON]

Purpose Get/Set the state of BELL (ON/OFF)

Syntax is_bell([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET BELL ON
M->bell = is_bell(.F.)    ===> SET BELL OFF
? M->bell                 ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_cent() [SET ... ON]

Purpose Get/Set the state of CENTURY (ON/OFF)

Syntax is_cent([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET CENTURY ON
M->centur = is_cent(.F.)  ===> SET CENTURY OFF
? M->centur               ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_confirm() [SET ... ON]

Purpose Get/Set the state of CONFIRM (ON/OFF)

Syntax is_confirm([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET CONFIRM ON
M->conf = is_confirm(.F.) ===> SET CONFIRM OFF
? M->conf                 ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_consol() [SET ... ON]

Purpose Get/Set the state of CONSOLE (ON/OFF)

Syntax is_consol([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical : .T. if it's ON;
.F. if it's OFF.

Usage

Example

```
SET CONSOLE ON
M->cons = is_consol(.F.)  ===> SET CONSOLE OFF
? M->cons                 ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_cursor() [SET ... ON]

Purpose Get/Set the state of CURSOR (ON/OFF)

Syntax is_cursor([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET CURSOR ON
M->curs = is_cursor(.F.)  ===> SET CURSOR OFF
? M->curs                 ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_delete() [SET ... ON]

Purpose Get/Set the state of DELETED (ON/OFF)

Syntax is_delete([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET DELETED ON
M->del = is_delete(.F.)   ===> SET DELETED OFF
? M->del                  ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_delim() [SET ... ON]

Purpose Get/Set the state of DELIMITERS (ON/OFF)

Syntax is_delim([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET DELIMITERS ON
M->delim = is_delim(.F.)  ===> SET DELIMITERS OFF
? M->delim                ===> .T. (ON)
```

Library INTNALS.LIB

See also ldelim_to(), rdelim_to().

Function LOGICAL is_escape() [SET ... ON]

Purpose Get/Set the state of ESCAPE (ON/OFF)

Syntax is_excape([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET ESCAPE ON
M->esc = is_escape(.F.)   ===> SET ESCAPE OFF
? M->esc                  ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_exact() [SET ... ON]

Purpose Get/Set the state of EXACT (ON/OFF)

Syntax is_exact([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET EXACT ON
M->exac = is_exact(.F.)   ===> SET EXACT OFF
? M->exac                 ===> .T. (ON)
```

Library INTNALS.LIB

See also

Function LOGICAL is_exclus() [SET ... ON]

Purpose Get/Set the state of EXCLUSIVE (ON/OFF)

Syntax is_exclus([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET EXCLUSIVE OFF
M->exclu = is_exclus(.T.) ===> SET EXCLUSIVE ON
? M->exclu                ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_fixed() [SET ... ON]

Purpose Get/Set the state of FIXED (ON/OFF)

Syntax is_fixed([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example
```
SET FIXED OFF
M->fix = is_fixed(.T.)    ===> SET FIXED ON
? M->fix                  ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_insert()

Purpose Get/Set the state of the readinsert flag.

Syntax is_insert([expL])

Parameters expL = a logical expression.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

Library INTNALS.LIB

See also

Function LOGICAL is_intens() [SET ... ON]

Purpose Get/Set the state of INTENSITY (ON/OFF)

Syntax is_intens([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET INTENSITY OFF
M->inten = is_intens(.T.) ===> SET INTENSITY ON
? M->inten                ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_print() [SET ... ON]

Purpose Get/Set the state of PRINT (ON/OFF)

Syntax is_print([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET PRINT OFF
M->prt = is_print(.T.)    ===> SET PRINT ON
? M->prt                  ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_scoreb() [SET ... ON]

Purpose Get/Set the state of SCOREBOARD (ON/OFF)

Syntax is_scoreb([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET SCOREBOARD OFF
M->score = is_scoreb(.T.) ===> SET SCOREBOARD ON
? M->score                ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_softsek() [SET ... ON]

Purpose Get/Set the state of SOFTSEEK (ON/OFF)

Syntax is_softsek([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET SOFTSEEK OFF
M->soft = is_softsek(.T.) ===> SET SOFTSEEK ON
? M->soft                 ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_unique() [SET ... ON]

Purpose Get/Set the state of UNIQUE (ON/OFF)

Syntax is_unique([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET UNIQUE OFF
M->unik = is_unique(.T.)  ===> SET UNIQUE ON
? M->unik                 ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function LOGICAL is_wrap() [SET ... ON]

Purpose Get/Set the state of WRAP (ON/OFF)

Syntax is_wrap([expL])

Parameters expL = a logical.

.T. to set it ON;
.F. to set it OFF.

Returns a logical (same as preceding).

Usage

Example

```
SET WRAP OFF
M->wrap = is_wrap(.T.)    ===> SET WRAP ON
? M->wrap                 ===> .F. (OFF)
```

Library INTNALS.LIB

See also

Function NUMERIC dbr_count() [RELATION]

Purpose Determine the number of relations defined in the specified work area.

Syntax dbr_count(<expN>,@<expC>)

Parameters
expN = work area number (0..255)
(0 for current work area)
expC = memvar for the returned error code
(passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area.

Usage The maximum number of relations returned by this function is eight (8).

dbr_count() will return 9 if the returned error code is 1.

Example

```
PUBLIC our_error

SELECT 3
USE C
SET INDEX TO C
SELECT 2
USE B
SET INDEX TO B
SELECT 1
USE A
SET RELATION TO XXX INTO B, TO YYY INTO C

? dbr_count(1,@our_error)     ===> 2 and our_error = 0
? dbr_count(2,@our_error)     ===> 0 and our_error = 0
? dbr_count(4,@our_error)     ===> 9 and our_error = 1
```

Library INTNALS.LIB

See also dbr_find(), dbr_relat(), dbr_select()

Function NUMERICAL dbr_find() [RELATION]

Purpose Determine if a relation defined in the specified work area is based on the specified key and return its ordinal position in the list of defined relations.

Syntax dbr_find(<expN>,<expC1>,@<expC2>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expC1 = key expression used to define the searched relation in the specified work area.
expC2 = memvar for the returned error code
(passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no relation defined in the specified work area.

Usage dbr_find() will return 99 if there's no relation in the <expN> work area which is defined using the <expC1> expression.

dbr_find() will return 99 if the returned error code is 1 or 2.

Microsoft C library LLIBCA.LIB must be linked to the program in order to use this function.

Example
```
PUBLIC our_error

SELECT 3
USE C
SET INDEX TO C
SELECT 2
USE B
SET INDEX TO B
SELECT 1
USE A
SET RELATION TO XXX INTO B, TO YYY INTO C

? dbr_find(1,"XXX",@our_error)  ===> 1 and our_error = 0
? dbr_count(2,"ZZZ",@our_error) ===> 99 and our_error = 0
? dbr_count(4,@our_error)       ===> 99 and our_error = 1
```

Library INTNALS.LIB

See also dbr_count(), dbr_relat(), dbr_select()

Function CHARACTER dbr_relat() [RELATION]

Purpose Determine the linking expression of the specified relation in the specified work area.

Syntax dbr_relat(<expN1>,<expN2>,@<expC>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expN2 = ordinal position of the relation in the list of relations defined in the specified work area (0..7).
expC = memvar for the returned error code (passed by reference)

Returns a string.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no relation defined in the specified work area;
3 if the specified ordinal position of the relation is out of the range (0 .. number of relations defined).

Usage This function is alike DBRELATION().

dbr_relat() will return a null string ("") with an error code of 1 or 2 or 3.

Example
```
PUBLIC our_error

SELECT 3
USE C
SET INDEX TO C
SELECT 2
USE B
SET INDEX TO B
SELECT 1
USE A
SET RELATION TO XXX INTO B, TO YYY INTO C

? dbr_relat(1,1,@our_error)   ===> XXX and our_error = 0
? dbr_relat(1,2,@our_error)   ===> YYY and our_error = 0
? dbr_relat(1,3,@our_error)   ===> "" and our_error = 3
? dbr_relat(2,1,@our_error)   ===> "" and our_error = 2
? dbr_relat(4,1,@our_error)   ===> "" and our_error = 1
```

Library INTNALS.LIB

See also dbr_count(), dbr_find(), dbr_select()

Function NUMERIC dbr_select() [RELATION]

Purpose Determine the target work area of a specified relation defined in the specified work area

Syntax dbr_select(<expN1>,<expN2>,@<expC>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expN2 = ordinal position of the relation in the list of relations defined in the specified work area (0..7).
expC = memvar for the returned error code
(passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no relation defined in the specified work area;
3 if the specified ordinal position of the relation is out of the range (0 .. number of relations defined).

Usage This function is alike DBRSELECT().

dbr_select() will return 999 if the returned error code is 1 or 2 or 3.

Example

```
PUBLIC our_error

SELECT 3
USE C
SET INDEX TO C
SELECT 2
USE B
SET INDEX TO B
SELECT 1
USE A
SET RELATION TO XXX INTO B, TO YYY INTO C

? dbr_select(1,1,@our_error)  ===> 1 and our_error = 0
? dbr_select(1,2,@our_error)  ===> 2 and our_error = 0
? dbr_select(1,3,@our_error)  ===> 999 and our_error = 3
? dbr_select(2,1,@our_error)  ===> 999 and our_error = 2
? dbr_select(4,1,@our_error)  ===> 999 and our_error = 1
```

Library INTNALS.LIB

See also dbr_count(), dbr_find(), dbr_relat()

Function NUMERICAL dbx_count() [INDEX]

Purpose Determine the number of indexes defined in the specified work area.

Syntax dbx_count(<expN>,@<expC>)

Parameters
expN = work area number
expC = memvar for the returned error code
(passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area.

dbx_count() will return a null value (0) if the returned error code is 1.

Usage

Example

```
PUBLIC our_error

SELECT 1
USE A
SET INDEX TO A,A1,A2,A3
? dbx_count(1,@our_error)     ===> 4
```

Library INTNALS.LIB

See also

Function LOGICAL dbx_new() [INDEX]

Purpose Determine if the index file opened in the specified work area is a newly created one.

Syntax dbx_new(<expN1>,<expN2>,@<expC>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expN2 = ordinal position of the index in the list of indexes opened in the specified work area (1 and up).
expC = memvar for the returned error code (passed by reference)

Returns a logical value.

.T. if the index is newly created;
.F. if it isn't.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no controlling index in the specified work area;
3 if the specified ordinal position of the index is out of the range (1..15).

Usage dbx_new() will return FALSE (.F.) if the returned error code is 1 or 2 or 3.

Example
```
PUBLIC our_error

SELECT 1
USE A
INDEX ON XXX TO A
? dbx_new(1,1,@our_error)     ===> .T.

CLOSE INDEX
SET INDEX TO A,A1,A2
? dbx_new(1,1,@our_error)     ===> .F.
```

Library INTNALS.LIB

See also dbx_count(),dbx_order()

Function CHARACTER dbx_order() [INDEX]

Purpose Determine the ordinal position of the controlling index in the list index files opened in the specified work area.

Syntax dbx_order(<expN>,@<expC>)

Parameters
expN = work area number (0..255)
(0 for current work area)
expC = memvar for the returned error code
(passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no controlling index in the specified work area.

Usage dbx_order() will return a null value (0) if the returned error code is 1 or 2.

Example
```
PUBLIC our_error

SELECT 1
USE A
SET INDEX TO A,A1,A2
SELECT 2
USE B
? dbx_order(1,@our_error)     ===> 1 and our_error = 0
```

Library INTNALS.LIB

See also

Function LOGICAL dbx_uniq() [INDEX]

Purpose Determine if an open index file is created with SET UNIQUE ON.

An index created with SET UNIQUE ON can be open in SET UNIQUE OFF mode.

Syntax dbx_uniq(<expN1>,<expN2>,@<expC>)

Parameters
- expN1 = work area number (0..255) (0 for current work area)
- expN2 = ordinal position of the index in the list of indexes opened in the specified work area (1 and up).
- expC = memvar for the returned error code (passed by reference)

Returns a logical.

AND

an error code :
- 0 if there's no error occured;
- 1 if there isn't a dbf opened in the specified work area;
- 2 if there's no controlling index in the specified work area;
- 3 if the specified ordinal position of the index is out of the range (1..15).

dbx_uniq() will return FALSE (.F.) if the returned error code is 1 or 2 or 3.

Usage

Example

Library INTNALS.LIB

See also

Function NUMERICAL dbx_pgnkey() [INDEX]

Purpose Determine the maximum number of keys which can be hold in a page of the index opened in the specified work area.

Syntax dbx_pgnkey(<expN1>,<expN2>,@<expC>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expN2 = ordinal position of the index in the list of indexes opened in the specified work area (1 and up).
expC = memvar for the returned error code (passed by reference)

Returns a numeric.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no controlling index in the specified work area;
3 if the specified ordinal position of the index is out of the range (1..15).

dbx_pgnkey() will return a null value (0) if the returned error code is 1 or 2 or 3.

Usage

Example

Library INTNALS.LIB

See also dbx_hpnkey().

Function NUMERICAL dbx_hpnkey() [INDEX]

Purpose Determine the maximum number of keys which can be hold in half of a page of the index opened in the specified work area.

Syntax dbx_hpnkey(<expN1>,<expN2>,@<expC>)

Parameters
expN1 = work area number (0..255)
(0 for current work area)
expN2 = ordinal position of the index in the list of indexes opened in the specified work area (1 and up).
expC = memvar for the returned error code (passed by reference)

Returns a string.

AND

an error code :
0 if there's no error occured;
1 if there isn't a dbf opened in the specified work area;
2 if there's no controlling index in the specified work area;
3 if the specified ordinal position of the index is out of the range (1..15).

dbx_hpnkey() will return a null value (0) if the returned error code is 1 or 2 or 3.

Usage

Example

Library INTNALS.LIB

See also dbx_pgnkey().

→