

NatStar

Version 6.00 Edition 1

NS-DK

Version 6.00 Edition 1

**Services Libraries Reference
Volume 2
Kernel APIs**

Information in this document is subject to change without notice as a result of changes in the product. The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms of the agreement. It is illegal to reproduce the software on any medium unless specifically authorized within the terms of the agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of Nat System.

© 2008 Nat System. All rights reserved.

Ref. n° NS600SRL02001

Contents

About this manual	v
Organisation of the manual	v
Conventions	vi

Chapter 1 NSDate Date and Time Library

Installation.....	1-3
NCL Language Extensions for Dates and Times	1-4
Dates	1-4
Days	1-4
Months.....	1-5
Times	1-5
NSDate.NCL File.....	1-6
Functional categories in the NSDate library	1-7
NSDate library reference	1-9

Chapter 2 NSMath Mathematics Library

Installation.....	2-3
NCL Language Extensions for Mathematical Calculations	2-4
NSMATH.NCL File.....	2-5
Functional categories in the NSMath library	2-6
NSMath library reference.....	2-8

Chapter 3 NSFexe Library

Installation.....	3-3
NCL Language Extensions for executing programs	3-4
NSFEXE.NCL File	3-5
Functional categories in the NSFexe library	3-6
NSFexe Library Reference.....	3-7

Chapter 4 NSMisc Miscellaneous Library

Installation.....	4-3
NSMISC.NCL File.....	4-4
Functional categories in the NSMisc library.....	4-5
Searching for a file in a list of directories	4-10
Handling controls	4-11
Managing Windows 32 controls.....	4-11
Managing the anchoring controls	4-11
Managing traces	4-11
Handling List Boxes	4-11
Handling lists of images and assigning images to menu items.....	4-12
NSMisc library reference	4-15

Chapter 5 NSDynstr Library

Installation.....	5-3
NSDynstr.NCL File	5-4
List of functions and instructions in the NSDynstr library	5-5
NSDynstr library reference	5-6
Example of saving a string content in a file	5-12

Chapter 6 NSMLE Library

Installation.....	6-3
NSMLE.NCL File.....	6-4
Functional categories in the NSMLE library	6-5
NSMLE library reference.....	6-6

Chapter 7 NSDYNCTR Library

Installation.....	7-3
NSDYNCTR Library Reference	7-4

Appendix A Display Formats

About formats for STRING\$ and ESTRING\$	A-3
Date formats	A-3
Time formats	A-3
Formats and meanings.....	A-4
Examples.....	A-7

Appendix B NSMisc Error Codes

NSMisc Error Codes	B-3
--------------------------	-----

About this manual

This is Volume 2 of the NatStar, NS-DK "Service Libraries Reference". The purpose of the Service Libraries Reference is to describe in detail the contents of the standard add-on libraries for the NCL language and explain how to use them.

This volume covers the following libraries: date and time formatting, program execution, mathematical, miscellaneous functions.

Organisation of the manual

Relationship to other manuals



For more information about the NatStar documentation set, see the "Documentation Road Map, Glossary and Global Indexes."



Before reading the Service Libraries Reference you are expected to have read the "Overview" and "Getting Started" manuals. You should not need to use the Reference unless you have been advised to do so in a "User Manual" or if you are already an experienced NatStar (or NS-DK) developer. If this is the case, you can use this Reference to learn in detail about the components it describes.

The Service Libraries Reference enables you to find and use one or more add-on libraries. Each of these contains a set of NCL language extensions which assist your application development by providing ready-to use coding. You can use each library in many applications, and choose those you will incorporate ("installation") in your workspace as needed. The Reference provides a brief description of each library, indicates how to install it, and provides various ways to find and look up the details of the components of each library.

The "User Manual" or "Development Guide" will usually indicate which library to use for a specific need. Once your chosen library is incorporated in your workspace, you can use the Reference to decide on the components you need, and look up the syntax and coding notes for each. The two indexes in this volume enable you to look up components based on either a functional theme, or the name of the component (function, instruction, etc.) to find its page number.



The complete set of functions, instructions, events, segments, operators and constants contained in NCL and in the add-on libraries is listed alphabetically in the API index of the "Documentation Road Map, Glossary and Global Indexes", which also contains a full topic index.

Organization of the manual

This manual contains several chapters.

Chapter 1	NSDate Library This library handles dates and times (current date and time, date and time format, day and month names, etc.).
Chapter 2	NSMath Library This library performs mathematical calculations (trigonometry, logarithms, etc.).
Chapter 3	NSFexe Library This library allows you to launch executables or start « command line » sessions from your application.
Chapter 4	NSMisc Library This library contains a large number of diverse functions and instructions,
Chapter 5	NSDynstr Library This library contains functions to handle character strings of Dynstr type.
Chapter 6	NSMLE Library This library contains functions to handle text in MLE controls or windows.
Chapter 7	NSDYNCTR Library This library contains functions allowing dynamic handling of controls.

Conventions

Typographic conventions

Important term	Important terms are printed in bold .
<i>Interface component</i>	The names of windows, dialog boxes, controls, buttons, menus and options are printed in <i>italics</i> .
[F9]	Keyboard key names appear in square brackets.

FILENAME	Filenames are printed in UPPERCASE.
syntax example	Syntax examples are printed in a fixed-width font.

Notational conventions

- A round bullet is used for lists
- ◆ A diamond is used for alternatives
- 1. Numbers are used to mark the steps in a procedure to be carried out in sequence

definition

A **definition** has a special presentation. It explains the term in a single paragraph. The term appears in the first column, then once in bold in the definition.

Operating conventions

Choose XXX \ YYY	This means you need to open the XXX menu, then choose the YYY command (option) from this menu. You can perform this action using the mouse or mnemonic characters on the keyboard.
Click the XXX \ YYY button	This means you need to display the tool bar named XXX, then click the YYY button in this tool bar (the name of each button is shown by its help bubble). You can only perform this action with the mouse.
Choose the XXX button	This means you need to choose the XXX button in a dialog box. You can perform this action using the mouse or mnemonic characters on the keyboard.

Icon codes



Comment, note, etc.



Reference to another part of the documentation



Danger: precaution to be taken, irreversible action, etc.



Suggestion: helpful hints, etc.



To go a step further: level of detail or expertise greater than the average level of the document

Chapter 1

NSDATE Date and Time Library



This library handles dates and times (current date and time, date and time format, day and month names, etc.).

This chapter explains

- How to install this library
- General aspects of its functions and instructions
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation	1-3
NCL Language Extensions for Dates and Times	1-4
Dates 1-4	
Days 1-4	
Months 1-5	
Times 1-5	
NSDATE.NCL File	1-6
Functional categories in the NSDate library.....	1-7
Retrieving current date, day and time 1-7	
Setting date and time format 1-7	
Retrieving format details 1-7	
Retrieving day and month names 1-7	
Format checking 1-8	
Conversion to and from integers and strings 1-8	
Milestone year 1-8	
Error management (constants returned) 1-8	
NSDate library reference	1-9

Installation

Declare NSDATE.NCL in the libraries required to develop your application.

Ensure that the NSxxDATE.DLL file has been placed in a PATH directory under Windows or LIBPATH directory under OS/2.

Ensure that the system date and time are correct (DATE and TIME commands under DOS or OS/2).

NCL Language Extensions for Dates and Times

Dates

The default date format is the one set in the Control Panel.

This format can be modified by SETDATEFORMAT, which accepts the following formats:

DD_MM_YY%, MM_DD_YY%, YY_MM_DD%

DD_MM_YYYY%, MM_DD_YYYY%, YYYY_MM_DD%

The separator used between months, days and years (e.g. "-" or "/") can be modified by SETDATESEPARATOR. GETDATEFORMAT% and GETDATESEPARATOR\$ return the format used.

The initial reference date is January 1, 1900. However, the NSDate library can handle "negative" dates, prior to 1900 and starting from January 1, 1583.

CURRENTDATE% returns the current date on the system that is executing the instruction. This is expressed as the number of days since the initial reference date.

DATE\$ and DATE% convert an integer into a date and vice versa. The integer expresses the number of days since the initial reference date, which is negative for dates before 1/1/1900. The DATE_ERROR% constant is returned by DATE% if an error occurs.

ISDATE% tests whether a string is in a valid date format.

Days

The CURRENTDAY% function returns the day of the week. 0 corresponds to Sunday.

The SETSHORTDAY / SETLONGDAY instructions and the GETSHORTDAY\$ / GETLONGDAY\$ functions provide a simple way to associate the integers 0 to 6 with character strings that identify the corresponding day in full (Sunday to Saturday).

Months

The SETSHORTMONTH / SETLONGMONTH instructions and the GETSHORTMONTH\$ / GETLONGMONTH\$ functions provide a simple way to associate the integers 0 to 11 with character strings that identify the corresponding month in full (January to December).

Times

The default time format is the one set in the Control Panel. This format can be modified by SETTIMEFORMAT, which accepts the following formats:

HH_MM%, HH_MM_AMPM%, HH_MM_SS%, HH_MM_SS_AMPM%

The separator used between hours, minutes and seconds (e.g. ":") can be modified by SETTIMESEPARATOR. GETTIMEFORMAT% and GETTIMESEPARATOR\$ return the format used.

For the formats HH_MM_AMPM% and HH_MM_SS_AMPM% (see HH_MM*%), the character strings displayed to differentiate A.M. from P.M. can be modified by SETTIMEAMPM. GETTIMEAMS\$ and GETTIMEPPM\$ return the strings used.

The initial reference time is 0 hours, 0 minutes, 0 seconds.

CURRENTTIME% returns the current time on the machine that is executing the instruction. This is expressed as the number of seconds since the initial reference time.

TIME\$ and TIME% convert an integer into a time and vice versa. The integer is expressed as the number of seconds since the initial reference time. The TIME_ERROR% constant is returned if an error occurs.

ISTIME% tests whether a string is in a valid time format.

NSDate.NCL File

The verbs in this library (NSDate) are described below. They are declared in a text file, written in NCL, called NSDate.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSDate NCL file :

1. Navigate to the <NATSTAR>\NCL, <NSDK>\NCL or <NATWEB>\NCL directory.

NB : <NATSTAR> means the directory in which NatStar was installed.

NB : <NSDK> means the directory in which NS-DK was installed.

NB : <NATWEB> means the directory in which NS-DK was installed.

2. Open the NSDate.NCL file with any text editor to see its contents.

Functional categories in the NSDate library

Here is a list, arranged by functional category, of the instructions, functions and constants in the NSDate date and time library.

Retrieving current date, day and time

CURRENTDATE%	1-10
CURRENTDAY%	1-11
CURRENTTIME%	1-12

Setting date and time format

DD_MM_YY*%	1-17
HH_MM%	1-29
MM_DD_YY*%	1-32
SETDATEFORMAT%	1-33

Retrieving format details

GETDATEFORMAT%	1-18
GETDATEFORMAT2%	1-19
GETDATESEPARATOR\$	1-20
GETTIMEAM\$	1-25
GETTIMEFORMAT%	1-26
GETTIMEPM\$	1-27
GETTIMESEPARATOR\$	1-28

Retrieving day and month names

GETLONGDAY\$	1-21
GETLONGMONTH\$	1-22
GETSHORTDAY\$	1-23
GETSHORTMONTH\$	1-24

Format checking

ISDATE%	1-30
ISTIME%	1-31

Conversion to and from integers and strings

DATES\$	1-13
DATE%	1-15
TIMES\$	1-43
TIME%	1-44

Milestone year

SETMILESTONEYEAR	1-47
GETMILESTONEYEAR%	1-48

Error management (constants returned)

DATE_ERROR%	1-16
-------------------	------

NSDate library reference

CURRENTDATE% Function

Returns the current date.

Syntax **CURRENTDATE%**

Return value INT(4)

Note The value returned is the number of days since 01/01/1900.

Example

```
MESSAGE "The current date is:", DATE$(CURRENTDATE%)
```

See also CURRENTTIME%, DATE\$, DATE%

CURRENTDAY% Function

Returns the current day of the week.

Syntax **CURRENTDAY%**

Return value INT(1)

Note The value returned lies between 0 (Sunday) and 6 (Saturday).

Example

```
MESSAGE "Today's day is:", GETLONGDAY$(CURRENTDAY%)
```

See also GETSHORTDAY\$, GETLONGDAY\$

CURRENTTIME% Function

Returns the current time.

Syntax **CURRENTTIME%**

Return value INT(4)

Note The value returned is the number of seconds since 00:00:00.

Example

```
MESSAGE "The time now is:", TIME$(CURRENTTIME%)
```

See also CURRENTDATE%, TIME\$, TIME%

DATE\$ Function

Converts an integer into a date.

Syntax **DATE\$** (*number*)

Parameters *number* INT(4) I integer to convert into a date

Return value CSTRING

Notes

1. *number* contains the number of days since 01/01/1900.
2. The date is returned in the current date format set by SETDATEFORMAT and SETDATESEPARATOR or, by default, the one set in the Control Panel.
3. DATE\$ returns "#ERR!" if the value passed is invalid.
4. The STRING\$ function in the NSMisc library can also be used to convert an integer into a date. See example.
5. The NSDate library can handle all dates starting from 1/1/1583 (start of the Gregorian calendar) by accepting negative integers for dates prior to 1/1/1900. For example:

DATE\$ (-115784)	returns "#ERR!"
DATE\$ (-115783)	returns "#ERR!"
DATE\$ (-115782)	returns "1/1/1583"
DATE\$ (-115781)	returns "2/1/1583"
...	
DATE\$ (-1)	returns "31/12/1899"
DATE\$ (0)	returns "1/1/1900"
DATE\$ (1)	returns "2/1/1900"
DATE\$ (2)	returns "3/1/1900"
...	

Example 1

```
MOVE DATE$(CURRENTDATE% + 1) TO S$ ; S$ contains tomorrow's date
```

Example 2

```
; Set date format
; Date format: year month day
SETDATEFORMAT YYYY MM DD%
; A hyphen is used as the separator
```

1-14 NSDATE Date and Time Library

```
SETDATESEPARATOR "-"  
  
MOVE DATE$(0) TO S$           ; S$ is set to "1900-1-1"  
MOVE DATE$(33602) TO S$      ; S$ is set to "1992-1-1"
```

Example 3

```
; STRING$ in the NSMisc library can also be used  
MOVE STRING$(33602, "yyyy-m-d") TO S$ ; S$ is set to "1992-1-1"
```

See also DATE%, DATE_ERROR%, CURRENTDATE%, TIMES\$, TIME%
 STRING\$ (NSMisc library)

DATE% Function

Converts an integer into a date.

Syntax **DATE%** (*date*)

Parameters *date* INT(4) I date to convert into an integer

Return value INT(4)

Notes

1. The date must be passed in the current date format set by SETDATEFORMAT and SETDATESEPARATOR or, by default, the one set in the Control Panel.
2. The integer returned contains the number of days since 01/01/1900.
3. DATE% returns DATE_ERROR% if the format of the date passed is incorrect.
4. The NSDate library can handle all dates starting from 1/1/1583 (start of the Gregorian calendar) by accepting negative integers for dates prior to 1/1/1900.

Example

```
; set date format
; date format : year month day
SETDATEFORMAT YYYY MM DD%
; a hyphen is used as the separator
SETDATESEPARATOR « _ »

MOVE DATE% (« 1990-1-1 ») TO I% ;I% is set to 32872
MOVE DATE% (« 1991-1-1 ») TO I% ;I% is set to 33237
```

See also DATE\$, DATE_ERROR%, CURRENTDATE%, TIME\$, TIME%

DATE_ERROR% Constant

Error code returned when an invalid date is supplied.

Syntax **DATE_ERROR%**

Notes

1. This value is returned by the DATE% function if the string passed to it as a parameter is not in date format.

2. It is declared internally as follows:

```
CONST DATE_ERROR%      -115783
```

See also DATE%, TIME_ERROR%

DD_MM_YY*% Constants

Logical values used to define the date format.

Syntax **DD_MM_YY%**
 DD_MM_YYYY%

Notes

1. These values are used by SETDATEFORMAT and returned by GETDATEFORMAT%.
2. Their meaning is as follows:
DD_MM_YY% <Day>, <Month>, <Year in 2 digits>
DD_MM_YYYY% <Day>, <Month>, <Year in 4 digits>
3. Two additional formats are available:
MM_DD_YY% or MM_DD_YYYY%
YY_MM_DD% or YYYY_MM_DD%
4. They are declared internally as follows:

CONST DD_MM_YY%	1
CONST DD_MM_YYYY%	4

See also MM_DD_YY*%, YY*_MM_DD%

GETDATEFORMAT% Function

Gets the format used for dates within the application.

Syntax **GETDATEFORMAT%**

Return value INT(1)

Notes

1. The format returned is the one set in the Control Panel, if it has not been modified by SETDATEFORMAT.
2. The following formats are returned by GETDATEFORMAT%:
DD_MM_YY%
DD_MM_YYYY%
MM_DD_YY%
MM_DD_YYYY%
YY_MM_DD%
YYYY_MM_DD%

Example

```
IF GETDATEFORMAT% = MM_DD_YYYY%  
  IF GETDATESEPARATOR$ = "/"  
    MESSAGE "Date", "The format is MM/DD/YYYY"  
  ENDIF  
ENDIF
```

See also GETDATESEPARATOR\$, SETDATEFORMAT, SETDATESEPARATOR
 GETTIMEFORMAT%, GETTIMESEPARATOR\$

GETDATEFORMAT2% Function

Gets the format used for dates within the application and the display format for days and months.

Syntax	GETDATEFORMAT2% (<i>day-leading-zero</i> , <i>month-leading-zero</i>)			
Parameters	<i>day-leading-zero</i>	INT(1)	I/O	indicates whether day numbers are displayed with leading zeros
	<i>month-leading-zero</i>	INT(1)	I/O	indicates whether month numbers between 1 and 9 are displayed with leading zeros

Return value INT(1)

Notes

1. The date format returned by this function is identical to the one returned by GETDATEFORMAT%. See the notes for this function.
2. The values of *day-leading-zero* and *month-leading-zero* (TRUE% or FALSE%) indicate whether day and month numbers are displayed with leading zeros.

Example

```
LOCAL INT DLZ%(1)      ; day-leading-zero flag
LOCAL INT MLZ%(1)      ; month-leading-zero flag

IF GETDATEFORMAT2%(DLZ%,MLZ%) = MM DD YYYY%
  IF DLZ% AND MLZ%
    MESSAGE "Date", \
      "The date format is Month,Day,Year with" & \
      "leading zeros for days and months"
  ENDIF
ENDIF
```

See also SETDATEFORMAT2, SETDATESEPARATOR, GETDATESEPARATOR\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$

GETDATESEPARATOR\$ Function

Gets the separator used for dates within the application.

Syntax **GETDATESEPARATORS**

Return value CSTRING

Notes

1. The separator returned is the one set in the Control Panel, if it has not been modified by SETDATESEPARATOR.
2. GETDATESEPARATOR\$ can return any character string as a separator.

Example

```
IF GETDATEFORMAT% = MM_DD_YYYY%  
  IF GETDATESEPARATOR$ = "/"  
    MESSAGE "Date", "The format is MM/DD/YYYY"  
  ENDIF  
ENDIF
```

See also GETDATEFORMAT%, SETDATEFORMAT, SETDATESEPARATOR
 GETTIMEFORMAT%, GETTIMESEPARATOR\$

GETLONGDAY\$ Function

Returns a string containing the day of the week corresponding to the integer passed as a parameter.

Syntax **GETLONGDAY\$(day)**

Parameters *day* INT(1) I integer corresponding to the day of the week.

Return value CSTRING

Notes

1. If *day* lies outside the authorized limits (from 0 for Sunday to 6 for Saturday), this function returns an empty string.
2. By default, the string returns the name of the day in English day string returned

0	"Sunday"
1	"Monday"
2	"Tuesday"
3	"Wednesday"
4	"Thursday"
5	"Friday"
6	"Saturday"
>=7	""
3. Use the SETLONGDAY instruction beforehand to set the days of the week in another language.

Example

```
MESSAGE "Today's day is:", GETLONGDAY$(CURRENTDAY%)
```

See also GETLONGMONTH\$, SETLONGDAY, SETLONGMONTH
GETSHORTDAY\$, GETSHORTMONTH\$

GETLONGMONTH\$ Function

Returns a string containing the day of the month corresponding to the integer passed as a parameter.

Syntax	GETLONGMONTH\$(month)		
Parameters	<i>month</i>	INT(1)	I integer corresponding to the month
Return value	CSTRING		
Notes			

1. If *month* lies outside the authorized limits (from 0 for January to 11 for December), this function returns an empty string.
2. By default, the string returns the name of the month in English:

month	string returned
0	"January"
1	"February"
2	"March"
3	"April"
4	"May"
5	"June"
6	"July"
7	"August"
8	"September"
9	"October"
10	"November"
11	"December"
>=12	""

3. Use the SETLONGMONTH instruction beforehand to set month names in another language.

Example

```
MOVE "/" TO SEP$
SETDATEFORMAT MM DD YY%
SETDATESEPARATOR SEP$
MOVE DATE$(CURRENTDATE%) TO TODAY$
MESSAGE "The current month is:", \
      GETLONGMONTH$(INT COPY$(TODAY$,1,POS$(SEP$,TODAY$)-1)-1)
```

See also	GETLONGDAY\$, SETLONGDAY, SETLONGMONTH GETSHORTDAY\$, GETSHORTMONTH\$
-----------------	--

GETSHORTDAY\$ Function

Returns the three first letters of the day of the week corresponding to the integer passed as a parameter.

Syntax **GETSHORTDAY\$(day)**

Parameters *day* INT(1) I integer corresponding to the day of the week.

Return value CSTRING

Notes

1. If *day* lies outside the authorized limits (from 0 for Sunday to 6 for Saturday), this function returns an empty string.
2. By default, the string returns the initial letters of the day in English:

day	string returned
0	"Sun"
1	"Mon"
2	"Tue"
3	"Wed"
4	"Thu"
5	"Fri"
6	"Sat"
>=7	""

3. Use the SETSHORTDAY instruction beforehand to set the days of the week in another language or to modify any of the strings returned.

Example

```
MESSAGE "Today's day is:", GETSHORTDAY$(CURRENTDAY%)
```

See also GETSHORTMONTH\$, SETSHORTDAY, SETSHORTMONTH
GETLONGDAY\$, GETLONGMONTH\$

GETSHORTMONTH\$ Function

Return the three first letters of the month corresponding to the integer passed as a parameter.

Syntax	GETSHORTMONTH\$(month)		
Parameters	<i>month</i>	INT(1)	I integer corresponding to the month
Return value	CSTRING		
Notes			

1. If *month* lies outside the authorized limits (from 0 for January to 11 for December), this function returns an empty string.
2. By default, the string returns the initial letters of the month in English:

month	string returned
0	"Jan"
1	"Feb"
2	"Mar"
3	"Apr"
4	"May"
5	"Jun"
6	"Jul"
7	"Aug"
8	"Sep"
9	"Oct"
10	"Nov"
11	"Dec"
>=12	""

3. Use the SETSHORTMONTH instruction beforehand to set month names in another language or to modify any of the strings returned.

Example

```
MOVE "/" TO SEP$
SETDATEFORMAT MM DD YY%
SETDATESEPARATOR SEP$
MOVE DATE$(CURRENTDATE%) TO TODAY$
MESSAGE "The current month is:", \
      GETSHORTMONTH$(INT COPY$(TODAY$,1,POS$(SEP$,TODAY$)-1)-1)
```

See also	GETSHORTDAY\$, SETSHORTDAY, SETSHORTMONTH GETLONGDAY\$, GETLONGMONTH\$
-----------------	---

GETTIMEAM\$ Function

Returns the A.M. symbols used by the application for times in 12-hour format.

Syntax **GETTIMEAM\$**

Return value CSTRING

Notes

1. The string returned uses the settings in the Control Panel if SETTIMEAMPM has not been used.
2. This function returns an empty string if the time display format set in the Control Panel is not expressed in 12 hour format or if the symbols have not been set by SETTIMEAMPM.

Example

```
SETTIMEAMPM "AM", "PM"  
MESSAGE GETTIMEAM$, GETTIMEPM$ ; displays "AM" and "PM"
```

See also GETTIMEFORMAT%, GETTIMEPM\$, GETTIMESEPARATORS
 SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR
 GETDATEFORMAT%, GETDATESEPARATOR\$

GETTIMEFORMAT% Function

Returns the format used for times within the application.

Syntax **GETTIMEFORMAT%**

Return value INT(1)

Notes

1. The value returned is the format set in the Control Panel if SETTIMEFORMAT has not been used.
2. The following time format constants are returned by GETTIMEFORMAT%:
HH_MM%
HH_MM_AMPM%
HH_MM_SS%
HH_MM_SS_AMPM%

Example

```
IF GETTIMEFORMAT% = HH MM SS%  
  IF GETTIMESEPARATOR$ = ":"  
    MESSAGE "Time", "The format is HH:MM:SS"  
  ENDIF  
ENDIF
```

See also GETTIMEAM\$, GETTIMEPM\$, GETTIMESEPARATOR\$
 SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR
 GETDATEFORMAT%, GETDATESEPARATOR\$

GETTIMEPM\$ Function

Returns the P.M. string used for times within the application.

Syntax **GETTIMEPM\$**

Return value CSTRING

Notes

1. The string returned uses the settings in the Control Panel if SETTIMEAMPM has not been used.
2. This function returns an empty string if the time display format set in the Control Panel is not expressed in 12 hour format or if the symbols have not been set by SETTIMEAMPM.

Example

```
SETTIMEAMPM "AM", "PM"  
MESSAGE GETTIMEAM$, GETTIMEPM$ ; displays "AM" and "PM"
```

See also GETTIMEAM\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$
 SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR
 GETDATEFORMAT%, GETDATESEPARATOR\$

GETTIMESEPARATOR\$ Function

Returns the separator used for times within the application.

Syntax **GETTIMESEPARATORS**

Return value CSTRING

Note The string returned contains the separator set in the Control Panel if SETTIMESEPARATOR has not been used.

Example

```
IF GETTIMEFORMAT% = HH MM SS%
  IF GETTIMESEPARATOR$ = ":"
    MESSAGE "Time", "The format is HH:MM:SS"
  ENDIF
ENDIF
```

See also GETTIMEAM\$, GETTIMEFORMAT%, GETTIMEPM\$
SETTIMEAMPM, SETTIMEFORMAT, SETTIMESEPARATOR
GETDATEFORMAT%, GETDATESEPARATORS

HH_MM*% Constants

Logical values used to define the time format.

Syntax

HH_MM%
HH_MM_AMPM%
HH_MM_SS%
HH_MM_SS_AMPM%

Notes

1. These values are used by SETTIMEFORMAT and returned by GETTIMEFORMAT%.
2. Their meaning is as follows:

HH_MM%	Time expressed in hours and minutes (24 hour format).
HH_MM_AMPM%	Time expressed in hours and minutes (12 hour format) with additional A.M. or P.M. symbols
HH_MM_SS%	Same as HH_MM% with seconds
HH_MM_SS_AMPM%	Same as HH_MM_AMPM% with seconds
3. You must use SETTIMEAMPM to specify the A.M. and P.M. symbols for the HH_MM_AMPM% or HH_MM_SS_AMPM% formats if they have not been set in the Control Panel.
4. They are declared internally as follows:

CONST HH_MM%	1
CONST HH_MM_AMPM%	2
CONST HH_MM_SS%	3
CONST HH_MM_SS_AMPM%	4

ISDATE% Function

Checks whether a string is in the current date format.

Syntax **ISDATE%** (*date*)

Parameters *date* CSTRING I string to test

Return value INT(1)

value	meaning
TRUE%	the string is a date in the correct format
FALSE%	the string is not in date format.

Note The reference date's format is the one specified by SETDATEFORMAT / SETDATESEPARATOR or, by default, the one set in the Control Panel.

See also ISTEIME%

ISTIME% Function

Checks whether a string is in the current time format.

Syntax **ISTIME%** (*time*)

Parameters *time* CSTRING I string to test

Return value INT(1)

value	meaning
TRUE%	the string is a time in the correct format.
FALSE%	the string is not in time format.

Note The reference time's format is the one specified by SETTIMEFORMAT / SETTIMESEPARATOR or, by default, the one set in the Control Panel.

See also ISDATE%

MM_DD_YY*% Constants

Logical values used to define the date format.

Syntax **MM_DD_YY%**
 MM_DD_YYYY%

- Notes**
- 1. These values are used by SETDATEFORMAT and returned by GETDATEFORMAT%.
 - 2. Their meaning is as follows:
MM_DD_YY% <Month>, <Day>, <Year in 2 digits>
MM_DD_YYYY% <Month>, <Day>, <Year in 4 digits>
 - 3. Two additional formats are available:
DD_MM_YY% or DD_MM_YYYY%
YY_MM_DD% or YYYY_MM_DD%
 - 4. They are declared internally as follows:

CONST MM_DD_YY%	2
CONST MM_DD_YYYY%	5

See also DD_MM_YY*%, YY*_MM_DD%

SETDATEFORMAT Instruction

Sets the format used for dates.

Syntax	SETDATEFORMAT <i>format</i>
Parameters	<i>format</i> INT(1) I date format
Notes	

1. This format will only be used within the application and will not modify the default format set in the Control Panel, which remains effective for any other applications that have not set a specific format.
2. The following formats are accepted by SETDATEFORMAT:
DD_MM_YY%
DD_MM_YYYY%
MM_DD_YY%
MM_DD_YYYY%
YY_MM_DD%
YYYY_MM_DD%
3. SETDATEFORMAT 0 uses the format set in the Control Panel.

Example 1

```
; Let's suppose that the date is January 2, 1995
SETDATEFORMAT YY_MM_DD%
SETDATESEPARATOR " / "
MOVE DATE$(CURRENTDATE%) TO A$ ; A$ is set to "95 / 1 / 2"
```

Example 2

```
; Return to the format set in the Control Panel
SETDATEFORMAT 0
```

See also SETDATESEPARATOR, GETDATEFORMAT%, GETDATESEPARATOR\$,
SETTIMEFORMAT, SETTIMESEPARATOR

SETDATEFORMAT2 Instruction

Sets the format used for dates and the display format used for days and month.

Syntax	SETDATEFORMAT2 <i>format, day-leading-zero, month-leading-zero</i>			
Parameters	<i>format</i>	INT(1)	I	date format
	<i>day-leading-zero</i>	INT(1)	I	indicates whether day numbers are displayed with leading zeros
	<i>month-leading-zero</i>	INT(1)	I	indicates whether month numbers between 1 and 9 are displayed with leading zeros

Notes

- 1. The date format set by this instruction is identical to the one set by SETDATEFORMAT. See the notes for this instruction.
- 2. The values of *day-leading-zero* and *month-leading-zero* (TRUE% or FALSE%) indicate whether day numbers and month numbers between 1 and 9 should be displayed with leading zones.

Example

```
; Let's suppose that the date is July 2, 1995
SETDATEFORMAT2 YYYY_MM_DD%, TRUE%, FALSE%
SETDATESEPARATOR " - "
MOVE DATE%(CURRENTDATE%) TO A$ ; A$ is set to "1995 / 8 / 04"
```

See also GETDATEFORMAT2%, SETDATESEPARATOR

SETDATESEPARATOR Instruction

Sets the separator used for dates.

Syntax **SETDATESEPARATOR** *separator*

Parameters *separator* CSTRING I date separator

Notes

1. This format will only be used within the application and will not modify the default separator set in the Control Panel, which remains effective for any other applications that have not set a specific separator.
2. SETDATESEPARATOR accepts any character string as a separator.
3. SETDATESEPARATOR « ! » uses the separator set in the Control Panel.

Example 1

```
; Lets suppose that the data is January 20, 1995
SETDATEFORMAT YY MM DD%
SETDATESEPARATOR « / »
MOVE DATE$(CURRENTDATE%) TO A$ ; A$ is set to « 95 / 01 / 20 »
```

Example 2

```
; Return to the separator set in the Control Panel
SETDATESEPARATOR « ! »
```

See also SETDATEFORMAT, GETDATEFORMAT%, GETDATESEPARATOR\$,
SETTIMEFORMAT, SETTIMESEPARATOR

SETLONGDAY Instruction

Associates a string with a day of the week

Syntax **SETLONGDAY** *day-integer, day-string*

Parameters	<i>day-integer</i>	INT(1)	I	day of the week
	<i>day-string</i>	CSTRING	I	day's name

Notes

1. *day-integer* must lie between 0 (for Sunday) and 6 (for Saturday).
2. The name set by this instruction will be returned by GETLONGDAY\$.
3. You can use this function to set the name of each day of the week in a language other than English, which is the default language for GETLONGDAY\$.

Example

```
; Days in German are:
SETLONGDAY 0, "Sonntag"
SETLONGDAY 1, "Montag"
SETLONGDAY 2, "Dienstag"
SETLONGDAY 3, "Mittwoch"
SETLONGDAY 4, "Donnerstag"
SETLONGDAY 5, "Freitag"
SETLONGDAY 6, "Samstag"
MESSAGE "Day #3 is: ", GETLONGDAY$(3) ; displays Mittwoch
```

See also SETLONGMONTH, GETLONGDAY\$, GETLONGMONTH\$, SETSHORTDAY,
 SETSHORTMONTH

SETLONGMONTH Instruction

Associates a string with a month.

Syntax **SETLONGMONTH** *month-integer*, *month-string*

Parameters *month-integer* INT(1) I month
 month-string CSTRING I name of the month

Notes

1. *month-integer* must lie between 0 (for January) and 11 (for December).
2. The month name set by this instruction will be returned by GETLONGMONTH\$.
3. You can use this function to set the name of each month of the year in a language other than English, which is the default language for GETLONGMONTH\$.

Example

```
; Months in French are:
SETLONGMONTH 0, "Janvier"
SETLONGMONTH 1, "Février"
SETLONGMONTH 2, "Mars"
SETLONGMONTH 3, "Avril"
SETLONGMONTH 4, "Mai"
SETLONGMONTH 5, "Juin"
SETLONGMONTH 6, "Juillet"
SETLONGMONTH 7, "Août"
SETLONGMONTH 8, "Septembre"
SETLONGMONTH 9, "Octobre"
SETLONGMONTH 10, "Novembre"
SETLONGMONTH 11, "Décembre"
MESSAGE "Month #5 is: ", GETLONGMONTH$(5) ; displays Juin
```

See also SETLONGDAY, GETLONGDAY\$, GETLONGMONTH\$,
 SETSHORTDAY, SETSHORTMONTH

SETSHORTDAY Instruction

Associates a short string with a day of the week.

Syntax **SETSHORTDAY** *day-integer*, *day-string*

Parameters	<i>day-integer</i>	INT(1)	I	day of the week
	<i>day-string</i>	CSTRING	I	day's short name

Notes

1. *day-integer* must lie between 0 (for Sunday) and 6 (for Saturday).
2. The short name set by this instruction is returned by GETSHORTDAY\$. It contains a maximum of the first five characters in *day-string*.
3. You can use this function to modify the default short names returned by GETSHORTDAY\$ or set a short name in a language other than English.

Example

```
; Short days in Spanish are:
SETSHORTDAY 0, "Dom"
SETSHORTDAY 1, "Lun"
SETSHORTDAY 2, "Mar"
SETSHORTDAY 3, "Mie"
SETSHORTDAY 4, "Jue"
SETSHORTDAY 5, "Vie"
SETSHORTDAY 6, "Sab"
MESSAGE "Day #4 is: ", GETSHORTDAY$(4)    ; displays Jue
```

See also SETSHORTMONTH, GETSHORTDAY\$, GETSHORTMONTH\$, SETLONGDAY, SETLONGMONTH

SETSHORTMONTH Instruction

Associates a short string with a month.

Syntax **SETSHORTMONTH** *month-integer*, *month-string*

Parameters *month-integer* INT(1) I month
 month-string CSTRING I month's short name

Notes

1. *month-integer* must lie between 0 (for January) and 11 (for December).
2. The short name set by this instruction will be returned by GETSHORTMONTH\$. It contains a maximum of the first five characters in *month-string*.
3. You can use this function to modify the default short names returned by GETSHORTMONTH\$ or set a short name in a language other than English.

Example

```
; Short months in Italian are:
SETSHORTMONTH 0, "Gen"
SETSHORTMONTH 1, "Feb"
SETSHORTMONTH 2, "Mar"
SETSHORTMONTH 3, "Apr"
SETSHORTMONTH 4, "Mag"
SETSHORTMONTH 5, "Giu"
SETSHORTMONTH 6, "Lug"
SETSHORTMONTH 7, "Ago"
SETSHORTMONTH 8, "Set"
SETSHORTMONTH 9, "Ott"
SETSHORTMONTH 10, "Nov"
SETSHORTMONTH 11, "Dic"
MESSAGE "Month #7 is: ", GETSHORTMONTH$(7)    ; displays Ago
```

See also SETSHORTDAY, GETSHORTDAY\$, GETSHORTMONTH\$, SETLONGDAY,
 SETLONGMONTH

SETTIMEAMPM Instruction

Sets the AM/PM strings used for times.

Syntax	SETTIMEAMPM <i>am, pm</i>			
Parameters	<i>am</i>	CSTRING	I	a.m. symbol
	<i>pm</i>	CSTRING	I	p.m. symbol

- Notes**
1. These symbols will only be used within the application and will not modify the symbols set in the Control Panel, which remain effective for any other applications that have not set specific symbols.
 2. SETTIMEAMPM accepts any character strings.
 3. The symbols set by this instruction will only be displayed if a 12 hour date format has been set in the Control Panel or if the date format set by SETTIMEFORMAT is HH_MM_AMPM% or HH_MM_SS_AMPM%.
 4. SETIMEAMPM "!", "!" uses the settings in the Control Panel.

Example

```
SETTIMEAMPM "AM", "PM"
SETTIMEFORMAT HH_MM_SS_AMPM%
MOVE TIME$(CURRENTTIME%) TO A$ ; A$ is set to "08: 30: 21 PM"

; Return to the format set in the Control Panel
SETTIMEAMPM "!", "!"
```

See also SETTIMEFORMAT, SETTIMESEPARATOR, GETTIMEAM\$,
GETTIMEPM\$,GETTIMEFORMAT%, GETTIMESEPARATOR\$,
SETDATEFORMAT, SETDATESEPARATOR

SETTIMEFORMAT Instruction

Sets the format used for times.

Syntax `SETTIMEFORMAT format`

Parameters *format* INT(1) I time format

Notes

1. This format will only be used within the application and will not modify the format set in the Control Panel, which remains effective for any other applications that have not set a specific time format.
2. The following formats are accepted by SETTIMEFORMAT:
HH_MM%
HH_MM_SS%
HH_MM_AMPM%
HH_MM_SS_AMPM%
For HH_MM% and HH_MM_SS%, hours may vary from 0 to 23.
For HH_MM_SS_AMPM% and HH_MM_SS_AMPM%, hours may vary from 0 to 11 only and the two strings set by SETTIMEAMPM are placed at the end of the string to differentiate AM from PM.
3. SETTIMEFORMAT 0 uses the settings in the Control Panel.

Example

```
; Let's suppose that the time is 20h 30mn 21s
SETTIMESEPARATOR ": "
SETTIMEFORMAT HH MM%
MOVE TIME$(CURRENTTIME%) TO A$ ; A$ is set to "20: 30"

; Return to the format set in the Control Panel
SETTIMEFORMAT 0
```

See also SETTIMEAMPM, SETTIMESEPARATOR, GETTIMEAM\$, GETTIMEPM\$, GETTIMEFORMAT%, GETTIMESEPARATOR\$, SETDATEFORMAT, SETDATESEPARATOR

SETTIMESEPARATOR Instruction

Sets the separator used for times.

Syntax **SETTIMESEPARATOR** *separator*

Parameters *separator* CSTRING I time separator

Notes

1. This separator will only be used within the application and will not modify the one set in the Control Panel, which remains effective for any other applications that have not set a specific time separator.
2. SETTIMESEPARATOR accepts any character strings.
3. SETTIMESEPARATOR "!" uses the settings in the Control Panel.

Example

```
; Let's suppose that the time is 20h 30mn 21s
SETTIMESEPARATOR ": "
SETTIMEFORMAT HH MM%
MOVE TIME$(CURRENTTIME%) TO A$    ; A$ is set to "20: 30"

; Return to the format set in the Control Panel
SETTIMESEPARATOR "!"
```

See also SETTIMEAMPM, SETTIMEFORMAT, GETTIMEAM\$, GETTIMEPM\$,
GETTIMEFORMAT%, GETTIMESEPARATOR\$, SETDATEFORMAT,
SETDATESEPARATOR

TIME\$ Function

Converts an integer into a time.

Syntax `TIME$ (number)`

Parameters *number* INT(4) I integer to convert into a time

Return value CSTRING

Notes

1. *number* must contain the number of seconds since 0h 0mn 0s.
2. The time is returned in the current time format set by SETTIMEFORMAT and SETTIMESEPARATOR or, by default, the one set in the Control Panel.
3. This function returns "#ERR!" if the integer passed is invalid.
4. The STRING\$ function in the NSMisc library can also be used to convert an integer to a time. See examples.
5. The NSDate library can handle all times between 00:00:00 and 23:59:59. TIME\$ returns "#ERR!" if the integer passed is negative (< 00:00:00) or exceeds 86399 (> 23:59:59).

Example 1

```
SETTIMEFORMAT HH MM SS%
SETTIMESEPARATOR ":"
MOVE TIME$(0) TO S$ ; S$ is set to "00:00:00"
MOVE TIME$(43200) TO S$ ; S$ is set to "12:00:00"
```

Example 2

```
; STRING$ in the NSMisc library can also be used
MOVE STRING$(43200, "hh:mm:ss") TO S$ ; S$ is set to "12:00:00"
```

See also TIME%, TIME_ERROR%, CURRENTTIME%, DATE\$, DATE%, STRING\$ (NSMisc library)

TIME% Function

Converts an integer into a time.

Syntax **TIME%** (*time*)

Parameters *time* CSTRING I time to convert into an integer

Return value INT(4)

Notes

1. The time must be passed in the current time format set by SETTIMEFORMAT and SETTIMESEPARATOR or, by default, the format set in the Control Panel.
2. The time returned contains the number of seconds since 0h 0mn 0s.
3. This function returns TIME_ERROR% if the time passed does not have the correct format.
4. The NSDate library can handle all times between 00:00:00 and 23:59:59.

Example

```
SETTIMEFORMAT HH MM SS%
SETTIMESEPARATOR ":"
MOVE TIME% («09:00:00») TO I%    ; I% is set to 32400
MOVE TIME% («23:59:59») TO I%    ; I% is set to 86399 (maximum value that can be
                                   ; returned by TIME%)
```

See also TIME\$, TIME_ERROR%, CURRENTTIME%, DATE\$, DATE%, STRING\$
 (NSMisc library)

TIME_ERROR% Constant

Error code returned when an invalid time is supplied.

Syntax **TIME_ERROR%**

Notes

1. This value is returned by the TIME% function if the string passed to it is not a valid time.
2. It is declared internally as follows:

```
CONST TIME_ERROR%    -1
```

See also TIME%, DATE_ERROR%

YY*_MM_DD% Constants

Logical values used to define the date format.

Syntax **YY_MM_DD%**

YYYY_MM_DD%

Notes

1. These values are used by SETDATEFORMAT and returned by GETDATEFORMAT%.

2. Their meaning is as follows:

YY_MM_DD% <Year in 2 digits>, <Month>, <Day>

YYYY_MM_DD% <Year in 4 digits>, <Month>, <Day>

3. Two additional formats are available:

DD_MM_YY% or DD_MM_YYYY%

MM_DD_YY% or MM_DD_YYYY%

4. They are declared internally as follows:

CONST YY_MM_DD% 3

CONST YYYY_MM_DD% 6

See also DD_MM_YY*%, MM_DD_YY*%

SETMILESTONEYEAR Instruction

Specifies the milestone year.

Syntax	SETMILESTONEYEAR <i>MilestoneYear</i>			
Parameter	<i>MilestoneYear</i>	INT(4)	I	The milestone year (between 0 and 99)

Notes

1. Before NatStar 2.16, dates using a two digit year format (for example 01/01/13), the system assumed that the date was for the current century (01/01/1913).
2. You can now specify a milestone year that lets you change the current century for dates. The algorithm is as follows:

For two digit year formats:

- If the year is less than the milestone year, the year is interpreted as being after2000.
- If the year is greater than or equal to the milestone year, the year is interpreted as being before 2000.

For example, if the milestone year is set to 30:

- The date 010129 is 01/01/2029
- The date 010130 is 01/01/1930
- The date 010131 is 01/01/1931

3. You can also specify the milestone year in the [Year2000] section of the NSLIB.INI configuration file using the following format:

```
[Year2000]
;Force4Digits=False
MilestoneYear=30
```

4. This value is read at startup. If no value is specified, the default value is -1. If you do not set a milestone year, NatStar (or NS-DK or NatWeb) behaves as before, interpreting dates as being in the current century.

GETMILESTONEYEAR% Function

Returns the milestone year.

Syntax **GETMILESTONEYEAR%**

Return value INT(4) milestone year

Chapter 2

NSMath Mathematics Library



This library performs mathematical calculations (trigonometry, logarithms, etc.)

This chapter explains

- How to install this library
- General aspects of the functions and instructions
- In which NCL file, the verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation 2-3

NCL Language Extensions for Mathematical Calculations 2-4

NSMATH.NCL File 2-5

Functional categories in the NSMath library 2-6

NSMath library reference..... 2-8

Installation

Declare NSMATH.NCL in the libraries required to develop your application.

Ensure that the file NSxxMATH.DLL has been placed in a PATH directory under Windows.

NCL Language Extensions for Mathematical Calculations

Trigonometric calculations are performed with SIN#, COS#, ARCTAN#.

Logarithmic calculations are performed with LN# and EXP#.

SQR# returns the square and SQRT# returns the square root.

POWER# raises a number to a power.

FRAC# returns the decimal portion and INT# returns the integer portion.

Lastly, PI# returns the value of the real number Pi.

All the above functions apply to NUM(8) type integers. They each have an equivalent function that applies to NUM(10) “extended” real numbers. The name of each of these functions is prefixed by E. (ESIN#, ELN#, EFRAC#, etc).

NSMATH.NCL File

The verbs in this library (NSMath) are described below. They are declared in a text file, written in NCL, called NSMATH.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSMATH.NCL file :

1. Navigate to the <NATSTAR>\NCL, <NSDK>\NCL or <NATWEB>\NCL directory.

NB : <NATSTAR> means the directory in which NatStar was installed.

NB : <NSDK> means the directory in which NS-DK was installed.

NB : <NATWEB> means the directory in which NatWeb was installed.

2. Open the NSMATH.NCL file with any text editor to see its contents.

Functional categories in the NSMath library

Here is a list, arranged by functional category, of the functions in the NSMath mathematics (calculations) library.

Integer and decimal part of a real number

EFRAC#	2-14
EINT#	2-15
FRAC#.....	2-23
INT#.....	2-24

Squares, roots and powers

EPOWER#	2-18
ESQR#	2-20
ESQRT#.....	2-21
POWER#	2-27
SQR#.....	2-29
SQRT#	2-30

"Pi" constant

EPI#	2-17
PI#.....	2-26

Trigonometric functions

ARCTAN#.....	2-9
COS#.....	2-10
EARCTAN#.....	2-11
ECOS#	2-12
ESIN#	2-19
SIN#.....	2-28

Logarithms and exponents

EEXP#.....	2-13
ELN#.....	2-16
EXP#.....	2-22
LN#	2-25

NSMath library reference

ARCTAN# Function

Returns the arc tangent of a real number.

Syntax **ARCTAN#** (*x*)

Parameters *x* NUM(8) I real number in radians

Return value NUM(8)

Note Use EARCTAN# if *x* is a NUM(10) real number.

Example

```
LOCAL N#  
MOVE ARCTAN# (1) TO N# ; N# is set to Pi/4, i.e. 0.78539...
```

See also EARCTAN#, COS#, SIN#

COS# Function

Returns the cosine of a real number.

Syntax **COS#** (*x*)

Parameters *x* NUM(8) I real number in radians

Note Use ECOS# if *x* is a NUM(10) real number.

Return value NUM(8)

Example

```
LOCAL N#  
  
MOVE COS#(0)        TO N#    ; N# is set to 1  
MOVE COS#(PI#/4) TO N#    ; N# is set to (square root of 2) / 2,  
                             ; i.e. 0.70710...  
MOVE COS#(1)        TO N#    ; N# is set to 0.54030...  
MOVE COS#(PI#/2) TO N#    ; N# is set to 0
```

See also ARCTAN#, SIN#, COS#

EARCTAN# Function

Returns the arc tangent of a real number.

Syntax **EARCTAN#** (*x*)

Parameters *x* NUM(10) I real number in radians

Return value NUM(10)

Note Identical to ARCTAN# except that it manipulates NUM(10) real numbers.

See also ARCTAN#, ECOS#, ESIN#

ECOS# Function

Returns the cosine of a real number.

Syntax	ECOS# (<i>x</i>)
Parameters	<i>x</i> NUM(10) I real number in radians**
Return value	NUM(10)
Note	Identical to COS# except that it manipulates NUM(10) real numbers.
See also	COS#, EARCTAN#, ESIN#

EEXP# Function

Returns the exponent of a real number.

Syntax **EEXP#**(x)

Parameters x NUM(10) I real number

Return value NUM(10)

Note Identical to EXP# except that it manipulates NUM(10) real numbers.

See also EXP#, ELN#

EFRAC# Function

Returns the decimal portion of a real number.

Syntax	EFRAC# (<i>x</i>)
Parameters	<i>x</i> NUM(10) I real number
Return value	NUM(10)
Note	Identical to FRAC# except that it manipulates NUM(10) real numbers.
See also	FRAC#, EINT#

EINT# Function

Returns the integer part of a real number.

Syntax **EINT#**(x)

Parameters x NUM(10) I real number

Return value NUM(10)

Note Identical to INT# except that it manipulates NUM(10) real numbers.

See also INT#, EFRAC#

ELN# Function

Returns the Naperian logarithm of a real number (strictly positive).

Syntax	ELN# (<i>x</i>)		
Parameters	<i>x</i>	NUM(10) I	real number
Return value	NUM(10)		
Notes	<div><div>1.</div><div>2.</div></div> This function returns 0 (zero) if <i>x</i> is negative or null. Identical to LN# except that it manipulates NUM(10) real numbers.		
See also	LN#, EEXP#		

EPI# Function

Returns the value of Pi, i.e. 3.14159...

Syntax	EPI#
Return value	NUM(10)
Note	Identical to PI# except that it returns a NUM(10) real number.
See also	PI#

EPOWER# Function

Returns the value of X raised to the power of Y (real numbers).

Syntax	EPOWER# (<i>x</i> , <i>y</i>)			
Parameters	<i>x</i>	NUM(10)	I	real number
	<i>y</i>	NUM(10)	I	power
Return value	NUM(10)			
Note	Identical to POWER# except that it manipulates NUM(10) real numbers.			
See also	POWER#			

ESIN# Function

Returns the sine of a real number.

Syntax **ESIN#** (*x*)

Return value NUM(10)

Parameters *x* NUM(10) I real number in radians

Note Identical to SIN# except that it manipulates NUM(10) real numbers.

See also EARCTAN#, ECOS#, SIN#

ESQR# Function

Returns the square of a real number.

Syntax	ESQR#(<i>x</i>)		
Parameters	<i>x</i>	NUM(10) I real number	
Return value	NUM(10)		
Note	Identical to SQR# except that it manipulates NUM(10) real numbers.		
See also	ESQRT#, SQR#		

ESQRT# Function

Returns the square root of a (positive) real number.

Syntax **ESQRT#**(x)

Parameters x NUM(10) I real number

Return value NUM(10)

Note Identical to SQRT# except that it manipulates NUM(10) real numbers.

See also ESQR#, SYRT#

EXP# Function

Returns the exponent of a real number.

Syntax	EXP#(<i>x</i>)
Parameters	<i>x</i> NUM(8) I real number
Return value	NUM(8)
Note	Use EEXP# if <i>x</i> is a NUM(10) real number.

Example

```
LOCAL N#  
  
MOVE EXP#(0) TO N# ; N# is set to 1  
MOVE EXP#(1) TO N# ; N# is set to 2.71828...  
MOVE EXP#(2) TO N# ; N# is set to 7.38905...  
                     ; (square of the previous number)  
  
; Calculate 23  
  
MOVE EXP#(3*LN#(2)) TO N# ; N# is set to 8  
                        ; better to use POWER#(2,3)
```

See also EEXP#, LN#

FRAC# Function

Returns the decimal portion of a real number.

Syntax **FRAC#**(x)

Parameters x NUM(8) I real number

Return value NUM(8)

Note Use EFRAC# if x is a NUM(10) real number.

Example

```
LOCAL N#  
MOVE FRAC# (PI#) TO N# ; N# is set to 0.14159...
```

See also EFRAC#, INT#

INT# Function

Returns the integer part of a real number.

Syntax INT#(*x*)

Parameters *x* NUM(8) I real number

Return value NUM(8)

Note Use EINT# if *x* is a NUM(10) real number.

Example

```
LOCAL N#  
MOVE INT# (PI#) TO N# ; N# is set to 3
```

See also EINT#, FRAC#

LN# Function

Returns the Naperian logarithm of a real number (strictly positive).

Syntax **LN#(*x*)**

Parameters *x* NUM(8) I real number

Return value NUM(8)

Notes

This function returns 0 (zero) if *x* is negative or null.

Use ELN# if *x* is a NUM(10) real number.

Example

```
LOCAL N#  
  
MOVE LN#(0) TO N#    ; Error !  
MOVE LN#(1) TO N#    ; N# is set to 0  
MOVE LN#(2) TO N#    ; N# is set to 0.69314...  
  
; Calculate 23  
  
MOVE EXP# (3*LN#(2)) TO N# ; N# contains 8  
                             ; Better to use POWER#(2,3)
```

See also ELN#, EXP#

PI# Function

Returns the value of Pi, i.e. 3.14159...

Syntax	PI#
Return value	NUM(8)
Note	Use EPI# if x is a NUM(10) real number.
Example	<pre>LOCAL N# ; Calculation of a circle's area MOVE PI# * SQR#(2) TO N# ; N# is set to 12.56637...</pre>
See also	EPI#

POWER# Function

Returns the value of X raised to the power of Y (real numbers).

Syntax	POWER#(x,y)			
Parameters	x	NUM(8)	I	real number
	y	NUM(8)	I	power
Return value	NUM(8)			
Note	Use EPOWER# if x and y are NUM(10) real numbers.			

Example

```
LOCAL N#  
  
; Calculate 23  
MOVE POWER#(2,3) TO N#    ; N# is set to 8
```

See also EPOWER#

SIN# Function

Returns the sine of a real number.

Syntax	SIN# (<i>x</i>)
Parameters	<i>x</i> NUM(8) I real number in radians
Return value	NUM(8)
Note	Use ESIN# if <i>x</i> is a NUM(10) real number.

Example

```
LOCAL N#
MOVE SIN#(0)            TO N#    ; N# is set to 0
MOVE SIN#(PI#/4) TO N#    ; N# is set to (square root of 2) / 2,
                             ; i.e. 0.70710...
MOVE SIN#(1)            TO N#    ; N# is set to 0.84147...
MOVE SIN#(PI#/2) TO N#    ; N# is set to 1
```

See also ARCTAN#, COS#, ESIN#

SQR# Function

Returns the square of a real number.

Syntax	SQR#(<i>x</i>)
Parameters	<i>x</i> NUM(8) I real number
Return value	NUM(8)
Notes	

Do not confuse with SQRT#.
Use ESQR# if *x* is a NUM(10) real number.

Example

```
LOCAL N#  
MOVE SQR#(3) TO N# ; N# is set to 9
```

See also	ESQR#, SQRT#
----------	--------------

SQRT# Function

Returns the square root of a (positive) real number.

Syntax	SQRT# (<i>x</i>)
Parameters	<i>x</i> NUM(8) I real number
Return value	NUM(8)
Notes	

Do not confuse with SQR#.
Use ESQRT# if *x* is a NUM(10) real number.

Example

```
LOCAL N#  
MOVE SQRT# (-1) TO N#    ; Error !  
MOVE SQRT# (9) TO N#    ; N# is set to 3
```

See also ESQR#, SQR#

Chapter 3

NSFexe Library



This NSFexe library allows you to launch executables or start « command line » sessions from your application. You can run these executables in synchronous or asynchronous mode and, if required, retrieve their return code in a variable or via a notification sent in a user message to one of the application's window.

In the remainder of this chapter, the term « console » designates a « command line » session used to type a system command or run a batch file. This session is displayed in a window and often called a « DOS box » under Windows 3.x.

This chapter explains

- How to install this library
- General aspects of its functions and instructions
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation	3-3
NCL Language Extensions for executing programs.....	3-4
NSFEXE.NCL File	3-5
Functional categories in the NSFexe library	3-6
Opening and closing DOS consoles	3-6
Console opening modes and program starting modes (constants)	3-6
Starting / stopping a program on a console	3-6
Starting / stopping a program	3-6
NSFexe Library Reference	3-7

Installation

Declare NSFEXE.NCL in the libraries required to develop your application.

Ensure that the NSxxFEXE.DLL file has been placed in a PATH directory under Windows.

NCL Language Extensions for executing programs

The NSFexe library includes the following verbs and constants:

OPENCONSOLE% opens a DOS console used to execute programs. Any notifications are sent to a NatStar (or NS-DK or NatWeb) window.
CLOSECONSOLE% closes this console.

STARTCONSOLEEXECEX% starts a program in a DOS console.
STOPCONSOLEEXEC% stops the program started in this console.

STARTEXCUTEEX% starts a program and redirects its output to a NatStar (or NS-DK or NatWeb) window. KILLEXCUTE and STOPEXCUTEEX terminate the program started by this function.

STARTEXCUTE2% launches a program and redirects its output to an NS-DK window. STOPEXCUTE2 stops the program started by this function.

CON_NFY_% and EXEC_NFY_TERMINATE% are constants used to notify the NatStar (or NS-DK or NatWeb) window of the program's status.

The FEXE_% constants indicate how a console will be opened or a program will be started.

NSFEXE.NCL File

The verbs in this library (NSFexe) are described below. They are declared in a text file, written in NCL, called NSFEXE.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSFEXE.NCL file :

1. Navigate to the <NATSTAR>\NCL,<NSDK>\NCL or <NATWEB> directory.
NB : <NATSTAR> means the directory in which NatStar was installed.
NB : <NSDK> means the directory in which NS-DK was installed.
NB : < NATWEB> means the directory in which NatWeb was installed.
2. Open the NSFEXE.NCL file with any text editor to see its contents.

Functional categories in the NSFexe library

Here is a list, arranged by functional category, of the instructions, functions and constants in the NSFexe program execution library.

Opening and closing DOS consoles

CLOSECONSOLE%	3-8
CON_NFY_ *%	3-9
OPENCONSOLE%	3-13

Console opening modes and program starting modes (constants)

FEXE_ *%	3-11
----------------	------

Starting / stopping a program on a console

STARTCONSOLEEXECEX%	3-15
STOPCONSOLEEXEC%	3-22

Starting / stopping a program

EXEC_NFY_TERMINATE%	3-10
KILLEXECUTE	3-12
STARTEXECUTEEX%	3-17
STARTEXECUTE2%	3-20
STOPEXECUTEEX	3-23
STOPEXECUTE2	3-24

NSFexe Library Reference

CLOSECONSOLE% Function

Closes a console.

Syntax	CLOSECONSOLE% (<i>cons-handle</i>)		
Parameters	<i>cons-handle</i>	INT(4)	I handle of the console to close
Return value	INT(1)		
	value	meaning	
	TRUE%	the console has been closed successfully	
	FALSE%	an error occurred while closing the console	
Note	The console must have been opened by a prior call to OPENCONSOLE% , which returned the console's handle.		

Example

```
; See the example for OPENCONSOLE%
```

See also OPENCONSOLE%, CON_NFY_**% constants

CON_NFY_*% Constants

These constants represent the notifications sent by a console in a user-defined event.

Syntax **CON_NFY_CLOSED%**
 CON_NFY_PROGENDED%
 CON_NFY_PROGOUTPUT%

Notes

1. The user-defined event is the one specified when the console was opened by OPENCONSOLE%.
2. These constants are sent in the PARAM12% parameter of the USERx event, together with additional information passed in PARAM34% or PARAM2\$:
 CON_NFY_CLOSED%
 Indicates that the console has been closed, either by calling CLOSECONSOLE% or by the user (via the console's system menu).
 CON_NFY_PROGENDED%
 Indicates that the program executed in the console has ended. PARAM34% contains the program exit code.
 CON_NFY_PROGOUTPUT%
 Indicates that information has been output by the program. In this case, PARAM2\$ is a CSTRING that contains the message.
3. The only notification that can be used for a program started by STARTEXECUTEEX% is EXEC_NFY_TERMINATE%.

4. They are declared internally as follows:

CONST CON_NFY_PROGENDED%	3
CONST CON_NFY_PROGOUTPUT%	4
CONST CON_NFY_CLOSED%	5

See also OPENCONSOLE%, CLOSECONSOLE%

EXEC_NFY_TERMINATE% Constant

Indicates that the program has terminated.

Syntax **EXEC_NFY_TERMINATE%**

Notes

1. This constant is sent in the PARAM12% parameter of the user-defined event specified when the program was started by STARTEXECUTEEX%. It notifies the NatStar (NS-DK or NatWeb) window that the program has terminated. This constant is the only notification that can be sent by a program started with STARTEXECUTEEX%.
2. The PARAM34% parameter of the USERx event will contain the program exit code.
3. It is declared internally as follows:

```
CONST EXEC_NFY_TERMINATE%     0
```

See also STARTEXECUTEEX%

FEXE_*% Constants

These constants indicate how a console is opened or a program is started.

Syntax

FEXE_HIDDEN%
FEXE_MAXIMIZED%
FEXE_MINIMIZED%
FEXE_NORMAL%
FEXE_USEPOSITIONSIZE%

Notes

1. The meaning of these constants is as follows:


FEXE_HIDDEN% the console or the program's main window is hidden when it is opened.

FEXE_MAXIMIZED % the console or the program's main window is maximized when it is opened.

FEXE_MINIMIZED% the console or the program's main window is minimized when it is opened.

FEXE_NORMAL% the console or the program's main window is opened with a size and position set by the system.

FEXE_USEPOSITIONSIZE % the console or the program's main window is opened with the dimensions specified by the OPENCONSOLE% or STARTEXECUTEEX% function.

2.  These options may be ignored by a program if it is not designed to accept this information when it is started up.

3. They are declared internally as follows:

CONST FEXE_NORMAL%	\$0000
CONST FEXE_HIDDEN%	\$0001
CONST FEXE_MAXIMIZED%	\$0002
CONST FEXE_MINIMIZED%	\$0004
CONST FEXE_USEPOSITIONSIZE %	\$8000

See also

OPENCONSOLE%, STARTEXECUTEEX%

KILLEXECUTE Instruction

Terminates a program.

Syntax **KILLEXECUTE** *program-ID*

Parameters *program-ID* INTEGER I ID of the program to terminate

Notes

1. The program must have been started by a prior call to STARTEXECUTEEX%, which returned the program ID.
2. A program aborted by KILLEXECUTE cannot prevent itself from terminating, unlike STOPEXECUTEEX, which sends a prior request to the program.

Example

EXECUTED event for the 'start' button

```
ProgID% = STARTEXECUTEEX%("APP.EXE" , " " , " " , \
                           SELF% , 2 , FALSE% , \
                           ErrCode% , RetCode% , \
                           FEEXE_NORMAL% , 0 , 0 , 0 , 0)
```

USER2 event for the window

```
IF PARAM12% = EXEC_NFY_TERMINATE%
    MESSAGE "The program has terminated" , \
           "Exit code =" && PARAM34%
ENDIF
```

EXECUTED event for the 'stop' button

```
; Terminate the program immediately
KILLEXECUTE ProgID%
```

See also STOPEXECUTEEX, STARTEXECUTEEX%

OPENCONSOLE% Function

Opens a console and returns its handle.

Syntax	OPENCONSOLE% (<i>title, create-flag, win-handle, event-nbr, mode, xpos, ypos, width, height</i>)			
Parameters	<i>title</i>	CSTRING	I	title of the console
	<i>create-flag</i>	INT(1)	I	indicates whether a new console should be opened
	<i>win-handle</i>	INT(4)	I	handle of the window that will receive any notifications
	<i>event-nbr</i>	INTEGER	I	ID of the user-defined event used to send notifications
	<i>mode</i>	INTEGER	I	console start-up mode
	<i>xpos</i>	INTEGER	I	
	<i>ypos</i>	INTEGER	I	position of the console
	<i>width</i>	INTEGER	I	
	<i>height</i>	INTEGER	I	size of the console
Return value	INT(4)			
	value	meaning		
	0	error opening the console		
	not null	console handle		

Notes

- title* is only displayed if the console appears in a non-maximized window. It is displayed in the window's title bar.
- If *create-flag* is set to FALSE%, the program will run in an existing console. If no console has been opened, OPENCONSOLE% will create a console. If *create-flag* is set to TRUE%, a new console will be created.
- Any notifications sent by the console (CON_NFY_%% constants) are redirected to the window identified by *win-handle*. They will be sent in the user-defined event specified by *event-nbr* (0 for the USER0 event, 1 for the USER1 event, and so on up to 15).
Notifications are sent in the following cases:
 - When information is output by the program and displayed in the console (this information can be received line by line or in blocks depending on the receive mode specified with STARTCONSOLEEXEC%).
 - When the program started by the console terminates.

- When the console itself terminates.

No notifications are sent if 0 is passed in *win-handle*.

4. *mode* should contain a combination of FEXE_%% constants. It specifies the size of the console when it is displayed: normal, maximized, minimized, etc.
5. *xpos*, *ypos*, *width* and *height* specify the position and size of the console opened. They are expressed in pixels and are only relevant if *mode* has been set to FEXE_USEPOSITIONSIZE%.

Example

```
LOCAL INT HD_Cons%(4)
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%

; Create a new console with a notification request
; in the USER 2 event for the current window
HD_Cons% = OPENCONSOLE$("Console title" , \
                        TRUE%, SELF%, 2, \
                        FEXE_NORMAL%, \
                        0, 0, 0, 0)

IF HD_Cons% <> 0
    ; Start a program in synchronous mode
    IF STARTCONSOLEEXECEX%(HD_Cons%, \
                           "APP.EXE", "", ". " \
                           TRUE%, TRUE%, TRUE%, \
                           ErrCode%, RetCode%)

        ; The return from STARTCONSOLEEXECEX% indicates that
        ; the message started in synchronous mode has
        ; terminated.
        ; Display its return code
        MESSAGE "Program terminated", "Return code =" && RetCode%
    ELSE
        MESSAGE "Cannot start program", \
                "System error =" && ErrCode%
    ENDIF
    IF NOT CLOSECONSOLE%(HD_Cons%)
        MESSAGE "Warning", "Error closing the console"
    ENDIF
ELSE
    MESSAGE "Warning", "Error opening the console"
ENDIF
```

See also

STARTCONSOLEEXECEX%, STOPCONSOLEEXEC%,
CLOSECONSOLE%

STARTCONSOLEEXECEX% Function

Starts a program (executable or batch) in an open console.

Syntax	STARTCONSOLEEXECEX% (<i>cons-handle, prog-name, params, working-dir, mode, echo, synch, err-code, return-code</i>)		
Parameters	<i>cons-handle</i>	INT(4)	I handle of the console in which the program will be executed
	<i>prog-name</i>	CSTRING	I program name
	<i>params</i>	CSTRING	I program's parameters
	<i>working-dir</i>	CSTRING	I program's working directory
	<i>mode</i>	INT(1)	I mode for receiving program output
	<i>echo</i>	INT(1)	I display/hide console
	<i>synch</i>	INT(1)	I synchronous/asynchronous mode
	<i>err-code</i>	INTEGER	I/Osystem error code
	<i>return-code</i>	INTEGER	I/Oprogram's return code
Return value	INT(1)		
	value	meaning	
	TRUE%	the program was started successfully	
	FALSE%	an error occurred while starting the program in the console. <i>err-code</i> indicates the system error number that identifies the problem.	

Notes

1. The program will run in the console opened by OPENCONSOLE% and identified by *cons-handle*.
2. *working-dir* specifies the working directory for *prog-name*. This is equivalent to the information in the Directory field of the *Properties* box in the Windows.
3. *mode* indicates how the program's output will be received: raw (*mode* set to FALSE%) or line by line (*mode* set to TRUE%).
4. Depending on the value of *echo*, the program's output will either be displayed in the console (*echo* set to TRUE%) or hidden (*echo* set to FALSE%).
5. The program can be started in synchronous mode (*synch* set to TRUE%) or asynchronous mode (*synch* set to FALSE%).
When a program is started in synchronous mode:

- The application that started the program is blocked until the program terminates. This applies to keyboard and mouse input but not to messages, which continue to be processed.
- The return code sent by the message when it terminates is received in *return-code*.

When a program is started in asynchronous mode:

- The program and the application that started it run simultaneously.
- The only way of knowing that the program has terminated is via a notification – *return-code* is not relevant in this case.

6. The *err-code* variable should only be tested if STARTCONSOLEEXECEX% returns FALSE%, otherwise its value is meaningless.

The *return-code* variable is only relevant for a program started in synchronous mode that returns a value when it terminates. As a general rule, the program should return the value 0 (zero) to indicate that it terminated normally; any other values are specific to the program.

7. The notification and return code are equivalent for a program started in synchronous mode, since the notification message is received by the application before it can retrieve the return code. How this code is retrieved depends entirely on the application developer.

Example

```
LOCAL INT HD_Cons%(4)
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%

; Create a new console with a notification request
; in the USER 2 event for the current window
HD_Cons% = OPENCONSOLE%("Console title" ,\
                        TRUE%, SELF%, 2, \
                        FEFE_NORMAL%, \
                        0, 0, 0, 0)

IF HD_Cons% <> 0
    ; Start a program in synchronous mode
    IF STARTCONSOLEEXECEX%(HD_Cons%, \
                          "APP.EXE", "", "." \
                          TRUE%, TRUE%, TRUE%, \
                          ErrCode%, RetCode%)

        ; The return from STARTCONSOLEEXECEX% indicates that
        ; the message launched in synchronous mode has
        ; terminated.
        ; Display its return code
        MESSAGE "Program terminated", "Return code =" && RetCode%
    ELSE
        MESSAGE "Cannot start program", \
              "System error =" && ErrCode%
    ENDIF
    IF NOT CLOSECONSOLE%(HD_Cons%)
        MESSAGE "Warning", "Error closing the console"
    ENDIF
ELSE
    MESSAGE "Warning", "Error opening the console"
ENDIF
```

See also

STOPCONSOLEEXEC%

STARTEXECUTEEX% Function

Starts a program and returns its ID.

Syntax	STARTEXECUTEEX% (<i>prog-name, params, working-dir, win-handle, event-nbr, synch, err-code, return-code, mode, xpos, ypos, width, height</i>)			
Parameters	<i>prog-name</i>	CSTRING	I	program name
	<i>params</i>	CSTRING	I	program's parameters
	<i>working-dir</i>	CSTRING	I	program's working directory
	<i>win-handle</i>	INT(4)	I	handle of the window that will receive any notifications
	<i>event-nbr</i>	INTEGER	I	ID of the user-defined event used to send notifications
	<i>synch</i>	INT(1)	I	synchronous/asynchronous
	<i>err-code</i>	INTEGER	I/O	system error code
	<i>return-code</i>	INTEGER	I/O	program's return code
	<i>mode</i>	INTEGER	I	program's display mode
	<i>xpos</i>	INTEGER	I	
	<i>ypos</i>	INTEGER	I	position of the program's main window
	<i>width</i>	INTEGER	I	
	<i>height</i>	INTEGER	I	size of the program's main window
Return value	INTEGER			
	value	meaning		
	0	an error occurred while starting the program. <i>err-code</i> indicates the system error number that identifies the problem.		
	not null	ID of the program started up		

Notes

1. *working-dir* specifies the working directory for *prog-name*. This is equivalent to the information in the Directory field of the *Properties* box in the Windows.
2. The program's messages will be sent to the window identified by *win-handle* in the user-defined event identified by *event-nbr* (0 for the USER0 event, 1 for the USER1 event, and so on up to 15).

The notification is not sent until the program terminates, whether or not the program was started in synchronous or asynchronous mode. In this case, PARAM12% in the user message is set to EXEC_NFY_TERMINATE% and PARAM34% contains the program's return code.

The notification and return code are equivalent for a program started in synchronous mode, since the notification message is received by the application before it can retrieve the return code. How this code is retrieved depends entirely on the application developer.

Pass 0 in *win-handle* if you don't want a notification to be sent, for example, if you want to retrieve the program's return code in the *return-code* variable.

3. Depending on the value of *synch*, the program will either be executed asynchronously (*synch* set to FALSE%) or synchronously (*synch* set to TRUE%).

When a program is started in synchronous mode:

- The application that started the program is blocked until the program terminates. This applies to keyboard and mouse input but not to messages, which continue to be processed.
- The return code sent by the message when it terminates is received in *return-code*.

When a program is started in asynchronous mode:

- The program and the application that started it run simultaneously.
- The only way of knowing that the program has terminated is via a notification – *return-code* is not relevant in this case.

4. The *err-code* variable should only be tested if STARTCONSOLEEXECEX% returns FALSE%, otherwise its value is meaningless.

The *return-code* variable is only relevant for a program started in synchronous mode that returns a value when it terminates. As a general rule, the program should return the value 0 (zero) to indicate that it terminated normally; any other values are specific to the program.

5. *mode* should contain a combination of FEXE_ *% constants. It indicates the size of the program's main window when it is displayed: normal, maximized, minimized, etc.

Warning: these options may be ignored by a program if it is not designed to accept this information when it is started up. For a program developed with NatStar (or NS-DK or NatWeb), the main window's *shell position* option must be selected for this mode to take effect.

A precise position and size can be indicated using *xpos*, *ypos*, *width* and *height* if mode is set to FEXE_USEPOSITIONSIZE%. These values must be expressed in pixels.



6. The mode FEXE_USEPOSITIONSIZE% is ignored under Windows 3.x.

Example

EXECUTED event for the 'Start' button

```
ProgID% = STARTEXECUTEEX("APP.EXE" , "", "", \
                           SELF%, 2, FALSE%, \
                           ErrCode%, RetCode%, \
                           FEEXE_NORMAL%, 0, 0, 0, 0)
```

USER2 event for the window

```
IF PARAM12% = EXEC_NFY_TERMINATE%
    MESSAGE "The program has terminated", \
           "Exit code =" && PARAM34%
ENDIF
```

See also

STOPEXECUTEEX

STARTEXCUTE2% Function

Launches the program *prog-name* with the parameters specified in *params*. The program's messages will be sent to the window identified by *win-handle* in the user-defined event identified by *msg-nbr* (0 for the USER0 event, 1 for the USER1 event, and so on.).

working-dir specifies the working directory for *prog-name*. This is equivalent to the information in the Directory field of the *Properties* box in the Windows.

Depending on the value of *synch*, the program will either be executed asynchronously (*synch* set to FALSE%) or synchronously (*synch* set to TRUE%). In the latter case, any code written after the STARTEXCUTE2% function will not be executed until the program has terminated.

If the program was launched successfully, STARTEXCUTE2% returns an integer that identifies the session started up. Otherwise, STARTEXCUTE2% returns 0. If errors occurred while starting the program, *return-code* indicates the error number that describes the problem.

Syntax	STARTEXCUTE2% (<i>prog-name</i> , <i>params</i> , <i>working-dir</i> , <i>win-handle</i> , <i>msg-nbr</i> , <i>synch</i> , <i>return-code</i>)			
Parameters	<i>prog-name</i>	CSTRING	I	program name
	<i>params</i>	CSTRING	I	parameters
	<i>working-dir</i>	CSTRING	I	working directory
	<i>win-handle</i>	INT(4)	I	handle of the window that will receive any notifications
	<i>msg-nbr</i>	INTEGER	I	ID of the user-defined event
	<i>synch</i>	INT(1)	I	synchronous/asynchronous
	<i>return-code</i>	INTEGER	I	!V! system error

Return value INTEGER

Example

EXECUTED event for the 'Start' button

```
Local INTEGER MyErrCode%
Local INT IDSession%

IDSession% = STARTEXCUTE2%("APP.EXE", "", "c:\", self%, 2, False%, MyErrCode%)
```

USER2 event for the window

```
IF PARAM1% = CON_NFY_PROGOUTPUT%  
    ...  
ELSE  
    STOPEXECUTE2 SessionID%  
ENDIF
```

STOPCONSOLEEXEC% Function

Terminates the program started in a console.

Syntax STOPCONSOLEEXEC% (*cons-handle*)

Parameters *cons-handle* INT(4) I handle of the console in which the program was executed

Return value INT(1)

value	meaning
TRUE%	the program was terminated successfully
FALSE%	an error occurred while terminating the program

Note The program must have been started by a prior call to STARTCONSOLEEXECEX%.

Example

EXECUTED event for the Start button

```
LOCAL INTEGER Errcode%
LOCAL INTEGER Retcode%
; Create a new console with a notification request
; in the USER 2 event for the current window
HD_Cons% = OPENCONSOLE%("Console title", \
                        TRUE%, SELF%, 2, \
                        FEFE_NORMAL%, \
                        0, 0, 0, 0)

IF HD_Cons% <> 0
    ; Start a program in asynchronous mode
    IF STARTCONSOLEEXECEX%(HD_Cons%, \
                          "APP.EXE", "", "" \
                          TRUE%, TRUE%, FALSE%, \
                          ErrCode%, RetCode%)
        ...
    ELSE
        MESSAGE "Cannot start program", \
                "System error =" && ErrCode%
    ENDIF
ELSE
    MESSAGE "Warning", "Error opening the console"
ENDIF
```

EXECUTED event for the Stop button

```
IF STOPCONSOLEEXEC% (HD_Cons%)
    MESSAGE "Console", "The program was terminated successfully"
ELSE
    MESSAGE "Warning", \
            "An error occurred while terminating the program"
ENDIF
```

See also STARTCONSOLEEXECEX%

STOPEXECUTEEX Instruction

Sends a termination request to a program.

Syntax `STOPEXECUTEEX program-ID`

Parameters `program-ID` INTEGER I ID of the program to terminate

Notes

1. The program must have been started by a prior call to STARTEXECUTEEX%.



2. Sends the 'close' system message to the program started in the session identified by *program-ID*. The program can then manage its own termination procedure, for example, by proposing to save any changes before closing.



The program's termination request is not sent if its main window is not active, which is often the case when the program's active window is a modal dialog box (identical behavior to the Task Manager).

A program designed with a Nat System tool will receive a CHECK event indicating whether or not the program has actually terminated (RETURN 0 or 1).

Example

EXECUTED event for the 'Start' button

```
ProgID% = STARTEXECUTEEX%("APP.EXE", "", "", \
                           SELF%, 2, FALSE%, \
                           ErrCode%, RetCode%, \
                           FEEXE_NORMAL%, 0, 0, 0, 0)
```

EXECUTED event for the 'Stop' button

```
STOPEXECUTEEX ProgID%
```

CHECK event for the program's main window

```
; Check whether the user really wants to stop the program
IF ASK2%("Warning", "Do you really want to quit APP?")= YES%
    ; Confirmed: close the application
    RETURN 0
ELSE
    ; Refused: don't close the application
    RETURN 1
ENDIF
```

See also `STARTEXECUTEEX%`

STOPEXECUTE2 Instruction



Sends the 'close' system message to the program started in the session identified by *Session-ID*. The program can then handle its own termination procedure, for example, by proposing to save any changes before closing.

This session was started by a prior call to STARTEXECUTE2%, which returned this session ID.

Syntax **STOPEXECUTE2** *Session-ID*

Parameter *Session-ID* INTEGER I ID of the session to terminate

Example

EXECUTED event for the 'Start' button

```
SessionID% = STARTEXECUTE2%("APP.EXE" \,  
                             " ", SELF%, 2, \  
                             FALSE%)
```

USER2 event for the window

```
IF PARAM1% = CON_NFY_PROGOUTPUT  
    ...  
ELSE  
    STOPEXECUTE2 SessionID%  
ENDIF
```


Chapter 4

NSMisc Miscellaneous Library

This miscellaneous library contains a large number of diverse functions and instructions, covering areas such as file handling, disk handling (directories, available space, file searches, etc.), random number generation, ASCII/EBCDIC conversion, and so on.

This chapter explains

- How to install this library
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation	4-3
NSMISC.NCL File.....	4-4
Functional categories in the NSMisc library.....	4-5
Searching for a file in a list of directories	4-10
Handling controls.....	4-11
Managing Windows 32 controls	4-11
Managing the anchoring controls	4-11
Managing traces	4-11
Handling List Boxes.....	4-11
Handling lists of images and assigning images to menu items.....	4-12
NSMisc library reference	4-15

Installation

Declare NSMISC.NCL in the libraries required to develop your applications.

Ensure that the NSxxMISC.DLL file has been placed in a PATH directory under Windows.

NSMISC.NCL File

The verbs in this library (NSMisc) are described below. They are declared in a text file, written in NCL, called NSMISC.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSMISC.NCL file:

- 1.** Navigate to the <NATSTAR>\NCL, <NSDK>\NCL or <NATWEB>\NCL directory.
 NB : <NATSTAR> means the directory in which NatStar was installed.
 NB : <NSDK> means the directory in which NS-DK was installed.
 NB : <NATWEB> means the directory in which NatWeb was installed.
- 2.** Open the NSMISC.NCL file with any text editor to see its contents.

Functional categories in the NSMisc library

Here is a list, arranged by functional category, of the functions, instructions and constants in the NSMisc miscellaneous library.

Starting up an Application

To start up another application:

EXECPROG	4-57
MODE_ *%	4-146

To obtain the start-up parameters for the current application:

PARAMCOUNT%	4-158
PARAMSTR\$	4-159
RemoveCmdLineParam	4-256
HideRemovedCmdLineParams%	4-257

DOS Environment

To obtain the environment parameters (SET commands):

ENVCOUNT%	4-55
ENVSTR\$	4-56
GETENV\$	4-116
GETLONGENV\$	4-117
PUTENV	4-118

Directory and Disk Handling

To change, create, remove directories or return the current directory:

CHDIR	4-35
GETDIR\$	4-112
MKDIR	4-145
RMDIR	4-166

To return the free size and total size of a disk, change disk, return the number of the current disk or to return the available logical volumes:

CHDISK	4-36
DISKFREE%	4-45
DISKFREE#	4-46
DISKSIZE%	4-47
DISKSIZE#.....	4-48
DT_ *%	4-51
GETDISK%	4-113
GETDISKS%	4-114
GETDISKTYPE%	4-115
GETVOLUMENAMES\$	4-129
NSSEARCHPATHS\$	4-221

Simple File Handling

To rename or delete a file:

FERASE	4-86
FRENAME	4-104

To copy a file under a new name:

FCOPYFILE	4-85
-----------------	------

To return a file's attributes and modify them:

FGETATTR%	4-93
FSETATTR.....	4-105

To return the size of a file:

FGETSIZE%	4-99
-----------------	------

To return the full pathname, directory, name and extension of a file:

FEXPAND\$	4-87
FGETDIR\$.....	4-94
FGETTEXT\$.....	4-95
FGETNAME\$.....	4-96

To search the disk for all files those match various criteria:

FFINDFIRST\$	4-88
FFINDNEXT\$.....	4-89
FFINDFIRSTEX\$	4-90
FFINDNEXTEX\$	4-91
FFINDCLOSEEX\$	4-92
FIND_ *%	4-100

Allows you to know or modify the date and the hour of a file:

FGETTIME	4-97
FSETTIME.....	4-98

Error Handling

To return the error code of the last NSMisc library function or instruction used:

MISCERROR%	4-144
------------------	-------

Handling Text Files

To open, create, append or close a text file:

T_APPEND%	4-181
T_CLOSE.....	4-182
T_CREATE%	4-183
T_OPEN%	4-187

To write (simple) and write a line to a text file:

T_WRITE.....	4-195
T_WRITEEX	4-196
T_WRITELN	4-197
T_WRITELNEX	4-198

To read (simple) an integer, real number or string:

T_READ%	4-188
T_READ#	4-189
T_READ\$	4-190

To line read an integer, real number or string

T_READLN%	4-191
T_READLN#.....	4-192
T_READLN\$.....	4-193
T_READLNEX\$	4-194

Error code, end of file, end of line:

T_EOF%	4-184
T_EOL%	4-185
T_ERROR%	4-186

Handling Binary Files

To open, create and close a binary file:

F_CLOSE	4-67
F_CREATE%	4-68
F_OPEN%	4-73
F_ROOPEN%	4-77

To read and write to a binary file in blocks of records or single records:

F_BLOCKREAD.....	4-60
F_BLOCKWRITE	4-62
F_READ	4-76
F_WRITE	4-82
F_LoadFile\$.....	4-83
F_SaveFile	4-84

Read, Write of a binary file by byte:

F_BYTEREAD.....	4-64
F_BYTEWRITE	4-66

Pointer positioning and truncation of a binary file:

F_BYTESEEK.....	4-65
F_SEEK	4-78
F_TRUNCATE.....	4-80

To lock and unlock a binary file:

F_LOCK.....	4-72
F_UNLOCK.....	4-81

Error code, end of file, size and pointer position of a binary file:

F_EOF%.....	4-70
F_ERROR%	4-71
F_POS%	4-75
F_SIZE%	4-79

Control and Window Manipulation

Constants used for the .KIND dynamic parameter, returning the control category:

DI_ *%	4-43
--------------	------

To list all controls in a window:

FIRSTCONTROL	4-102
NEXTCONTROL	4-148

To return the control name or the window name from a window-handle:

CONTROLNAME\$	4-41
WINDOWNAME\$.....	4-213

To return the handle of the window to which a control belongs:

WINDOWFROMCONTROL%	4-212
--------------------------	-------

Determine if a number is a valid handle:

ISVALIDSELFHANDLE%	4-132
--------------------------	-------

To return the control that has the focus:

FOCUSCONTROL	4-103
--------------------	-------

To return the control that will get the focus after the current control or to modify the focus sequence:

GETTABCONTROL	4-126
---------------------	-------

SETTABCONTROL.....4-176

To prevent the current control from losing the focus:

GETCHECKMODE%4-107

SETCHECKMODE4-170

To indicate which control is associated with the [Esc] and [Enter] keys respectively, and to modify these associations:

CHK_ *%4-37

GETESCAPECONTROL4-119

GETRETURNCONTROL4-125

SETESCAPECONTROL4-173

SETRETURNCONTROL4-175

Returns the handle of the window that has the mouse pointer:

GETWNDCAPTURE4-130

MDI Window Management

To return and modify menu option titles on MDI windows:

GETMDISTR\$.....4-124

MDI_ *%4-138

SETMDISTR4-174

To arrange MDI child windows in the parent window area:

MDIARRANGEWINDOWS.....4-137

MDI_ARRANGE*%4-140

MDISHOWHIDDENWINDOWS4-141

Searching for a file in a list of directories

NSSEARCHPATHS\$4-221

Handling controls

GETCONTROLTEMPLATE	4-214
GETCONTROLID%	4-215
MAKECONTROL	4-216
SELECTBYVALUE%	4-217
SETFOCUSCONTROLATTRIBUTES%	4-219

Managing Windows 32 controls

WCCSEND%	4-204
WCCSEND0%	4-205
WCCSEND1%	4-206
WCCSENDPTR%	4-207
WCCSENDSTR%	4-208

Managing the anchoring controls

NS_AS_ *%	4-252
-----------------	-------

Managing traces

SetNSGenTrace	4-246
MyTrace	4-247
NS_TRACE	4-248
SET_NSGENTRACE	4-250
GTE_ *%	4-251

Handling List Boxes

The series of functions/instructions ***EDITINPLACE** allows a temporary dynamic control to be displayed within a List Box:

ACTIVATEEDITINPLACE	4-16
BEGINEDITINPLACE	4-22
QUERYEDITINPLACE	4-160
TERMINATEEDITINPLACE	4-199

You can, for example, display Combo Boxes within List Box cells.

Temporary controls allow the enrichment of the GUI and maximization of the use of the client space of the windows.

Two new functions allow handling of List Boxes:

ADJUSTLISTBOXCOLUMNS	4-18
GETLISTBOXCELL	4-121

Handling lists of images and assigning images to menu items

Nat System has decided to publish the Window Image List programming interface (from the NSMISC service library) in order to allow images or bitmaps to be associated with menu choices.

This new interface relies on the creation, updating and handling of lists of images attached to windows called Window Image Lists or WILs. Image formats accepted are JPEG, GIF and BMP (JPEGs do not allow transparency).



The lists of images will be referred to in the rest of this document by the term WIL.

WILs are lists of bitmaps of identical size. Images of different sizes cannot exist in the same WIL.

It is possible to create several WILs and attach them to a window but it is only from the first WIL, indexed as 0, that images can be assigned to menu items. WILs with an index greater than 0 are able to serve for drawing in a PS% (Presentation Space) but not to enrich menu items with images.

Predefined images

DI_ *%	4-43
GETMENUITEMPICTURE%	4-225
SETMENUITEMPICTURE	4-226
WIL_GETCOUNT%	4-227
WIL_GETPICTURESCOUNT%	4-228
WIL_SETCOUNT	4-229
WIL_DESTROY	4-230
WIL_APPENDFROMBMP%	4-231
WIL_DELETEPICTURES	4-233

WIL_GETSIZE	4-234
WIL_DRAWPICTURE	4-235
WIL_FINDPICTURE%	4-237
WIL_FREEHASH.....	4-239
WIL_FINDORADDONEPICTURE%	4-240
SMIP_ *%	4-242

Sort arrays

NSSORT	4-209
--------------	-------

Others

To generate random numbers:

RANDOM%	4-162
RANDOM#.....	4-163
RANDOMIZE.....	4-164

To convert an ASCII code to an EBCDIC code and vice versa:

ASCII2EBCDIC\$.....	4-21
EBCDIC2ASCII\$.....	4-52

To convert a number into a character string (accepts all Microsoft Excel formats):

STRING\$	4-180
----------------	-------

To display a message box with an icon and parameterizable buttons:

MB_ *%	4-134
MESSAGE%	4-142

The default format for numbers is the one set in the Control Panel. To obtain and modify the decimal and thousands separators used by the STRING\$ function and the Entry field and CBE controls with "Number" format:

GETDECIMALSEPARATOR\$.....	4-111
GETTHOUSANDSSEPARATORS\$.....	4-127
SETDECIMALSEPARATOR	4-172

SETTHOUSANDSSEPARATOR4-177

To handle regular expressions:

COMPILEREGREXPR%4-39

EXECUTEREGREXPR%4-59

Allows you to know and modify the default behavior of bitmap controls:

GETBITMAPCONTROLSDEFAULTS%4-106

SETBITMAPCONTROLSDEFAULTS4-169

Allocation of shared memory:

SHAREDMEMALLOCERROR%4-178

To convert a buffer from one character set to another and to return the current character set:

CHS_ *%4-38

GETCURRENTCHARSET%4-110

GETHOSTCHARSET%4-120

TRANSLATECHARS4-201

TRANSLATECHARS2%4-202

TRANSLATECHS\$4-203

Allows you to know the state of the code:

GETVIRTUALKEYSTATE%4-128

Callback:

INSTALL_CALLBACK4-32

CALL_PREVCALLBACK4-33

REMOVE_CALLBACK4-34

Sort:

nsStrCmp%4-224

NSMisc library reference

ACTIVATEEDITINPLACE instruction

Makes the active editing field of the control visible.

Syntax **ACTIVATEEDITINPLACE** *wnd, x%, y%*

Parameters	<i>wnd</i>	POINTER	I	pointer returned by the function BEGINEDITINPLACE
	<i>x%</i>	INTEGER	I	coordinate X
	<i>y%</i>	INTEGER	I	coordinate Y

Comments

1. The parameters *x%* and *y%* should be greater than or equal to 0.
2. The cursor in the entry field is enabled at the X and Y coordinates.
3. The X,Y coordinates are relative to the bottom left corner of the temporary control EDITINPLACE.
4. If X or Y are equal to 0, the whole temporary control field will be selected.
5. Editing finishes when:
 - focus passes to another window (click outside the editor, [Alt]+[Tab]...),
 - the [Escape] or Enter keys are pressed
 - if the control to which the entry field is attached (or its parent window) receives a VSCROLL, HSCROLL, CHANGED or TERMINATE message.
 - in all cases, except where the [Escape] key is pressed, editing is considered as being accepted.

Example

```
local POINTER HDATA%
local ret%
local retour%
NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
    MESSAGE "ERREUR", "Handle of NULL segment"
```



```

EXIT
ENDIF
FILL HDATA%, SIZEOF SEG_EIPINFO, 0
MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
MOVE 1 TO SEG_EIPINFO(HDATA%).X ; abscissa in pixels of the editing rectangle
MOVE 1 TO SEG_EIPINFO(HDATA%).Y ; ordinate in pixels of the editing rectangle
MOVE 68 TO SEG_EIPINFO(HDATA%).W ; width of the editing rectangle
MOVE 25 TO SEG_EIPINFO(HDATA%).H ; height of the editing rectangle
;The temporary control is an EditComboBox DI_COMBBOXE%
MOVE DI_COMBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text
ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)
move ret% to EF
IF ret%=0
    MESSAGE "ERREUR","Traitement terminé"
ELSE
    ; once the edit in place has been called you can access the temporary Ctrl
    INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL
    ;displays the temporary Control
    ACTIVATEDITINPLACE ret%,1,2
ENDIF
DISPOSE HDATA%

; The event User0 of the control EF_TEST
Insert at END "User0 of EF_TEST " to LISTBOX1
IF param12% <> -1
    IF param12% = 0
        Insert at END " Escape pressed" to LISTBOX1
    ELSE
        Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
        Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
    ENDIF
ENDIF
ENDIF

```

See also

BEGINEDITINPLACE

ADJUSTLISTBOXCOLUMNS instruction

This instruction allows the space between the columns of the List Box to be defined.

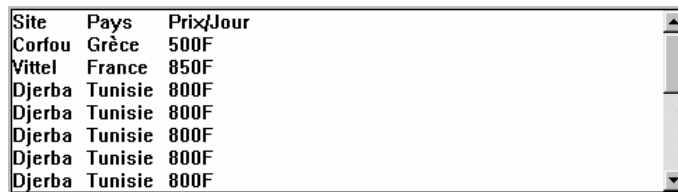
Syntax	ADJUSTLISTBOXCOLUMNS			<i>lb, line%, nbl%, col%, nbc%, widthinc%</i>
Parameters	<i>lb</i>	CONTROL	I	name of the List Box
	<i>line%</i>	INTEGER	I	first line to be taken into account for determining the adjustment
	<i>nbl%</i>	INTEGER	I	number of lines to adjust
	<i>col%</i>	INTEGER	I	first column
	<i>nbc%</i>	INTEGER	I	number of columns to adjust
	<i>widthinc%</i>	INTEGER	I	number of pixels between the columns

Example

```
ADJUSTLISTBOXCOLUMNS LB1, 0, LineCount%(LB1), 1, 3, 100
;Adjusts the width of columns 1 to 3 of the List Box LB1 taking account of
;the text in these columns for all the lines of the List Box and adding 100
;pixels (space between the columns)
```

Site	Pays	Prix/Jour
Corfou	Grèce	500F
Vittel	France	850F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F

```
ADJUSTLISTBOXCOLUMNS LB1, 0, LineCount%(LB1), 1, 3, 10  
;Adjusts the width of columns 1 to 3 of the List Box LB1 taking account of  
;the text in these columns for all the lines of the List Box and adding 10  
;pixels (space between the columns)
```



A screenshot of a list box containing a table of travel data. The table has three columns: 'Site', 'Pays', and 'Prix/Jour'. The data is as follows:

Site	Pays	Prix/Jour
Corfou	Grèce	500F
Vittel	France	850F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F
Djerba	Tunisie	800F

ADJUST_ *% Constant

ADJUST_ *% constants are used jointly with the new segment SEG_SIZEORPOS and the ADJUSTSIZEORPOS event. This event informs a window that a user wants to modify its position or size and enables it change position and size values.

ADJUST_MOVE% indicates user intention to modify window position

ADJUST_SIZE% indicates user intention to modify window size

Syntax **ADJUST_MOVE %**

ADJUST_SIZE%

Note

1. The constants are declared internally as follows:

```
CONST ADJUST_MOVE%       8
```

```
CONST ADJUST_SIZE%      16
```

See also SEG_SIZEORPOS

ASCII2EBCDIC\$ Function

Converts from ASCII to EBCDIC and returns the EBCDIC equivalent of an ASCII string.

Syntax **ASCII2EBCDIC\$** (*ASCII-string*)

Parameters *ASCII-string* CSTRING I ASCII string to convert

Return value CSTRING

See also EBCDIC2ASCII\$
NCL language: ASC%, CHR\$

BEGINEDITINPLACE function

Prepares the editing of a value in an entry control within a standard control (in particular a List Box).

The EDITINPLACE controls are of three types: Combo Box, Combo Box Edit and Entry Field. They allow the enrichment of the GUI and maximization of the use of the client space of the windows.

The first function to call to create an EDITINPLACE control is the function BEGINEDITINPLACE which does the work of preparation, initializes the editing field and returns a handle for it. This handle will subsequently be used by the other *EDITINPLACE functions.

Syntax	BEGINEDITINPLACE	<i>(info, usermsg%)</i>		
Parameters	<i>info</i>	SEGMENT	I/O	segment SEG_EIPINFO
	<i>usermsg%</i>	INTEGER	I	number of USERx message

Value returned

POINTER to a temporary EDITINPLACE control.

Comments

1. The segment SEG_EIPINFO defined in the file NSMISC.NCL:

```

SEGMENT SEG_EIPINFO ; Edit in Place properties
    INTEGER VER      ; Version = 0
    CONTROL CTRL     ; Contains the address of the parent control
    INTEGER X        ; abscissa in pixels of the editing rectangle (in relation to
                      ; the bottom left corner of CTRL)
    INTEGER Y        ; ordinate in pixels of the editing rectangle (in relation to
                      ; the bottom left corner of CTRL)
    INTEGER W        ; Width in pixels of the EDITINPLACE control
    INTEGER H        ; Height in pixels of the EDITINPLACE control
    INTEGER KIND     ; Type of EDITINPLACE control(DI_ENTRYFIELD%, DI_COMBBOX%,
                      ; DI_COMBBOXE%)
    CSTRING TEXT     ; Text to be assigned to the EDITINPLACE (ENTRYFIELD,
                      ; COMBBOXE)
ENDSEGMENT ; SEG_EIPINFO
;CTRL contains the address of the control in which you want to insert a
;temporary control KIND the type of temporary control you want to display
;use the constant DI_*
;TEXT the initialization text of the temporary control

```

2. The only permissible types of temporary controls in the variable `KIND` of the segment `SEG_EIPINFO` are Entry-field, Combo-box and Combo-box edit (ie the following constants respectively: `DI_ENTRYFIELD%`, `DI_COMBBOX%` and `DI_COMBBOXE%`).
3. The parameter *info* should be initialized with the required values before calling this function.
4. The parameter *usermsg%* indicates the number of a `USERx` message to notify the container control that editing has finished.
5. If editing is finished by pressing the [Escape] key, the event `USERx` is sent with `PARAM12%` at 0, otherwise either `PARAM1$` contains the changed text, or `PARAM12%` is the number of the item selected if the temporary control is an editable Combo Box.
6. If the temporary control is an Entry Field or an editable Combo Box, it is essential to check that `PARAM12%` is different from -1 before reading `PARAM1$`.
7. The function returns a pointer to be used as the parameter *wnd* of the other `*EDITINPLACE` functions/instructions, except if its value is 0 (error preventing the continuation of processing).
8. On the return of the call to `BEGINEDITINPLACE`, the variable `CTRL` of the segment `SEG_EIPINFO` is replaced by the editing temporary control (of the type indicated in `Info.Kind`), which allows its initialization (`INSERT` if it is a Combo Box, editable or not, dynamic parameters `CASE`, `CHARACTERS`, `FILLTEXT`, `FORMAT`, `JUSTIFICATION`, `MAXLEN`, `NOBLANKS`, `REQUIRED`, `SKIPBLANKS` and/or `TYPE` for an Entry Field or an editable Combo Box).

Example

```
local POINTER HDATA%
local ret%
local retour%

NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
    MESSAGE "ERREUR", "Handle of NULL segment"
    EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_EIPINFO, 0

MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
```

```

MOVE 1 TO SEG_EIPINFO(HDATA%).X ; abscissa in pixels of the editing rectangle
MOVE 1 TO SEG_EIPINFO(HDATA%).Y ; ordinate in pixels of the editing rectangle
MOVE 68 TO SEG_EIPINFO(HDATA%).W ; width of the editing rectangle
MOVE 25 TO SEG_EIPINFO(HDATA%).H ; height of the editing rectangle

;The temporary control is an EditComboBox DI_COMBBBOXE%
MOVE DI_COMBBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text

ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)

move ret% to EF
IF ret%=0
    MESSAGE "ERROR","Process is over"
ELSE
    ; once the edit in place has been called you can access the temporary Ctrl
    INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL
    ;displays the temporary Control
    ACTIVATEEDITINPLACE ret%,1,2
ENDIF
DISPOSE HDATA%

; The event User0 of the control EF_TEST
Insert at END "User0 of EF_TEST " to LISTBOX1
IF param12% <> -1
    IF param12% = 0
        Insert at END " Escape pressed" to LISTBOX1
    ELSE
        Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
        Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
    ENDIF
ENDIF
ENDIF

```

See also GETLISTBOXCELL

CHANGEWINDOWSTYLE instruction

Changes in a dynamically manner attributes and styles of windows. *RemoveStyles%* parameter is a combination of styles to retrieve from the window. *AddStyles%* parameter is a combination of styles to add to the window.

Syntax **CHANGEWINDOWSTYLE** *Win, RemoveStyles%, AddStyles%*

Parameters	<i>win</i>	POINTER	I	handle of the window that we want to modify the style attributes
	<i>RemoveStyles%</i>	INT(4)	I	constant combination WS_%% of styles to be removed from window
	<i>AddStyles%</i>	INT(4)	I	constant combination WS_%% of styles to be added to window.

Examples

```
; Removes the title bar of the window and adds a vertical and a horizontal
;scroll-bar
ChangeWindowState hWnd%, WS_TITLE%, WS_HORZSCROLLBAR% \
    bor WS_VERTSCROLLBAR%
; Switches the border on resizable
ChangeWindowState hWnd%, WS_BORDER%, WS_SIZEBORDER%
```

Note

- 1.The WS_TITLE% parameter is not taken in account.

See also WS_%%

WS_ *% constant

WS_ *% constant are used jointly with CHANGEWINDOWSTYLE instruction to dynamically change window style.

Syntax

- WS_TITLE%**
- WS_BORDER%**
- WS_SIZEBORDER%**
- WS_DLGBORDER%**
- WS_VERTSCROLLBAR%**
- WS_HORZSCROLLBAR%**
- WS_NOBYTEALIGN%**
- WS_QUIT%**
- WS_MINIMIZE%**
- WS_SYSMENU%**
- WS_MINMAX%**

Notes

1.The definition of the constants are :

WS_TITLE%	Shows or hide the title bar.
WS_BORDER%	Positions fine border.
WS_SIZEBORDER	Positions modifyable border.
WS_DLGBORDER%	Positions Dialog box border
WS_VERTSCROLLBAR%	Shows or hide the vertical bar.
WS_HORZSCROLLBAR%	Shows or hide the horizontal bar.
WS_NOBYTEALIGN%	Must be present so that the horizontal position is not a multiple of 8 pixels (constant to be used only in addition, you cannot remove it).
WS_QUIT%	Positions or removes « Quit on Close » option
WS_MINIMIZE%	Shows the "Minimize" key

WS_SYSMENU%	Shows or hide system menu.
WS_MINMAX%	Shows or hide « Minimize/Maximize » keys

2.The other constants are not taken in consideration : WS_OVER%, WS_BELOW%, WS_DEFAULTICON%, WS_SHELLPOSITION%, WS_SIZEREDRAW%, WS_TASKLIST%, WS_SAVEBITS%, WS_MENUBAR%, WS_SECONDARY%, WS_MDI%.

3.Internal declaration is:

CONST WS_TITLE%	\$00000001
CONST WS_BORDER%	\$00000002
CONST WS_SIZEBORDER%	\$00000004
CONST WS_DLGBORDER%	\$00000008
CONST WS_VERTSCROLLBAR%	\$00000010
CONST WS_HORZSCROLLBAR%	\$00000020
CONST WS_NOBYTEALIGN%	\$00002000
CONST WS_QUIT%	\$00008000
CONST WS_MINIMIZE%	\$00010000
CONST WS_SYSMENU%	\$04000000
CONST WS_MINMAX%	\$08000000

See also CHANGEWINDOWSTYLE

GETDESKTOPWIDTH% Function

This function returns the usable screen width, taking into account the tool bars

Syntax **GETDESKTOPWIDTH%**

Return value INT(4) usable screen width expressed in pixels

See also SCREENWIDTH%, GETDESKTOPHEIGHT%, GETDESKTOPX%,
GETDESKTOPY%

GETDESKTOPHEIGHT% Function

Returns the usable screen height, taking into account the tool bars.

Syntax **GETDESKTOPHEIGHT%**

Return value INT(4) usable screen height expressed in pixels

See also SCREENHEIGHT%, GETDESKTOPWIDTH%, GETDESKTOPX%,
GETDESKTOPY%

GETDESKTOPX% Function

Returns the first usable horizontal screen position, taking into account the tool bars. Position 0 is found at the extreme left of the screen.

Syntax **GETDESKTOPX%**

Return value INT(4) first usable horizontal screen position expressed in pixels

See also GETDESKTOPWIDTH%, GETDESKTOPHEIGHT%, GETDESKTOPY%

GETDESKTOPY% Function

Returns the first usable vertical screen position, taking tool bars into account. Position 0 is situated at the bottom of the screen.

Syntax **GETDESKTOPY%**

Return value INT(4) first usable vertical screen position expressed in pixels

See also GETDESKTOPWIDTH%, GETDESKTOPHEIGHT%, GETDESKTOPX%

INSTALL_CALLBACK function

Returns a handle (*hCallback*) that must be used by CALL_PREVCALLBACK and REMOVE_CALLBACK instructions.

Syntax **INSTALL_CALLBACK** (*pfnCallback*)

Parameter *pfnCallback* POINTER I pointer to a function

Return value POINTER

Notes

1. The function of which the pointer is passed in parameter must be prototyped in the following way. The name of the function can be different :

```
FUNCTION MyCallback(POINTER mySelf%, INTEGER Id%, INTEGER Msg%, POINTER  
Parm1, POINTER Parm2) RETURN INTEGER
```

MyCallback function is then called for each event concerning a window or a control of the application, mySelf% and Id% indicate the window or the control concerned, Msg% the event, Parm1 and Parm2 the associated parameters.

2. Each using of MyCallback function must call once and only once CALL_PREVCALLBACK.

```
i% = CALL_PREVCALLBACK(hCallback, mySelf%, Id%, Msg%, Parm1, Parm2)  
;... additional processing  
RETURN i%
```

or

```
RETURN CALL_PREVCALLBACK(hCallback, mySelf%, Id%, Msg%, Parm1, Parm2)
```

See also CALL_PREVCALLBACK , REMOVE_CALLBACK

CALL_PREV CALLBACK function

The CALL_PREV CALLBACK function passed the control to another callback added by INSTALL_CALLBACK or to the standard processing of events.

Syntax	CALL_PREV CALLBACK (<i>hCallBack</i> , <i>win</i> , <i>id</i> , <i>msg</i> , <i>parm1</i> , <i>parm2</i>)			
Parameters	<i>hCallBack</i>	POINTER	I	Handle sent by INSTALL_CALLBACK
	<i>win</i>	POINTER	I	Window handle
	<i>id</i>	INTEGER	I	Id of the window or the control
	<i>msg</i>	INTEGER	I	Event
	<i>parm1</i>	POINTER	I	Parameter associated to the event
	<i>parm2</i>	POINTER	I	Parameter associated to the event
Return value	INTEGER			
See also	INSTALL_CALLBACK, REMOVE_CALLBACK			

REMOVE_CALLBACK instruction

Remove the callback added by `INSTALL_CALLBACK`.

Syntaxe **REMOVE_CALLBACK** *hCallBack*

Paramètre *hCallBack* `POINTER` `I` handle sent by `INSTALL_CALLBACK`

Voir aussi `INSTALL_CALLBACK`, `CALL_PREVCALLBACK`

CHDIR Instruction

Changes the current directory.

Syntax **CHDIR** *directory-name*

Parameters *directory-name* CSTRING I name of the new current directory

Note The CHDIR instruction is equivalent to the DOS or CHDIR commands.

Example

```
CHDIR " .. "        ; changes to the parent directory

CHDIR " TEST "      ; changes to the TEST directory located
; under the current directory
```

See also CHDISK, MKDIR, RMDIR, GETDIR\$

CHDISK Instruction

Changes the current logical disk.

Syntax **CHDISK** *disk-number*

Parameters *disk-number* INT(1) I number of the new current disk

Note The disk-number integer passed to CHDISK must lie between 0 and 26, where:

Value	Stands for the following disk
0	current disk
1	A :
2	B :
3	C :
4 ... 25	D : ... Y :
26	Z :

Example

```
CHDISK 5 ; E becomes the current disk
IF MISCERROR% <> 0
MESSAGE " ERROR ", "The requested logical disk does not exist! "
ENDIF
```

See also GETDISK%, GETDISKS%

CHK_*% Constants

Values used to indicate the reason for the CHECK event.

Syntax **CHK_LOSEFOCUS%**
 CHK_ENDSESSION%
 CHK_PUSHBUTTON%

Notes

1. These values are returned in PARAM1% of the CHECK event.
2. These constants have the following meaning:
CHK_LOSEFOCUS% the current control has lost the focus.
CHK_ENDSESSION% the Windows session has ended.
CHK_PUSHBUTTON% a Push button has been pressed.
3. CHK_ENDSESSION% can only be returned under Windows.
4. They are declared internally as followed:
CONST CHK_LOSEFOCUS% 0
CONST CHK_ENDSESSION% 1
CONST CHK_PUSHBUTTON% 2

See also SETCHECKMODE
 NCL language: CHECK event.

CHS_ *% Constants

Constants used to identify character sets.

Syntax **CHS_ASCII%**
 CHS_ASCIIIMAC%
 CHS_ANSI%
 CHS_EBCDIC%

Notes

1. These constants have the following meaning:
 CHS_ASCII% ASCII character set (PC)
 CHS_ASCIIIMAC% ASCII character set (Macintosh)
 CHS_ANSI% ANSI character set
 CHS_EBCDIC% EBCDIC character set

2. They are declared internally as follows:

```
CONST CHS_ASCII%     0
CONST CHS_ANSI%       1
CONST CHS_ASCIIIMAC% 2
CONST CHS_EBCDIC%    3
```

See also GETHOSTCHARSET%, GETCURRENTCHARSET%, TRANSLATECHARS

COMPILEGREXPR% Function

Validates an expression mask, compiles it and stores it in a buffer in condensed form.

Syntax **COMPILEGREXPR%** (*expression-mask, buffer-address, buffer-size*)

Parameters	<i>expression-mask</i>	CSTRING	I	expression to check
	<i>buffer-address</i>	POINTER	I	storage buffer address
	<i>buffer-size</i>	INTEGER	I	buffer size

Return value INT(1)

value	meaning
0	No errors.
1	Buffer size exceeded
2	Syntax error detected

Notes

1. The expression mask is applied to each individual character in the string.
2. Syntax for describing regular expressions:
 - [-] **designates a set of authorized characters.**
 - e.g.:
 - * [A-Z] allows upper case A - Z.
 - * [a-k] allows lower case a-k
 - * [A-JS-Ws-w] allows upper case A - J and S- W as well as lower case s-w.
 - When a letter is specified on its own, this means that only that letter is allowed.**
 - e.g.
 - * Mrs only allows " Mrs " to be input
 - * whereas, [Mm]r allows Mr and mr.
 - ^ allows all characters EXCEPT those that follow the ^.**
 - e.g.
 - * [^A-Z0] allows everything except A - Z and zero.
 - * [^EJMP] allows everything except E J M and P.
 - % allows 0 - n occurrences of any characters.**
 - e.g.
 - * %Mr% allows any sentence containing " Mr ", hence " Dear Mr Klein " is correct.

_ allows any character.

e.g.

00__99 allows any 6-letter string starting with 00 and ending in 99 (002499, 00az99, etc.).

***** indicates that the preceding character can be repeated 0 - N times.

e.g.

* [A-J0-3]* allows 0 - n characters between A and J and 0 and 3.

* [^EL246]* allows 0 - n characters other than E,L,2,4 or 6.

**** is used as an escape character.

this character should be used whenever a character that is part of the regular expression syntax is to be interpreted as a 'normal' character.

e.g.

C:\\[A-Z]* allows string beginning with " C:\ ", followed by an upper case letter and ending in " * ".

Example

```
SEGMENT BUFFER
CHAR DATA[300]
ENDSEGMENT
LOCAL BUF%
NEW BUFFER,BUF%
IF COMPILEREGREXPR%(' %[Mm]r%',BUF%,300)=0
    ...
IF EXECUTEREGREXPR%('Dear Mr Klein',BUF%)
...;process valid string
ELSE
...;process invalid string
ENDIF
ELSE
...;error processing
ENDIF
DISPOSE BUF%
```

See also EXECUTEREGREXPR%

CONTROLNAME\$ Function

Returns the name of a control.

Syntax **CONTROLNAME\$** (*control*)

Parameters *control* CONTROL I control

Return value CSTRING

Notes

1. The name of a control corresponds to the contents of the " Name " field in its Info box.
2. **IMPORTANT NOTE.** For CONTROLNAME\$ to work in generated mode, the /NAMES option must be used with the generation module or the " Symbols info " option must be checked in the Generation box.

Example 1

```
LOCAL CONTROL CTRL
FIRSTCONTROL SELF%, CTRL
MESSAGE " Name of the first control: ", CONTROLNAME$(CTRL)
```

Example 2

```
LOCAL CONTROL CTRL
MOVE ENTRY00001 TO @CTRL
MESSAGE " Name of the control ", CONTROLNAME$(CTRL)
; displays ENTRY00001
```

See also WINDOWFROMCONTROL%, WINDOWNAME\$

CURRENCY\$ Function

Returns a string containing the number formatted according to the currency definitions set in the Control Panel.

Syntax **CURRENCY\$** (*number*)

Parameters *number* NUM(8) I number to format

Return value CSTRING

Notes

1. The format applied is the one set in the Control Panel, which may be modified with the separators specified by SETDECIMALSEPARATOR and SETTHOUSANDSSEPARATOR (if these instructions have been called).
2. Use the ECURRENCY\$ function if the type of the real number you want to format is NUM(10).

Example

```
; If " $ " is the currency symbol defined as a suffix in the
; Control Panel:
SETDECIMALSEPARATOR " . "
SETTHOUSANDSSEPARATOR " , "
MOVE CURRENCY$(2371.4) TO S$ ; S$ contains " 2,371.40 $ "
```

See also ECURRENCY\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, STRING\$

DI_*% Constants

Values used to indicate a control's category.

Syntax	DI_BITMAP%
	DI_CHECKBOX%
	DI_CHECKBOX3%
	DI_COMBBOX%
	DI_COMBBOXE%
	DI_CONTROL%
	DI_CUSTOM%
	DI_ENTRYFIELD%
	DI_GROUPBOX%
	DI_HSCROLL%
	DI_ICON%
	DI_LISTBOX%
	DI_MENUITEM%
	DI_MLE%
	DI_PUSHBUTTON%
	DI_RADIOBUTTON%
	DI_STATICTEXT%
	DI_VSCROLL%

Notes

1. These values are returned when a control's .KIND dynamic parameter is read.
2. DI_CHECKBOX3% is a Check box control with " 3 States " checked in its Info box.
3. DI_CONTROL% identifies a Control Template and DI_CUSTOM% identifies a Custom Control.
4. They are declared internally as follows:
CONST DI_STATICTEXT% 1
CONST DI_GROUPBOX% 2
CONST DI_PUSHBUTTON% 3
CONST DI_RADIOBUTTON% 4
CONST DI_CHECKBOX% 5
CONST DI_CHECKBOX3% 6

CONST DI_ENTRYFIELD%	7
CONST DI_VSCROLL%	8
CONST DI_HSCROLL%	9
CONST DI_LISTBOX%	10
CONST DI_COMBBOX%	11
CONST DI_COMBBOXE%	12
CONST DI_ICON%	13
CONST DI_MLE%	14
CONST DI_BITMAP%	15
CONST DI_CONTROL%	16
CONST DI_MENUITEM%	23
CONST DI_CUSTOM%	24

Example 1

```
GLOBAL CONTROL CTRL
...
IF CTRL.KIND = DI_ENTRYFIELD%
MESSAGE " The control CTRL ", " is an Entry field "
ENDIF
```

Example 2

```
; A control's category cannot be changed
MOVE DI_PUSHBUTTON% TO ENTRY00001.KIND ; no effect!
```

See also .KIND dynamic parameter (NCL language)

DISKFREE% Function

Return the free size in bytes of a disk.

Syntax **DISKFREE%** (*disk-number*)

Parameters *disk-number* INT(1) I disk number

Return value INT(4)

Notes

1. The function returns -1 if the disk does not exist or is inaccessible.
2. *disk-number* must be an integer between 0 and 26, where:

Value	Stands for the following disk:
-------	--------------------------------

0	current disk
---	--------------

1	A:
---	----

2	B:
---	----

3	C:
---	----

4 ... 25	D: ... Y:
----------	-----------

26	Z:
----	----

3. The size of the disk must be limited to 2 GB.

Example

```
IF DISKFREE%(0) > SIZEMYLIST%  
SAVE MYLIST TO " MYLIST.TXT "  
ELSE  
MESSAGE " Warning! ", " Insufficient disk space "  
ENDIF
```

See also DISKSIZE%, FGETSIZE%, GETDIR\$, GETDISK%

DISKFREE# Function

Return the free size for a specific drive as a real.

Syntax **DISKFREE#** (*disk-number*)

Parameter *disk-number* INT(1) I disk number

Return value REAL

Notes

1. The function returns -1 if the disk does not exist or is inaccessible.
2. *disk-number* must be an integer between 0 and 26, where:

Value	Stands for the following disk:
0	current disk
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:
3. The size of the disk is not limited to 2 GB unlike DISKSIZE# function.

See also DISKFREE%, DISKSIZE%, DISKSIZE#, FGGETSIZE%, GETDIR\$, GETDISK%

DISKSIZE% Function

Return the total size in bytes of a disk.

Syntax **DISKSIZE%** (*disk-number*)

Parameters *disk-number* INT(1) I disk number

Return value INT(4)

Notes

1. The function returns -1 if the disk does not exist or is inaccessible.
2. *disk-number* must be an integer between 0 and 26, where:

Value	Stands for the following disk:
0	current disk
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:
3. The size of the disk must be limited to 2 GB.

Example

```
MESSAGE " Disk C's capacity: ", DISKSIZE%(3)
```

See also DISKFREE%, FGETSIZE%, GETDIR\$, GETDISK%

DISKSIZE# Function

Return the total size for a specific drive as a real.

Syntax **DISKSIZE#** (*disk-number*)

Parameter *disk-number* INT(1) I disk number

Return value REAL

Notes

1. The function returns -1 if the disk does not exist or is inaccessible.
2. *disk-number* must be an integer between 0 and 26, where:

Value	Stands for the following disk:
0	current disk
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z:
3. The size of the disk is not limited to 2 GB unlike DISKSIZE% function.

Example

```
MESSAGE " Disk C's capacity: ", DISKSIZE#(3)
```

See also DISKSIZE%, DISKFREE%, DISKFREE#, FGETSIZE%, GETDIR\$,
GETDISK%

DLG_CHECKCONTROLS function

This function enables to check all controls of a window

Syntax **DLG_CHECKCONTROLS** (*dlg*)

Parameter *dlg* POINTER I Handle of the window that we want to check all controls

Returned value INT(4) Different of 0 if the control is not focus, FALSE% if it can't.

Example

```
if DLG_CHECKCONTROLS(self%) <> 0
    ...
endif
```

Note

1. Calling the function, a check is made on all control but the CHECK event is not called upon.

DLG_CHECKCONTROLFOCUS% Function

Check that the specified Custom Control can take the focus.

Syntax **DLG_CHECKCONTROLFOCUS%** (*ctrl*)

Parameter *ctrl* CONTROL I name of the Custom Control

Return value INT(2)

Note

1. Returns TRUE% if all is OK, FALSE% if the check failed.

See also SETFOCUS, DLG_CHECKCONTROLS

DT_ *% Constants

The VK_APPS% constant is used in the CHARACTER event to support the Applications key on keyboards with Windows keys.

The DT_ *% constants are used in the GETDISKTYPE function to specify the type of disk.

Syntax

DT_REMOVABLE%
DT_FIXED%
DT_REMOTE%
DT_CDROM%
DT_RAMDISK%

Notes

1. The definition of the constants :

DT_REMOVEABLE %	Removable disk or diskette
DT_FIXED%	Hard disk drive
DT_REMOTE%	Remote or network drive
DT_CDROM%	CD-ROM
DT_RAMDISK %	RAM Disk or memory

2. They are declared internally as follows:

```
const DT_REMOVABLE% 2
const DT_FIXED%      3
const DT_REMOTE%     4
const DT_CDROM%      5
const DT_RAMDISK%    6
```

See also GETDISKTYPE

EBCDIC2ASCII\$ Function

Converts from EBCDIC to ASCII and returns the ASCII equivalent of the converted EBCDIC string.

Syntax **EBCDIC2ASCII\$** (*EBCDIC-string*)

Parameters *EBCDIC-string* CSTRING I EBCDIC string to convert

Return value CSTRING

See also ASCII2EBCDIC\$
NCL language: ASC%, CHR\$

ECURRENCY\$ Function

Returns a string containing the number formatted according to the currency definitions set in the Control Panel.

Syntax **ECURRENCY\$** (*number*)

Parameter *number* NUM(10) I number to format

Return value CSTRING

Note See notes on the CURRENCY\$ function. ECURRENCY\$ is identical to this function except that it formats NUM(10) real numbers.

See also CURRENCY\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, STRING\$

ESTRING\$ Function

Formats a real number and returns the resulting string.

Syntax **ESTRING\$** (*real-number*, *string-format*)

Parameters	<i>real-number</i>	NUM(10)	I	real number to format
	<i>format-string</i>	CSTRING	I	format to apply

Return value CSTRING

Note See notes on the STRING\$ function. ESTRING\$ is identical to this function except that it formats NUM(10) real numbers.

See also STRING\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, CURRENCY\$, DATE\$, TIME\$ (NSDATE library)

ENVCOUNT% Function

Returns the number of environment parameters declared in the DOS session.

Syntax **ENVCOUNT%**

Return value INTEGER

Note An environment parameter is declared through the DOS SET command.

Example

```
; Display all environment parameters
MOVE 1 TO I%
WHILE I% <= ENVCOUNT%
    MESSAGE " Parameter " && I%, ENVSTR$(I%)
    MOVE I%+1 TO I%
ENDWHILE
```

See also ENVSTR\$, GETLONGENV\$, GETENV\$, PUTENV

ENVSTR\$ Function

Returns the environment parameter whose index is passed as a parameter.

Syntax **ENVSTR\$(index)**

Parameters *index* INTEGER I index of the environment parameter

Return value CSTRING

Notes

1. The index of the first environment parameter is 1.
2. The total number of environment variables declared is returned by the ENVCOUNT% function.

Example

```
; Display all environment parameters
MOVE 1 TO I%
WHILE I% <= ENVCOUNT%
    MESSAGE " Parameter " && I%, ENVSTR$(I%)
    MOVE I%+1 TO I%
ENDWHILE
```

See also ENVCOUNT%, GETLONGENV\$, GETENV\$, PUTENV

EXECPROG Instruction

Starts up an application.

Syntax **EXECPROG** *exe-filename, param-string, mode, xpos, ypos, width, height*

Parameters	<i>exe-filename</i>	CSTRING	I	name of the executable file to run
	<i>param-string</i>	CSTRING	I	parameters to add to the command line
	<i>mode</i>	INTEGER	I	application start-up mode
	<i>xpos</i>	INTEGER	I	x position of the application's main window
	<i>ypos</i>	INTEGER	I	y position of the application's main window
	<i>width</i>	INTEGER	I	width of the application's main window
	<i>height</i>	INTEGER	I	height of the application's main window

Notes

1. *mode* must be a combination of `MODE_*` constants and is used to specify the way in which the application is to be started: normal, maximized, minimized, etc.
2. EXECPROG allows you to start a .EXE generated by NatStar (or NS-DK) with `MODE_MINIMIZE%` or `MODE_MAXIMIZE%`. For this, the Main Window should be defined with " Shell Position & Size " and a Size Border. It does not allow you to start up a .EXE generated with `MODE_INVISIBLE%`.
3. Test `MISCERROR%` to find out whether the application was started up successfully. See Appendix B of this manual for details on the error codes that can be returned.



4. *xpos, ypos, width* and *height* are not used under Windows.
5. The parent application that performs the EXECPROG and the child application started up by EXECPROG run simultaneously, owing to Windows environment. The executables are independent of one another: the child application is not stopped when the parent application terminates.
6. The programs started up by EXECPROG can be either Windows or DOS executables or .COM, .BAT or .PIF files. Parameters can only be passed to .EXE executable files.

All programs other than Windows executables are run according to the information provided in the corresponding .PIF file or the default .PIF file. Similarly, program termination (e.g. closure of the DOS window in which the program was started up) must also be specified in the corresponding .PIF file.

Example 1

```

; APP.EXE is a graphical mode application designed
; using NatStar
EXECPROG " C:\APP\APP.EXE ", " ", MODE_NORMAL%, 0, 0, 0, 0
IF MISCERROR% = 0
MESSAGE " OK ", " Application APP.EXE started up! "
ELSE
MESSAGE " Error " && MISCERROR%, \
" Application APP.EXE not started up "
ENDIF

```

Example 2



```

; Starting up NOTEPAD minimized
EXECPROG " C:\WINDOWS\notepad.exe ", " ", \
MODE_MINIMIZED%, 0, 0, 0, 0

```

Example 3



```

; Starting up a .CMD command file
; (a character application such as CMD.EXE run with
; MODE_VIO_SCREEN% is executed and is visible)
; Warning. Using /K instead of /C leaves the CMD.EXE
; session open after execution of BATCH.CMD.
EXECPROG " C:\OS2\CMD.EXE ", " /K BATCH.CMD ", \
MODE_VIO_SCREEN%, 0, 0, 0, 0

```

Example 4

```

; Starting up an editor that runs under DOS
EXECPROG " C:\EDIT.EXE ", " C:\FIC\INFO.TXT ", \
MODE_NORMAL, 0, 0, 0, 0

```

See also

MODE_*, PARAMCOUNT%, PARAMSTR\$, STARTEXECUTE2%,
STARTCONSOLEEXEC% (NSFEXE library)

EXECUTEREGREXPR% Function

Tests whether a character string conforms with a regular expression.

Syntax EXECUTEREGREXPR%(*string*, *buffer-handle*)

Parameters	<i>string</i>	CSTRING	I	string to test
	<i>buffer-handle</i>	POINTER	I	address of the buffer that stores the compiled regular expression

Return value INT(1)

value	meaning
TRUE%	the string is valid
FALSE%	the string is not valid

Note The regular expression must previously have been compiled with COMPILEREGREXPR% and stored in the buffer specified by *buffer-handle*.

Example

```
SEGMENT BUFFER
CHAR DATA[300]
ENDSEGMENT
LOCAL BUF%
NEW BUFFER, BUF%
IF COMPILEREGREXPR%('%[Mm]r%', BUF%, 300)=0
    ...
IF EXECUTEREGREXPR%('Dear Mr Klein', BUF%)
    ...;process valid string
ELSE
    ...;process invalid string
ENDIF
ELSE
    ...;error processing
ENDIF
DISPOSE BUF%
```

See also COMPILEREGREXPR%

F_BLOCKREAD Instruction

Reads several records (or blocks) at a time from a binary file.

Syntax **F_BLOCKREAD** *file-handle, buffer-address, nbr-blocks-buffer, nbr-blocks-read*

Parameters	<i>file-handle</i>	INT(4) I	handle of the file to read
	<i>buffer-address</i>	INT(4) I	address of the buffer used to copy the records read
	<i>nbr-blocks-buffer</i>	INT(2) I	number of records to read
	<i>nbr-blocks-read</i>	INT(2) I/O	number of records actually read

Notes

1. Records are read starting from the current pointer position. The pointer is then positioned after the last record actually read. Any records read are copied into the buffer (or memory area) specified by *buffer-address*.
2. The size of the storage area must be greater than or equal to the size of the segment specified by F_OPEN% or F_CREATE%.
3. The number of records read must always be less than or equal to the number of records that can appear in the buffer.
4. *nbr-blocks-read* is updated by the instruction once the read and copy operations are complete. This variable contains the number of records actually read from the file.

Example

```
; Declare the segment used
; INIT event for the test window
SEGMENT INDIVIDUAL
STRING LAST_NAME(31)
STRING FIRST_NAME(31)
ENDSEGMENT
; EXECUTED event for the write test button
LOCAL H%, DATA%
; Define a buffer for 100 records
LOCAL INDIVIDUAL BUFF[100]
LOCAL INT NB_BLOCKS_BUFF%(2)
LOCAL INT NB_BLOCKS%(2)
; Opens the binary file " CLIENTS.CLI "
MOVE F_OPEN%(SIZEOF INDIVIDUAL, " CLIENTS.CLI ") TO H%
IF F_ERROR%<>0
MESSAGE " Error ", " Cannot open file "
```

```
EXIT
ENDIF
; Read the first two blocks
F_BLOCKREAD H%,@BUFF, 2, NB_BLOCKS%
IF F_ERROR% <> 0
MESSAGE " Error ", \
" Cannot read first two blocks "
ELSE
MESSAGE " Read OK ", \
NB_BLOCKS% && " blocks have been read "
ENDIF
; Closes the file
F_CLOSE H%
```

See also

F_READ, F_BLOCKWRITE, F_WRITE

F_BLOCKWRITE Instruction

Writes several records (or blocks) at a time to a binary file.

Syntax F_BLOCKWRITE *file-handle, buffer-address, nbr-blocks-buffer, nbr-blocks-written*

Parameters	<i>file-handle</i>	INT(4) I	handle of the file to update
	<i>buffer-address</i>	INT(4) I	address of the buffer containing the records to write
	<i>nbr-blocks-buffer</i>	INT(2) I	number of records to write
	<i>nbr-blocks-written</i>	INT(2) I/O	number of records actually written

Notes

1. The records to write to the file must previously have been written to the buffer (or memory area) specified by *buffer-address*.
2. The size of the storage area must be greater than or equal to the size of the segment specified by F_OPEN% or F_CREATE%.
3. Records are written to the file starting from the current pointer position. The pointer is then positioned after the last record written.
4. The number of records to write to the file must match the number of records actually written to the buffer. This number is therefore always less than or equal to the number of records that appear in the buffer.
5. *nbr-blocks-written* is updated by the instruction once the records have been written to the file. This variable contains the number of records actually written to the file.

Example

```

; Declare and open the file in the same
; way as the files in the example for F_BLOCKREAD
; Write records to the buffer
BUFF[0].LAST_NAME = " Aldrin "
BUFF[0].FIRST_NAME = " Buzz "
BUFF[1].LAST_NAME = " Armstrong "
BUFF[1].FIRST_NAME = " Neil "
; Position the pointer at the end of the file to append
; the records to the file
F_SEEK H%,F_SIZE%(H%)
; Write to the file
F_BLOCKWRITE H%,@BUFF, NB_BLOCKS_BUFFER%, NB_BLOCKS%
IF F_ERROR% <> 0

```

```
MESSAGE " Error ", \
" Cannot add blocks "
ELSE
MESSAGE " Write OK ", \
NB_BLOCS% && " blocks have been written "
ENDIF
; Closes the file
F_CLOSE H%
```

See also F_BLOCKREAD, F_READ, F_WRITE

F_BYTEREAD Instruction

Read a specified number of bytes in a file starting at the current position of the pointer. The active is then repositioned according to the number of bytes read.

Syntax **F_BYTEREAD** *h%*, *data*, *nbbytes%*, *nbread%*

Parameters	<i>h%</i>	INT(4)	I	Handle of file opened using F_OPEN
	<i>data%</i>	POINTER	I	Address of target buffer
	<i>nbbytes%</i>	INTEGER	I	Number of bytes to read
	<i>nbread%</i>	INTEGER	O	Number of bytes read

Notes

1. *data%* is the memory address where the blocks are written to when read. The size of the allocated memory must be equal to or greater than the minimum size specified by *nbbytes%*.
2. *nbbytes%* is the number of bytes to read. Must be smaller than or equal to *data%*.
3. *nbbytes%* and *nbread%* are integers and are 2 bytes in 16 bit systems and 4 in 32 bit systems and specifies the real number of bytes read.
4. The number of bytes read is independent of the size returned by the F_OPEN% function.

Example

```
local integer NB_BYTES_READ%
local BUFF%, hFile%
hFile% = F_Open% (10, 'c:\autoexec.bat')
if MiscError% <> 0
  Message 'Error', 'Cannot open c:\autoexec.bat'
  Exit
endif
new 100, BUFF%
F_Bytread hFile%, BUFF%, 100, NB_BYTES_READ%
if (MiscError% <> 0) or (NB_BYTES_READ% <> 100)
  Message 'ERREUR', 'Error ' & MiscError% & ' bytes read:' \
    & NB_BYTES_READ%
endif
dispose BUFF%
F_Close (hFile%)
```


F_BYTESEEK Instruction

F_BYTESEEK sets the active pointer to the specified position.

Syntax **F_BYTESEEK** *h%, p%*

Parameters	<i>h%</i>	INT(4)	I	Handle of file opened using F_OPEN
	<i>p%</i>	INT(4)	I	New position of pointer

Note

1. The new position of the pointer is independent of the size returned by the F_OPEN% function.

F_BYTEWRITE Instruction

Write a specified number of bytes in a file starting at the current position of the pointer. The active is then repositioned according to the number of bytes written.

Syntax **F_BYTEWRITE** *h%, data, nbbytes%, nbread%*

Parameters	<i>h%</i>	INT(4)	I	Handle of file opened using F_OPEN
	<i>data%</i>	POINTER	I	Address of target buffer
	<i>nbbytes%</i>	INTEGER	I	Number of bytes to write
	<i>nbread%</i>	INTEGER	O	Number of bytes written

Notes

1. *data%* is the memory address where the blocks are copied from when writing..
The size of the allocated memory must be equal to or greater than the minimum size specified by *nbbytes%*.
2. *nbbytes%* is the number of bytes to write. Must be smaller than or equal to *data%*.
3. *nbbytes%* and *nbread%* are integers and are 2 bytes in 16 bit systems and 4 in 32 bit systems and specifies the real number of bytes written.
4. The number of bytes write is independent of the size returned by the F_OPEN% function.

Example

```
local integer NB_BYTES_READ%
local BUFF%, hFile%
hFile% = F_Open% (10, 'c:\autoexec.bat')
if MiscError% <> 0
    Message 'Error', 'Cannot open c:\autoexec.bat'
    Exit
endif
new 100, BUFF%
F_Byteread hFile%, BUFF%, 100, NB_BYTES_READ%
if (MiscError% <> 0) or (NB_BYTES_READ% <> 100)
    Message 'ERREUR', 'Error ' & MiscError% & ' bytes read:' \
        & NB_BYTES_READ%
endif
dispose BUFF%
F_Close (hFile%)
```

F_CLOSE Instruction

Closes a binary file.

Syntax `F_CLOSE file-handle`

Parameters *file-handle* INT(4) I file handle

Note The binary file must have been opened by F_OPEN% or F_CREATE%

Example

```
; *** See example for F_CREATE% and F_OPEN%
```

See also F_CREATE%, F_OPEN%

F_CREATE% Function

Creates a file in read/write mode and return its handle.

Syntax **F_CREATE%** (*segment-size*, *filename*)

Parameters	<i>segment-size</i>	INTEGER	I	size of the segment used in the file
	<i>filename</i>	CSTRING	I	name of the file to create

Return value INT(4)

Notes

1. A binary file is a file made up of a series of NCL segments that always have the same structure.
2. F_CREATE% creates a new file. If the file already exists, it deletes it and creates a new file.
3. The SIZEOF operator provides a simple way to specify the segment size.
4. You can subsequently use the handle returned to access the file with the other F_%% read/write functions and instructions.
5. Brackets used in the filename string enable environment parameters to be used:
; Creates the SAMPLE.TXT file located in the directory
; specified by the environment parameter NS-LST
MOVE F_CREATE%(..., " (NS-LST)SAMPLE.TXT ") TO H%

Example

```
; Declaration of the segment used
; INIT Event for the window
SEGMENT INDIVIDUAL
STRING LASTNAME (31)
STRING FIRSTNAME (31)
ENDSEGMENT

; EXECUTED event for the button that writes to the file
; Allocate memory and "map" it to the segment
LOCAL H%, DATA%
NEW INDIVIDUAL, DATA%

; Create the binary file " CLIENTS.CLI "
; (the file is opened if it already exists)
MOVE F_CREATE%(SIZEOF INDIVIDUAL, " CLIENTS.CLI ") TO H%
IF F_ERROR%<>0
```

```
MESSAGE " Error ", " Cannot open file "
EXIT
ENDIF

; Write new data to memory
MOVE " Smith " TO INDIVIDUAL(DATA%).LAST_NAME
MOVE " John " TO INDIVIDUAL(DATA%).FIRST_NAME
; Write contents of the memory area to the binary file
F_WRITE H%, DATA%
MOVE " Mouse " TO INDIVIDUAL(DATA%).LAST_NAME
MOVE " Mickey " TO INDIVIDUAL(DATA%).FIRST_NAME
F_WRITE H%, DATA%
; Closes the file
F_CLOSE H%
; Free the allocated memory
DISPOSE DATA%
```

See also

F_CLOSE, F_EOF%, F_ERROR%, F_LOCK, F_OPEN%, F_POS%, F_READ,
F_SEEK, F_SIZE%, F_TRUNCATE, F_UNLOCK, F_WRITE

T_* Functions and Instructions

NCL language: LOAD and SAVE instructions

F_EOF% Function

Indicates whether the current pointer position is at the end of the binary file.

Syntax **F_EOF%** (*file-handle*)

Parameters *file-handle* INT(4) I file handle

Return value INT(1)

value	meaning
TRUE%	the current pointer position is at the end of the file.
FALSE%	the current pointer position is not at the end of the file.

Example

```
; *** See examples for F_CREATE% and F_OPEN%
```

See also F_POS%

F_ERROR% Function

Returns the last error code resulting from binary file handling.

Syntax F_ERROR%

Return value INT(4)

value	meaning
0	No error
not null	Error in the last F_ *% function or instruction used. The various error codes returned are shown in appendix B of this manual.

Note We advise you to always call F_ERROR% after calling an F_ *% instruction or function to check that it was executed correctly.

Example

```
; *** See examples for F_CREATE% and F_OPEN%
```

See also F_CLOSE, F_CREATE%, F_EOF%, F_LOCK, F_OPEN%, F_POS%, F_READ, F_SEEK, F_SIZE%, F_TRUNCATE, F_UNLOCK, F_WRITE, FCOPYFILE

F_LOCK Instruction

Locks the record at the current pointer position in a binary file.

Syntax **F_LOCK** *file-handle*

Parameters *file-handle* INT(4) I file handle

Note Once a record is locked, it can no longer be modified. Use F_UNLOCK to unlock and modify the record.

Example

```
; *** See example for F_UNLOCK
```

See also F_ERROR%, F_UNLOCK

F_OPEN% Function

Opens a binary file in read/write mode and return its handle.

Syntax **F_OPEN%** (*segment-size*, *filename*)

Parameters	<i>segment-size</i>	INTEGER	I	size of the segment used in the file
	<i>filename</i>	CSTRING	I	name of the file to open

Return value INT(4)

Notes

1. A binary file is a file made up of a series of records that always have the same structure. A record is described using an NCL segment.
A binary file cannot be read using a text editor.
2. The SIZEOF operator provides a simple way to specify the segment size.
3. You can subsequently use the handle returned to access the file with the other F_*% read/write functions and instructions.
4. Once the file is opened, the pointer is positioned on the first record. To modify a record or write after the last record, you need to reposition the pointer using F_SEEK.
5. If the file does not exist, F_OPEN% returns an error. The error must be retrieved using F_ERROR%.
6. Brackets used in the filename string enable environment parameters to be used:
 ; Opens the SAMPLE.TXT file located in the directory
 ; specified by the environment parameter NS-LST
 MOVE F_OPEN%(..., " (NS-LST)SAMPLE.TXT ") TO H%

Example

```

; Declare the segment used
SEGMENT INDIVIDUAL
STRING LASTNAME(31)
STRING FIRSTNAME(31)
ENDSEGMENT
NEW INDIVIDUAL, DATA%

; Read the binary file " CLIENTS.CLI "
MOVE F_OPEN%(SIZEOF INDIVIDUAL, " CLIENTS.CLI ") TO H%
IF F_ERROR%<>0
MESSAGE " Error ", " Cannot open file "

```

```
EXIT
ENDIF
MOVE 0 TO I%
WHILE NOT F_EOF%(H%)
  F_READ H%, DATA%
  MOVE I%+1 TO I%
  MESSAGE " Client number " && I%, \
  INDIVIDUAL(DATA%).LASTNAME && INDIVIDUAL(DATA%).FIRSTNAME
ENDWHILE
F_CLOSE H%
DISPOSE DATA%
```

See also F_ROOPEN%, F_CREATE%, F_READ, F_WRITE

F_POS% Function

Returns the value of the current position pointer in a binary file.

Syntax **F_POS%** (*file-handle*)

Parameters *file-handle* INT(4) I file handle

Return value INT(4)

Notes

1. The position of the record read from or written to the binary file is stored in a pointer. This position is the sequence number of the segment in the file-the first segment is numbered 0.
2. The current position is automatically incremented following a read or write operation. It therefore corresponds to the next record after the one that has just been read or written.

Example

```
; Let's assume that the segment has been declared
; and the file has been opened as shown in the example
; for F_OPEN%
MOVE 0 TO I%
; Search for John COSTER
; and change the first name to Peter
WHILE NOT F_EOF%(H%)
  F_READ H%, DATA%
  IF INDIVIDUAL(DATA%).LASTNAME = "COSTER" \
  AND INDIVIDUAL(DATA%).FIRSTNAME = "John"
    ; Reposition the pointer
    ; to the record that has just been read
    F_SEEK H%,F_POS%(H%) -1
    MOVE "Peter" TO INDIVIDUAL (DATA%).FIRSTNAME
    F_WRITE H%,DATA%
    BREAK;
  ENDIF
ENDWHILE
F_CLOSE H%
DISPOSE DATA%
```

See also F_EOF%, F_SEEK

F_READ Instruction

Reads a segment at the current position in a binary file.

Syntax **F_READ** *file-handle, segment-address*

Parameters	<i>file-handle</i>	INT(4) I	file handle
	<i>segment-address</i>	INT(4) I	address of the segment that will receive the data read from the file

Notes

1. The segment whose address is passed in segment-address must be identical to the one used when the file is created or opened.
2. F_READ automatically increments the current position. The pointer will therefore move to the next record once the read operation is complete.
3. You can read several records at a time using F_BLOCKREAD.

Example

```
; Let's assume that the segment has been declared
; and the file has been opened as shown in the example
; for F_OPEN%
; Read all records
WHILE NOT F_EOF%(H%)
  F_READ H%, DATA%
  MOVE I%+1 TO I%
  MESSAGE " Client number " && I%, \
  INDIVIDUAL(DATA%).LAST_NAME && \
  INDIVIDUAL(DATA%).FIRST_NAME
ENDWHILE
F_CLOSE H%
DISPOSE DATA%
```

See also F_BLOCKREAD, F_POS%, F_SEEK, F_WRITE

F_ROOPEN% Function

Opens a binary file in read only mode and returns its handle.

Syntax **F_ROOPEN%** (*segment-size*, *filename*)

Parameters	<i>segment-size</i>	INTEGER	I	size of the segment used in the file
	<i>filename</i>	CSTRING	I	name of the file to open

Return value INT(4)

Notes

1. This function is identical to F_OPEN% except that it opens files in read-only mode. See notes on F_OPEN%.

See also F_OPEN%, F_CREATE%, F_READ, F_WRITE

F_SEEK Instruction

Modifies the current position in a binary file.

Syntax **F_SEEK** *file-handle*, *position*

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>position</i>	INT(4)	I	new current position in the file

Note This position is the sequence number of the record in the file-the first record is numbered 0 (zero).

Example

```
; Let's assume that the segment has been declared
; and the file has been opened as shown in the example
; for F_OPEN%
; Position the pointer on the last record
F_SEEK H%, F_SIZE%(H%)-1
; Position the pointer at the end of the file to write
; after the last record
F_SEEK H%, F_SIZE%(H%)
```

See also F_POS%, F_EOF%

F_SIZE% Function

Returns the size of a binary file.

Syntax **F_SIZE%** (*file-handle*)

Parameters *file-handle* INT(4) I file handle

Return value INT(4)

Note The size returned is the number of records in the file.

See also F_CREATE%, F_OPEN%

F_TRUNCATE Instruction

Truncates a binary file starting from the current position.

Syntax **F_TRUNCATE** *file-handle*

Parameter *file-handle* INT(4) I file handle

Note Truncate means that the contents of the file are deleted starting from the current position and up to the end of the file. After this, the file will only contain the records located before the current position.

See also F_CREATE%, F_OPEN%, F_ERROR%, F_POS%, F_READ, F_SEEK, F_SIZE%

F_UNLOCK Instruction

Unlocks the record at the current pointer position in a binary file

Syntax **F_UNLOCK** *file-handle*

Parameters *file-handle* INT(4) I file handle

Note The record must have been locked by F_LOCK. Once it is unlocked, you can modify it again.

Example

```
; Let's assume that the segment has been declared
; and the file has been opened as shown in the example
; for F_OPEN%
; Store the current position
MOVE POS%(H%) TO MEMPOS%
; Lock the current record
F_LOCK H%
; *** processing
; Reposition the pointer on the locked record
F_SEEK H%, MEMPOS%
; Unlock the record
F_UNLOCK H%
```

See also F_ERROR%, F_LOCK

F_WRITE Instruction

Writes a segment at the current position in a binary file.

Syntax **F_WRITE** *file-handle*, *segment-address*

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>segment-address</i>	INT(4)	I	address of the segment to write

Notes

1. The segment to write must be identical to the one used when the file was created or opened.
2. F_WRITE automatically increments the current position. The pointer will therefore move to the next record once the write operation is complete.
3. You can write several records at a time using F_BLOCKWRITE.

Example

```
; *** See examples for F_CREATE% and F_OPEN%
```

See also F_BLOCKWRITE, F_POS%, F_READ, F_SEEK

F_LoadFile\$ Function

Return the totality of a content file, indicated in parameter, in only one character string of the DynStr type. Thus, the jumps of line are present like control characters.

Syntax	F_LoadFile\$ <i>FileName\$</i>
Parameter	<i>Filename\$</i> CSTRING I name of a file
Return value	DYNSTR

Note

1. We recommend that you call MISCERROR% function after calling the F_LoadFile\$ function to check that it was executed correctly.

Example

```
F_LoadFile$ "File02.txt"
IF MISCERROR% = 0
    MESSAGE "OK", "Load of the file is completed!"
ELSE
    MESSAGE "Error" && MISCERROR%, "Load of the file is not completed"
ENDIF
```

See also F_SaveFile

F_SaveFile Instruction

Allows you to save text in a file.

Syntax **F_SaveFile** *FileName\$, Text\$*

Parameters	<i>FileName\$</i>	CSTRING	I	name of a file
	<i>Text\$</i>	DYNSTR	I	content of a file

Return value DYNSTR

Note

1. We recommend that you call MISCERROR% function after calling the F_SaveFile instruction to check that it was executed correctly.

Example

```
F_SaveFile "Fichier1.txt", "The F_SaveFile instruction allows you to \
save text in a file"
IF MISCERROR% = 0
    MESSAGE "OK", "Save is completed!"
ELSE
    MESSAGE "Error" && MISCERROR%, "Save is not completed"
ENDIF
```

See also F_LoadFile\$

FCOPYFILE Instruction

Copies a file under another name or to another file.

Syntax **FCOPYFILE** *source-filename, destination-filename*

Parameters	<i>source-filename</i>	CSTRING	I	name of the file to copy
	<i>destination-filename</i>	CSTRING	I	name of the destination file

Notes

1. This instruction only applies to a source file. It is therefore only partially equivalent to the COPY command in DOS: the '*' and '?' characters are not recognized.
2. The destination file is created if it does not exist; otherwise its contents are completely overwritten by the contents of the source file.
3. This instruction can be applied to any type of file, so the result of the copy can be retrieved using T_ERROR%, F_ERROR%.
4. Regardless of the type of error encountered (invalid destination directory, not enough disk space, etc.), only one value is returned by the functions mentioned in note 3: ERROR_FAILED_TO_COPY_FILE% (See appendix B of this manual).
5. Use FRENAME if you only want to rename a file.

Example

```
; Copy the contents of the file HELLO.TXT in the
; current directory to the file GOODBYE.TXT located in the
; same directory
FCOPYFILE " HELLO.TXT ", "GOODBYE.TXT " ;
IF F_ERROR% <> 0
    BEEP
    MESSAGE "Error","Copy impossible : error code : " && F_ERROR%
ENDIF
```

See also FRENAME

FERASE Instruction

Deletes a file.

Syntax **FERASE** *filename*

Parameters *filename* CSTRING I name of the file to delete

Note This instruction is equivalent to the DOS DEL command.

Example 1

```
FERASE " TEST.TXT "        ; deletes the TEST.TXT file located  
; in the current directory
```

Example 2

```
MOVE " \NATSTAR\SAMPLES\TEST.SCR " TO S$  
FERASE S$
```

See also RMDIR

FEXPAND\$ Function

Returns a string containing the full pathname of a file.

Syntax **FEXPAND\$** (*filename*)

Parameters *filename* CSTRING I filename to expand

Return value CSTRING

Notes

1. FEXPAND\$ does not check whether the file exists, it concatenates the name of the current directory (including the name of the disk volume) with the filename passed as a parameter.
2. If filename is an empty string (""), FEXPAND\$ returns a string containing the drive and the current directory followed by a \.
3. If filename contains only the file name without the directory, FEXPAND\$ returns a string containing the drive and the current directory followed by the file name.

Example 1

```
MESSAGE " The current directory is: ", FEXPAND$(" ")
```

Example 2

```
; If the current directory is D:\NATSTAR\SAMPLES
MOVE FEXPAND$("..\TEST.TXT ") TO S$
; This sets S$ to " D:\NATSTAR\TEST.TXT "
```

See also FGETDIR\$, FGETTEXT\$, FGETNAME\$, GETDIR\$, GETPATHNAME\$ (NSPATH library)

FFINDFIRST\$ Function

Returns the name of the first file with the specified attributes.

Syntax **FFINDFIRST\$** (*filename*, *attributes*)

Parameters	<i>filename</i>	CSTRING	I	file name
	<i>attributes</i>	INTEGER	I	file attributes

Return value CSTRING

Notes

1. The wildcard characters * and ? are accepted in the filename string.
2. The attributes are specified using a combination of FIND_%% constants.
3. FFINDFIRST\$ returns an empty string when no file is found.
4. See the notes on the FIND_%% constants for details on the various attributes that can be specified.
5. Use FFINDNEXT\$ to search for more files with the same attributes.

Example 1

```
; Searches for a file with a precise name
IF FFINDFIRST$(" C:\TEST.TXT ", FIND_ANYFILE%) = " "
MESSAGE " Warning ", " The file C:\TEST.TXT does not exist "
ENDIF
```

Example 2

```
; Searches for the size and modification date of a file
MOVE FFINDFIRST$(" C:\CONFIG.SYS ", FIND_ANYFILE% + \
FIND_RETURN_SIZE% + FIND_RETURN_DATE%) TO F$
; F$ may contain the following string, for example:
; " CONFIG.SYS,2296,33259 "
```

See also FFINDNEXT\$, FGETATTR%, FIND_%%, GETPATHNAME\$ (NSPATH library)

FFINDNEXT\$ Function

Following a successful search with FFINDFIRST\$, this function returns the name of the next file that has the same attributes as the first file found.

Syntax **FFINDNEXT\$**

Return value CSTRING

Notes

1. FFINDNEXT\$ returns an empty string when all files have been found.
2. See the notes on the FIND_%% constants for details on the various attributes that can be specified.
3. The date and time returned with the additional information (FIND_RETURN_%% constants) are in the form of integers. The DATE\$ and TIME\$ functions in the NSDATE library can be used to convert them into character strings.

Example 1

```
; Looks for all DLL files
; in the directory C:\WINDOWS
MOVE FFINDFIRST$(" C:\WINDOWS\*.DLL ", FIND_ANYFILE%) TO F$
WHILE F$ <> " "
MESSAGE " File found ", F$
MOVE FFINDNEXT$ TO F$
ENDWHILE
ENDIF
```

Example 2

```
; Searches for directories in volume C's root directory
; i.e. files with the Directory attribute only
MOVE FFINDFIRST$(" C:\*.* ", FIND_DIRECTORY% + \
FIND_ORED_ONLY%) TO F$
WHILE F$ <> " "
MESSAGE " Directory found ", F$
MOVE FFINDNEXT$ TO F$
ENDWHILE
```

See also FFINDFIRST\$, FGETATTR%, FIND_%%, GETPATHNAMES\$ (NSPATH library)

FFINDFIRSTEX\$ Function

This function is used in the same way as the FFINDFIRST\$ function. The only difference being that the *path\$* parameter can contain a long name. If the error code MISCERROR% equals 0, the returned string will be the first long filename corresponding to the searched values. The *hDir%* parameter will be positioned, allowing it to be reused when FFINDNEXTEX\$ and FFINDCLOSEEX functions are called. This supplementary parameter authorizes simultaneous searches in several directories or sub-directories.

Syntax **FFINDFIRSTEX\$** (*path\$, attr%, hDir%*)

Parameters	<i>path\$</i>	CSTRING	I	filename string (can contain wildcard characters * and ?)
	<i>attr%</i>	INT(4)	I	search attributes
	<i>hdir%</i>	INT(4)	O	returned handle for the following functions : FINDNEXTEX\$ and FFINDCLOSEEX\$

Return value CSTRING first long name of a file corresponding to the sought after values

Example

```
; Search in the directory "c:\my directory"
local int hDir%
local s$
s$ = FFINDFIRSTEX$("c:\mon répertoire\mes document*", \  FIND_ANYFILE%,hDir%)
if miscError% <> 0
    exit
endif
while miscError% = 0
    insert at end s$ to LB_RESULT
    s$ = FFINDNEXTEX$ (hDir%)
endWhile
FFINDCLOSEEX (hDir%)
```

See also FFINDFIRST\$, FFINDNEXTEX\$, FFINDCLOSEEX, GETSHORTFILENAME\$ from the NSWIN library.

FFINDNEXTEX\$ Function

Following a successful search with FFINDFIRSTEX\$, this function returns the name of the next file that has the same attributes as the first file found.

Syntax **FFINDNEXTEX\$** (*hDir%*)

Parameter *hDir%* INT(4) I handle for files obtained by the FFINDFIRSTEX\$

Returned value CSTRING the following long filename corresponding to the desired values

Notes

1. FFINDNEXT\$ returns an empty string when all files have been found.
2. See the notes on the FIND_ *% constants for details on the various attributes that can be specified.
3. The date and time returned with the additional information (FIND_RETURN_ *% constants) are in the form of integers. The DATE\$ and TIME\$ functions in the NSDATE library can be used to convert them into character strings.

See Also FFINDNEXT\$, FFINDFIRSTEX\$, FFINDCLOSEEX, GETSHORTFILENAME
from the NSWIN library

FFINDCLOSEEX Instruction

Following a successful search with the FFINDFIRSTEX\$ function this instruction must be called in order to free allocated system resources.

Syntax **FFINDNEXTTEX\$** *hDir%*

Parameter *hdir%* INT(4) I handle of the directory obtained by the
FFINDFIRSTTEX\$.

See also FFINDFIRSTTEX\$, FFINDNEXTTEX\$

FGETATTR% Function

Returns the attributes of a file.

Syntax **FGETATTR%** (*filename*)

Parameters *filename* CSTRING I file name

Return value INTEGER

Note The attributes are specified using a combination of FIND_ *% constants.

Example

```
MOVE " C:\WINDOWS " TO F$
IF FGETATTR%(F$) BAND FIND_DIRECTORY% <> 0
MESSAGE F$, " is a directory "
ENDIF
```

See also FSETATTR, FFINDFIRST\$, FFINDNEXT\$, FIND_ *%

FGETDIR\$ Function

Returns the directory of a file.

Syntax **FGETDIR\$** (*filename*)

Parameters *filename* CSTRING I file name

Return value CSTRING

Note

1. *filename* must specify the file's full name (disk, directory, filename)
2. This function never accesses the disk. It is only applied to the character string passed as a parameter, from which it extracts the sub-string that specifies the directory.
3. If the file's directory is not specified in filename, FGETDIR\$ returns an empty string, regardless of the actual location of the file on the disk.
4. Similarly, FGETNAME\$ and FGETTEXT\$ can be used to obtain the file's name and extension.
5. FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETTEXT\$(F\$) is always strictly equal to F\$, whatever the value of F\$.

Example

```
MOVE " C:\NATSTAR\SCREENS\SAMPLE.SCR " TO F$  
MESSAGE " Directory ", FGETDIR$(F$) ; displays " C:\NATSTAR\SCREENS\ "
```

See also FGETTEXT\$, FGETNAME\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (NSPATH Library)

FGETTEXT\$ Function

Returns the extension of a file.

Syntax **FGETTEXT\$** (*filename*)

Parameters *filename* CSTRING I file name

Return value CSTRING

Notes

1. *filename* must specify the file's full name (disk, directory, filename)
2. This function never accesses the disk. It is only applied to the character string passed as a parameter, from which it extracts the sub-string that specifies the file extension.
3. Similarly, FGETDIR\$ and FGETNAME\$ can be used to obtain the file's directory and name.
4. FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETTEXT\$(F\$) is always strictly equal to F\$, whatever the value of F\$.

Example

```
MOVE " C:\NSNATSTAR\SCREENS\SAMPLE.SCR " TO F$  
MESSAGE " Extension ", FGETTEXT$(F$) ; displays " .SCR "
```

See also FGETDIR\$, FGETNAME\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (NSPATH Library)

FGETNAME\$ Function

Returns the name of a file.

Syntax **FGETNAME\$** (*filename*)

Parameters *filename* CSTRING I file name

Return value CSTRING

Notes

1. filename must specify the file's full name (disk, directory, file name)
2. This function never accesses the disk. It is only applied to the character string passed as a parameter, from which it extracts the sub-string that specifies the filename.
3. Similarly, FGETDIR\$ and FGETTEXT\$ can be used to obtain the file's directory and extension.
4. FGETDIR\$(F\$) & FGETNAME\$(F\$) & FGETTEXT\$(F\$) is always strictly equal to F\$, whatever the value of F\$.

Example

```
MOVE " C:\NATSTAR\SCREENS\SAMPLE.SCR " TO F$  
MESSAGE " Name ", FGETNAME$(F$)        ; displays " SAMPLE "
```

See also FGETDIR\$, FGETTEXT\$, GETDIR\$, FEXPAND\$, GETPATHNAME\$ (NSPATH Library)

FGETTIME Instruction

Returns the specified file's timestamp and assigns the corresponding variables *file\$*, *year%*, *month%*, *day%*, *hour%*, *min%* and *sec%*.

Syntax **FGETTIME** *file\$*, *year%*, *month%*, *day%*, *hour%*, *min%*, *sec%*

Parameters	<i>file\$</i>	CSTRING	I	Name of file
	<i>year%</i>	INTEGER	O	Year
	<i>month%</i>	INTEGER	O	Month
	<i>day%</i>	INTEGER	O	Day
	<i>hour%</i>	INTEGER	O	Hour
	<i>min%</i>	INTEGER	O	Minutes
	<i>sec%</i>	INTEGER	O	Seconds

Notes

1. If the specified file is not found or cannot be read, the MISCERROR% function returns ERROR_CANNOT_ACCESS_TO_FILE%.

See Also FSETTIME

FSETTIME Instruction

Modifies the file's timestamp using the variables YEAR%, MONTH%, DAY%, HOUR%, MIN% and SEC%.

Syntax **FSETTIME** *file\$, year%, month%, day%, hour%, min%, sec%*

Parameters	<i>file\$</i>	CSTRING	I	Name of file
	<i>year%</i>	INTEGER	I	Year
	<i>month%</i>	INTEGER	I	Month
	<i>day%</i>	INTEGER	I	Day
	<i>hour%</i>	INTEGER	I	Hour
	<i>min%</i>	INTEGER	I	Minutes
	<i>sec%</i>	INTEGER	I	Seconds

Notes

1. If the specified file is not found or cannot be read, the MISCERROR% function returns ERROR_CANNOT_ACCESS_TO_FILE%.

See also FGETTIME

FGETSIZE% Function

Returns the size of a file.

Syntax **FGETSIZE%** (*filename*)

Parameter *filename* CSTRING I file name

Return value INT(4)

value	meaning
0	file does not exist or was not found
not null	size of the file in bytes

Note This function searches for the file in the current directory if filename does not contain a path.

Example

```
MESSAGE " Size of CONFIG.SYS ", FGETSIZE%( " C:\CONFIG.SYS " )
```

See also DISKSIZE%, F_SIZE%

FIND_ *% Constants

File attributes and search criteria.

Syntax

- FIND_ANYFILE%**
- FIND_ARCHIVE%**
- FIND_DIRECTORY%**
- FIND_HIDDEN%**
- FIND_READONLY%**
- FIND_SYSFILE%**
- FIND_ORED_ONLY%**
- FIND_RETURN_ATTR%**
- FIND_RETURN_SIZE%**
- FIND_RETURN_DATE%**
- FIND_RETURN_TIME%**

Notes

1. FIND_ANYFILE% up to FIND_SYSFILE% are the file attributes used by the FFINDFIRST\$ and FGETATTR% functions and the FSETATTR instruction.
2. FIND_ORED_ONLY% is used by FFINDFIRST\$. When combined with the file attributes, FIND_ORED_ONLY% indicates that the search is only to include files whose attributes exactly match the combination.
3. FIND_RETURN_*% are used by FFINDFIRST\$. When combined with the file attributes, these constants specify the additional information (attribute, size, modification date, modification time) to be returned by FFINDFIRST\$ and FFINDNEXT\$ after the file name.

4. These constants have the following meaning:

FIND_READONLY%	Read only (write-protected)
FIND_HIDDEN%	File hidden (not displayed with a DIR)
FIND_SYSFILE%	System file
FIND_DIRECTORY%	Directory
FIND_ARCHIVE%	Archive attribute under DOS



This attribute does not correspond to the attribute directories under the Windows explorer.

FIND_ANYFILE%	Any file type
---------------	---------------

- | | |
|-------------------|--|
| FIND_ORED_ONLY% | Search using exact attribute combination |
| FIND_RETURN_ATTR% | Search returns attribute |
| FIND_RETURN_SIZE% | Search returns size |
| FIND_RETURN_DATE% | Search returns date |
| FIND_RETURN_TIME% | Search returns time |
5. The FIND_READONLY%, FIND_HIDDEN%, FIND_SYSFILE%, FIND_DIRECTORY% and FIND_ARCHIVE% constants may be combined (e.g. FIND_HIDDEN% + FIND_DIRECTORY% corresponds to a hidden directory). FIND_ANYFILE% is the sum of the first five constants.
 6. The result of a search always includes " normal " files. Hence, a search performed with FIND_DIRECTORY% not only returns directories but also normal files. A strict distinction can be made by using FIND_ORED_ONLY%.
 7. The FIND_RETURN_*% constants can be used to obtain additional information on the file searched for. This information is placed after the file name returned by FFINDFIRST\$ and FFINDNEXT\$ and always appears in the following order : attribute, size, date modified, time modified. The information is separated by commas. It is only provided if the corresponding constant has been specified in the attribute parameter of FFINDFIRST\$.
 8. They are declared internally as follows:

CONST FIND_READONLY%	\$01
CONST FIND_HIDDEN%	\$02
CONST FIND_SYSFILE%	\$04
CONST FIND_DIRECTORY%	\$10
CONST FIND_ARCHIVE%	\$20
CONST FIND_ANYFILE%	\$37
CONST FIND_ORED_ONLY%	\$0100
CONST FIND_RETURN_ATTR%	\$0200
CONST FIND_RETURN_SIZE%	\$0400
CONST FIND_RETURN_DATE%	\$0800
CONST FIND_RETURN_TIME%	\$1000

See also FFINDFIRST\$, FGETATTR%, FSETATTR

FIRSTCONTROL Instruction

Returns the first control in a window.

Syntax **FIRSTCONTROL** *window-handle, control-var*

Parameters	<i>window-handle</i>	POINTER	I	window handle
	<i>control-var</i>	CONTROL	I/O	first control in the window

Notes

1. The sequence of controls returned is simply the order in which they were saved, which can vary after each save. This order is not related to the physical layout of the window or the focus sequence. We therefore advise you not to make any assumptions about the sequence of controls. The FIRSTCONTROL and NEXTCONTROL instructions should only be used to obtain the names of all the controls in the window and not to retrieve their position, which can be obtained with the .X and .Y dynamic qualifiers.
2. To obtain the next control, use NEXTCONTROL.

Example

```
GLOBAL CONTROL CTRL
FIRSTCONTROL SELF%, CTRL
MESSAGE " The first control found is located at ", \
CTRL.X && CTRL.Y
```

See also NEXTCONTROL, CONTROLNAME\$, WINDOWFROMCONTROL%

FOCUSCONTROL Instruction

Returns the control that has the focus in a window.

Syntax **FOCUSCONTROL** *window-handle, control-var*

Parameters	<i>window-handle</i>	POINTER	I	window handle
	<i>control-var</i>	CONTROL	I/O	control that has the focus in the window

Note

1. If an error occurs, the MISCERROR% function returns ERROR_INVALID_HANDLE% or ERROR_NO_MORE_CONTROL%.

Example

```
LOCAL CONTROL CTRL_FOCUS
; Modifies the background color
; of the control that has the focus
FOCUSCONTROL SELF%, CTRL_FOCUS
MOVE COL_RED% TO CTRL_FOCUS.BACKCOLOR
```

See also GETTABCONTROL, SETTABCONTROL

FRENAME Instruction

Renames a file.

Syntax **FRENAME** *old-filename, new-filename*

Parameters	<i>old-filename</i>	CSTRING	I	old filename
	<i>new-filename</i>	CSTRING	I	new filename

Note This instruction is equivalent to the DOS REN or RENAME command.

Example

```
; Rename HELLO.TXT as HOWDY.TXT
; in the current directory
FRENAME " HELLO.TXT " , " HOWDY.TXT "
```

See also FCOPYFILE

FSETATTR Instruction

Modifies the attributes of a file.

Syntax **FSETATTR** *filename*, *attribute*

Parameters	<i>filename</i>	CSTRING	I	name of the file to modify
	<i>attribute</i>	INTEGER	I	new file attributes

Note The attributes of the file are specified by a combination of FIND_ *% constants.

Example

```
; Hides the TEST.TXT file and protects it against writes
FSETATTR " TEST.TXT ", FIND_READONLY% + FIND_HIDDEN%
```

See also FGETATTR%, FFINDFIRST\$, FFINDNEXT\$, FIND_ *%

GETBITMAPCONTROLSDEFAULTS% Function

Allows you to know the default behavior of bitmap controls. The value returned using this verb are a sum of the following NSMISC flags : DEFBMPTRANSPARENT% et DEFBMPBUTTONIZED%.

Syntax **GETBITMAPCONTROLSDEFAULTS%**

Notes

1. The definition of the flags are :

- **DEFBMPTRANSPARENT%**
Activate transparency by default. Also used with `PictureButton` and `PrintButton` NatStar (or NS-DK) custom controls.
- **DEFBMPBUTTONIZED%**
Activates the automatic drawing of borders and simulated Pressed and Disabled `PushButton` bitmap controls with only one bitmap (Released).

See Also **SETBITMAPCONTROLSDEFAULTS**

GETCHECKMODE% Function

Returns an indicator stating whether the mode that prevents a control from losing the focus is enabled.

Syntax **GETCHECKMODE%** (*window-handle*)

Parameters *window-handle* POINTER I window handle

Return value INT(1)

value	meaning
TRUE%	check mode is enabled
FALSE%	check mode is disabled

Example

```
; Reverse the " Check Mode " for the current mode  
IF GETCHECKMODE%(SELF%)  
  SETCHECKMODE SELF%, FALSE%  
ELSE  
  SETCHECKMODE SELF%, TRUE%  
ENDIF
```

See also SETCHECKMODE

GETCONTROLKIND function

This function returns the same value as the .KIND dynamic qualifier. But if the control in question is a custom-control type its GETVALUE event will not be called.

Syntax **GETCONTROLKIND** (*ctrl*)

Parameter *ctrl* CONTROL I Control to test

Returned value INT(2) this value is one of the DI_STATICTEXT% to DI_CUSTOM% constant.

Example

```
local control ctrl
move PB_TEST to @ctrl
message "Type du control PB_TEST", getControlKind (ctrl)
```

See also DI_*, .KIND

GETCONTROLSUBKIND function

This function returns the same value as the .SUBKIND dynamic qualifier. It enables identification for a bitmap, an icon, a push-button or a check-box...

Syntax **GETCONTROLSUBKIND** (*ctrl*)

Parameter *ctrl* CONTROL I Bitmap control type to testfor subkind

Returned Value INT(2) This value is the following constants : DI_ICON%,
DI_PUSHBUTTON% or DI_CHECKBOX%

Example

```
local control ctrl
move BMP_TEST to @ctrl
message "Sous type de la bitmap PB_TEST", getControlSubKind (ctrl)
```

See also DI_*

GETCURRENTCHARSET% Function

Returns the character set used by the application.

Syntax **GETCURRENTCHARSET%**

Return value INTEGER

value	meaning
CHS_*	Character set used

Note This function is particularly useful when developing multi-target applications designed to run on different systems.

Example

```
LOCAL S$(100)
MOVE ENTRY TO S$
; Convert the string into ASCII if the current
; character set is ANSI
IF GETCURRENTCHARSET% = CHS_ANSI%
TRANSLATECHARS @S$, length S$, CHS_ANSI%, CHS_ASCII%
ENDIF
```

See also GETHOSTCHARSET%, TRANSLATECHARS and CHS_*

GETDECIMALSEPARATOR\$ Function

Returns the character used as the decimal separator within the application.

Syntax **GETDECIMALSEPARATOR\$**

Return value CSTRING

Note If SETDECIMALSEPARATOR has not been executed, this function will return the default separator set in the Control Panel.

See also STRING\$, SETDECIMALSEPARATOR, GETTHOUSANDSSEPARATOR\$, SETTHOUSANDSSEPARATOR

GETDIR\$ Function

Returns the current directory of a disk.

Syntax **GETDIR\$** (*disk-number*)

Parameters *disk-number* INT(1) I disk number

Return value CSTRING

Notes

1. *disk-number* must be an integer between 0 and 26, where:

Value	Stands for the following disk:
0	current disk
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z

2. GETDIR\$(0) and GETDIR\$(GETDISK%) are equivalent but GETDISK% never returns 0!

Example

```
MESSAGE " The current directory is ", GETDIR$(0)
```

See also CHDIR, MKDIR, RMDIR, GETPATHNAME\$ (NSPATH library)

GETDISK% Function

Returns the number of the current disk.

Syntax **GETDISK%**

Return value INT(4)

Note The disk number is an integer between 1 and 26, where:

Value	Stands for the following disk:
0	current disk
1	A:
2	B:
3	C:
4 ... 25	D: ... Y:
26	Z

Example

```
MESSAGE " The current disk is ", CHR$(64 + GETDISK%)
```

See also GETDISKS%, DISKFREE%, DISKSIZE%

GETDISKS% Function

Returns an integer indicating which logical disks are available.

Syntax **GETDISKS%**

Return value INT(4)

Note Each bit in the integer returned represents a logical disk (bit with order 1 for volume A, bit with order 2 for volume B, and so on). When a bit is set to 1, this indicates that the logical volume exists.

Example

```
LOCAL I%,J%,LDISK%
; Displays the available logical disks
; in the LB_DISK List box
MOVE GETDISKS% TO I%
MOVE 1 TO J%
MOVE ASC%(" A ") TO LDISK%
REPEAT
IF (I% BAND J%) = J%
INSERT AT END CHR$(LDISK%) TO LB_DISK
ENDIF
MOVE J%*2 TO J%
MOVE LDISK%+1 TO LDISK%
UNTIL LDISK% = ASC%(" Z ")
```

See also GETDISK%, DISKFREE%, DISKSIZE%

GETDISKTYPE% Function

Returns the disk type of the associated letter passed as a parameter.

Syntax **GETDISKTYPE%** (*volume%*)

Parameter *volume%* INT(1) I Disk of known type

Return value INT DT_ *% constant corresponding to disk type

Notes

1. Number of disk :

0 current disk

1 A:

2 B:

3 C:

26 Z:

2. If not found, the MISCERROR% function returns
ERROR_CANNOT_GET_DISK%.

See Also GETVOLUMENAMES

GETENV\$ Function

Returns the value of an environment parameter.

Syntax **GETENV\$** (*env-string*)

Parameter *env-string* CSTRING I environment parameter

Return value CSTRING

Example

```
MESSAGE " PATH variable ", GETENV$(" PATH ")
MESSAGE " NS-INI variable ", GETENV$(" NS-INI ")
```

See also GETLONGENV\$, ENVCOUNT%, ENVSTR\$, PUTENV

GETLONGENV\$ Function

Returns the value of an environment parameter.

Syntax **GETENV\$** (*env-string*)

Parameter *env-string* CSTRING I environment parameter

Return value DYNSTR

Note

1. This function returns character strings whatever its size. It is thus often more practical to use this function instead of GETENV\$.

Example

```
MESSAGE " PATH variable ", GETLONGENV$(" PATH ")  
MESSAGE " NS-INI variable ", GETLONGENV$(" NS-INI ")
```

See also GETENV\$, ENVCOUNT%, ENVSTR\$, PUTENV

PUTENV Instruction

Allows you to allocate a value to an environment variable.

Syntax **PUTENV** *name\$*, *value\$*

Parameters	<i>name\$</i>	CSTRING	I	environment variable
	<i>value\$</i>	CSTRING	I	value of the environment variable

See also GETENV\$, GETLONGENV\$

GETESCAPECONTROL Instruction

Specifies which control in a Dialog box will be activated when the end-user presses the [Esc] key.

Syntax **GETESCAPECONTROL** (*window-handle*, *control*)

Parameters	<i>window-handle</i>	POINTER	I	Dialog box handle
	<i>control</i>	CONTROL	I/O	control activated

Note

1. The specified control is the one that has been associated statically with the [Esc] key in the window's Info box or dynamically using SETESCAPECONTROL. The associated control is usually a Push button that receives the EXECUTED event when the end-user presses [Esc].

Example

```
LOCAL CONTROL CTRL
GETESCAPECONTROL SELF%, CTRL
IF CTRL = PB_CANCEL
MESSAGE "Info",\
"The [Esc] key is associated with the Cancel button
ENDIF
```

See also SETESCAPECONTROL, GETRETURNCONTROL, SETRETURNCONTROL

GETHOSTCHARSET% Function

Returns the character set used by the system of the machine running the application.

Syntax **GETHOSTCHARSET%**

Return value	INTEGER
	value meaning
	CHS_%% Character set used.

Note This function is particularly useful when developing multi-target applications designed to run on different systems.

See also GETCURRENTCHARSET%, TRANSLATECHARS and CHS_%% constants

GETLISTBOXCELL instruction

Calculates the coordinates of a List Box cell or of a pixel depending on the situation and stores the result in the fields X, Y, W, H, L and C of the segment SEG_LBCELL.

This function is very useful for editing in an entry field in a List Box.

Syntax	GETLISTBOXCELL <i>lb, cell, xc%, yl%, flags%</i>			
Parameters	<i>lb</i>	CONTROL	I	name of the List Box
	<i>cell</i>	SEGMENT	I/O	segment SEG_LBCELL
	<i>xc%</i>	INTEGER	I	x coordinate of a pixel or number of the column
	<i>yl%</i>	INTEGER	I	y coordinate of a pixel or number of the line
	<i>flags%</i>	INTEGER	I	LBCF_%% constants

Comments

1. The segment SEG_LBCELL defined in the file NSMISC. NCL:

```

SEGMENT SEG_LBCELL ; coordinates of a List Box cell
    INTEGER X ; The abscissa in pixels from the bottom left corner of the cell
; in relation to the bottom left corner of the client field of the List Box.
    INTEGER Y ; The ordinate in pixels from the bottom left corner of the cell
in
; relation to the bottom left corner of the client field of the List
Box.
    INTEGER W ;the width in pixels of the cell or that of the line if
;LBCF_WHOLELINE% is used
    INTEGER H ; Height of the cell in pixels
    INTEGER L ; Cell line. Lines start at 0
    INTEGER C ; Cell column. Columns start at 1
    INTEGER CLIP ; Indicates if the cell is clipped or hidden (combination of
; LBCC_%% constants)
ENDSEGMENT ; SEG_LBCELL

```

2. The parameters *xc%* and *yl%* indicate:

- ◆ either the coordinates (x, y) of a pixel in a client field of the List Box, if the flag LBCF_FROMXY% is present in the parameter *flags%*. In this situation, you could pass the parameters PARAM1% and PARAM2% as parameters xc and xy of the events MOUSEMOVE, BUTTONDOWN, BUTTONUP and BUTTONDBLCLK of the List Box.
- ◆ or the number of the column (*xc%*) and of the line (*yl%*). The columns and lines being numbered from 0.

3. The constants LBCF_*% being able to be added to the parameter *flags%* have the following role:

LBCF_CLIPCOORDS%	clips the rectangle to the borders of the visible area of the List Box.
LBCF_ENLARGERIGHTCOL%	if the cell indicated is in the last column, enlarges it (if possible) up to the right border of the visible area.
LBCF_WHOLELINE%	gives a rectangle taking up all the line (or the whole width of the visible area if LBCF_CLIPCOORDS% is present).
LBCF_FROMXY%	<i>xc%</i> and <i>yl%</i> correspond to the x/y coordinates of a pixel.

4. The field CLIP of the segment SEG_LBCELL indicates if one or more sides of the rectangle are clipped (to the border of the visible area). If the field CLIP is different from 0, the borders of the rectangle are clipped.

5. The constants LBCC_*% correspond to the field CLIP of the parameter *cell*.

LBCC_CLIPLEFT%	\$01	Left border clipped.
LBCC_CLIPRIGHT%	\$02	Right border clipped.
LBCC_CLIPTOP%	\$04	Top clipped.
LBCC_CLIPBOTTOM%	\$08	Bottom clipped.
LBCC_CLIPALL%	\$10	All sides clipped.
LBCC_BELOWLASTLINE%	\$20	Clipped after the last line.
LBCC_RIGHTOFLASTCOL%	\$40	Clipped to the right of the right border.

LBCC_TOPLOCKED%	\$80	The cell belongs to the top locked lines of a List Box.
-----------------	------	---

Example

```
LOCAL POINTER HDATA%
NEW SEG_LBCELL, HDATA%
IF HDATA%=0
    MESSAGE "ERREUR","Handle of NULL segment"
    EXIT
ENDIF
FILL HDATA%, SIZEOF SEG_LBCELL, 0
GETLISTBOXCELL LB01, SEG_LBCELL(HDATA%), 1, 2, LBCF_CLIPCOORDS%
MOVE SEG_LBCELL(HDATA%).X TO EF_X
MOVE SEG_LBCELL(HDATA%).Y TO EF_Y
MOVE SEG_LBCELL(HDATA%).W TO EF_W
MOVE SEG_LBCELL(HDATA%).H TO EF_H
MOVE SEG_LBCELL(HDATA%).L TO EF_L
MOVE SEG_LBCELL(HDATA%).C TO EF_C
MOVE SEG_LBCELL(HDATA%).CLIP AND LBCC_CLIPRIGHT% TO EF_CLIP

DISPOSE HDATA%
```

See also

BEGINEDITINPLACE

GETMDISTR\$ Function

Returns the title of an item on the system menu or arrangement menu (Cascade/Tile) in an MDI window.

Syntax	GETMDISTR\$ (<i>menu-item</i>)			
Parameters	<i>menu-item</i>	INT(1)	I	menu item
Return value	CSTRING			

Notes

- 1. The required menu item is passed as a parameter in the form of an MDI_ *% constant.
- 2. If the title contains a keyboard accelerator character when the menu is displayed, this is preceded by a tilde (~) character in the string returned by GETMDISTR\$.

Example

```
; Read the title for tile
LOCAL A$(40)
MOVE GETMDISTR$(MDI_TILE%) TO A$
```

See also SETMDISTR, MDI_ *%

GETRETURNCONTROL Instruction

Specifies which control in a Dialog box will be activated when the end-user presses the [Enter] key.

Syntax **GETRETURNCONTROL** (*window-handle*, *control*)

Parameters	<i>window-handle</i>	POINTER	I	dialog box handle
	<i>control</i>	CONTROL	I/O	control activated

Note The specified control is the one that has been associated statically with the [Enter] key in the window's property pane or dynamically using SETRETURNCONTROL. The associated control is usually a Push button that receives the EXECUTED event when the end-user presses [Enter].

Example

```
LOCAL CONTROL CTRL
GETRETURNCONTROL SELF%, CTRL
IF CTRL = PB_OK
MESSAGE "Info",\
"The [Enter] key is associated with the OK button"
ENDIF
```

See also SETRETURNCONTROL, GETESCAPECONTROL, SETESCAPECONTROL

GETTABCONTROL Instruction

Indicates which control in a Dialog class window will get the focus when the end-user presses [Tab] from a specified control.

Syntax **GETTABCONTROL** (*source-ctrl*, *target-ctrl*)

Parameters	<i>source-ctrl</i>	CONTROL	I	control that loses the focus
	<i>target-ctrl</i>	CONTROL	I/O	control that gets the focus

Note The target control is the one that has been associated statically with the Tab field in the property pane for source-ctrl or specified dynamically using SETTABCONTROL.

Example

```
local CONTROL CTRLSRC
local CONTROL CTRLCIBLE
; Determine which control has the focus
FOCUSCONTROL SELF%, CTRLSRC
; Retrieve the name of the control that will get the focus
GETTABCONTROL CTRLSRC, CTRLCIBLE
MESSAGE " GetTabControl ", \
CONTROLNAME$(CTRLCIBLE) && \
" will get the focus after " && \
CONTROLNAME$(CTRLSRC)
; This example code can appear in the EXECUTED event for a
; test Push button if the button's No Focus option is
; checked in its Info box.
```

See also SETTABCONTROL, FOCUSCONTROL

GETTHOUSANDSSEPARATOR\$ Function

Returns the character used as the thousands separator within the application.

Syntax **GETTHOUSANDSSEPARATOR\$**

Return value CSTRING

Note If a SETTHOUSANDSSEPARATOR has not been performed, the returned character will be the default separator set in the Control Panel.

See also STRING\$, SETTHOUSANDSSEPARATOR, GETDECIMALSEPARATOR\$, SETDECIMALSEPARATOR

GETVIRTUALKEYSTATE% Function

Determines whether or not a key (VK_*) is pressed (TRUE%) or not (FALSE%).

Syntax **GETVIRTUALKEYSTATE%** (VKEY%)

Return value INT(1)

GETVOLUMENAME\$ Function

Returns the disk label of the associated number passed as a parameter.

Syntax **GETVOLUMENAME\$** (*volume%*)

Parameter *volume%* INT (1) Disk to retrieve the label

Return value CSTRING label of disk

Notes

1. Number of disk

0 current disk

1 A:

2 B:

3 C:

... ..

26 Z:

2. If not found, the MISCERROR% function returns
ERROR_CANNOT_GET_DISK%.

See Also GETDISKTYPE%

GETWNDCAPTURE Function

Returns the handle of the window that has the mouse pointer. This handle is not an NSDK handle, but the OS handle.

Syntax **GETWNDCAPTURE**

Return value **POINTER** **I** handle of window having captured the mouse pointer

Note

1. You convert this handle into a NSDK handle, use GETSELFFROMHWND% in NSWIN.NCL for Windows.

See also GETSELFFROMHWND%

INSTALL_CALLBACK Function

Returns a handle (hCallback) that must be passed by CALL_PREVALLCALLBACK and REMOVE_CALLBACK.

Syntax **INSTALL_CALLBACK** (*pfnCallback*)

Parameter *pfnCallback* POINTER I pointer to a function

Return value POINTER

Notes

1. The function of which the pointer is passed in parameter must be prototyped in the following way. The name of the function can be different :

```
FUNCTION MyCallback(POINTER Self%, INTEGER Id%, INTEGER Msg%, POINTER Parm1,  
POINTER Parm2) RETURN INTEGER
```

MyCallback function is then called for each event concerning a window or a control of the application, Self% and Id% indicate the window or the control concerned, Msg% the event, Parm1 and Parm2 the associated parameters.

2. Each using of MyCallback function must call once and only once CALL_PREVALLCALLBACK.

```
i% = CALL_PREVALLCALLBACK(hCallback, Self%, Id%, Msg%, Parm1, Parm2)  
;... additional processing  
RETURN i%
```

or

```
RETURN CALL_PREVALLCALLBACK(hCallback, Self%, Id%, Msg%, Parm1, Parm2)
```

See also CALL_PREVCALLBACK , REMOVE_CALLBACK

ISVALIDSELFHANDLE% Function

Determines if the number passed as a parameter is a valid handle.

This handle is a NatStar (or NS-DK) handle and not a system handle. For example, NatStar (or NS-DK) handles are the handles returned by SELF%.

Syntax **ISVALIDSELFHANDLE%** (*WinHand%*)

Parameters *WinHand%* INT(4) I Handle of the window

Return value INT(1) True% or False%

Note

1. This function can be used to see if a window is still open or has been closed by the user.

Example

```
IF ISVALIDSELFHANDLE%(HANDLE%)  
  . . .  
ENDIF
```

LGETENV function

Equivalent of GETENV\$ but returns a pointer on a CSTRING longer than 255 characters.

Syntax **LGETENV** (*name*\$)

Parameter *name*\$ CSTRING I Environment variable to read

Return value POINTER Pointer CSTRING

Use

A structure type containing a long enough cString must be declared which is then used to access the function result

Example

INIT event of the window

```
segment lCString
  cstring cs (500)
endSegment
```

EXECUTED event while push-button control

```
local cString res (500)
res=lCstring (LGetEnv ("PATH")).cs
```

See also GETENV\$, PUTENV

MB_ *% Constants

Values used to represent the items in a message box.

Syntax	MB_ABORT%
	MB_APPLICATION%
	MB_CANCEL%
	MB_DEFBUTTON1%
	MB_DEFBUTTON2%
	MB_DEFBUTTON3%
	MB_ENTER%
	MB_ERROR%
	MB_HELP% (obsolete)
	MB_ICONASTERISK%
	MB_ICONEXCLAMATION%
	MB_ICONHAND%
	MB_ICONQUESTION%
	MB_IGNORE%
	MB_MOVEABLE%
	MB_NO%
	MB_OK%
	MB_RETRY%
	MB_SYSTEM% (obsolete)
	MB_YES%

Notes

1. These constants are used by the last parameter and by the return code of the MESSAGE% function.
2. The buttons to be displayed in the box are specified using the following constants:

MB_ENTER%	Enter button
MB_CANCEL%	Cancel button
MB_ABORT%	Abort button
MB_IGNORE%	Ignore button
MB_OK%	OK button
MB_RETRY%	Retry button

MB_YES% Yes button

MB_NO% No button

The language of the text in these buttons depends on the language set for the system.

You cannot combine more than three MB_%% Push buttons. In addition, not all combinations of 1, 2 or 3 buttons are supported.

Here are the Push button combinations supported:

First	Second	Third
MB_OK%		
MB_OK%	MB_CANCEL%	
MB_ENTER%		
MB_ENTER%	MB_CANCEL%	
MB_RETRY%	MB_CANCEL%	
MB_YES%	MB_CANCEL%	
MB_YES%	MB_NO%	
MB_YES%	MB_NO%	MB_CANCEL%
MB_ABORT%	MB_RETRY%	MB_IGNORE%

3. The MB_DEFBUTTON%% constants identify the button selected by default:

MB_DEFBUTTON1% First button selected by default

MB_DEFBUTTON2% Second button selected by default

MB_DEFBUTTON3% Third button selected by default

There can only be one default button when the message box is opened. If no MB_DEFBUTTON%%s are specified, the first button is selected.

The order of a button does not depend on the order specified when MB_%% Push button constants are added (since A+B is equal to B+A, and so on). The order actually depends on the column containing the MB_%% constant in the list of supported Push button combinations shown above. Hence, the "second" Button for an MB_CANCEL% + MB_ENTER% combination is MB_CANCEL%.

4. The icon displayed in the message box is specified using one of the MB_ICON%% constants:

MB_ICONASTERISK% Information " i " (equivalent to SPTR_ICONINFORMATION%)

MB_ICONEXCLAMATION% Exclamation mark (equivalent to SPTR_ICONWARNING%)

MB_ICONHAND% Stop sign (equivalent to SPTR_ICONERROR%)

MB_ICONQUESTION% Question mark (equivalent to SPTR_ICONQUESTION%)

There can only ever be one icon in a message box. If no MB_ICON%% is specified, the box appears without an icon.

5. The message box's behavior can be specified using three constants:

- `MB_SYSTEM%` specifies that the message box is modal for the system: all applications are suspended until the box is closed.



The `MB_SYSTEM%` constant is obsolete. It used to work with Windows 3.1.

- `MB_APPLICATION%` specifies that the message box is modal for the application: only the application that performed the `MESSAGE%` is suspended until the box is closed.

If neither of these flags is specified, the box is modal for the application.

- `MB_MOVEABLE%` displays the message box with a title bar (and a system menu) that can be used to move it. By default, if `MB_MOVEABLE%` is not specified, the box cannot be moved.

6. `MB_ERROR%` error code

This constant is the value returned by `MESSAGE%` if the function failed. It therefore returns an error detected during the processing of `MESSAGE%` due to extreme circumstances, such as insufficient memory available to display the box.

This is the only `MB_*` constant that cannot be used as a parameter of `MESSAGE%`.

7. They are declared internally as follows:

```
CONST MB_ENTER%      1
CONST MB_CANCEL%     2
CONST MB_ABORT%      4
CONST MB_IGNORE%     8
CONST MB_OK%         16
CONST MB_RETRY%      32
CONST MB_YES%        64
CONST MB_NO%         128
CONST MB_ERROR%      256
CONST MB_ICONASTERISK% 1024
CONST MB_ICONEXCLAMATION% 2048
CONST MB_ICONHAND%   4096
CONST MB_ICONQUESTION% 8192
CONST MB_SYSTEM%     16384
CONST MB_APPLICATION 32768
CONST MB_MOVEABLE%   65536
CONST MB_DEFBUTTON1% 131072
CONST MB_DEFBUTTON2% 262144
CONST MB_DEFBUTTON3% 524288
```

See also

`MESSAGE%`

MDIARRANGEWINDOWS Instruction

Arranges MDI child windows in the parent window area:

Syntax **MDIARRANGEWINDOWS** *window-handle, type*

Parameters	<i>window-handle</i>	POINTER	I	handle of the MDI parent window
	<i>type</i>	INT(4)	I	requested arrangement type

Notes

1. This instruction only applies to MDIWINDOW type windows.
2. *type* must be one of the MDI_ARRANGE*% constants:

MDI_ARRANGE_CASCADE%	cascade arrangement
MDI_ARRANGE_HORZTILE%	horizontal tile arrangement
MDI_ARRANGE_VERTTILE%	vertical tile arrangement

See also MDI_ARRANGE*%

MDI_ *% Constants

Identify an item on an MDI window's system menu or organization menu.

Syntax

MDI_CANCEL%
MDI_CASCADE%
MDI_CLOSE%
MDI_MAXIMIZE%
MDI_MINIMIZE%
MDI_MORE%
MDI_MOVE%
MDI_NEXT%
MDI_RESTORE%
MDI_SELECT%
MDI_SIZE%
MDI_TILE%

Notes

1. The titles of the options represented by these constants can be retrieved or modified using GETMDISTR\$ and SETMDISTR
2. The MDI window system menu items are identified by:

Constants	English title
MDI_RESTORE%	Restore
MDI_NEXT%	Next
MDI_MOVE%	Move
MDI_SIZE%	Size
MDI_MINIMIZE%	Miimize
MDI_MAXIMIZE%	Maximize
MDI_CLOSE%	Close

3. The MDI window organization menu items are identified by:

Constants	English title
MDI_CASCADE%	Cascade
MDI_TILE%	Tile
MDI_MORE%	More
MDI_SELECT%	Select

MDI_CANCEL%	Cancel
-------------	--------

4. The item corresponding to MDI_MORE% is displayed in the menu after the names of the MDI windows opened when the number of windows exceeds nine. MDI_SELECT% and MDI_CANCEL% correspond to the titles of the buttons in the dialog box displayed when the MDI_MORE% option is selected.
5. The system menu titles cannot be modified. None of the MDI_ *% constants associated with the system menu have any effect when they are used with SETMDISTR
6. They are declared internally as follows:
CONST MDI_RESTORE% 1
CONST MDI_NEXT% 2
CONST MDI_MOVE% 3
CONST MDI_SIZE% 4
CONST MDI_MINIMIZE% 5
CONST MDI_MAXIMIZE% 6
CONST MDI_CLOSE% 7
CONST MDI_CASCADE% 8
CONST MDI_TILE% 9
CONST MDI_MORE% 10
CONST MDI_SELECT% 11
CONST MDI_CANCEL% 12

See also GETMDISTR\$, SETMDISTR

MDI_ARRANGE*% Constants

Constants to arrange MDI child windows in the parent window area.

Syntax **MDI_ARRANGE_CASCADE%**
 MDI_ARRANGE_HORZTILE%
 MDI_ARRANGE_VERTTILE%

Notes

1. These constants are used as parameters of the MDIARRANGEWINDOWS instruction.
2. This is their meaning:
 - MDI_ARRANGE_CASCADE% cascade arrangement
 - MDI_ARRANGE_HORZTILE% horizontal tile arrangement
 - MDI_ARRANGE_VERTTILE% vertical tile arrangement

See also MDIARRANGEWINDOWS

MDISHOWHIDDENWINDOWS Instruction


Allows to show or not the hidden child windows MDI list in the windows menu.

Syntax	MDISHOWHIDDENWINDOWS <i>showMode</i>		
Parameter	<i>showMode</i>	INT(1) I	display mode of the child windows MDI list
See also	MDIARRANGEWINDOWS, MDI_ARRANGE*%		

MESSAGE% Function

Opens a message box consisting of Push buttons, an optional icon and a Help button. The function returns the value of a pressed Push button.

Syntax **MESSAGE%** (*window-handle*, *title-string*, *message-string*, *help-item*, *box-style*)

Parameters	<i>window-handle</i>	POINTER	I	handle of the window that displays the message
	<i>title-string</i>	CSTRING	I	title of the message box
	<i>message-string</i>	CSTRING	I	message displayed in the box
	<i>help-item</i>	INTEGER	I	help item called if the end-user presses the Help button.
				 This parameter is not operational.
	<i>box-style</i>	INT(4)	I	box style

Return value INT(4)

value **meaning**

MB_%% button pressed by the end-user

Notes

1. *window-handle* is the handle of the window that performed the MESSAGE%, i.e. usually SELF%.
2. The parameter *help-item* is not operational because the MB_HELP% constant is obsolete.
3. *box-style* is a combination of MB_%% constants used to specify the Push buttons displayed and, if applicable, an icon and *Help* button.

If MB_MOVEABLE% is specified in *box-style*, a title bar and system menu are added. In this case, when the box is closed through its system menu (double-click, [Alt]+[F4], or selection of the Close item), the MESSAGE% function returns MB_CANCEL%.

Example

```
IF MESSAGE%(SELF%, " Title ", " This is a message ", 0, \
MB_RETRY% + MB_CANCEL% + MB_ICONHAND% + MB_MOVEABLE%) \
= MB_RETRY%
MESSAGE " MESSAGE% box closed ", " by pressing Retry "
ELSE
MESSAGE " MESSAGE% box closed ", \
" by pressing Cancel or through its system menu "
ENDIF
```

See also

MB_%% constants

NSHELP library

NCL language: MESSAGE, ASK2%, ASK3%

MISCERROR% Function

Returns the error code of the last NSMisc library function or instruction used.

Syntax **MISCERROR%**

Return value INT(4)

Notes

- 1.** If no error occurred, MISCERROR% returns zero.
- 2.** The F_* and T_* file handling functions and instructions (except F_ERROR%) affect the returned value by MISCERROR%.
- 3.** The various error codes returned by MISCERROR% are shown in Appendix B of this manual.

MKDIR Instruction

Creates a directory.

Syntax **MKDIR** *directory-name*

Parameters *directory-name* CSTRING I name of the directory to create

Notes

1. This instruction is equivalent to the DOS MD or MKDIR command.
2. If no path has been specified in *directory-name*, the directory will be created as a sub-directory of the current directory.

Example 1

```
MKDIR " TEST " ; Creates a directory TEST located in  
; the current directory
```

Example 2

```
MOVE " C:\NATSTAR\SAMPLES " TO S$  
MKDIR S$
```

See also CHDIR, RMDIR, GETDIR\$, GETDISK%

MODE_*% Constants

Application start-up modes.

Syntax

MODE_INVISIBLE%
MODE_MAXIMIZE%
MODE_MINIMIZE%
MODE_NORMAL%

Notes

1. These constants indicate how EXECPROG will start up the application it executes. Do not confuse with the COMMAND_*% constants in the NSWIN or the NSPM library.
2. There are three start-up modes used by the Windows environments:
 - MODE_INVISIBLE% starts up the application and hides it (it is up to the application to perform a SHOW).
 - MODE_MAXIMIZE% starts up the application maximized.
 - MODE_MINIMIZE% starts up the application minimized.

MODE_INVISIBLE% is not implemented for applications generated using NatStar (or NS-DK).

The effect of the other start-up modes depends on the system used to run the application.

- MODE_NORMAL% starts up the application normally, without forcing a state, a position or a size. The application started up may be a Windows or DOS executable or a .COM, .BAT or .PIF file.
- MODE_NOAUTOCLOSE%, MODE_USE_POSITION%, MODE_PM_SCREEN%, MODE_FULL_SCREEN% and MODE_VIO_SCREEN% are not supported under Windows.

The way in which a non-Windows executable is to be started up must be specified in the corresponding .PIF file. If this type of program has been started up in a window, the behavior of the window once the program stops must also be specified in the .PIF file.

- MODE_NORMAL% starts up the application normally, without forcing a state, a position or a size. The application started up must be a Presentation Manager application.

3. They are declared internally as follows:

CONST MODE_NORMAL%	0
CONST MODE_INVISIBLE%	1
CONST MODE_MAXIMIZE%	2
CONST MODE_MINIMIZE%	4

See also

EXECPROG

NEXTCONTROL Instruction

Returns the next control in the search started by the previous FIRSTCONTROL and NEXTCONTROL instructions.

Syntax **NEXTCONTROL** *control-var*

Parameters *control-var* CONTROL I/O next control in the window

Notes

1. When there are no more controls in the window, the MISCERROR% function returns a non-zero error code.
2. The order in which the controls are retrieved by NEXTCONTROL is not related to the physical layout of the window or the focus sequence.
3. The control variable used with each NEXTCONTROL must be the one used by FIRSTCONTROL since the control it contains is used as a reference for obtaining the next control.

Example

```
; Fill a table of controls with the
; names of all the controls in a window
; and display the names of these controls.
GLOBAL CONTROL CTRL[100]
LOCAL I%, CONTROL TEMPCTRL
FIRSTCONTROL SELF%, TEMPCTRL
MOVE 0 TO I%
WHILE (MISCERROR% = 0) AND (I% < 100)
MESSAGE " Control number " && I%, CONTROLNAME$(TEMPCTRL)
MOVE TEMPCTRL TO @CTRL[I%]
MOVE I%+1 TO I%
; Warning!
; Do not use NEXTCONTROL CTRL[I%] directly
; Cf. Note 3
NEXTCONTROL TEMPCTRL
ENDWHILE
MESSAGE " There are ", I% && " controls "
```

See also FIRSTCONTROL, CONTROLNAME\$, WINDOWFROMCONTROL%

NSEXISTDIR% Function

This function lets you verify the existence of a directory on a disk.

Syntax **NSEXISTDIR%** (*file*)

Parameter *file* CSTRING I name of the directory whose existence is to be verified

Return value INT(2) TRUE% if the directory exists FALSE% if not-existent

See also NSEXISTFILE%

NSEXISTFILE% Function

This function lets you verify the existence of a file on a disk.

Syntax **NSEXISTFILE%** (*file*)

Parameter *file* CSTRING I name of the file needing existence verification

Return value INT(2)
 TRUE% if the file exists
 FALSE% if not in existence

See also NSEXISTDIR%

NSGETMENUWINDOW function

Returns the handle of the window containing the menu.

Syntax **NSGETMENUWINDOW** (*wnd*)

Parameter *wnd* POINTER I handle of the window (self%)

Return value POINTER

Notes

1. If it is a MDI child window, NSGETMENUWINDOW retrieves the handle of the MDI parent window.
2. Allows the partition of one Image List only attached to the MDI parent window for MDI child windows. Example :
WIL_GETCOUNT%(NSGetMenuWindow(Self%), ...) where Self% is the handle of a MDI child window or another kind of window.

See also WIL_DELETEPICTURES, WIL_DESTROY, WIL_DRAWPICTURE,
WIL_FREEHASH, WIL_GETSIZE, WIL_SETCOUNT,
WIL_APPENDFROMBMP%, WIL_FINDORADDONEPICTURE%,
WIL_FINDPICTURE%, WIL_GETCOUNT%, WIL_GETPICTURESCOUNT%

NSMEMCOMP% function

Compares two memory blocks.

Syntax **NSMEMCOMP%** (*pMem1*, *pMem2*, *size%*)

Parameters	<i>pMem1</i>	POINTER	I	Pointer on a memory block
	<i>pMem2</i>	POINTER	I	Pointer on another memory block
	<i>size%</i>	INTEGER	I	Number of bytes to compare

Return value INT(1)

Note

1. Returns 0 if the two memory blocks are identical, otherwise returns the difference between the two bytes.

See also NSMEMPOS%

NSMEMPOS% Function

Returns the position of a string in another string.

Syntax **NSMEMPOS%** (*pMem1*, *pMem2*, *size%*, *size2%*, *last%*)

Parameters	<i>pMem1</i>	POINTER	I	Pointer on the searched memory block
	<i>pMem2</i>	POINTER	I	Pointer on the memory block in which we carry out the search.
	<i>size%</i>	INTEGER	I	Number of bytes in the memory block searched.
	<i>size2%</i>	INTEGER		Number of bytes in the memory block in which we execute the search.
	<i>last%</i>	INTEGER	I	Direction of the search.

Return value INTEGER

Notes

1. Indicate in the *last%* parameter the FALSE% value to execute the search from the beginning and TRUE% to execute the search from the end.
2. The return value is the offset (from the start of the second block), where begins the sequence of bytes searched or -1 if the search failed.
3. NSMEMPOS% has the same fonctionnalities of the POS% but only for the blocks of bytes which don't end by 0.

See also POS% (NCL), NSMEMCOMP%

SETZORDER instruction

Changes the Z-Order of a window. In other words, positioning the window in the foreground or in the background or in front of another window (even if it is no longer the focus window).

If ZOK_BEHIND% constant is used, the behind% parameter must be the window behind which the window in question (referenced by win%) will be placed.

Syntax	SETZORDER win%, kind%, behind%			
Parameters	win%	POINTER	I	window handle for which the Z-ORDER is to be changed
	kind%	INTEGER	I	constant specifying the desired type of arrangement
	behind%	POINTER	I	handle of the second window if the parameter kind% equals ZOK_BEHIND%

Example

```
; puts the referenced window by hWnd% to the foreground
SETZORDER hWnd%, ZOK_TOP%, 0
```

Note

1. The ZOK_TOPMOST% parameter is recognized as ZOK_ TOP% and the ZOK_NOTOPMOST% parameter has no effect.

See also ZOK_*

ZOK_*% constant

These constants are passed as a SETZORDER instruction parameter.

ZOK_TOP%	indicates putting the window in the foreground
ZOK_BOTTOM%	indicates putting the window in the background
ZOK_BEHIND%	indicates putting the window behind another window
ZOK_TOPMOST%	indicates keeping the window in the foreground even when no longer in focus

Syntax

ZOK_TOP%
ZOK_TOPMOST%
ZOK_NOTOPMOST%
ZOK_BOTTOM%
ZOK_BEHIND%

Notes

1. ZOK_TOPMOST% parameter is recognized as ZOK_TOP % and ZOK_NOTOPMOST % parameter has no effect.
2. Declared internally as follows:

CONST	ZOK_TOP%	0
CONST	ZOK_TOPMOST%	1
CONST	ZOK_NOTOPMOST%	2
CONST	ZOK_BOTTOM%	3
CONST	ZOK_BEHIND%	4

See also SETZORDER

DEV_ADJUSTX function

This function transforms a value to its equivalent value within the new data system. This transformation takes in account the developer workstation resolution and the window's horizontal zooming factors during execution.

Syntax **DEV_ADJUSTX** (*win%*, *x%*)

Parameters	<i>win%</i>	POINTER	I	handle of the window to ajust.
	<i>x%</i>	INTEGER	I	value to be adjusted

Returned value integer value after adjustment

Example

```
PB_TEST.X = DEV_ADJUSTX (self%,50)
PB_TEST.WIDTH = DEV_ADJUSTX (self%,100)
```

In this example, push-button sizes and positions in X will always be correct, regardless of the window resolution and size differences between the development and the execution.

See also DEV_ADJUSTY

DEV_ADJUSTY function

This function transforms a value to its equivalent value within the new data system. This transformation takes in account the developer workstation resolution and the window's vertical zooming factors during execution.

Syntax **DEV_ADJUSTY** (*win%*, *y%*)

Parameters	<i>win%</i>	POINTER	I	handle of the window to adjust
	<i>y%</i>	INTEGER	I	value to be adjusted

Returned value INTEGER value after adjustment.

Example

```
PB_TEST.Y = DEV_ADJUSTY (self%,40)
PB_TEST.HEIGHT = DEV_ADJUSTX (self%,24)
```

In this example, push-button sizes and positions in Y will always be correct, regardless of the window resolution and size differences between the development and the execution.

See also DEV_ADJUSTX

PARAMCOUNT% Function

Returns the number of parameters passed on the command line when the application was started up.

Syntax **PARAMCOUNT%**

Return value INTEGER

Notes

1. If PARAMCOUNT% returns 0, the application was started up without any parameters.
2. Let's suppose we have an application named APP.EXE. If this application is started up from the command line in the Windows File/Run menu:
APP NAT SYSTEM
then PARAMCOUNT% will return 2,
3. The application's name and start-up parameters can be obtained with PARAMSTR\$.

Example

```
; Display the name of the application
; together with any parameters
MESSAGE " Name of the application ", PARAMSTR$(0)
MOVE 1 TO I%
WHILE I% <= PARAMCOUNT%
MESSAGE " Parameter " && I%, PARAMSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

See also PARAMSTR\$, EXECPROG

PARAMSTR\$ Function

Returns one of the application's start-up parameters.

Syntax **PARAMSTR\$** (*index*)

Parameter *index* INTEGER I parameter index

Return value CSTRING

Notes

1. PARAMSTR\$(0) returns the name of the application.
The index of the first start-up parameter is 1. The total number of start-up parameters is equal to PARAMCOUNT%.
2. Let's suppose we have an application named APP.EXE. If this application is started up from the command line in the Windows File/Run menu:
APP NAT SYSTEM
Then
 - PARAMSTR\$(0) will return " APP ",
 - PARAMSTR\$(1) will return " NAT ",
 - PARAMSTR\$(2) will return " SYSTEMS ".

Example

```
; Display the name of the application
; together with any parameters
MESSAGE " Name of the application ", PARAMSTR$(0)
MOVE 1 TO I%
WHILE I% <= PARAMCOUNT%
MESSAGE " Parameter " && I%, PARAMSTR$(I%)
MOVE I%+1 TO I%
ENDWHILE
```

See also PARAMCOUNT%, EXECPROG

QUERYEDITINPLACE function

Enables the contents of the parameter *info* to be recovered via the pointer *wnd*.

Syntax	QUERYEDITINPLACE (<i>wnd</i> , <i>info</i>)			
Parameters	<i>wnd</i>	POINTER	I	pointer returned by the function BEGINEDITINPLACE
	<i>info</i>	SEGMENT	I/O	segment SEG_EIPINFO
Returned value	INT(1)			

Comment

1. The segment SEG_EIPINFO defined in the file NSMISC.NCL:

```
SEGMENT SEG_EIPINFO ; Edit In Place Information
    INTEGER VER    ; Version = 0
    CONTROL CTRL   ;
    INTEGER X      ;     INTEGER Y
    INTEGER W
    INTEGER H      ;
    INTEGER KIND   ; Kind of the editor control (DI_*)
    CSTRING TEXT   ; Init edit text (STATICTEXT, ENTRYFIELD, COMBBX)
ENDSEGMENT ; SEG_EIPINFO

; CTRL contains the address of the control in which we want to insert a control
    INTEGER X      ; abscissa in pixels of the editing rectangle
    INTEGER Y      ; ordinate in pixels of the editing rectangle
    INTEGER W      ; width of the editing rectangle
    INTEGER H      ; height of the editing rectangle
;KIND the type of the temporary control which we want to display uses DI_*
;constants
;TEXT the initialisation text of the temporary control
```

2. If the control EDITINPLACE is no longer valid, the instance is terminated. The function QUERYEDITINPLACE may then cause a crash. The elements Kind and Text of the segment SEG_EIPINFO will not be filled in.

Example

```
Local SEG_EIPINFO INFO
LOCAL H%
local INT Retour%(1)
Insert at END "User0 du controle " to LISTBOX1
IF param12% <> -1
    IF param12% = 0
        Insert at END "On a appuyé sur la touche Escape " to LISTBOX1
    ELSE
        Insert at END "PARAM12$ "&&PARAM1$ to LISTBOX1
        Insert at END "PARAM12% "&&PARAM12% to LISTBOX1
    ENDIF
ENDIF
h% = @INFO
; getdata% returns the handle of the temporary control
; then QUERYEDITINPLACE fills the SEG_EIPINFO segment
retour% = QUERYEDITINPLACE (GETDATA%, SEG_EIPINFO(h%))
INSERT AT END 'X =' && INFO.X TO LISTBOX1
INSERT AT END 'Y =' && INFO.Y TO LISTBOX1
```

RANDOM% Function

Returns an integer random number.

Syntax **RANDOM%** (*maximum-integer*)

Parameter *maximum-integer* INTEGER I maximum value of the integer

Return value INTEGER

Notes

1. The integer returned lies between 0 and (maximum-integer - 1)
2. Maximum must lie between 1 and 32767.
3. RANDOM% can only be used after initializing the random number generator with RANDOMIZE.

Example

```
; Example that simulates a dice being thrown
; until a 6 is obtained.
; The value of a dice can only lie between 1 and 6.
; Here, RANDOM%(6) can only return integer values
; between 0 and 5.
RANDOMIZE
REPEAT
MOVE RANDOM%(6)+1 TO I%
MESSAGE " Throwing the dice ", I%
UNTIL I% = 6
MESSAGE " You win! ", " It landed on six. "
```

See also RANDOM#, RANDOMIZE

RANDOM# Function

Returns a real random number.

Syntax **RANDOM#**

Return value NUM(8)

Notes

1. The real number returned lies between 0 (inclusive) and 1 (exclusive).
2. RANDOM# can only be used after initializing the random number generator with RANDOMIZE.

See also RANDOM%, RANDOMIZE

RANDOMIZE Instruction

Initializes the random number generator.

Syntax **RANDOMIZE**

Note You must first call RANDOMIZE before you can use RANDOM% or RANDOM# to obtain a random number.

Example

```
; See example for RANDOM%
```

See also RANDOM%, RANDOM#

REMOVE_CALLBACK Instruction

Removes the callback added by INSTALL_CALLBACK.

Syntax **REMOVE_CALLBACK** *hCallback*

Parameter *hCallback* POINTER I handle envoyé par INSTALL_CALLBACK

See also INSTALL_CALLBACK, CALL_PREVCALLBACK

RMDIR Instruction

Deletes a directory.

Syntax **RMDIR** *directory-name*

Parameter *directory-name* CSTRING I name of directory to delete

Notes

1. This instruction is equivalent to the DOS RD or RMDIR command.
2. If the directory's path is not specified in *directory-name*, this instruction will delete the *directory-name* sub-directory below the current directory.

Example 1

```
RMDIR " TEST " ; Deletes the directory, TEST, located in
; the current directory
```

Example 2

```
MOVE " \NATSTAR\SAMPLES " TO S$
RMDIR S$
```

See also CHDIR, MKDIR, GETDIR\$, GETDISK%, GETDISKS%

SEG_SIZEORPOS Segment

This segment is used jointly with the ADJUST_%% constant and ADJUSTSIZEORPOS event. This event informs a window that a user wants to modify its size or position and enables it to change position and size values.

Definition **segment SEG_SIZEORPOS**
 int posX (4)
 int posY (4)
 int Width (4)
 int Height (4)
 endSegment

See also ADJUST_%%

SELFFROMCONTROL% Function

Returns the *self%* of a control.

Syntax **SELFFROMCONTROL%** (*ctrl*)

Parameter *ctrl* CONTROL I name of a control

Return value POINTER

SETBITMAPCONTROLSDEFAULTS Instruction

Allows you to modify the default behavior of bitmap controls. The value passed using this verb are a sum of the following NSMISC flags : DEFBMPTRANSPARENT% and DEFBMPBUTTONIZED%.

Syntax **SETBITMAPCONTROLSDEFAULTS** *default%*

Parameter *default%* INT I DEFBMP* flag

Notes

1. The definition of the flags are :

- DEFBMPTRANSPARENT%
Activate transparency by default. Also used with PictureButton and PrintButton NatStar (or NS-DK) custom controls.
- DEFBMPBUTTONIZED%
Activates the automatic drawing of borders and simulated Pressed and Disabled PushButton bitmap controls with only one bitmap (Released).

See also GETBITMAPCONTROLSDEFAULTS%

SETCHECKMODE Instruction

Enables or disables " Check Mode ", which prevents the current control in a window from losing the focus.

Syntax **SETCHECKMODE** *window-handle, mode*

Parameters	<i>window-handle</i>	POINTER	I	window handle
	<i>mode</i>	INT(1)	I	enables/disables check mode

Notes

1. This mode is enabled when " mode " is set to TRUE%.
2. The current control receives the CHECK event instead of the LOSEFOCUS event when it loses the focus. A RETURN 1 in the code for the CHECK event prevents the control from losing the focus.
3. FALSE% disables "Check Mode", after which the LOSEFOCUS event will be received by all controls that lose the focus.
4. A Push button in 'No Checking' mode does not send the CHECK event when the focus changes. The same applies to the Push button associated with the [Esc] key.

Example

INIT event for the current window

```
; Enable " Check Mode "
SETCHECKMODE SELF%, TRUE%
```

CHECK event for the EF_NAME Entry field

```
IF PARAM1% = CHK_LOSEFOCUS%
; focus is lost
IF EF_NAME = " "
BEEP
MESSAGE " Error " , " Name is a required field "
; prevent the focus from being lost
RETURN 1
ELSE
RETURN 0
ENDIF
ENDIF
```

See also GETCHECKMODE%, CHK_%% constants

SETCURRENTCHARSET instruction

Allows to change the active code page (charset).

Syntax **SETCURRENTCHARSET** *chs%*

Parameter *chs%* INTEGER I CHS_ *% constant

Note

1. Use this instruction cautiously and only if any application's window is open.

See also CHS_ *%, GETCURRENTCHARSET%, GETHOSTCHARSET%,
TRANSLATECHARS, TRANSLATECHARS2%

SETDECIMALSEPARATOR Instruction

Defines the character to be used as the decimal separator.

Syntax	SETDECIMALSEPARATOR <i>string</i>		
Parameter	<i>string</i>	CSTRING	I separator
Notes	<ol style="list-style-type: none">1. This format is only used within the application: it does not modify the default set in the Control Panel, which remains effective for any other applications that have not set a specific separator.2. The decimal separator defined in this way is used by the STRING\$ and CURRENCY\$ functions and by Entry field and CBE controls with " Number " format.3. SETDECIMALSEPARATOR " ! " uses the separator set in the Control Panel.		

Example 1

```
SETDECIMALSEPARATOR " . "           ; US and UK decimal separator
MOVE STRING$ (0.2, " 0.0 ") TO S$    ; S$ is set to " 0.2 "
```

Example 2

```
SETDECIMALSEPARATOR " , "           ; European decimal separator
MOVE STRING$ (0.2, " 0,0 ") TO S$    ; S$ is set to " 0,2 "
```

See also STRING\$, CURRENCY\$, GETDECIMALSEPARATOR\$,
 GETTHOUSANDSSEPARATOR\$, SETTHOUSANDSSEPARATOR

SETESCAPECONTROL Instruction

Dynamically sets the control in a dialog box that will be activated when the end-user presses the [Esc] key.

Syntax **SETESCAPECONTROL** (*window-handle*, *control*)

Parameters	<i>window-handle</i>	POINTER	I	dialog box handle
	<i>control</i>	CONTROL	I	control to activate

Note The associated control is usually a Push button that receives the EXECUTED event when the end-user presses [Esc].

Example

```
IF ISDISABLED% (PB_CANCEL)
SETESCAPECONTROL SELF%,PB_EXIT
ELSE
SETESCAPECONTROL SELF%, PB_CANCEL
ENDIF
```

See also GETESCAPECONTROL, GETRETURNCONTROL, SETRETURNCONTROL

SETMDISTR Instruction

Modifies the title of an item on an MDI window's system menu or organization menu (Cascade, Title).

Syntax **SETMDISTR** *menu-item*, *title-string*

Parameters	<i>menu-item</i>	INT(1)	I	menu item to modify
	<i>title-string</i>	CSTRING	I	new title

Notes

1. The required menu item is specified using an MDI_**% constant.
2. A character within the title will be considered as a keyboard accelerator for the menu item if it is preceded by the tilde (~) character. It will be underlined when the menu is displayed.
3. A menu title can only be modified by SETMDISTR before the window containing the menu is displayed. Hence, this instruction should be coded in the window's INIT event.

Example

```
; French translation of the " Cascade " title
SETMDISTR MDI_CASCADE%, " ~Chevauchement "
```

See also GETMDISTR\$, MDI_**%

SETRETURNCONTROL Instruction

Dynamically sets the control in a dialog box that will be activated when the end-user presses the [Enter] key.

Syntax **SETRETURNCONTROL** (*window-handle*, *control*)

Parameters	<i>window-handle</i>	POINTER	I	dialog box handle
	<i>control</i>	CONTROL	I	control to activate

Note The associated control is usually a Push button that receives the EXECUTED event when the end-user presses [Enter].

Example

```
IF ISDISABLED% (PB_OK)
SETRETURNCONTROL SELF%,PB_HELP
ELSE
SETRETURNCONTROL SELF%, PB_OK
ENDIF
```

See also GETRETURNCONTROL, GETESCAPECONTROL, SETESCAPECONTROL

SETTABCONTROL Instruction

Sets the control in a Dialog window that will get the focus when the end-user presses the [Tab] key and a specified control has the focus.

Syntax **SETTABCONTROL** (*source-ctrl*, *target-ctrl*)

Parameters	<i>source-ctrl</i>	CONTROL	I	control that will lose the focus
	<i>target-ctrl</i>	CONTROL	I	control that will get the focus

Note This instruction is the dynamic equivalent of the Tab field in the Info box for source-ctrl.

Example

```
; If the person is married and female,  
; the "Maiden name" field in the input window  
; will get the focus when the end-user presses Tab in the "Name" field  
IF CK_MARRIED% = CHECKED%  
IF RB_SEX = 2  
SETTABCONTROL EF_NAME, EF_MAIDENNAME  
SETTABCONTROL EF_MAIDENNAME, EF_FIRSTNAME  
ELSE  
SETTABCONTROL EF_NAME, EF_FIRSTNAME  
ENDIF  
ENDIF
```

See also GETTABCONTROL, FOCUSCONTROL

SETTHOUSANDSSEPARATOR Instruction

Defines the character to be used as the thousands separator.

Syntax **SETTHOUSANDSSEPARATOR** *string*

Parameters *string* CSTRING I separator

Notes

1. This format is only used within the application: it does not modify the default set in the Windows Control Panel, which remains effective for any other applications that have not set a specific separator.
2. The thousands separator defined in this way is used by the STRING\$ and CURRENCY\$ functions and by Entry field and CBE controls with " Number " format.
3. When using the SETTHOUSANDSSEPARATOR instruction with STRING%, the thousands separator must be indicated in the format string passed to STRING% in order to be taken into account. The format string uses the comma (,) to indicate the thousands separator, and the comma refers to the separator defined by SETTHOUSANDSSEPARATOR.
4. SETTHOUSANDSSEPARATOR " ! " uses the separator set in the Control Panel.

Example 1

```
SETTHOUSANDSSEPARATOR " " ; Space between thousands  
MOVE STRING$ (2300, " #,##0") TO S$ ; S$ is set to "2 300"
```

Example 2

```
SETTHOUSANDSSEPARATOR "" ; No separator  
MOVE STRING$ (2300, "###0") TO S$ ; S$ is set to "2300"
```

See also

STRING\$, CURRENCY\$, GETTHOUSANDSSEPARATOR\$,
GETDECIMALSEPARATOR\$, SETDECIMALSEPARATOR

SHAREDMEMALLOCERROR% Function

Verifies whether or not shared memory has already been allocated (by another process for example).

Syntax **SHAREDMEMALLOCERROR%**

Return value INT(4)

- 0 if allocation has been successful
- 1 if allocation failed (specified shared memory already allocated or error).

Example

```
NEW SEGSTR, SHMEMPTR%, 'SHARED'
IF SHMEMPTR% = 0
    ; pointer null was returned
    MESSAGE 'Allocation error', \
        'An error has occurred when using NEW'
ELSE
    ; pointer returned, but has the memory been allocated ?

IF SHAREDMEMALLOCERROR% <> 0
    ; memory already allocated and initialized
ELSE
    ; not allocated so initialize here mémoire
...
ENDIF
ENDIF
```

SHORTENPATHNAME Instruction

Replaces a long file name passed by modifiable parameter by its short equivalent. For example, "My test document.txt" becomes "MYDOC~1.TXT").

Syntax **SHORTENPATHNAME** *pathname*\$

Parameter *pathname*\$ CSTRING I/O name of the file

Note

1. If the name of the file includes a path, the directories names will be replaced too by their short equivalents.

STRING\$ Function

Formats a real number and returns the resulting string.

Syntax **STRING\$** (*real-number*, *format-string*)

Parameters	<i>real-number</i>	NUM(8)	I	real number to format
	<i>format-string</i>	CSTRING	I	format to apply

Return value CSTRING

Notes

1. The string passed in *format-string* must always be in American format. However, the string returned by STRING\$ is based on the separators defined in the Control Panel which can be modified by the SETDECIMALSEPARATOR and SETTHOUSANDSSEPARATOR instructions. This ensures that applications are fully portable, regardless of the separators defined in the Control Panel: only the displayed result varies according to the machine used to run the application.
2. The formats accepted are those shown in Appendix A of this manual.
3. Use the ESTRING\$ function if the type of the real number that you want to format is NUM(10).
4. The DATE\$ and TIME\$ functions in the NSDATE library can also be used to format dates and times.

Example

```
; In this example (see note 1), we will assume that
; a space is used as the thousands separator and that a
; period is used as the decimal separator.

MOVE STRING$(1234.567, " 0 ")      TO S$ ; S$ = " 1235 "
MOVE STRING$(1234.567, " 0.0 ")    TO S$ ; S$ = " 1234.6 "
MOVE STRING$(1234.567, " # ##0.00 ") TO S$ ; S$ = " 1 234.57 "
MOVE STRING$(1234.567, " 0.0E+00 ") TO S$ ; S$ = " 1.2E+03 "
MOVE STRING$(0.12345, " 0% ")      TO S$ ; S$ = " 12% "
MOVE STRING$(0, " yyyy-m-d ") TO S$ ; S$ = " 1900-1-1 "
MOVE STRING$(0, " hh:mm:ss ") TO S$ ; S$ = " 00:00:00 "
```

See also ESTRING\$, SETDECIMALSEPARATOR, SETTHOUSANDSSEPARATOR, CURRENCY\$, DATE\$, TIME\$ (NSDATE library)

T_APPEND% Function

Opens an ASCII text file in write mode with the pointer at the end of the file.

Syntax **T_APPEND%** (*filename*)

Parameters *filename* CSTRING I name of the file to open

Return value INT(4)

Notes

1. This function allows you to append text directly to the end of the file starting from the next T_WRITE instruction.
2. Using brackets within the filename string enables environment parameters to be used:

; Opens the SAMPLE.TXT file located in the directory

; specified by the NS-LST environment parameter

MOVE T_APPEND%(" (NS-LST)SAMPLE.TXT ") TO H%

See also T_OPEN%, T_CLOSE

F_* Functions and Instructions

NCL language: LOAD and SAVE Instructions

T_CLOSE Instruction

Closes an ASCII text file.

Syntax **T_CLOSE** *file-handle*

Parameters *file-handle* INT(4) I handle of the file to close

Note The various error codes returned by T_ERROR% are shown in Appendix B of this manual.

Example

```
; Write to the text file " TEST.TXT "
MOVE T_CREATE%(" TEST.TXT ") TO H%
IF T_ERROR%<>0
MESSAGE " T_CREATE% error ", " Cannot create the file "
EXIT
ENDIF
T_WRITELN H%, " Hello "
T_WRITELN H%, " Bye "
T_CLOSE H%
```

See also T_APPEND%, T_CREATE%, T_EOF%, T_EOL%, T_ERROR%, T_OPEN%,
T_READ%, T_READ#, T_READ\$, T_READLN%, T_READLN#, T_READLN\$,
T_WRITE, T_WRITELN

F_* Functions and Instructions

NCL language: LOAD and SAVE Instructions

T_CREATE% Function

Creates an ASCII text file in write mode and returns its handle.

Syntax **T_CREATE%** (*filename*)

Parameters *filename* CSTRING I name of the file to create

Return value INT(4)

Notes

1. T_CREATE% only creates the text file if it does not exist; otherwise T_CREATE% opens the file and deletes its contents.
2. You can subsequently use the handle returned to access the file with the other T_*% read/write functions and instructions.
3. Using brackets within the filename string enables environment parameters to be used:
 ; Creates the SAMPLE.TXT file in the directory
 ; specified by the NS-LST environment parameter
 MOVE T_CREATE%(" (NS-LST)SAMPLE.TXT ") TO H%

Example

```
MOVE T_CREATE%(" TEST.TXT ") TO H%
IF T_ERROR%<>0
MESSAGE " T_CREATE% error ", " Cannot create the file "
EXIT
ENDIF
T_WRITELN H%, " Hello "
T_WRITELN H%, " Bye "
T_CLOSE H%
```

See also T_APPEND%, T_CLOSE, T_EOF%, T_EOL%, T_ERROR%,
 T_OPEN%, T_READ*, T_WRITE*

F_* Functions and Instructions

NCL language : LOAD and SAVE Instructions

T_EOF% Function

Indicates if the current pointer position is at the end of the ASCII text file.

Syntax	T_EOF% (<i>file-handle</i>)		
Parameters	<i>file-handle</i>	INT(4)	I handle of the file
Return value	INT(1)		
	value	meaning	
	TRUE%	The pointer is at the end of the file.	
	FALSE%	The pointer is not at the end of the file.	

Example

```
; Read the text file " TEST.TXT "  
MOVE T_OPEN%(" TEST.TXT ") TO H%  
IF T_ERROR%<>0  
MESSAGE " T_OPEN% error ", " Cannot open the file "  
EXIT  
ENDIF  
MOVE 0 TO I%  
WHILE NOT T_EOF%(H%)  
MOVE I%+1 TO I%  
MESSAGE " Line number " && I%, T_READLN$(H%)  
ENDWHILE  
T_CLOSE H%
```

See also T_EOL%

T_EOL% Function

Indicates if the current pointer position is at the end of the line in a text file.

Syntax **T_EOL%** (*file-handle*)

Parameters *file-handle* INT(4) I file handle

Return value INT(1)

value	meaning
-------	---------

TRUE%	The pointer is at the end of the line.
-------	--

FALSE%	The pointer is not at the end of the line.
--------	--

See also T_EOF%

T_ERROR% Function

Returns the last error code generated by handling text files.

Syntax **T_ERROR%**

Return value INT(4)

value	meaning
0	No error
not null	Error in the last T_%% function or instruction used. The various error codes returned are listed in Appendix B of this manual.

Note

1. We recommend that you always call T_ERROR% after calling a T_%% function or instruction to check that it was executed correctly.

Example

```
; *** See example for T_OPEN%
```

See also T_APPEND%, T_CLOSE, T_CREATE%, T_EOF%, T_EOL%, T_OPEN%,
T_READ*, T_WRITE*

T_OPEN% Function

Opens an ASCII text file in read mode and returns its handle.

Syntax **T_OPEN%** (*filename*)

Parameters *filename* CSTRING I name of the file to open

Return value INT(4)

Notes

1. A text file is made up of a series of printable characters. CR-LF characters (Carriage Return - Line Feed) can be inserted into the file to split it into lines of characters similar to the lines displayed when the file is read with a text editor.
2. You can subsequently use the handle returned to access the file with the other T_.*% read functions or instructions.
3. If the file does not exist, T_OPEN% generates an error. This error must be retrieved using T_ERROR%.
4. Using brackets within the filename string enables environment parameters to be used:
; Opens the SAMPLE.TXT file located in the directory
; specified by the NS-LST environment parameter
MOVE T_OPEN%(" (NS-LST)SAMPLE.TXT ") TO H%

Example

```
MOVE T_OPEN%(" TEST.TXT ") TO H%
IF T_ERROR%<>0
MESSAGE " T_OPEN% error ", " Cannot open the file "
EXIT
ENDIF
MOVE 0 TO I%
WHILE NOT T_EOF%(H%)
MOVE I%+1 TO I%
MESSAGE " Line number " && I%, T_READLN$(H%)
ENDWHILE
T_CLOSE H%
```

See also

T_APPEND%, T_CLOSE, T_CREATE%, T_EOF%, T_EOL%, T_ERROR%,
T_READ*, T_WRITE*

T_READ% Function

Returns a word from the current line in an ASCII text file and converts it to an integer.

Syntax **T_READ%** (*file-handle*)
Parameters *file-handle* INT(4) I file handle
Return value INT(4)

Notes

1. A word comprises a series of characters preceded by a start of line marker or space character and followed by a space character or end of line marker. The word is read starting from the current cursor position.
2. The value 0 (zero) is returned if the word cannot be converted to an integer.
3. After the read operation, the cursor is positioned directly after the word. The next T_READ% will read the next word on the line or the first word on the next line if the end of the current line has been reached.
4. T_READ# is used to convert to a real number.
5. We recommend that you always call T_ERROR% after calling a T_**% function or instruction to check that it was executed correctly.

Example

```
; Let's assume that the file has been opened and that its  
; handle has been copied into the H% variable  
; Each line in the file should contain  
; the following information:  
; age    salary    last name    first name  
; e.g. 24   2000   Anderson   Sarah  
; Read the entire file  
; and display its information  
WHILE NOT T_EOF%(H%)  
MESSAGE "Information", "Age:" && T_READ%(H%) && \  
"Salary:" && T_READ#(H%) && \  
"Last name:" && T_READLN$(H%)  
ENDWHILE  
T_CLOSE H%
```

See also T_READ#, T_READ\$, T_READLN%, T_READLN#, T_READLN\$,
 T_READLNEX\$

T_READ# Function

Returns a word from the current line in an ASCII text file and converts it to a real number.

Syntax **T_READ#** (*file-handle*)

Parameters *file-handle* INT(4) I file handle

Return value NUM(8)

Notes

1. A word comprises a series of consecutive characters preceded by a start of line marker or space character and followed by a space character or end of line marker. The word is read starting from the current cursor position.
2. The value 0 (zero) is returned if the first word cannot be converted to a real number.
3. After the read operation, the cursor is positioned directly after the word. The next T_READ# will read the next word on the line or the first word on the next line if the end of the current line has been reached.
4. T_READ% is used to convert to an integer.
5. We recommend that you always call T_ERROR% after calling a T_.*% function or instruction to check that it was executed correctly.

Example

```
; See example for T_READ%
```

See also T_READ%, T_READ\$, T_READLN%, T_READLN#, T_READLN\$,
T_READLNEX\$

T_READ\$ Function

Returns the contents of the current line in an ASCII text file.

Syntax	T_READ\$ (<i>file-handle</i>)		
Parameter	<i>file-handle</i>	INT(4)	I file handle
Return value	CSTRING(255)		

Notes

1. The line is read starting from the current pointer position, which may not be at the beginning of a line if the T_READ% or T_READ# functions have just been executed.
2. The string returned contains up to 255 characters, excluding end of line characters. T_READ\$ must be executed several times in succession to read a line containing more than 255 characters. The new T_READLNEX\$ function allows you to use more easily strings containing more than 255 characters.
3. Even if the whole line has been read, T_READ\$ does not automatically position the read pointer at the beginning of the next line. For lines containing over 255 characters, you must check if the end of the line has been reached before using T_READLN\$ to read and skip to the next line. The new T_READLNEX\$ function allows you to use more easily strings containing more than 255 characters.
4. We recommend that you always call T_ERROR% after calling a T_*\$ function or instruction to check that it was executed correctly.

Example

```
; See example for T_READ%
```

See also T_READ%, T_READ#, T_READLN%, T_READLN#, T_READLN\$, T_READLNEX\$

T_READLN% Function

Returns a word from the current line in an ASCII text file, converts the word to an integer, and positions the read pointer at the beginning of the next line.

Syntax **T_READLN%** (*file-handle*)

Parameter *file-handle* INT(4) I file handle

Return value INT(4)

Notes

1. The only difference between T_READ% and T_READLN% is that T_READLN% positions the read pointer at the beginning of the next line immediately after the read operation.
2. See notes 1 and 2 for T_READ%.
3. We recommend that you always call T_ERROR% after calling a T_.*% function or instruction to check that it was executed correctly.

See also T_READ%, T_READ#, T_READ\$, T_READLN#, T_READLN\$, T_READLNEX\$

T_READLN# Function

Returns a word from the current line in an ASCII file, converts it to a real number, and positions the read pointer at the beginning of the next line.

Syntax **T_READLN#** (*file-handle*)

Parameter *file-handle* INT(4) I file handle

Return value NUM(8)

Notes

1. The only difference between T_READ# and T_READLN# is that T_READLN# positions the read pointer at the beginning of the next line immediately after the read operation.
2. See notes 1 and 2 for T_READ#.
3. We recommend that you always call T_ERROR% after calling a T_**% function or instruction to check that it was executed correctly.

See also T_READ%, T_READ\$, T_READ#, T_READLN%, T_READLN\$, T_READLNEX\$

T_READLN\$ Function

Returns the contents of the current line in an ASCII text file and positions the read pointer at the beginning of the next line.

Syntax **T_READLN\$** (*file-handle*)

Parameter *file-handle* INT(4) I file handle

Return value CSTRING

Notes

1. The only difference between T_READ\$ and T_READLN\$ is that T_READLN\$ positions the read pointer at the beginning of the next line immediately after the read operation.
2. If your string contains more than 255 characters use the new T_READLNEX\$ function instead.
3. See notes for T_READ\$.
4. We recommend that you always call T_ERROR% after calling a T_ *% function or instruction to check that it was executed correctly.

See also T_READ%, T_READ\$, T_READ#, T_READLN%, T_READLN#, T_READLNEX\$

T_READLNEX\$ Function

Returns the contents of the current line in an ASCII text file and positions the read pointer at the beginning of the next line.

Syntax **T_READLNEX\$** (*file-handle*)

Parameter *file-handle* INT(4) I file handle

Return value DYNSTR

Notes

1. The only difference between T_READLN\$ and T_READLNEX\$ is that T_READLNEX\$ allows you to return a string containing more than 255 characters.
2. See notes for T_READ\$.
3. We recommend that you always call T_ERROR% after calling a T_*% function or instruction to check that it was executed correctly.

See also T_READLN\$, T_READ%, T_READ\$, T_READ#, T_READLN%, T_READLN#

T_WRITE Instruction

Writes a specified expression to an ASCII text file.

Syntax **T_WRITE** *file-handle, expression*

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>expression</i>	CSTRING	I	expression to write

Notes

1. The expression is automatically converted to a character string.
2. After writing to the file, the pointer will be positioned after the last character.
3. T_WRITE returns an error if the file was opened in read-only mode by T_OPEN%.
4. If your expression to write contains more than 255 characters, use the T_WRITEEX instruction instead.
5. We recommend that you always call T_ERROR% after calling a T_%% function or instruction to check that it was executed correctly.

Example

```
; Let's assume that the file has been opened and that its  
; handle has been copied into the H% variable  
T_WRITE H%, " Hello "  
T_WRITE H%, " Good-bye "  
; The current line in the file contains the string  
; " HelloGood-bye "  
T_CLOSE H%
```

See also T_WRITEEX, T_WRITELN, T_WRITELNEX

T_WRITEEX Instruction

Writes a specified expression to an ASCII text file.

Syntax **T_WRITEEX** *file-handle, expression*

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>expression</i>	DYNSTR	I	expression to write

Notes

1. The only difference between T_WRITE and T_WRITEEX is that T_WRITEEX allows you to write an expression of more than 255 characters.
2. The expression is automatically converted to a character string.
3. After writing to the file, the pointer will be positioned after the last character.
4. T_WRITEEX returns an error if the file was opened in read-only mode by T_OPEN%.
5. We recommend that you always call T_ERROR% after calling a T_ *% function or instruction to check that it was executed correctly.

See also T_WRITE, T_WRITELN, T_WRITELNEX

T_WRITELN Instruction

Writes a specified expression to an ASCII text file and includes a line feed character at the end of the expression.

Syntax `T_WRITELN file-handle, expression`

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>expression</i>	CSTRING	I	expression to write

Notes

1. The expression is automatically converted to a character string.
2. After writing to the file, the pointer will be positioned at the beginning of the next line.
3. T_WRITELN returns an error if the file was opened in read-only mode by T_OPEN%.
4. If your expression to write contains more than 255 characters, use the T_WRITELNEX instruction instead.
5. We recommend that you always call T_ERROR% after calling a T_*% function or instruction to check that it was executed correctly.

Example

```
; Let's assume that the file has been opened and that its  
; handle has been copied into the H% variable  
T_WRITELN H%, " Hello "  
T_WRITELN H%, " Good-bye "  
; The last line in the file contains the string " Good-bye  
; and the previous line contains the string " Hello "  
T_CLOSE H%
```

See also T_WRITELNEX, T_WRITE, T_WRITEEX

T_WRITELNEX Instruction

Writes a specified expression to an ASCII text file and includes a line feed character at the end of the expression.

Syntax **T_WRITELNEX** *file-handle, expression*

Parameters	<i>file-handle</i>	INT(4)	I	file handle
	<i>expression</i>	DYNSTR	I	expression to write

Notes

1. The only difference between T_WRITELN and T_WRITELNEX is that T_WRITELNEX allows you to write an expression of more than 255 characters.
2. The expression is automatically converted to a character string.
3. After writing to the file, the pointer will be positioned at the beginning of the next line.
4. T_WRITELNEX returns an error if the file was opened in read-only mode by T_OPEN%.
5. We recommend that you always call T_ERROR% after calling a T_**% function or instruction to check that it was executed correctly.

Example

```
; Let's assume that the file has been opened and that its
; handle has been copied into the H% variable
T_WRITELNEX H%, " Hello Hello Hello Hello Hello Hello Hello Hello Hello
Hello Hello "
T_WRITELNEX H%, " Good-bye "
; The last line in the file contains the string " Good-bye
; and the previous line contains the string " Hello Hello Hello Hello
;Hello ... "
T_CLOSE H%
```

See also T_WRITELN, T_WRITE, T_WRITEEX

TERMINATEEDITINPLACE instruction

Closes the temporary editing field EDITINPLACE. Equivalent to the user pressing the [Enter] key.

Syntax	TERMINATEEDITINPLACE				<i>wnd, wantnewvalue%</i>
Parameters	<i>wnd</i>	POINTER	I	pointer returned by the function BEGINEDITINPLACE	
	<i>wantnewvalue%</i>	INTEGER	I/O	boolean indicator	

Comment

1. If *wantnewvalue%* is TRUE%, the USERx event specified is sent, just as though editing had been terminated normally (accepted).

Example

```
local POINTER HDATA%
local ret%
local retour%

NEW SEG_EIPINFO, HDATA%
IF HDATA%=0
    MESSAGE "ERREUR", "Handle de segment NULL"
    EXIT
ENDIF

FILL HDATA%, SIZEOF SEG_EIPINFO, 0

MOVE 0 TO SEG_EIPINFO(HDATA%).VER
;The container control is the EntryField EF_TEST
@SEG_EIPINFO(HDATA%).CTRL=EF_TEST
MOVE 1 TO SEG_EIPINFO(HDATA%).X
MOVE 150 TO SEG_EIPINFO(HDATA%).Y
MOVE 120 TO SEG_EIPINFO(HDATA%).W
MOVE 40 TO SEG_EIPINFO(HDATA%).H
;The temporary control is an EditComboBox DI_COMBBOXE%
MOVE DI_COMBBOXE% TO SEG_EIPINFO(HDATA%).kind
MOVE "Initial Value" to SEG_EIPINFO(HDATA%).text

ret%= BEGINEDITINPLACE (SEG_EIPINFO(HDATA%), 0)
```

```
SETDATA ret%

move ret% to EF
IF ret%=0
    MESSAGE "ERREUR","Traitement terminé"
ELSE
    ; once the editin place has been called you can access the temporary Ctrl
    INSERT AT END 'Choix1' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix2' to SEG_EIPINFO(HDATA%).CTRL
    INSERT AT END 'Choix3' to SEG_EIPINFO(HDATA%).CTRL
    ;displays the temporary Control and select all the contents of the control
    ;if you want to place the cursor at a certain position in the temporary
    ;control
    ;specify the xy coordinates like ACTIVATEEDITINPLACE ret%,50,1
    ACTIVATEEDITINPLACE ret%,50,0
    ; Terminates the control after 1 second
    WAIT 1000
    TERMINATEEDITINPLACE ret%, TRUE%
ENDIF
DISPOSE HDATA%
```


TRANSLATECHARS Instruction

Translates the contents of a buffer with a specified size from one character set to another.

Syntax **TRANSLATECHARS** *buffer-address*, *buffer-size*, *source-set*, *target-set*

Parameters	<i>buffer-address</i>	POINTER	I	address of a buffer containing the data to convert
	<i>buffer-size</i>	INTEGER	I	buffer size in bytes
	<i>source-set</i>	INTEGER	I	source character set
	<i>target-set</i>	INTEGER	I	target character set

Notes

1. The character sets must be specified using CHS_%% constants.
2. This instruction is particularly useful when developing multi-target applications designed to run under different environments.

Example

```
LOCAL S$(100)
MOVE ENTRY TO S$
; Convert the string into ASCII if the current character set
; is ANSI
IF GETCURRENTCHARSET% = CHS_ANSI%
TRANSLATECHARS @S$, length S$, CHS_ANSI%, CHS_ASCII%
ENDIF
```

See also GETHOSTCHARSET%, GETCURRENTCHARSET%, CHS_%% constants

TRANSLATECHARS2% Function

Translates the contents of an entry buffer in an exit buffer and in a different character set.

Syntax `TRANSLATECHARS2% (buf_in, len_in%, chs_in%, buf_out, size_out%, chs_out%)`

Parameters	<i>buf_in</i>	POINTER	I	address of a buffer containing the data to convert
	<i>len_in%</i>	INTEGER	I	length of the entry buffer
	<i>chs_in%</i>	INTEGER	I	source character set
	<i>buf_out</i>	POINTER	I	address of the exit buffer
	<i>size_out%</i>	INTEGER	I	maximal size in bytes of the exit buffer
	<i>chs_out%</i>	INTEGER	I	target character set

Return value INTEGER

Notes

1. The character sets must be specified using CHS_%% constants.
2. The return value corresponds to the number of bytes in exit.
3. This function must be use instead of TRANSLATECHARS when one of the two code pages is CHS_UTF8. Actually, in this case, the entry string has not always the same size than the exit string and TRANSLATECHARS doesn't allow that.
4. To set the "0 binary" termination character of the exit character set, it is necessary that the *len_in%* parameter is equal to the length of the entry string + 1. If not, the exit character set may have not a termination character.

See also TRANSLATECHARS, TRANSLATECHS\$, GETHOSTCHARSET%, GETCURRENTCHARSET%, SETCURRENTCHARSET, CHS_%%

TRANSLATECHS\$ Function

Translates the C character set in a different character set.

Syntax `TRANSLATECHS% (S$, chs_in%, chs_out%)`

Parameters	<i>S\$</i>	CSTRING	I	Chaîne de caractères C
	<i>chs_in%</i>	INTEGER	I	Jeu de caractères source
	<i>chs_out%</i>	INTEGER	I	Jeu de caractères cible

Return value CSTRING

Notes

1. The character sets must be specified using CHS_%% constants.
2. The C string is limited to 255 characters.

See also TRANSLATECHARS2% TRANSLATECHARS , GETHOSTCHARSET%,
GETCURRENTCHARSET%, SETCURRENTCHARSET, CHS_%%

WCCSEND% Function

Allows you to send Windows messages requiring two parameters (*wParam*, *pParam*) to the Windows 32-bit control implemented in NatStar and NS-DK.



This function should therefore be used when the Windows message indicated in the MSDN documentation requires two parameters.

e.g. : The DTM_SETRANGE message for the Date and Time picker control contains two parameters and therefore must be used with WCCSEND%.

MSDN definition:

DTM_SETRANGE

wParam = (WPARAM) flags;

lParam = (LPARAM) lpSysTimeArray;

Syntax WCCSEND% (*Pwcc*, *Msg*, *Wparam*, *lParam*)

Parameters	<i>Pwcc</i>	POINTER	I	pointer on the Windows control.
	<i>Msg</i>	INTEGER	I	Windows message to be sent
	<i>Wparam</i>	INTEGER	I	specifies the value of the message parameter
	<i>lParam</i>	INT(4)	I	specifies the value of the message parameter

Value returned INT(4)

Example

```
local Systemtime lsystemtime
local ret%
fill @lsystemtime, sizeof Systemtime, 0
lsystemtime.wyear = 2001
lsystemtime.wmonth = 4
lsystemtime.wdayofweek = 4
lsystemtime.wDay = 12
lsystemtime.wHour = 18
lsystemtime.wMinute = 10
lsystemtime.wSecond = 09
lsystemtime.wMilliseconds = 0
; assigns a value to the Date and Time Picker control
ret% = WCCSEND%(int DATETIMEPICKER.ctrl,DTM_SETSYSTEMTIME% \ ,GDT_VALID%,
@lSystemTime)
```

See also WCCSEND0%, WCCSEND1%, WCCSENDPTR%, WCCSENDSTR%

WCCSEND0% Function

Allows you to send Windows messages which do not require any parameters to the Windows 32-bit control implemented in NatStar and NS-DK.



This function should therefore be used when the Windows message indicated in the MSDN documentation does not require any parameters.

e.g.

The DTM_GETMCFONT message for the Date and Time picker control does not contain any parameters and therefore must be used with WCCSEND0%.

MSDN definition :

DTM_GETMCFONT

wParam = 0;

lParam = 0;

Syntax **WCCSEND0%** (*Pwcc*, *Msg*)

Parameters	<i>Pwcc</i>	POINTER	I	pointer on the Windows control.
	<i>Msg</i>	INTEGER	I	Windows message to be sent

Value returned INT(4)

Example

```
; read the value of a track bar control  
position%=wccsend0% (int wcctb.ctrl, TBM_GETPOS%)
```

WCCSEND1% Function

Allows you to send Windows messages requiring one parameter (wParam) to the Windows 32-bit control implemented in NatStar and NS-DK.



This function should therefore be used when the Windows message indicated in the MSDN documentation requires one parameter.

e.g.

The DTM_GETMCCOLOR message for the Date and Time picker control contains one parameter and therefore must be used with WCCSEND0%.

MSDN definition:

```
DTM_GETMCCOLOR wParam = (WPARAM)(INT)iColor;  
lParam = 0;
```

Syntax	WCCSEND1% (Pwcc, Msg, wParam)			
Parameters	<i>Pwcc</i>	POINTER	I	pointer on the Windows control.
	<i>Msg</i>	INTEGER	I	Windows message to be sent
	<i>wParam</i>	INTEGER	I	specifies the value of the message parameter
Value returned	INT(4)			

Example

```
local color%  
; imports the background colour for a Date Time Picker control  
; in RGB code  
color% = WCCSEND1% (int DATETIMEPICKER.ctrl, \  
                    DTM_GETMCCOLOR%, \  
                    MCSC_BACKGROUND%)
```

WCCSENDPTR% Function

Allows you to send Windows messages requiring a pointer type parameter to the Windows 32-bit control implemented in NatStar and NS-DK.

Syntax **WCCSENDPTR%** (*Pwcc*, *Msg*, *lParam*)

Parameters	<i>Pwcc</i>	POINTER	I	pointer on the Windows control.
	<i>Msg</i>	INTEGER	I	Windows message to be sent
	<i>lParam</i>	INTEGER	I	specifies the value of the message parameter

Value returned INT(4)

Example

```
local Systemtime lsystemtime
local ret%
; returns the value of a Date Time Picker control
fill @lsystemtime, sizeof Systemtime, 0
ret% = WCCSENDPTR%(int DATETIMEPICKER.ctrl,DTM_GETSYSTEMTIME%,\
@lSystemTime)
if ret% = GDT_VALID%
    efda = lsystemtime.wyear
    efdm = lsystemtime.wmonth
    efdjs = lsystemtime.wdayofweek
    efdj = lsystemtime.wDay
    efdh = lsystemtime.wHour
    efdmi = lsystemtime.wMinute
    efds = lsystemtime.wSecond
    efdms = lsystemtime.wMilliseconds
endif
```

WCCSENDSTR% Function

Allows you to send Windows messages requiring a character string type parameter to the Windows 32-bit control implemented in NatStar and NS-DK.

Syntax **WCCSENDSTR%** (*Pwcc*, *Msg*, *lParam*)

Parameters	<i>Pwcc</i>	POINTER	I	pointer on the Windows control.
	<i>Msg</i>	INTEGER	I	Windows message to be sent
	<i>lParam</i>	INTEGER	I	specifies the value of the message parameter

Value returned INT(4)

Example

```
; opens an .AVI file in an Animation control  
local ret%  
ret% = WCCSENDSTR%(INT WCCANIM.CTRL, ACM_OPEN%, \ "c:\temp\f.avi")
```


NSSORT Instruction

Lets you sort arrays automatically.

Syntax `NSSORT pdata, Count%, CompFunc, SwapInstr`

Parameters	<i>pData</i>	POINTER	I	address of the pointer array
	<i>count%</i>	INTEGER	I	number of elements in the array
	<i>CompFunc</i>	POINTER	I	pointer to the comparison routine
	<i>SwapInstr</i>	POINTER	I	pointer to the Swap routine

Comments

1. The *pData* parameter must be an integer array (default) or of any type.
2. The *CompFunc* parameter equals 0 if you want to use the standard comparison routine.
3. The *CompSwap* parameter equals 0 if you want to use the standard swap routine.

Prototype of the comparison function

The *pData* parameter corresponds to the address of the array to be sorted, the array will be accessed via the *i1%* and *i2%* indices.

The function should return:

- a negative value if element *i1%* < element *i2%*
- 0 if they are equal.
- a positive value if element *i1%* > element *i2%*

```
FUNCTION CompFunc%(POINTER pdata, INTEGER i1%, INTEGER i2%) RETURN
INTEGER
```

Prototype of the Swap instruction (inversion)

This function should invert the *i1%* and *i2* elements.

```
INSTRUCTION SwapInstr POINTER pdata, INTEGER i1%, INTEGER i2%
```

Example 1

Sorting a array of integers

```
const MAX_ITEMS 1000
```

```

const MAX_ITEMS 1000
; array of numbers
global Numbers% [1000]

; fill the array
RANDOMIZE
for u% = 0 to MAX_ITEMS -1
numbers%[u%] = random% (10000)
endFor
; sort the array
NSSORT @numbers%, MAX_ITEMS,0,0

```

Example 2*Sorting a array of client segments*

```

segment sCLIENT
cstring name$ (31)
int AGE%
EndSegment

; array of clients
global sCLIENT clients [1000]

; fill the array
local u%

RANDOMIZE
for u% = 0 to MAX_ITEMS -1
clients[u%].name$ = "Dupont" && string$ (u%,"0000")
clients[u%].age% = random% (10000)
endFor

; Sort the array (by age)
; call NSSORT routine with custom routine addresses (CompClient,
; swapClient)
nssort @clients, MAX_ITEMS, @CompClient%, @SwapClient

; Display the array
local u%

for u% = 0 to MAX_ITEMS -1
insert at end "Client" && u% && "=" && clients[u%].name$ && "(age=" & \
clients[u%].age% & ")" to LISTBOX
endFor

;*****
;
;          Callback Routines
;*****

;
; CompClient%()
; callback function for age comparaison
; return a negative value if element i1% < element i2%
; return 0 if they are identical
; return strictly positive value if element i1% > element i2%
;
function CompClient% (sClient @pData[], integer i1%, integer i2%) Return

```

```
Integer
return pdata[i1%].age% - pdata[i2%].age%
endFunction ; CompClient%

;
; SwapClient%()
;
; callBack function for Client Swap
;
instruction SwapClient sClient @pdata[], integer i1%, integer i2%
local sClient tmpCli

tmpCli = pdata[i1%]
pdata [i1%] = pdata [i2%]
pdata [i2%] = tmpcli
EndInstruction ; SwapClient
```

WINDOWFROMCONTROL% Function

Returns the handle of the window containing a specified control.

Syntax **WINDOWFROMCONTROL%** (*control*)

Parameters *control* CONTROL I control

Return value POINTER

Note A variable of type CONTROL is stored over 6 bytes, 4 of which contain the window-handle. WINDOWFROMCONTROL% returns these 4 bytes.

Example

```
GLOBAL CONTROL CTRL
; Suppose we have a window named SCREEN. Its handle is H%
; and it contains a control named ENTRY1
MOVE SCREEN(H%).ENTRY1 TO @CTRL
; WINDOWFROMCONTROL% retrieves the window's handle (H%)
MESSAGE " Handle of the window containing CTRL ", \
WINDOWFROMCONTROL%(CTRL)
; WINDOWNAME$ retrieves the window's name
MESSAGE " Name of the window containing CTRL ", \
WINDOWNAME$(WINDOWFROMCONTROL%(CTRL))
; CONTROLNAME$ finds ENTRY1
MESSAGE " Name of the control stored in CTRL ", \
CONTROLNAME$(CTRL)
```

See also CONTROLNAME\$, WINDOWNAME\$

WINDOWNAME\$ Function

Returns the name of a window.

Syntax `WINDOWNAME$ (window-handle)`

Parameter *window-handle* POINTER I window handle

Return value CSTRING

Notes

1. The name of a window corresponds to the contents of the "Name" field in the Window Info box. This name is used by CALL or OPEN to identify the window.
2. **IMPORTANT.** For WINDOWNAME\$ to work in generated mode, the /NAMES option **MUST** be used with the generator module or the "Symbols info" option **MUST** be checked in the Generation box.

Example 1

```
MESSAGE " The name of the current window is: ", \
WINDOWNAME$(SELF%)
```

Example 2

```
MESSAGE " The name of the main window is: ", \
WINDOWNAME$(MAINWINDOW%)
```

See also CONTROLNAME\$, WINDOWFROMCONTROL%

GETCONTROLTEMPLATE Instruction

This instruction allows events and dynamic parameters to be sent (via the *tpl* parameter) to the generic model of the Custom Control instance indicated in the parameter *ctrl*.

Syntax	GETCONTROLTEMPLATE	<i>ctrl, tpl</i>		
Parameters	<i>ctrl</i>	CONTROL	I	name of the Custom Control instance
	<i>tpl</i>	CONTROL	O	variable

Example

```
; PPB1 is a Picture Button control
LOCAL CONTROL CT

GetControlTemplate PPB1, CT
Message ControlName$(CT), ControlName$(PPB1)
;displays a dialog box with "PICTBUT" (the generic anme of the Custom
;Control) in the title and "PPB1" in the box (name of the control). PPB1
;should be replaced by a CONTROL type variable (but you must initialize it)
```

GETCONTROLID% Function

The function GETCONTROLID% returns the number or index of the control passed as a parameter.

Syntax **GETCONTROLID%** *ctrl*

Parameter *ctrl* CONTROL I name of control

Value returned INTEGER

Comment

1. This function supplements the function WINDOWFROMCONTROL%. The function WINDOWFROMCONTROL% returns the window handle containing the control passed as a parameter.

Example

```
; PB04 = name of the Push Button control
MESSAGE "Number of Push Button control", GETCONTROLID%(PB04)
; returns 102
; EF01 = name of the Entry Field control
MESSAGE "Number of Entry Field control", GETCONTROLID%(EF01)
; returns 101
```

MAKECONTROL Instruction

Allows the creation of a control by initializing it via the control index.

Syntax **MAKECONTROL** *ctrl, parentwnd, ctrlid%*

Parameters	<i>ctrl</i>	CONTROL	I/O	name of control
	<i>parentwnd</i>	POINTER	I	parent window
	<i>ctrlid%</i>	INTEGER	I	control identifier ID

Example

```
MakeControl C, WindowFromControl%(X), GetControlID%(X)
; is the same as @C = X.
```


SELECTBYVALUE% function

This is used to search text or change the selection status of one or several lines of a control.

Syntax **SELECTBYVALUE%** (*ctl*, *mode%*, *start%*, *text\$*)

Parameters	<i>ctl</i>	CONTROL	I	name of the control
	<i>mode%</i>	INTEGER	I	combination of SBVM_*% constants
	<i>start%</i>	INTEGER	I	starting position of the text being searched
	<i>text\$</i>	CSTRING	I	searched text; to be written between quotation marks

Comments

1. SBVM_*% constants indicate the search options and the action to be carried out
2. SBVM_STRP*% and SBVM_EXPTP% constants can only be used with List Boxes or Combo Boxes.
3. To be able to use this function on a Custom Control or a template, it should accordingly implement the SELECTED, LINECOUNT and GETVALUE events (especially when (PARAM2% = 0) AND (PARAM1% <> DEFRET%) for GETVALUE).
4. The possible actions are as follows (only one action possible at a time):

SBVM_FIND% returns the index of the first element found.

SBVM_SEL% selects the corresponding lines.

SBVM_UNSEL% unselects the corresponding lines.

SBVM_CHANGE% reverses the selection of the corresponding lines.

The comparison options are as follows (the final three options are excluded):

SBVM_NOCASE% case insensitive.

SBVM_PART%	the string being searched for should be present in the line.
SBVM_EXACT%	the string being searched for should be identical to the line.
SBVM_REGEX%	the string being searched for is treated with COMPILEREGREXPR% and compared to the lines of the control with EXECUTEREGREXPR%.

The SBVM_ONE% option shows an action on the first corresponding line, while SBVM_ALL% (forbidden with SBVM_FIND%) shows that the action must be applied to all corresponding lines.

The SBVM_FWD% option shows an ascending search (forward in line numbers) while SBVM_BACK% shows a descending search (back in line numbers).

The following options can also be added for List Boxes and Combo Boxes:

SBVM_STRPPA% ignores the text of the presentation attributes (" {...}" at the beginning of the column).

SBVM_STRPBM% ignores the bitmap and icon handles.

SBVM_STRP0W% ignores the text and presentation attributes of columns of width 0.

SBVM_STRPTP% ignores cursors on text (presentation attribute "{P[Y]}").

SBVM_EXPTP% takes cursor-referred text into account as if it was present instead of the cursor.

Example

```
SELECTBYVALUE LB01, SBVM_SEL% AND SBVM_NOCASE%, "football"
```

See also

GETCONTROLTEMPLATE, GETCONTROLID%, MAKECONTROL

SETFOCUSCONTROLATTRIBUTES% Function

Determines the attributes set on a control taking the focus.

 This function is especially useful for an icon.

Syntax **SETFOCUSCONTROLATTRIBUTES%** (*focusctrlattrs*)

Parameter *focusctrlattrs* SEGMENT I/O segment FOCUSCTRLATTRS

Value returned INT(1)

Comments

1. The segment FOCUSCTRLATTRS defined in the file NSMISC.NCL:

```
SEGMENT FOCUSCTRLATTRS
  INT      Version      ; SIZEOF FOCUSCTRLATTRS (pour version du segment)
  INT      KindMask     ; a CKM_.*% constant
  CSTRING  Font         ; name of the control's font taking the focus
  INT      FontSize(2)  ; size of the control's font taking the focus
  INT      FontSels(2)  ; type of the font (bold,italic,...)
  INT      MinWidth(2)  ; minimal width in pixels
  INT      MinHeight(2) ; minimal height in pixels (for MLE control only)
ENDSEGMENT
```

2. The constants CKM_.*% definable in the field *KindMask* determine the type of control:

CKM_MLE%	MLE control.
CKM_EF%	Entry Field control.
CKM_FEF%	Filtered Entry Field control (no characters >= 255).
CKM_CBE%	Combo Box with Entry field control.
CKM_FCBE%	Filtered Combo Box with Entry Field control (no characters >= 255).
CKM_CB%	Combo Box control.
CKM_PB%	Push Button control.
CKM_CK%	Check box control.
CKM_RB%	Radio Button control.
CKM_LB%	List Box control.

CKM_DEF% MLE+EF+CBE control.

3. This function may be called several times with different values for the *KindMask* field to set the attributes specific to different types of controls.

Example

```
Local FOCUSCTRLATTRS theSeg
theSeg.Version = sizeof FOCUSCTRLATTRS
theSeg.KindMask = CKM_DEF% ; use any of the CKM_* constants
theSeg.Font="Simplified Arabic"
theSeg.FontSize = 14
;GFS_ITALIC% 1
;GFS_UNDERSCORE% 2 GFS_STRIKEOUT% 4 GFS_BOLD% 8
;GFS_OUTLINE% 16
theSeg.FontSels= GFS_OUTLINE%; use any of the GFS_* constants
;bold/italic/... of the focus font
theSeg.MinWidth=14 ; minimum width in pixels
theSeg.MinHeight= 14 ; minimum height in pixels (for MLE only)
; since we are in 3.00 there is no need to define a return variable
; for the function nor brackets for the parameters
SETFOCUSCONTROLATTRIBUTES% theSeg
```

NSSEARCHPATHS\$ function

Finds the file *name\$* in the directories indicated by the parameter *paths\$*. The main advantage of this function is that it allows the 255 character size limit of the search path to be avoided.

Syntax	NSSEARCHPATHS\$ (<i>name\$</i> , <i>paths\$</i>)			
Parameters	<i>name\$</i>	CSTRING	I	name of file being searched for
	<i>paths\$</i>	CSTRING	I	names of directories
Value returned	CSTRING (255) returns the path and name of the file being searched for. This string is limited to 255 characters.			

Comments

1. The names of the directories in *paths\$* are separated by semi-colons. The quotes (") surrounding the directory names are ignored.
2. The *path\$* parameter can be a CSTRING of more than 255 characters.
3. It is possible to specify one or more environment variables, themselves containing one or more directory names, by putting each variable name in parenthesis (also separated by semi-colons).
4. "(=EXE)" allows specification of the directory of the program currently being executed.
5. The PATH parameter should be composed as follows:

```
h% =@STIN$
s$="(<=&h%&")"
test$ = NSSEARCHPATHS$("MyFile.txt", s$)
```
6. The syntax (=Pointer) allows the address of a C string of whatever length containing a list of directories to be indicated.

Example 1

```
Local test$
Local S$
Local STIN$(1000)
local pointer h%
INSERT AT END 'Search for nsdesign' TO LISTBOX1
test$ = NSSEARCHPATHS$("nsdesign.exe", "(path)")
```

```
INSERT AT END test$ TO LISTBOX1
INSERT AT END 'Search for your information.cov' TO LISTBOX1
test$ = NSSEARCHPATHS$("Your information.cov", "C:\Documents and Settings\All
Users\Documents")
INSERT AT END test$ TO LISTBOX1
INSERT AT END 'STIN$ < 255 Result <=255' TO LISTBOX1
STIN$ = "D:\tests\This is a new folder to test the long names which goes beyond
the famous limit of 255 characters of Nat System products use nsSearchPaths and
verify the result\test\bmp\image\New Folder\New Folder2\New Folder3"
test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "The file is in" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ < 255 Result > 255 returns an empty string' TO LISTBOX1
test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "The file is in" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ > 255 Result <= 255' TO LISTBOX1
STIN$ = "(PATH);.;(=EXE);(NS-BIN);(NS-BCK);(NS-TXT);(NS-BMP);"
STIN$ = STIN$&"D:\tests\This is a new folder to test the long names which goes
beyond the famous limit of 255 characters of Nat System products use
nsSearchPaths and verify the result\test\bmp\image\New Folder\New Folder2\New
Folder3"

test$ = NSSEARCHPATHS$("toto.txt", STIN$)
INSERT AT END "The file is in" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1

INSERT AT END 'STIN$ > 255 Result <= 255 using (=h%) works fine' TO LISTBOX1
STIN$ = "(PATH);.;(=EXE);(NS-BIN);(NS-BCK);(NS-TXT);(NS-BMP);"
STIN$ = STIN$&"D:\tests\This is a new folder to test the long names which goes
beyond the famous limit of 255 characters of Nat System products use
nsSearchPaths and verify the result\test\bmp\image\New Folder\New Folder2\New
Folder3"

h% =@STIN$
s$="(&h%)"
test$ = NSSEARCHPATHS$("toto.txt", s$)
INSERT AT END "The file is in" TO LISTBOX1
INSERT AT END test$ TO LISTBOX1
INSERT AT END "Length of test$ = "&& LENGTH test$ TO LISTBOX1
```

Example 2

```
Test$=nsSearchPaths$("myfile.bmp", "(=EXE);(PATH);.(NS-BIN)")  
;Research the myfile.bmp in directories of the current program, then in PATH  
;directories, and finally in the directories indicated by NS_BIN  
;environment variable.
```

nsStrCmp% function

Compares and sorts two characters strings.

Syntax **nsStrCmp%(S1\$, S2\$, IgnoreCase%)**

Parameters	<i>S1\$</i>	CSTRING	I	First characters strings
	<i>S2\$</i>	CSTRING	I	Second characters strings
	<i>IgnoreCase%</i>	INT(2)	I	Boolean allowing you to make the distinction (or not) between capital letters and small letters.

Return value INTEGER

Notes

1. The function sorts alphabetically and when the parameter *IgnoreCase%* is set in TRUE%, the capital letters have priority on the small ones.
2. Example of a list sorted by nsStrCmp% :

Original list	List sorted by the nsStrCmp% function
Papy	le Gendre
Pépé	le gendre
Père	Le Guenn
Pêche	le Guenn
Peccadille	Léa
Pierre	Legendre
Portrait	Leguenn
public	Papy
Legendre	Peccadille
le gendre	Pêche
le Gendre	Pépé
leguenn	Père
le Guenn	Pierre
Le Guenn	Portrait
Léa	public

GETMENUITEMPICTURE% function

Returns the index of the image associated with a menu item or a figure between \$F000 and \$F014 if the image is one of the images predefined in the runtime environment or \$FFFF in the situation where an image associated with menu items is missing.

Syntax **GETMENUITEMPICTURE%** (*wnd*, *Id%*)

Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>Id%</i>	INTEGER	I	menu item identifier

Value returned INT(2)

Comments

- 1.The range \$F000...\$FFFE is reserved for runtime environment images. Currently, a small part of this range between \$F000 and \$F014 is composed of images as indicated in the introduction. Below this range are the indexes of images belonging to the first WIL belonging to the window (or of its parent window if it is an MDI child and does not have its own WIL) which are returned by GETMENUITEMPICTURE%.
- 2.The menu item identifier may be obtained by calling the function GETCONTROLID%.
- 3.This function is equivalent to calling the dynamic property Picture of the menuItem:

```
i% = MenuItem.picture
```

Example

```
local i%(2)
i%=GetMenuItemPicture% (self%,GETCONTROLID%(MNU_DEL))
SetMenuItemPicture (self%,GETCONTROLID%(MNU_CLOSE), i%)
```

See Also GETCONTROLID%, WIL_APPENDFROMBMP%

SETMENUITEMPICTURE instruction

Sets the image of a menu item.

Syntax	SETMENUITEMPICTURE			<i>wnd, Id%, pict%</i>
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>Id%</i>	INTEGER	I	menu item identifier
	<i>pict%</i>	INT(2)	I	index or image identifier

Comments

- 1.The menu item identifier may be obtained by calling the function
GETCONTROLID%.
- 2.This function is equivalent to calling the dynamic property Picture of the menuItem:

```
MnuItem.picture = $F014
```

Example

```
;First image of the first WIL  
SetMenuItemPicture (self%,GETCONTROLID%(MNU_CLOSE), 0)  
;Runtime Image $F00C  
SetMenuItemPicture (self%,GETCONTROLID%(MNU_ABOUT), $F00C)  
;Second image of the first WIL  
SetMenuItemPicture (self%,GETCONTROLID%(MNU_OUVR), 1)
```

See also GETCONTROLID%, WIL_APPENDFROMBMP%

WIL_GETCOUNT% function

Returns the number of WILs associated with a window. A WIL contains one or several images of the same size (generally 16x16 pixels, but not necessarily) with a transparency mask (like icons).

Syntax	WIL_GETCOUNT%	<i>wnd</i>
Parameter	<i>wnd</i>	POINTER I window handle
Value returned	INTEGER	Number of WILs for the window.

Comment

1. One or more WILs can be associated with a window:

Example

```
local index%  
index% = WIL_GetCount%(self%)  
insert AT END "the number of the list ="&&index% to MLE
```

See also WIL_APPENDFROMBMP%

WIL_GETPICTURESCOUNT% function

Returns the number of images in a WIL belonging to a window.

Syntax **WIL_GETPICTURESCOUNT%** (*wnd*, *Index%*)

Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>index%</i>	INTEGER	I	WIL index (from 0 to WIL_GetCount%(Self%) - 1)

Value returned INTEGER The number of images in the WIL.

Example

```
INSERT AT END "The number of images on the 1st list is " & \  
WIL_GetPicturesCount%(self%, 0) TO MLE
```

See also WIL_APPENDFROMBMP%, WIL_GETCOUNT%

WIL_SETCOUNT instruction

Sets the number of WILs belonging to a window. Its use is optional, the functions that add images to a WIL create it if need be.

Syntax **WIL_SETCOUNT** *wnd, Count%*

Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>count%</i>	INTEGER	I	number of the Image List

Comment

1. If the number of Image Lists is less than `WIL_GETCOUNT%(Self%)`, the extra Images Lists will be destroyed.

Example

```
local index%  
WIL_SetCount(self%, 1)  
index% = WIL_GetCount%(self%)  
insert AT END "The number of list ="&&index% to MLE ;1
```

See also **WIL_APPENDFROMBMP%**, **WIL_GETCOUNT%**

WIL_DESTROY instruction

Destroys the WIL that is indicated. The following WILs are not re-numbered (their index remains the same, the free index can be used for a new WIL).

Syntax	WIL_DESTROY <i>wnd, Index%</i>			
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>Index%</i>	INTEGER	I	WIL index

Example

```
local nb%
nb% = WIL_GetCount%(self%)
if nb% > 0
    WIL_DESTROY(self%,nb% - 1) ; destroy the last WIL
endif
```

See also **WIL_APPENDFROMBMP%**, **WIL_GETCOUNT%**

WIL_APPENDFROMBMP% function

Adds one or more images at the end of an Image List. If the Image List does not yet exist, it is created.

Syntax	WIL_APPENDFROMBMP% (<i>wnd</i> , <i>Index%</i> , <i>Bmp%</i> , <i>PicturesWidth%</i> , <i>TranspColor%</i>)			
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>index%</i>	INTEGER	I	WIL index. (or -1 if you want to add a new WIL after those that already exist for the window handle <i>wnd</i>)
	<i>bmp%</i>	POINTER	I	bitmap handle containing the image(s). (The height is the same as that of the WIL, the width should be a multiple of the width of the images in the WIL)
	<i>PicturesWidth%</i>	INTEGER	I	the width of the images in pixels (must be identical to that of the images in the WIL if they already exist)
	<i>TranspColor%</i>		I	the transparency color.
Value returned	The value returned is either the index of the first image added to the WIL (0 if it was initially empty or did not exist), or the index of the new WIL if the parameter giving the index of the WIL was equal to -1 (addition of a new WIL, in this case, its first image is always 0 in the index). Returns -1 if an error occurs.			

Comments

1. The transparent color is a 32-bit integer, if the high order byte is 0, it is one of the colors COL_*, the values -4...-1 indicate that you should use the color of the bottom right/top right/bottom left/top left pixel of the first image of the bitmap, -5 indicates that the images are opaque and a value of the type \$FErrvvbb allows specification of an RGB color where rr,vv,bb are the hexadecimal digits corresponding to the red, green and blue components.
2. It is henceforth possible to process JPEG and GIF image formats in addition to BMP, however, it is not possible to have a transparent color with JPEGs.

Example

```
Local hbmp%, index%
;this bmp has height of 32 and a width of 3x32 pixels
;therefore it makes a WIL of 3 bitmaps
MOVE CREATEBMP% ("(NS-BMP)\Natsystem.BMP") TO HBMP% ;Creates the bitmap hBMP%
index% = WIL_AppendFromBmp%(self%,-1,HBMP%,32,-1)
index% = WIL_GetCount%(self%)
insert AT END "the number of list ="&&index% to MLE
;a new WIL with Gifs
MOVE CREATEBMP% ("(NS-BMP)\Natsystem.gif") TO HBMP%
index% = WIL_AppendFromBmp%(self%,-1,HBMP%,32,-1)
index% = WIL_GetCount%(self%)
insert AT END " the number of list ="&&index% to MLE
```

See also WIL_FINDORADDONEPICTURE%

WIL_DELETEPICTURES instruction

Deletes images from a WIL. Images following images that have been deleted have their index reduced accordingly.

Syntax	WIL_DELETEPICTURES <i>wnd, Index%, First%, Count%</i>			
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>Index%</i>	INTEGER	I	WIL index.
	<i>First%</i>	INTEGER	I	index of the first image
	<i>Count%</i>	INTEGER	I	number of images to be deleted

Example

```
; deletes the first image from the first WIL
INTEGER COUNT%
WIL_DeletePictures self%,0,0,1
```

See also WIL_APPENDFROMBMP%, WIL_FINDORADDONEPICTURE%

WIL_GETSIZE instruction

Gives the dimensions of the images in the WIL indicated.

Syntax	WIL_GETSIZE	<i>wnd, Index%, Width%, Height%</i>		
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>Index%</i>	INTEGER	I	WIL index.
	<i>Width%</i>	INTEGER	I/O	width of the images in the WIL.
	<i>Height%</i>	INTEGER	I/O	height of the images in the WIL.

Example

```
local INTEGER index%, INTEGER width%, INTEGER height%
index% = 0 ; first WIL
WIL_GetSize self%, index%, width%, height%
insert AT END "The width of WIL's images WIL "&index%&&"est de "&width%&& \
"their height "&height% to MLE
```

WIL_DRAWPICTURE instruction

Draws the requested image.

Syntax	WIL_DRAWPICTURE <i>ps%, x%, y%, Style%, wnd, IndexIL%, IndexPict%</i>			
Parameters	<i>ps%</i>			Presentation Space
	<i>x%</i>	INTEGER	I	x coordinates of the bottom left corner of the image
	<i>y%</i>	INTEGER	I	y coordinates of the bottom left corner of the image
	<i>style%</i>	INTEGER	I	drawing mode constant ILD_TRANSPARENT% with the transparency bit taken into account, or IDL_MASK% to draw the outline of the image.
	<i>wnd</i>	POINTER	I	window handle
	<i>IndexIL%</i>	INTEGER	I	WIL index
	<i>IndexPict%</i>	INTEGER	I	image index

Example

```
INIT Main Window
GLOBAL POINTER handleFille%
GLOBAL int gPaint%(1)
gPaint% = False%
;Better to leave the window get initialized first
post WIL

;user Event WIL of the main WINDOW
Local hbmp%, index%
MOVE CREATEBMP% (" (NS-BMP)\MTINTIN.BMP" ) TO HBMP% ;Creates the bitmap BMP%
MOVE HBMP% TO BMP_SA ; Displays the bitmap
index% = WIL_AppendFromBmp%(self%,-1, HBMP%,32,-1) ; a list of 4 pictures
OpenH WDESSIN, 0, handleFille%
ATTACHCHILDWINDOW SELF%, CHILD_TOPAREA% , handleFille%
gPaint% = TRUE%
```

```
;Event Paint of childWindow WDESSIN
; it draws all the pictures of all the WILs attached to the parentWindow
LOCAL POINTER PS%, i%, j%
LOCAL INTEGER X%
LOCAL INTEGER Y%
local INTEGER width%, INTEGER height%
MOVE PARAM12% TO PS%
GPI_ERASE PS%
if gpaint%
    X% = 0; getclientwidth%
    for i% = 0 to wil_getcount% (parentwindow%) -1
        WIL_GetSize parentwindow%, i%, width%, height%
        Y% = getclientheight%/2 - height%/2
        for j% = 0 to wil_getpicturescount% (parentwindow%, i%) -1
            ; the Y is calculated to be the center
            ; while the X is incremented
            wil_DrawPicture PS%, X%,Y%,0,parentwindow%, i%, j%
            X% = X% + width%
        endfor
    endfor
endif
```

WIL_FINDPICTURE function

Finds an image in a WIL (identical pixels) and returns the index of the first image that corresponds to the search criteria or -1 where no picture is found.

Syntax	WIL_FINDPICTURE% (<i>Wnd, Index%, Bmp%, TranspColor%, ppHash</i>)			
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>index%</i>	INTEGER	I	WIL index
	<i>bmp%</i>	INTEGER	I	bitmap handle to be found (it must have the same dimensions as the images in the WIL)
	<i>TranspColor%</i>		I	the color should be considered as being transparent.
	<i>ppHash</i>	POINTER	I	pointer, passed by the address, changed by the function) which should be set to 0 (NULL) before a first search in a WIL and passed to WIL_FREEHASH after the last search or if the WIL is changed other than by WIL_FINDORADDONEPICTURE% (used to optimize the speed of comparisons when several images are being searched for/added.

Comment

1. The transparent color is a 32-bit integer, if the high order byte is 0, it is one of the colors COL_*, the values -4...-1 indicate that you should use the color of the bottom right/top right/bottom left/top left pixel of the first image of the bitmap, -5 indicates that the images are opaque and a value of the type \$FErrvvbb allows specification of an RGB color where rr,vv,bb are the hexadecimal digits corresponding to the red, green and blue components.

Example

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% (" (NS-BMP)\validoff.BMP") TO HBMP%
index% = WIL_FindPicture% (self%,0,HBMP%,-1, @pthash)
if index% = -1
    insert AT END "the image has not been found" to MLE
else
```

4-238 *NSMisc Miscellaneous Library*

```
        insert AT END "the image is at "&&index% &&"place" to MLE
endif
WIL_FreeHash pthash
```

See also WIL_FREEHASH

WIL_FREEHASH instruction

Frees the memory allocated by WIL_FINDPICTURE% or by WIL_FINDORADDONEPICTURE%.

Syntax	WIL_FREEHASH	<i>pHash</i>	
Parameter	<i>pHash</i>	POINTER	I/O pointer passed as the last parameter of one of these functions WIL_FINDPICTURE% or by WIL_FINDORADDONEPICTURE%

Example

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% ("(NS-BMP)\validoff.BMP") TO HBMP%
index% = WIL_FindPicture% (self%,0,HBMP%,-1, @pthash)
if index% = -1
    insert AT END "the image has not been found" to MLE
else
    insert AT END "the image is at "&&index% &&"place" to MLE
endif
WIL_FreeHash pthash
```

See also WIL_FINDPICTURE%, WIL_FINDORADDONEPICTURE%

WIL_FINDORADDONEPICTURE% function

Adds one or more images at the end of a WIL.

Syntax	WIL_FINDORADDONEPICTURE%. (<i>wnd</i> , <i>Index%</i> , <i>Bmp%</i> , <i>TranspColor%</i> , <i>ppHash</i>)			
Parameters	<i>wnd</i>	POINTER	I	window handle
	<i>index%</i>	INTEGER	I	WIL index. (or -1 if you want to add a new WIL after those that already exist for this window)
	<i>bmp%</i>	POINTER	I	bitmap handle containing the image(s). (the height is that of the WIL, the width should be a multiple of the width of the images)
	<i>PicturesWidth%</i>	INTEGER	I	the width of the images (in pixels, must be identical to that of the WIL if it already exists)
	<i>TranspColor%</i>		I	the color being considered as being transparent.
Value returned	Returns the index of the image found (if it already exists) or added, or -1 when an error occurs.			

Comment

1. The transparent color is a 32-bit integer, if the high order byte is 0, it is one of the colors COL_*, the values -4...-1 indicate that you should use the color of the bottom right/top right/bottom left/top left pixel of the first image of the bitmap, -5 indicates that the images are opaque and a value of the type \$FErrvvbb allows specification of an RGB color where rr,vv,bb are the hexadecimal digits corresponding to the red, green and blue components.

Example

```
Local hbmp%, index%
Local POINTER pthash
pthash = 0
MOVE CREATEBMP% (" (NS-BMP)\validoff.BMP") TO HBMP%
; finds the first occurrence of the picture
index% = WIL_FindOrAddOnePicture% (self%,0,HBMP%,-1, @pthash)
```



```
if index% = -1
    insert AT END "an error happens" to MLE
else
    insert AT END "the image is at "&&index% "&&"place" to MLE
endif
WIL_FreeHash pthash
```

See also WIL_FREEHASH, WIL_FINDPICTURE%

SMIP_ *% constants

These constants let you dynamically assign images predefined in the Nat System runtime environment to menu items. In previous versions, only the corresponding hexadecimal number could be used

Syntax	SMIP_FirstStdPict%
	SMIP_FileNew%
	SMIP_FileNew2%
	SMIP_FileOpen%
	SMIP_FileSave%
	SMIP_FileSaveAll%
	SMIP_FilePrint%
	SMIP_EditUndo%
	SMIP_EditRedo%
	SMIP_EditCut%
	SMIP_EditCopy%
	SMIP_EditPaste%
	SMIP_EditDelete%
	SMIP_Help%
	SMIP_Search%
	SMIP_FindNext%
	SMIP_FindPrev%
	SMIP_WinCascade%
	SMIP_WinSplitHorz%
	SMIP_WinSplitVert%
	SMIP_WinChoose%
	SMIP_ExitApp%
	SMIP_FileClose1%
	SMIP_FileClose2%
	SMIP_FileDelete%
	SMIP_Zoom%
	SMIP_Tools%
	SMIP_HelpContent%
	SMIP_HelpIndex%

SMIP_HelpSearch%**Notes**

1. In order to associate images with menu items the variable NoMenuPictures must not be equal to True in the NSLIB.INI file.
2. These images are all one size of 16x16 pixels, the recommended size for menu items.
3. Their signification is the following:

SMIP_FirstStdPict% or SMIP_FileNew%	new
SMIP_FileNew2%	new document (variant of \$F000)
SMIP_FileOpen%	open
SMIP_FileSave%	save
SMIP_FileSaveAll%	save all
SMIP_FilePrint%	print
SMIP_EditUndo%	undo
SMIP_EditRedo%	redo
SMIP_EditCut%	cut
SMIP_EditCopy%	copy
SMIP_EditPaste%	paste
SMIP_EditDelete%	delete
SMIP_Help%	help
SMIP_Search%	find
SMIP_FindNext%	find next
SMIP_FindPrev%	find previous
SMIP_WinCascade%	cascade (MDI)
SMIP_WinSplitHorz%	tile horizontal (MDI)
SMIP_WinSplitVert%	tile vertical (MDI)
SMIP_WinChoose%	select window (MDI)
SMIP_ExitApp%	exit

SMIP_FileClose1%	close a file
SMIP_FileClose2%	close a file
SMIP_FileDelete%	delete a file
SMIP_Zoom%	zoom
SMIP_Tools%	tools
SMIP_HelpContent%	help content
SMIP_HelpIndex%	help index
SMIP_HelpSearch%	help search

4. Their internal declaration is:

```

CONST SMIP_FirstStdPict% $F000
CONST SMIP_FileNew%      $F000
CONST SMIP_FileNew2%     $F001
CONST SMIP_FileOpen%     $F002
CONST SMIP_FileSave%     $F003
CONST SMIP_FileSaveAll%  $F004
CONST SMIP_FilePrint%    $F005
CONST SMIP_EditUndo%     $F006
CONST SMIP_EditRedo%     $F007
CONST SMIP_EditCut%      $F008
CONST SMIP_EditCopy%     $F009
CONST SMIP_EditPaste%    $F00A
CONST SMIP_EditDelete%   $F00B
CONST SMIP_Help%         $F00C
CONST SMIP_Search%       $F00D
CONST SMIP_FindNext%     $F00E
CONST SMIP_FindPrev%     $F00F
CONST SMIP_WinCascade%   $F010
CONST SMIP_WinSplitHorz% $F011
CONST SMIP_WinSplitVert% $F012
CONST SMIP_WinChoose%    $F013
CONST SMIP_ExitApp%      $F014
CONST SMIP_FileClose1%   $F015
CONST SMIP_FileClose2%   $F016
CONST SMIP_FileDelete%   $F017
CONST SMIP_Zoom%         $F018
CONST SMIP_Tools%        $F019

```

```
CONST SMIP_HelpContent%  $F01A
CONST SMIP_HelpIndex%    $F01B
CONST SMIP_HelpSearch%   $F01C
```

Example

```
MnuItem.picture = SMIP_FileNew%
```

Where MnuItem is the name of a menu item and \$F000 the image reference for « new document » in the runtime environment.

See also

WIL_APPENDFROMBMP%, WIL_FINDORADDONEPICTURE%, .PICTURE
dynamic parameter

SetNSGenTrace instruction

Allows you to modify the traces formatting by providing a pointer on a MyTrace function.

Syntax **SetNSGenTrace** *traceFunc*

Parameter *traceFunc* POINTER I pointer towards a MyTrace function

See Also myTrace

MyTrace instruction

Allow to modify the traces formatting.

Syntax **MyTrace** *trace, line%, info*

Parameters	<i>trace</i>	SEGMENT	I	reference on the SEG_NSGENTRACE segment
	<i>line%</i>	INT	I	line number of the source file
	<i>info</i>	POINTER	I	not used currently

Notes

1. The *line%* parameter corresponds to the line number where the event of trace occurred. The first line of the window event is 1.
2. The *info* pointer is reserved for a later use and could point on additional information. Currently, it is null for all traces.

See also SetNSGenTrace

NS_TRACE instruction

Allow to integrate the user traces in the NCL programs.:

Syntax **NS_TRACE** *message*\$

Parameter *message*\$ CSTRING I user traces

Note

- By default, the formatting of traces follows the next example:

```

NSDK:14:32:49
*****
**
NSDK:14:32:49 NSGENTRACE-<TRC> { ID in BINARY/FILE(LINE)
NSDK:14:32:49 NSGENTRACE-<TRC> } ID in BINARY/FILE(LINE)
(CLOCK=CPU)
NSDK:14:32:49          \ \ \          \      \
\          \_____CPU time between
NSDK:14:32:49          \ \ \          \      \
\          enter and leave
NSDK:14:32:49          \ \ \          \      \
\_____ line in FILE
NSDK:14:32:49          \ \ \          \
\_____ ressource FILE
NSDK:14:32:49          \ \ \          \
\_____ BINARY (dll or exe)
NSDK:14:32:49          \ \ \ _____ID of
function/instruction/event
NSDK:14:32:49          \ \ _____Enter
or leave funct/instr/event
NSDK:14:32:49          \_Generated trace reason
'instr', 'funct', 'event' for
NSDK:14:32:49          respectively instruction,
function and control event
NSDK:14:32:49
*****
**
NSDK:14:32:49 NSGENTRACE-event { SEQ.EXECUTED in
TEST.EXE/TEST.SCR(1)
NSDK:14:32:49 NSGENTRACE-instr  { GO3 in TEST.EXE/SEQ.NCL(137)
NSDK:14:32:50 NSGENTRACE-instr  } GO3 in TEST.EXE/SEQ.NCL(139)
(CLOCK=0001.047)

```



```
NSDK:14:32:50 NSGENTRACE-event } SEQ.EXECUTED in
TEST.EXE/TEST.SCR(2) (CLOCK=0001.047)
NSDK:14:32:55 NSGENTRACE-event { <CLIENT>.TERMINATE in
TEST.EXE/TEST.SCR(1)
NSDK:14:32:55 NSGENTRACE-event } <CLIENT>.TERMINATE in
TEST.EXE/TEST.SCR(2) (CLOCK=0000.000)
```

5. The NS_TRACE var\$ syntax authorizes from now on a string longer than 255 characters. However, the NS_TRACE var1\$&var2\$ syntax is still limited to 255 characters.

See also SEG_NSGENTRACE segment, GTE_ *% constants

SEG_NSAGENTTRACE segment

Allow to integrate the user traces in the NCL programs.

Syntax **SEGMENT SEG_NSAGENTTRACE**

version

trEvt

module

res

ctrl

id

descr

ENDSEGMENT

Fields	<i>version</i>	INTEGER	I	checking version of the segment
	<i>trEvt</i>	INTEGER	I	Trace event (GTE_%% constants)
	<i>module</i>	CSTRING	O	EXE or DLL name
	<i>res</i>	CSTRING	O	NCL/SCR/TPL resource file name
	<i>ctrl</i>	CSTRING	O	for the NCL events only, "<CLIENT>" for the window
	<i>id</i>	CSTRING	O	identifier (function, instruction, event)
	<i>descr</i>	POINTER	I	description of the information pointer content

See also GTE_%% constants, NS_TRACE

GTE_%% constants

Values for the GTE_%% trace events.

Syntax

GTE_ENTERINSTR%

GTE_LEAVEINSTR%

GTE_ENTERFUNC%

GTE_LEAVEFUNC%

GTE_ENTEREVENT%

GTE_LEAVEEVENT%

Notes

1. Their meaning is the following:

GTE_ENTERINSTR%	beginning of an instruction
GTE_LEAVEINSTR%	end of an instruction
GTE_ENTERFUNC%	beginning of a function
GTE_LEAVEFUNC%	end of a function
GTE_ENTEREVENT%	beginning of a window event
GTE_LEAVEEVENT%	end of a window event

2. Their internal declaration is :

```
; GTE = Generator Trace Event
CONST GTE_ENTERINSTR% 1
CONST GTE_LEAVEINSTR% 2
CONST GTE_ENTERFUNC% 3
CONST GTE_LEAVEFUNC% 4
CONST GTE_ENTEREVENT% 5
CONST GTE_LEAVEEVENT% 6
```

See also SEG_NSGENTRACE, NS_TRACE, MyTrace, SetNSGenTrace

NS_AS_* constants

Allow to anchor controls into a window.

Syntaxe

NS_AS_LEFT
NS_AS_RIGHT
NS_AS_TOP
NS_AS_BOTTOM
NS_AS_PLEFT
NS_AS_PRIGHT
NS_AS_PTOP
NS_AS_PBOTTOM
NS_AS_NONE
NS_AS_BOTTOM_LEFT
NS_AS_TOP_LEFT
NS_AS_TOP_RIGHT
NS_AS_BOTTOM_RIGHT
NS_AS_WIDTH_TOP
NS_AS_WIDTH_BOTTOM
NS_AS_HEIGHT_RIGHT
NS_AS_HEIGHT_LEFT
NS_AS_WIDTH_HEIGHT

Notes

1. Used with the dynamic parameter .ANCHOR.
2. Their meaning is :

NS_AS_LEFT	Fixation to the left
NS_AS_RIGHT	Fixation to the right
NS_AS_TOP	Fixation at the top
NS_AS_BOTTOM	Fixation at the bottom
NS_AS_PLEFT	Fixation proportionnally to the left
NS_AS_PRIGHT	Fixation proportionnally to the right
NS_AS_PTOP	Fixation proportionnally at the top
NS_AS_PBOTTOM	Fixation proportionnally at the bottom

NS_AS_NONE	No anchoring. The edges of the control are not attached to the edges of the window.
NS_AS_BOTTOM_LEFT	Fixation at the left bottom of the control.
NS_AS_TOP_LEFT	Fixation at the left top of the control.
NS_AS_BOTTOM_RIGHT	Fixation at the right bottom of the control.
NS_AS_TOP_RIGHT	Fixation at the right top of the control.
NS_AS_WIDTH_TOP	Fixation at the right, the left and the top.
NS_AS_WIDTH_BOTTOM	Fixation at the right, the left and the bottom.
NS_AS_WIDTH_HEIGHT	Fixation at the top, the bottom, the left and the right.
NS_AS_HEIGHT_LEFT	Fixation at the top, the bottom and the left.
NS_AS_HEIGHT_RIGHT	Fixation at the top, the bottom and the left.

3. Their internal declaration is:

```

; base anchor styles
const NS_AS_LEFT      "L "
const NS_AS_RIGHT     "R "
const NS_AS_TOP       "T "
const NS_AS_BOTTOM    "B "
const NS_AS_PLEFT     "%L "
const NS_AS_PRIGHT    "%R "
const NS_AS_PTOP      "%T "
const NS_AS_PBOTTOM   "%B "

; complex anchor styles
const NS_AS_NONE      " "
const NS_AS_BOTTOM_LEFT  "B L "
const NS_AS_TOP_LEFT    "T L "
const NS_AS_BOTTOM_RIGHT "B R "
const NS_AS_TOP_RIGHT   "T R "
const NS_AS_WIDTH_TOP   "T L R "
const NS_AS_WIDTH_BOTTOM "B L R "
const NS_AS_HEIGHT_LEFT "T B L "
const NS_AS_HEIGHT_RIGHT "T B R "
const NS_AS_WIDTH_HEIGHT "T B L R "

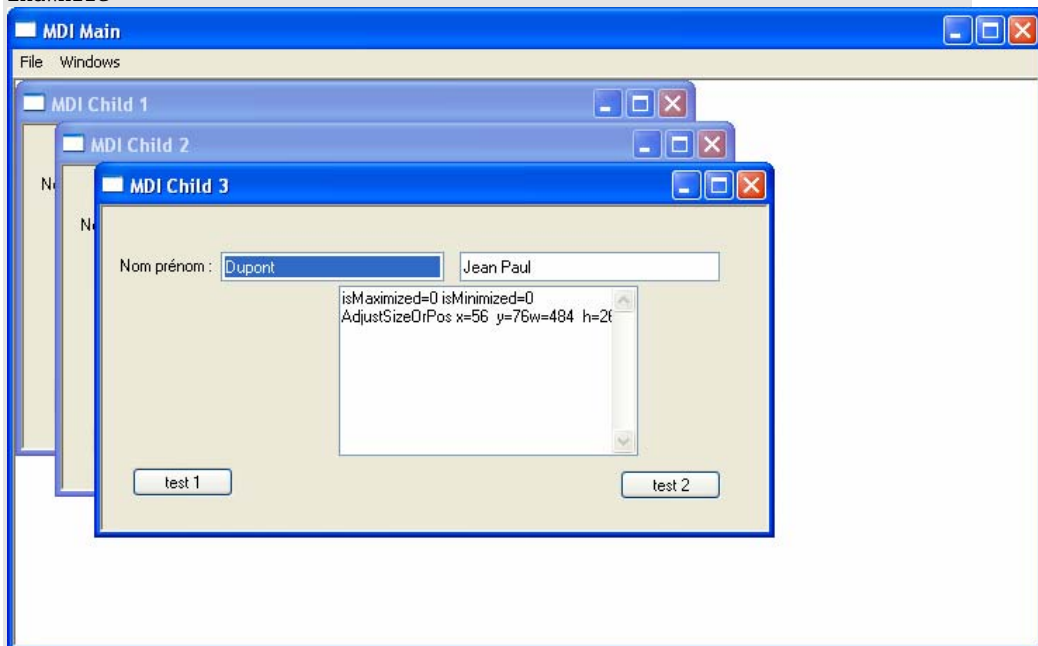
```

Example

```

local control c
firstcontrol (self%,C)
while miscerror% = 0
    c.ANCHOR = NS_AS_PTOP & NS_AS_PBOTTOM & NS_AS_PLEFT &
    NS_AS_PRIGHT
    nextcontrol (c)
EndWhile

```

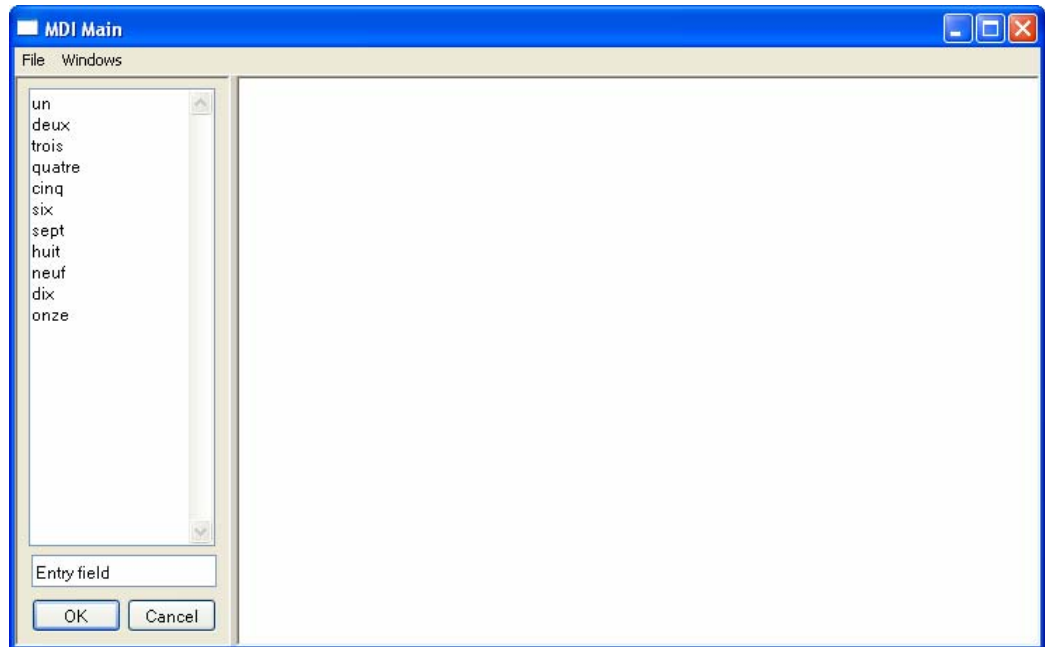


1. An attached window is a window of Dialog class opened without handle of the parent window and attached with ATTACHCHILDWINDOW instruction in order to be integrated into the parent window.

```

OpenH WATTACH, 0, WndAtt
AttachChildWindow Self%, CHILD_PrefixArea%, WndAtt

```

**2. Their internal declaration is:**

```
CONST ACF_MDIDeactivate% $01
CONST ACF_Resizable% $02
```

Example

```
Local Pointer WndAtt2

OpenH WATTACH2, 0, WndAtt2
WATTACH2(WndAtt2).T1 = "Bottom"
SetBackCol COL_LightGreen% To WndAtt2
AttachChildWindow MainWindow%, CHILD_BotArea%, WndAtt2
SetAttachedChildFlags MainWindow%, CHILD_BotArea%, \
ACF_Resizable%
```

See also

GetAttachedChildFlags%, SetAttachedChildFlags

RemoveCmdLineParam Instruction

Removes (hides) the command line parameter *Parm%*. The total number of start parameters (corresponding to the result of the PARAMCOUNT% function) is reduced by 1 and the next parameter takes its place in the PARAMSTR\$() function list and so on.

Syntax **RemoveCmdLineParam** *Parm%*

Parameter	<i>Parm%</i>	INTEGER	I	Start parameter (from 1 to PARAMCOUNT%)
------------------	--------------	---------	---	--

Comments

1. The PARAMSTR\$(0) line cannot be removed, because PARAMSTR\$(0) returns the name of the application. The first start parameter has an index of 1.
2. The total number of start parameters is equal to PARAMCOUNT%.

See also PARAMCOUNT%, PARAMSTR%, HideRemovedCmdLineParams%

HideRemovedCmdLineParams% Function

Lets you reveal all the parameters that have been hidden.

Syntax	HideRemovedCmdLineParams%	<i>Hidden%</i>
Parameter	<i>Hidden%</i>	INTEGER I Boolean (FALSE% or TRUE%)
Return value	INTEGER	
Comments		

1. To reveal parameters hidden by the RemoveCmdLineParam instruction, set the *Hidden%* parameter to FALSE%. To hide them again, set the *Hidden%* parameter to TRUE%.
2. This instruction lets you retrieve the former state of the start parameters so that you can save the active state during a first call to change them.

Example:

```
bak% = HideRemovedCmdLineParams%(FALSE%)
```

then restore the active state when you have finished using PARAMCOUNT% and PARAMSTR\$ in the modified state.

```
HideRemovedCmdLineParams%(bak%)
```

See also PARAMCOUNT%, PARAMSTR%, RemoveCmdLineParam

Chapter 5

NSDynstr Library



This library contains functions to handle character strings of Dynstr type.

***This chapter
explains***

- How to install this library
- General aspects of its functions and instructions
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation	5-3
NSDynstr.NCL File	5-4
List of functions and instructions in the NSDynstr library	5-5
NSDynstr library reference	5-6
Example of saving a string content in a file.....	5-12

Installation

Declare NSDYNSTR, NSERRORS and NSMISC in the libraries required to develop your application.

Ensure that the NSxxDYNSTR.DLL, NSxxERRORS.DLL and NSxxMISC.DLL files has been placed in a PATH directory under Windows.

NSDynstr.NCL File

The verbs in this library (NSDynstr) are described below. They are declared in a text file, written in NCL, called NSDYNSTR.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSDYNSTR NCL file:

1. Navigate to the <NATSTAR>\NCL, <NSDK>\NCL or <NATWEB>\NCL directory.
NB : <NATSTAR> means the directory in which NatStar was installed.
NB : <NSDK> means the directory in which NS-DK was installed.
NB : <NATWEB> means the directory in which NS-DK was installed.
2. Open the NSDYNSTR.NCL file with any text editor to see its contents.

List of functions and instructions in the NSDynstr library

Here is a list of the instructions, functions and constants in the NSDynstr library.

IsDSNull%	5-7
SetDSNull	5-8
ForceDSLlength	5-9

NSDynstr library reference

IsDSNull% function

Lets you find out if a character string has been changed or not.

Syntax **IsDSNull%** (*ds*)

Parameter *ds* DYNSTR I character string

Return value TRUE% or FALSE%

Comments

1. The value returned is a Boolean indicating whether it is a question of a variable that has been changed or not since its creation or since the last call to SetDSNull on this variable.

```
Local DynStr ds
```

```
Message "ds is null", IsDSNull%(ds) ; returns TRUE%, displayed as 1
```

```
ds = "" ; empty string but netherless modified
```

```
Message "ds is not null anymore", IsDSNull%(ds)
```

```
; returns FALSE%, displayed as 0
```

2. If you write `ds = F$(parameters)` and `F$` returns a null DynStr (but not an empty string, and `F$` must be declared as returning a DynStr), then `IsDSNull%(ds)` returns TRUE%, even if the DynStr returned is one of the parameters (DynStr) of `F$` which will have received a null DynStr variable. This is not true for `Copy$`, `Delete$`, `Insert$`, `Skip`, `LSkip`, `RSkip`, `UpCase`, `LowCase`, `&` and `&&`, even if they are passed one (or two for `Insert$`, `&` and `&&`) null DynStr(s), these functions always return a non-null DynStr.
3. The text of a null DynStr is identical to that of an empty DynStr, even for comparison operators, only `IsDSNull%` can distinguish them. This has been done in order to be able to carry out diagnosis of an uninitialized DynStr variable.

See also SetDSNull

SetDSNull Instruction

Lets you set a DynStr variable to the NULL state and also delete its content.

Syntax	SetDSNull	<i>ds</i>		
Parameter	<i>ds</i>	DYNSTR	O	character string
See also	IsDSNull%			

ForceDSLength Function

Lets you force the size of a character string.

Syntax **ForceDSLength** (*ds*, *Len%*)

Parameters	<i>ds</i>	DYNSTR	I	character string
	<i>Len%</i>	INTEGER	I	length of the character string

Return value POINTER

Comments

1. The ForceDSLength function modifies the character string *ds* so that it can contain the required number of characters (plus one character for the end of string zero, *Len%* gives the number of useable characters). *Len%* which contains the length requested on input, takes the length available on output. IT IS ESSENTIAL not to use more characters than the value contained in *Len%* on output (plus the zero end of string byte which is already present in the correct position after the call to ForceDSLength).
2. The function returns a pointer to the first character (uninitialized) of the string. This pointer should be used to fill the character string. The initial characters in the string may be lost.
3. The length *Len%* may be shortened on output in two situations: if the size requested is too great, or if the DynStr variable passed is in fact a parameter of the DynStr@ type function to which a CString(n) type variable has been passed instead of a DynStr (which is permitted but which limits the size of the variable) during a call to this function, in which case, the size of this variable may not exceed n (255 if not specified).
4. This function should only be used if you want to transfer a sequence of character buffers as efficiently as possible where you know the total size in advance but which you can only obtain once (in order to avoid concatenation that reallocates and copies DynStr each time), or if you need a direct pointer to the memory buffer of the DynStr characters in order to avoid copying and/or concatenation (for example, if you want to carry out one or more F_BLOCKREADs in order to transfer a text file (without an end of string 0 in the middle) into a DynStr.

Example

```
; Reads the contents of a text file with the name FileName$ into  
FileContent$.
```

```
; Returns one of the NSERRORS.NCL error codes, No_Error% if OK.
Function LoadFile%(DynStr FileName$, DynStr@ FileContent$)
    Local F%, Ret%(1), Cnt%(2), Integer SubSize%, Integer Size%,
    Pointer P

    F% = F_ROOpen%(1, FileName$)
    Ret% = F_Error%
    If Ret% = No_Error%
        Size% = F_Size%(F%)
        Ret% = F_Error%
        If Ret% = No_Error%
            ; tries to force the length of the DynStr FileContent$ to the
            ; size of the file FileName$, one additional byte is allocated and
            ; initialized to 0 for the end of the string. Size% may be modified
            ; if the DynStr FileContent$ in fact points to a CString(n%) and
            ; cannot
            ; take the desired size. Obtain a pointer to the area
            ; allocated which then needs to be filled with text.
            P = ForceDSLenth(FileContent$, Size%)
            If P = 0
                Ret% = Error_Not_Enough_Memory% ; failure of
;ForceDSLenth
            Else
                SubSize% = $4000 ; reads 16 kb at a time
                ; as long as there is room and the end of the file is not reached;
                While (Size% > 0) And Not F_EOF%(F%)
                    If Size% < SubSize%
                        SubSize% = Size% ; last block
;read smaller than 16 kb
                    EndIf
                    F_BlockRead F%, P, SubSize%, Cnt%
; reading the block, Cnt% bytes read
                    Ret% = F_Error%
                    If Ret% <> No_Error%
                        Break ; file read error
                    EndIf
                    P = P + Cnt% ; advances the pointer to
;the next block
                    Size% = Size% - Cnt% ; calculates the
;size read of the size allocated
                EndWhile
                If Size% > 0 ; have fewer characters been
;read than have been allocated?
; Truncates DynStr to the number of characters read
```

```
FileContent$ = Copy$(FileContent$, 1,  
\  
Length(FileContent$) - Size%)  
EndIf  
EndIf  
EndIf  
F_Close F%  
EndIf  
Return Ret%  
EndFunction ; LoadFile$
```

See also IsDSNull%, SetDSNull

Example of saving a string content in a file

```
; Save the content of the FileContent$ string in a file named
; FileName$. Return error code in NSERRORS.NCL,
; No_Error% if OK.
Function SaveFile%(DynStr FileName$, DynStr FileContent$)
    Local F%, Ret%(1), Cnt%(2), SubSize%, Size%, Pointer P

    F% = F_Create%(1, FileName$)
    Ret% = F_Error%
    If Ret% = No_Error%
        Size% = Length(FileContent$) ; taille du fichier
        ; obtains a pointer on the existing characters of the Dynstr,
        ; WE MUST NEVER MODIFY THEM BECAUSE SEVERAL DYNSTR MAY SHARE THE
        ; SAME TEXT IN MEMORY, except right away after a ForcedSLength,
        ; as long as the DynStr that have been passed isn't coied or passed
        ; by parameter to a function. Besides, THIS POINTER MAY BECOME
        ; INVALID if we change the Dynstr.
        P = @FileContent$
        SubSize% = $4000 ; write 16kb at the same time
        While Size% > 0
            If Size% < SubSize%
                SubSize% = Size%
                ; last written block smaller than 16kb
            EndIf
            F_BlockWrite F%, P, SubSize%, Cnt%
            ; writing a block, Cnt% bytes written
            Ret% = F_Error%
            If Ret% <> No_Error%
                Break
            EndIf
            If SubSize% <> Cnt%
                ; number of bytes written smaller than asked ?
                Ret% = Error_Cannot_Write_On_File%
                Break
            EndIf
            P = P + SubSize% ; put the pointer to the following
; block
            Size% = Size% - SubSize% ; deduce the written size
; to the total size
        EndWhile
        F_Close F%
```

```
EndIf
Return Ret%
EndFunction ; SaveFile%

; as T_ReadLn$ but without restriction to 255 characters !
Function T_ReadLnDS$(TF%) Return DynStr
    Local DynStr Ret$

    Ret$ = T_Read$(TF%)
    While (T_Error% = No_Error%) And Not T_EOL%(TF%)
        Ret$ = Ret$ & T_Read$(TF%)
    EndWhile
    Ret$ = Ret$ & T_ReadLn$(TF%)
    Return Ret$
EndFunction ; T_ReadLnDS$
```

Chapter 6

NSMLE Library



This library contains functions to handle text in MLE control or window.

***This chapter
explains***

- How to install this library
- General aspects of its functions and instructions
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation	6-3
NSMLE.NCL File	6-4
Functional categories in the NSMLE library	6-5
NSMLE library reference	6-6

Installation

Declare NSMLE in the libraries required to develop your application.

Ensure that the NSxxMLE.DLL file has been placed in a PATH directory under Windows.

NSMLE.NCL File

The verbs in this library (Nsmle) are described below. They are declared in a text file, written in NCL, called NSMLE.NCL. This file may also contain some complementary verbs which are not documented, so you can, if you wish, look at the file in order to have a complete reference for the library.

To look at the NSMLE NCL file:

1. Navigate to the <NATSTAR>\NCL, <NSDK>\NCL or <NATWEB>\NCL directory.
NB : <NATSTAR> means the directory in which NatStar was installed.
NB : <NSDK> means the directory in which NS-DK was installed.
NB : <NATWEB> means the directory in which NS-DK was installed.
2. Open the NSMLE.NCL file with any text editor to see its contents.

Functional categories in the NSMLE library

Here is a list of the instructions, functions and constants in the NSMLE library.

NSMLE_CUT.....	6-7
NSMLE_PASTE	6-9
NSMLE_CLEAR	6-10
NSMLE_SETTEXT%	6-11
NSMLE_GETTEXT.....	6-12
NSMLE_GETTEXTSIZE%	6-13
NSMLE_ISCLIPBOARD%	6-16
NSMLE_SELECTION\$.....	6-18
NSMLE_SEARCH%.....	6-19
NSMLE_CHANGE%	6-20

NSMLE library reference

NSMLE_CUT Instruction

Allows you to cut the selected text.

Syntax **NSMLE_CUT** *ctrl*

Parameter *ctrl* CONTROL I MLE control

Example

```
; ctrl is MLE control  
NSMLE_CUT ctrl
```

See also NSMLE_PASTE, NSMLE_COPY

NSMLE_COPY Instruction

Allows you to copy the selected text.

Syntax **NSMLE_COPY** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Example

```
; ctrl is a MLE control
NSMLE_COPY ctrl
```

See also NSMLE_PASTE, NSMLE_CUT

NSMLE_PASTE Instruction

Allows you to paste the selected text.

Syntax **NSMLE_PASTE** *ctrl*

Parameter *ctrl* CONTROL I MLE control

Example

```
Pointer pWin  
NSMLE_PASTE pWin
```

See also NSMLE_CUT, NSMLE_COPY

NSMLE_CLEAR Instruction

Allows you to clear the selected text.

Syntax **NSMLE_CLEAR** *ctrl*

Parameter *ctrl* CONTROL I MLE control

Example

```
Pointer pWin  
NSMLE_CLEAR pWin
```

See also NSMLE_CUT

NSMLE_SETTEXT% Function

Allows you to set a character string in the cursor position.

Syntax **NSMLE_SETTEXT%** (*ctrl*, *string*)

Parameters	<i>ctrl</i>	CONTROL	I	MLE control
	<i>string</i>	CSTRING	I	character string

Returned value INT(1)

Note

1. The *string* parameter can be a CSTRING or DYNSTR type (without truncature).

Example

```
LOCAL cstring string
string = « test »
Message "NSMLE_SetText%", NSMLE_SetText%(ctrl,string,length(chaine))
```

See also NSMLE_GETTEXT, NSMLE_GETTEXTSIZE%

NSMLE_GETTEXT Instruction

Allows you to retrieve the text contained in the maximum size defined in parameter.

Syntax **NSMLE_GETTEXT** *ctrl*

Parameter *ctrl* CONTROL I MLE control

Example

```
LOCAL DYNSTR ds
Ds = NSMLE_GetText (ctrl)
```

See also NSMLE_GETTEXTSIZE%, NSMLE_SETTEXT

NSMLE_GETTEXTSIZE% Function

Allows you to retrieve the size of the text contained in the window.

Syntax **NSMLE_GETTEXTSIZE%** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Returned value INTEGER

Exemple

```
Local int sz  
Sz = NSMLE_GetTextSize%(ctrl)
```

See also NSMLE_GETTEXT, NSMLE_GETTEXTSIZE%

NSMLE_ISSELECTED% Function

Allows you to know if there is a selected block.

Syntax **NSMLE_ISSELECTED%** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Returned value INT(1)

Example

```
NSMLE_ISSELECTED% ctrl
```

See also NSMLE_UNDO

NSMLE_ISUNDO% Function

Allows you to know if there is an action to cancel.

Syntax **NSMLE_ISUNDO%** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Returned value INT(1)

Example

```
NSMLE_ISUNDO% ctrl
```

See also NSMLE_UNDO%

NSMLE_ISCLIPBOARD% Function

Allows you to know if there is something to paste in the clipboard.

Syntax **NSMLE_ISCLIPBOARD%** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Returned value INT(1)

Exemple

```
NSMLE_ISCLIPBOARD% ctrl
```

See also NSMLE_PASTE, NSMLE_CUT, NSMLE_COPY

NSMLE_UNDO Instruction

Allows you to cancel the last action executed.

Syntax **NSMLE_UNDO** *ctrl*

Parameter *ctrl* CONTROL I MLE control

Example

```
NSMLE_UNDO ctrl
```

See also NSMLE_ISUNDO

NSMLE_SELECTION\$ Function

Allows you to select a character string.

Syntax **NSMLE_SELECTION\$** (*ctrl*)

Parameter *ctrl* CONTROL I MLE control

Returned value CSTRING

Example

```
Pattern$ = NSMLE_SELECTION$(ctrl)
If
    NSMLE_SEARCH%(ctrl, Pattern$, WholeWord%, CaseSensitive%)
EndIf
```

See also NSMLE_SEARCH%, NSMLE_CHANGE%

NSMLE_SEARCH% Function

Allows you to execute a search in the content of the MLE.

Syntax **NSMLE_SEARCH%** (*ctrl*, *pattern\$*, *wholeword%*, *casesensitive%*)

Parameters	<i>ctrl</i>	CONTROL	I	MLE control
	<i>pattern\$</i>	CSTRING	I	string to search
	<i>wholeword%</i>	INT(1)	I	search in the whole word
	<i>casesensitive%</i>	INT(1)	I	search by respecting the case sensitive

Returned value INT(1)

Example

```
Pattern$ = NSMLE_SELECTION$(ctrl)
If NSMLE_SEARCH%(ctrl, Pattern$, WholeWord%, CaseSensitive%)
EndIf
```

See also NSMLE_SELECTION\$, NSMLE_CHANGE%

NSMLE_CHANGE% Function

Allows you to replace a text by another one.

Syntax **NSMLE_CHANGE%** (*ctrl*, *pattern*\$, *replaceby*\$, *wholeword*%, *casesensitive*%, *changeall*%)

Parameters	<i>ctrl</i>	CONTROL	I	MLE control
	<i>pattern</i> \$	CSTRING	I	string to search
	<i>replaceby</i> \$	CSTRING	I	replacing search
	<i>wholeword</i> %	INT(1)	I	search in the whole word
	<i>casesensitive</i> %	INT(1)	I	search by respecting the case sensitive
	<i>changeall</i> %	INT(1)	I	change all the items found

Returned value INT(1)

Example

```
NSMLE_CHANGE%(mymlctrl, « search », « replaceby », wholeword%,
casesensitive%, changeall%)
```

See also NSMLE_SELECTION\$, NSMLE_SEARCH%

Chapter 7

NSDYNCTR Library



This chapter contains functions allowing dynamic handling of controls.

***This chapter
explains***

- How to install this library
- General aspects of its functions and instructions
- In which NCL file these verbs are declared
- The components in this library, arranged in functional categories
- The reference of the components in this library

Contents

Installation 7-3

NSDYNCTR Library Reference 7-4

Installation

Declare NSDYNCTR.NCL in the libraries required to develop your application.

Ensure that the NSxxDYNC.DLL file has been placed in a PATH directory under Windows.

NSDYNCTR Library Reference

The NSDYNCTR library includes the following verbs and constants:

DCA_EF_*	7-6
DCA_LB_*	7-10
DCA_MLE_*	7-12
DCA_CB_*	7-14
DCA_CHK_*	7-15
DCA_RB_*	7-16
DCA_ST_*	7-17
DCA_BMP_*	7-19
DCA_GB_*	7-21
DCA_VS_*	7-23
DCA_HS_*	7-24
DCA_MNU_*	7-25
DI_WINCMNCTRL%	7-27
DI_WCC_ANIMATION%	7-29
DI_WCC_DATETIMEPICKER%	7-30
DI_WCC_HOTKEY%	7-32
DI_WCC_LISTVIEW%	7-33
DI_WCC_MONTHCALENDAR%	7-36
DI_WCC_PROGRESSBAR%	7-37
DI_WCC_TRACKBAR%	7-38
DI_WCC_TREEVIEW%	7-40

DC_CONTROLPROPERTIES	7-42
ITEXISTS	7-47
GETTOTALDYNAMICCONTROLS%	7-48
DC_CREATECONTROL%	7-49
DC_DESTROYCONTROL%.....	7-50
DC_GETCONTROLPROPERTIESFROMID	7-51
DC_GETCONTROLPROPERTIES.....	7-53

DCA_EF_* Constants

These constants allow you to implement Dynamic Control Attributes for the Entry Field control.

Syntax	DCA_EF_AUTOSCROLL
	DCA_EF_HORZSCROLL
	DCA_EF_VERTSCROLL
	DCA_EF_EXPLODING
	DCA_EF_CENTER
	DCA_EF_LEFT
	DCA_EF_RIGHT
	DCA_EF_MARGIN
	DCA_EF_REQUIRED
	DCA_EF_DISABLE
	DCA_EF_LOCKTEXT
	DCA_EF_FULLTEXT
	DCA_EF_UPCASE
	DCA_EF_SKIPBLANKS
	DCA_EF_NOBLANKS
	DCA_EF_OVER
	DCA_EF_BELOW
	DCA_EF_AUTOTAB
	DCA_EF_HIDETEXT
	DCA_EF_INTEGER
	DCA_EF_NUMBER
	DCA_EF_DATE
	DCA_EF_TIME

DCA_EF_HIDDEN**Notes**

1. The meaning of these constants is as follows:

DCA_EF_AUTOSCROLL

The contents of the input field are automatically scrolled when the number of characters entered exceeds the size of this field. The characters that become invisible are not lost. Scrolling occurs during input.

DCA_EF_HORZSCROLL

Lets you display two horizontal arrows to the right of the Entry Field allowing the entry field to be scrolled when the number of characters entered exceeds the field size.

DCA_EF_VERTSCROLL

Lets you display two vertical arrows in the Entry Field allowing the entry field to be scrolled when the number of characters entered exceeds the field size.

DCA_EF_EXPLODING

Lets you change the height of the Entry Field during input if the text input exceeds the field width so that all the characters entered are visible.

On exiting input, that's to say when you exit the Entry Field, it reverts to its initial size.

DCA_EF_CENTER

Lets you centre the text.

DCA_EF_LEFT

Lets you justify the text at left.

DCA_EF_RIGHT

Lets you justify the text at right.

DCA_EF_MARGIN

Lets you draw a margin round an input field.

DCA_EF_REQUIRED

Lets you force entry in the Entry Field.

DCA_EF_DISABLE

Lets you disable the control.

DCA_EF_LOCKTEXT

Lets you lock input into a field.

DCA_EF_FULLTEXT

Lets you force the user to enter the number of characters specified in the *Max. length* field.

DCA_EF_UPCASE

Lets you convert characters entered in lower case into upper case automatically.

DCA_EF_SKIPBLANKS

Lets you remove spaces entered at the start and end of the field automatically.

DCA_EF_NOBLANKS

Lets you prevent input of spaces.

DCA_EF_OVER

Lets you display the control with a raised appearance.

DCA_EF_BELOW

Lets you display the control with a sunken appearance.

DCA_EF_AUTOTAB

To pass the focus to the next field automatically.

DCA_EF_HIDETEXT

Lets you enter passwords: the characters entered are not displayed but are replaced with '*'s'.

DCA_EF_INTEGER

Forces the input of an integer.

DCA_EF_NUMBER

Forces the input of a real number.

DCA_EF_DATE

Forces the input of a date.

DCA_EF_TIME

Forces the input of an hour.

DCA_EF_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_EF_AUTOSCROLL	1	
CONST DCA_EF_HORZSCROLL	2	
CONST DCA_EF_VERTSCROLL	4	
CONST DCA_EF_EXPLODING	8	
CONST DCA_EF_CENTER	16	
CONST DCA_EF_LEFT	32	
CONST DCA_EF_RIGHT	64	
CONST DCA_EF_MARGIN	128	
CONST DCA_EF_REQUIRED	256	
CONST DCA_EF_DISABLE	512	
CONST DCA_EF_LOCKTEXT	1024	
CONST DCA_EF_FULLTEXT	2048	;Force Fill
CONST DCA_EF_UPCASE	4096	
CONST DCA_EF_SKIPBLANKS	8192	
CONST DCA_EF_NOBLANKS	16384	
CONST DCA_EF_OVER	32768	
CONST DCA_EF_BELOW	65536	
CONST DCA_EF_AUTOTAB	131072	
CONST DCA_EF_HIDETEXT	262144	
CONST DCA_EF_INTEGER	524288	
CONST DCA_EF_NUMBER	1048576	
CONST DCA_EF_DATE	2097152	
CONST DCA_EF_TIME	4194304	
CONST DCA_EF_HIDDEN	8388608	

DCA_LB_* Constants

These constants let you implement Dynamic Control Attributes for the List Box control.

Syntax	DCA_LB_DISABLE
	DCA_LB_NOADJUSTPOS
	DCA_LB_MULTIPLESEL
	DCA_LB_USERDRAW
	DCA_LB_HORZSCROLLBAR
	DCA_LB_REALTIME
	DCA_LB_OVER
	DCA_LB_BELOW
	DCA_LB_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_LB_DISABLE

Lets you disable the control.

DCA_LB_NOADJUSTPOS

Lets you avoid adjusting the height of the list.

DCA_LB_MULTIPLESEL

Lets you select several lines.

DCA_LB_USERDRAW

DCA_LB_HORZSCROLLBAR

Lets you display a horizontal scroll bar in a list.

DCA_LB_REALTIME

Means that list scrolling is synchronized with the movement of the scroll box.

DCA_LB_OVER

Lets you display the control with a raised appearance.

DCA_LB_BELOW

Lets you display the control with a sunken appearance.

DCA_LB_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

```
CONST DCA_LB_DISABLE           1
CONST DCA_LB_NOADJUSTPOS      2
CONST DCA_LB_MULTIPLESEL      4
CONST DCA_LB_USERDRAW         8
CONST DCA_LB_HORZSCROLLBAR    16
CONST DCA_LB_REALTIME         32
CONST DCA_LB_OVER             64
CONST DCA_LB_BELOW           128
CONST DCA_LB_HIDDEN          8388608
```

Example

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
i% = 2000
new @Properties
Properties.Kind=DI_LISTBOX%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - 230 - GetTotalDynamicControls% *
25
Properties.Width=CtrlWidth%
Properties.Height=200
Properties.TAB=102
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Listbox"
Properties.Attributes = DCA_LB_HORZSCROLLBAR + DCA_LB_BELOW
Properties.SEPARATORS = " '|', #7 "
Properties.TABULATIONS ="0,100,100"
Properties.Text="Nom|Prenom"
j% =DC_CreateControl% (Properties)
insert at END "Jean|Dupont" to Properties.Ctrl
dispose @Properties
```

DCA_MLE_* Constants

These constants let you implement Dynamic Control Attributes for the MLE control.

Syntax

DCA_MLE_LOCKTEXT

DCA_MLE_DISABLE

DCA_MLE_UPCASE

DCA_MLE_ACTIVETAB

DCA_MLE_OVER

DCA_MLE_BELOW

DCA_MLE_WORDWRAP

DCA_MLE_NOADJUSTPOS

DCA_MLE_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_MLE_LOCKTEXT

Lets you lock input.

DCA_MLE_DISABLE

Lets you disable the control.

DCA_MLE_UPCASE

Lets you convert characters entered in lower case into upper case automatically.

DCA_MLE_ACTIVETAB

Lets you force the use of the [Tab] key for moving the focus to the next field. When it is not checked (by default), the Tab key lets you move the cursor within the MLE to the next tab position, the width of which is specified in Tab stop interval.

DCA_MLE_OVER

Lets you display the control with a raised appearance.

DCA_MLE_BELOW

Lets you display the control with a sunken appearance.

DCA_MLE_WORDWRAP

Lets you turn on automatic word wrap.

DCA_MLE_NOADJUSTPOS

Lets you avoid adjusting the height of the list.

DCA_MLE_HIDDEN

Lets you hide the control.

3. They are declared internally as follows:

CONST DCA_MLE_LOCKTEXT	1	
CONST DCA_MLE_DISABLE	2	
CONST DCA_MLE_UPCASE	4	
CONST DCA_MLE_ACTIVETAB	8	;Disable tab stops
CONST DCA_MLE_OVER	16	
CONST DCA_MLE_BELOW	32	
CONST DCA_MLE_WORDWRAP	64	
CONST DCA_MLE_NOADJUSTPOS	128	
CONST DCA_MLE_HIDDEN	8388608	

DCA_CB_* Constants

These constants let you implement Dynamic Control Attributes for the Combo Box control.

Syntax **DCA_CB_DISABLE**
DCA_CB_OVER
DCA_CB_BELOW
DCA_CB_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_CB_DISABLE

Disables the control. Determines the state of the control on start-up of the dialog box to which it belongs.

DCA_CB_OVER

Lets you display the control with a raised appearance.

DCA_CB_BELOW

Lets you display the control with a sunken appearance.

DCA_CB_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_CB_DISABLE	1
CONST DCA_CB_OVER	16
CONST DCA_CB_BELOW	32
CONST DCA_CB_HIDDEN	8388608

DCA_CHK_* Constants

These constants let you implement Dynamic Control Attributes for the Check Box control.

Syntax

DCA_CHK_AUTOSIZE

DCA_CHK_DISABLE

DCA_CHK_3STATES

DCA_CHK_BELOW

DCA_CHK_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_CHK_AUTOSIZE

Lets you adjust the size of the Check Box to the text displayed.

DCA_CHK_DISABLE

Permet de désactiver le contrôle.

DCA_CHK_3STATES

Lets you define a 3 state Check Box. Most Check Boxes are 2 state: checked or not checked. A 3 state Check Box can be obtained (checked, not checked, greyed out).

DCA_CHK_BELOW

Lets you display the control with a sunken appearance.

DCA_CHK_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_CHK_AUTOSIZE	1
CONST DCA_CHK_DISABLE	2
CONST DCA_CHK_3STATES	4
CONST DCA_CHK_OVER	8
CONST DCA_CHK_BELOW	16
CONST DCA_CHK_HIDDEN	8388608

DCA_RB_* Constants

These constants let you implement Dynamic Control Attributes for the Radio Button control.

Syntax

DCA_RB_DISABLE

DCA_RB_AUTOSIZE

DCA_RB_OVER

DCA_RB_BELOW

DCA_RB_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_RB_DISABLE

Lets you disable the control.

DCA_RB_AUTOSIZE

Lets you adjust the size of the Radio Button to the text displayed.

DCA_RB_OVER

Lets you display the control with a raised appearance.

DCA_RB_BELOW

Lets you display the control with a sunken appearance.

DCA_RB_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

```
CONST DCA_RB_DISABLE           1
CONST DCA_RB_AUTOSIZE          2
CONST DCA_RB_OVER               4
CONST DCA_RB_BELOW              8
CONST DCA_RB_HIDDEN            8388608
```

DCA_ST_* Constants

These constants let you implement Dynamic Control Attributes for the Static Text control.

Syntax	DCA_ST_CENTER
	DCA_ST_LEFT
	DCA_ST_RIGHT
	DCA_ST_HALFTONE
	DCA_ST_MNEMONIC
	DCA_ST_ERASERECT
	DCA_ST_WORDWRAP
	DCA_ST_AUTOSIZE
	DCA_ST_MARGIN
	DCA_ST_OVER
	DCA_ST_BELOW
	DCA_ST_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_ST_CENTER

Lets you centre the text.

DCA_ST_LEFT

Lets you justify the text at left.

DCA_ST_RIGHT

Lets you justify the text at right.

DCA_ST_HALFTONE

Lets you grey out the text in a text zone.

DCA_ST_MNEMONIC

Lets you assign a keyboard shortcut to a Static Text control. For example, in the case of a control ~Exit, the shortcut [Alt]+E lets you position the focus on the control associated with it.

DCA_ST_ERASERECT

Lets you erase the rectangular background behind a text zone.

DCA_ST_WORDWRAP

Lets you define a text zone with several lines.

DCA_ST_AUTOSIZE

Lets you make all the text visible.

DCA_ST_MARGIN

Lets you frame the Static Text control.

DCA_ST_OVER

Lets you display the control with a raised appearance.

DCA_ST_BELOW

Lets you display the control with a sunken appearance.

DCA_ST_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_ST_CENTER	1
CONST DCA_ST_LEFT	2
CONST DCA_ST_RIGHT	4
CONST DCA_ST_HALFTONE	8
CONST DCA_ST_MNEMONIC	16
CONST DCA_ST_ERASERECT	32
CONST DCA_ST_WORDWRAP	64
CONST DCA_ST_AUTOSIZE	128
CONST DCA_ST_MARGIN	256
CONST DCA_ST_OVER	512
CONST DCA_ST_BELOW	1024
CONST DCA_ST_HIDDEN	8388608

DCA_BMP_* Constants

These constants let you implement Dynamic Control Attributes for the Bitmap control.

Syntax

DCA_BMP_ADJUSTSIZE

DCA_BMP_DISABLE

DCA_BMP_ICON

DCA_BMP_PUSHBUTTON

DCA_BMP_CHECKBOX

DCA_BMP_NOCHECKING

DCA_BMP_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_BMP_ADJUSTSIZE

To adjust the size of a bitmap.

DCA_BMP_DISABLE

To disable the control.

DCA_BMP_ICON

To associate Icon control behaviour with a Bitmap.

DCA_BMP_PUSHBUTTON

To associate Push Button control behaviour with a Bitmap.

DCA_BMP_CHECKBOX

To associate Check Box control behaviour with a Bitmap.

DCA_BMP_NOCHECKING

To prevent a Bitmap exhibiting Check Box behaviour.

DCA_BMP_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

```
CONST DCA_BMP_ADJUSTSIZE      1
CONST DCA_BMP_DISABLE        2
CONST DCA_BMP_ICON            4
CONST DCA_BMP_PUSHBUTTON      8
CONST DCA_BMP_CHECKBOX        16
CONST DCA_BMP_NOCHECKING      32
CONST DCA_BMP_HIDDEN          8388608
```

Example

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - 25 - GetTotalDynamicControls% *
25
Properties.Kind=DI_BITMAP%
Properties.Tab =102
Properties.Width=32
Properties.Height=32
h% = createbmp%(" (NS-BMP)\ADD.BMP")
h1% = createbmp%(" (NS-BMP)\CLOSE.BMP")
h2% = createbmp%(" (NS-BMP)\CANCEL.BMP")
Properties.disabled=':&h%
Properties.pressed=':&h1%
Properties.released=':&h2%
Properties.Attributes = DCA_BMP_PUSHBUTTON+DCA_BMP_ADJUSTSIZE
j% =DC_CreateControl% (Properties)
Properties.CTRL.FORECOLOR=TRANSP_TOPLEFT%
Properties.CTRL.Relief=2
dispose @Properties
```

See also

DC_ControlProperties

DCA_GB_* Constants

These constants let you implement Dynamic Control Attributes for the Group Box control.

Syntax	DCA_GB_CENTER
	DCA_GB_LEFT
	DCA_GB_RIGHT
	DCA_GB_HALFTONE
	DCA_GB_MNEMONIC
	DCA_GB_ERASERECT
	DCA_GB_OVER
	DCA_GB_BELOW
	DCA_GB_OVER1
	DCA_GB_BELOW1
	DCA_GB_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_GB_CENTER

Lets you centre the text.

DCA_GB_LEFT

Lets you justify the text at left.

DCA_GB_RIGHT

Lets you justify the text at right.

DCA_GB_HALFTONE

Lets you grey out the text in a text zone.

DCA_GB_MNEMONIC

Lets you assign a keyboard shortcut to a Group Box control.

DCA_GB_ERASERECT

Lets you erase the rectangular background behind a text zone.

DCA_GB_OVER

Lets you display the control with a raised appearance.

DCA_GB_BELOW

Lets you display the control with a sunken appearance.

DCA_GB_OVER1

Lets you display the control with a raised appearance without text. Warning, the text will no longer be visible.

DCA_GB_BELOW1

Lets you display the control with a sunken appearance without text. Warning, the text will no longer be visible.

DCA_GB_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_GB_CENTER	1	; Center aligned Text
CONST DCA_GB_LEFT	2	; Left aligned Text
CONST DCA_GB_RIGHT	4	; Right aligned Text
CONST DCA_GB_HALFTONE	8	; Half Tone
CONST DCA_GB_MNEMONIC	16	; shortcut sensitive (always)
CONST DCA_GB_ERASERECT	32	; Erase Rectangle
CONST DCA_GB_OVER	64	; Shadow Light (raised)
CONST DCA_GB_BELOW	128	; Shadow Dark (sunken)
CONST DCA_GB_OVER1	256	; Shadow Light'opt (raised)
CONST DCA_GB_BELOW1	512	; Shadow Dark'opt (sunken)
CONST DCA_GB_HIDDEN	8388608	

DCA_VS_* Constants

These constants let you implement Dynamic Control Attributes for the Vscroll control (vertical scroll bar).

Syntax **DCA_VS_DISABLE**
 DCA_VS_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_VS_DISABLE

Lets you disable the control.

DCA_VS_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

CONST DCA_VS_DISABLE	1
CONST DCA_VS_HIDDEN	8388608

DCA_HS_* Constants

These constants let you implement Dynamic Control Attributes for the Hscroll control (horizontal scroll bar).

Syntax **DCA_HS_DISABLE**
 DCA_HS_HIDDEN

Notes

1. The meaning of these constants is as follows:

 DCA_HS_DISABLE

 Lets you disable the control.

 DCA_HS_HIDDEN

 Lets you hide the control.

2. They are declared internally as follows:

```
CONST DCA_VS_DISABLE            1
CONST DCA_VS_HIDDEN            8388608
```

DCA_MNU_* Constants

These constants let you implement Dynamic Control Attributes for the Menu control.

Syntax

DCA_MNU_RIGHTSIDE

DCA_MNU_SEPARATOR

DCA_MNU_DISABLED

DCA_MNU_CHECKED

DCA_MNU_SUBMENU

DCA_MNU_HIDDEN

Notes

1. The meaning of these constants is as follows:

DCA_MNU_RIGHTSIDE

Lets you display a menu item to the right of the menu bar.

DCA_MNU_SEPARATOR

Lets you display a separator bar after a menu option.

DCA_MNU_DISABLED

Lets you disable the control.

DCA_MNU_CHECKED

Lets you check a menu option when the window opens.

DCA_MNU_SUBMENU

Lets you define a sub-menu

DCA_MNU_HIDDEN

Lets you hide the control.

2. They are declared internally as follows:

```
CONST DCA_MNU_RIGHTSIDE      1
CONST DCA_MNU_SEPARATOR      2 ; as well as MK_SEPARATOR%
CONST DCA_MNU_DISABLED       4 ; as well as MK_DISABLED%
```

7-26 *NSDYNCTR Library*

CONST DCA_MNU_CHECKED	8 ; as well as MK_CHECKED%
CONST DCA_MNU_SUBMENU	16; as well as MK_SUBMENU%
CONST DCA_MNU_HIDDEN	8388608

DI_WINCMNCTRL% Constant

Constant used to indicate the value of a Common Control.

Syntax **DI_WINCMNCTRL%**

Note

1. When the dynamic parameter .KIND=DI_WINCMNCTRL, then the subtype (.SUBKIND) equals one of the following :
 - DI_WCC_Animation%
 - DI_WCC_DateTimePicker%
 - DI_WCC_HotKey%
 - DI_WCC_ListView%
 - DI_WCC_MonthCalendar%
 - DI_WCC_ProgressBar%
 - DI_WCC_TrackBar%
 - DI_WCC_TreeView%
2. The following constants correspond to the common styles of all Windows Common Controls:
 - WCC_Transparent
Allows you to combine the control's background colour with the main colour of the window.
 - WCC_ModalFrame
Draws the picture of the control border in relief.
 - WCC_ClientEdge
Draws the picture of the control border in sunken.

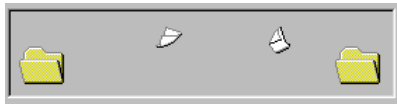


Figure -Example of an Animation control

- WCC_StaticEdge

Draws the picture of the control border in sunken.



Figure -Example of an Animation control

- **WCC_HIDDEN**

Lets you hide the control.

3. They are declared internally as follows:

```
CONST WCC_Transparent      1
CONST WCC_ModalFrame      32
CONST WCC_ClientEdge      512
CONST WCC_StaticEdge      131072
CONST WCC_HIDDEN          8388608
```

Example

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.Kind=DI_WINCMNCTRL%
Properties.Subkind =DI_WCC_DateTimePicker%
Properties.Style =DTS_LONGDATEFORMAT
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - Y%
Properties.Width=CtrlWidth%
Properties.Height=20
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Date Time Picker"
j% =DC_CreateControl% (Properties)

dispose @Properties
```

DI_WCC_Animation% Constant

Constant indicating an *Animation* control.

Syntax **DI_WCC_Animation%**

Note

1. The following constants associated with the DI_WCC_Animation% constant allow you to set the *Animation* control:
 - ACS_AUTOPLAY
Allows you to run the animation automatically.
 - ACS_TIMER
Allow you to set the number of seconds for which the animation should run.
 - ACS_CENTER
Allow you to center the animation in the *Animation* control window.
 - ACS_TRANSPARENT
Allows you to combine the control's background colour with the main colour of the window.

2. They are declared internally as follows:

CONST ACS_CENTER	\$0001
CONST ACS_TRANSPARENT	\$0002
CONST ACS_AUTOPLAY	\$0004
CONST ACS_TIMER	\$0008

DI_WCC_DateTimePicker% Constant

Constant indicating a *Date Time Picker* control.

Syntax **DI_WCC_DateTimePicker%**

Notes

1. The following constants associated with the DI_WCC_DateTimePicker% constant allow you to set the *Date Time Picker* control:
 - DTS_SHORTDATEFORMAT
Allows you to specify a short date format. (e.g. “26/03/01”)
 - DTS_LONGDATEFORMAT
Allows you to specify a long date format. (e.g. “Tuesday 27 March 2001”)
 - DTS_TIMEFORMAT
Allows you to specify a time format (e.g. “14:53:18”)
 - DTS_UPDOWN
Defines an UpDown control represented by a pair of arrows allowing you to remove or add a value to the associated Date control. This style can be used instead of the calendar, which is the default option.
 - DTS_SHOWNONE
Does not show any date selection in the control. This style displays a checkbox in the control allowing you to confirm a date entered or selected.
 - DTS_SHORTDATECENTURYFORMAT
Allows you to specify the year in four digits (e.g.: “26/03/2001”)
 - DTS_APPCANPARSE
Allows the user to analyse input and to take any necessary action. It authorises users to edit the control’s client area when they press the [F2] key.
 - DTS_RIGHTALIGN
Right alignment (by default, the *Date and Time Picker* control is aligned to

the left).

2. They are declared internally as follows:

Const	DTS_SHORTDATEFORMAT	0
Const	DTS_UPDOWN	1
Const	DTS_SHOWNONE	2
Const	DTS_LONGDATEFORMAT	4
Const	DTS_TIMEFORMAT	8
Const	DTS_SHORTDATECENTURYFORMAT	12
Const	DTS_APPCANPARSE	16
Const	DTS_RIGHTALIGN	32

DI_WCC_Hotkey% Constant

Constant indicating an *HotKey* control.

Syntax **DI_WCC_Hotkey%**

DI_WCC_ListView% Constant

Constant indicating a *ListView* control.

Syntax **DI_WCC_ListView%**

Notes

1. The following constants associated with the DI_WCC_ListView% constant allow you to set the *List View* control:
 - LVS_ICON
Displays the data in the form of large icons.
 - LVS_REPORT
Displays the data in detailed form.
 - LVS_SMALLICON
Displays the data in the form of small icons.
 - LVS_LIST
Displays the data in the form of a list.
 - LVS_TYPEMASK
Specifies an Image List type of mask.
 - LVS_TYPESTYLEMASK
Specifies the style of an Image List type of mask.
 - LVS_ALIGNMASK
Aligns the mask of an Image List.
 - LVS_SINGLESEL
Restricts the selection to a single element.
 - LVS_SHOWSELALWAYS
The selection is always visible, even if the control no longer has the focus.
 - LVS_SORTASCENDING

Sorts the data in ascending order.

- LVS_SORTDESCENDING

Sorts the data in descending order.

- LVS_SHAREIMAGELISTS

The image list is not deleted if the control is deleted. This style allows you to use a single image list for several *List View* controls.

- LVS_NOLABELWRAP

This style allows the control to display the text of the element on a single line in icon view.

- LVS_AUTOARRANGE

This style allows the control to keep an icon view.

- LVS_EDITLABELS

This style allows the control to edit the text for the element in place.

- LVS_OWNERDATA

This style specifies a virtual List View control. This style allows the control to manipulate millions of elements.

- LVS_NOSCROLL

This style disables the control's scrollbar. This style is not compatible with the list or detailed views.

- LVS_ALIGNTOP

The elements are aligned to the top of the list in icon view (large or small).

- LVS_ALIGNLEFT

The elements are aligned to the left of the list in icon view (large or small).

- LVS_OWNERDRAWFIXED.

The owner of the control can draw elements in a detailed view. The ListView control sends a WM_DRAWITEM message to paint each element. It does not send separate messages for each of the sub-elements.

- LVS_NOCOLUMNHEADER

This style allows you to disable the displaying of column headers in a detailed view.

- LVS_NOSORTHEADER.

The column headers are not clickable and do not therefore allow you to sort by column in a detailed view.

2. They are declared internally as follows:

CONST LVS_ICON	\$0000
CONST LVS_REPORT	\$0001
CONST LVS_SMALLICON	\$0002
CONST LVS_LIST	\$0003
CONST LVS_TYPEMASK	\$0003
CONST LVS_SINGLESEL	\$0004
CONST LVS_SHOWSELALWAYS	\$0008
CONST LVS_SORTASCENDING	\$0010
CONST LVS_SORTDESCENDING	\$0020
CONST LVS_SHAREIMAGELISTS	\$0040
CONST LVS_NOLABELWRAP	\$0080
CONST LVS_AUTOARRANGE	\$0100
CONST LVS_EDITLABELS	\$0200
CONST LVS_OWNERDATA	\$1000
CONST LVS_NOSCROLL	\$2000
CONST LVS_TYPESTYLEMASK	\$fc00
CONST LVS_ALIGNTOP	\$0000
CONST LVS_ALIGNLEFT	\$0800
CONST LVS_ALIGNMASK	\$0c00
CONST LVS_OWNERDRAWFIXED	\$0400
CONST LVS_NOCOLUMNHEADER	\$4000
CONST LVS_NOSORTHEADER	\$8000

DI_WCC_MonthCalendar% Constant

Constant indicating a *Month Calendar* control.

Syntax **DI_WCC_MonthCalendar%**

Notes

1. The following constants associated with the DI_WCC_MonthCalendar% constant allow you to set the *Month Calendar* control:
 - MCS_DAYSTATE

This style displays some dates in bold according to the NOTIFY MCN_GETDAYSTATE% messages sent by the control.
 - MCS_MULTISELECT

This style allows you to select a date interval inside the control. By default, the maximum interval is one week.
 - MCS_WEEKNUMBERS

This style indicates the week number (1-52) to the left of each line.
 - MCS_NOTODAYCIRCLE

This style deactivates the circle around the day's date.
 - MCS_NOTODAY

This style does not display the day's date at the bottom of the control.
2. They are declared internally as follows:

CONST MCS_DAYSTATE	1
CONST MCS_MULTISELECT	2
CONST MCS_WEEKNUMBERS	4
CONST MCS_NOTODAYCIRCLE	8
CONST MCS_NOTODAY	16

DI_WCC_ProgressBar% Constant

Constant indicating a *Progress Bar* control.

Syntax **DI_WCC_ProgressBar%**

Notes

1. The following constants associated with the DI_WCC_ProgressBar% constant allow you to set the *Progress Bar* control:

- PBS_SMOOTH

Displays a smooth progress bar.

- PBS_VERTICAL

Displays a progress bar with a vertical axis.

2. They are declared internally as follows:

CONST	PBS_SMOOTH	\$01
CONST	PBS_VERTICAL	\$04

DI_WCC_TrackBar% Constant

Constant indicating an *Track Bar* control.

Syntax **DI_WCC_TrackBar%**

Notes

1. The following constants associated with the DI_WCC_TrackBar% constant allow you to set the *Track Bar* control:
 - TBS_AUTOTICKS
The control displays a tick for each value.
 - TBS_VERT
This style orientates the *Track Bar* control vertically.
 - TBS_HORZ
This style orientates the *Track Bar* control horizontally.
 - TBS_TOP
The ticks are located above.
 - TBS_BOTTOM
The ticks are located below.
 - TBS_LEFT
The ticks are located to the left.
 - TBS_RIGHT
The ticks are located to the right.
 - TBS_BOTH
The ticks are located above and below the cursor.
 - TBS_NOTICKS
Allows you to display no ticks.
 - TBS_ENABLESELRANGE

The Track Bar control displays a single selection. The selection is highlighted.

- TBS_FIXEDLENGTH

The control allows you to set the size of the cursor.

- TBS_NOTHUMB

The control does not display a cursor.

- TBS_TOOLTIPS

This style automatically creates a tooltip displaying the current position of the cursor.

- TBS_REVERSED

Allows you to select the tinier value (most of the time it corresponds to the minimal value) as the higher value and the highest value (corresponding in general with the maximal value) as the tinier value.

- TBS_DOWNSIZELEFT

Allows you to define the bottom at left and the top at right (by default, it is reversed).

2. They are declared internally as follows:

CONST TBS_AUTOTICKS		\$0001
CONST TBS_VERT		\$0002
CONST TBS_HORZ		\$0000
CONST TBS_TOP		\$0004
CONST TBS_BOTTOM		\$0000
CONST TBS_LEFT		\$0004
CONST TBS_RIGHT		\$0000
CONST TBS_BOTH		\$0008
CONST TBS_NOTICKS	\$0010	
CONST TBS_ENABLESELRANGE		\$0020
CONST TBS_FIXEDLENGTH		\$0040
CONST TBS_NOTHUMB	\$0080	
CONST TBS_TOOLTIPS		\$0100
CONST TBS_REVERSED		\$0200
CONST TBS_DOWNSIZELEFT		\$0400

DI_WCC_TreeView% Constant

Constant indicating an *Tree View* control.

Syntax **DI_WCC_TreeView%**

Notes

1. The following constants associated with the DI_WCC_TreeView% constant allow you to set the *Tree View* control:
 - **TVS_HASBUTTONS**
Displays (+) and (-) alongside the list's parents elements. They allow you to expand or fold child elements.
 - **TVS_HASLINES**
Uses lines to allow a better view of the hierarchy of the elements.
 - **TVS_LINESATROOT**
Uses vertical lines to connect the elements to the root element of the list. This style is only effective if the *Has lines* style is also specified.
 - **TVS_EDITLABELS**
Editing the properties of the label of the selected element.
 - **TVS_DISABLEDRAHDROP**
Disables the drag and drop option.
 - **TVS_SHOWSELALWAYS**
 - **TVS_RTLREADING**
Specifies the display for a reading from right to left.
 - **TVS_NOTOOLTIPS**
Tooltips are disabled.
 - **TVS_CHECKBOXES**
Enables checkboxes for *Tree View* control elements. A checkbox is displayed only if an image is associated with the element.

- TVS_TRACKSELECT

Enables the highlighted displaying of nodes pointed to by the mouse in a *Tree View* control.

- TVS_SINGLEEXPAND

Leads to the extension of the selected element and folds all the others.

- TVS_INFOTIP

Imports the information from the tooltip.

- TVS_FULLROWSELECT

Selects the whole line. Enables the complete selection of the line in a tree structure view. The whole line of the selected element is highlighted whichever element on the line is selected. This style can not be used with the STV_HASLINES constant.

- TVS_NONEVENHEIGHT

Sets the height of the elements to an uneven height with the TVM_SETITEMHEIGHT% message. By default, the height of the elements has an even value.

- TVS_NOHSCROLL

Allows you to disable the horizontal scrolling.

- TVS_NOScroll

Disables horizontal and vertical scrolling. The control does not display any scrollbars.

2. They are declared internally as follows:

CONST TVS_HASBUTTONS		\$0001
CONST TVS_HASLINES		\$0002
CONST TVS_LINESATROOT		\$0004
CONST TVS_EDITLABELS		\$0008
CONST TVS_DISABLEDRAHDROP		\$0010
CONST TVS_SHOWSELALWAYS		\$0020
CONST TVS_RTLREADING		\$0040
CONST TVS_NOTOOLTIPS		\$0080
CONST TVS_CHECKBOXES		\$0100
CONST TVS_TRACKSELECT		\$0200
CONST TVS_SINGLEEXPAND		\$0400
CONST TVS_INFOTIP	\$0800	
CONST TVS_FULLROWSELECT		\$1000
CONST TVS_NOScroll		\$2000
CONST TVS_NONEVENHEIGHT		\$4000
CONST TVS_NOHSCROLL		\$8000

DC_ControlProperties SEGMENT

The segment DC_ControlProperties is passed as a parameter to the various DC_* functions. It lets you create dynamic controls according to the properties set.

Syntax **SEGMENT DC_ControlProperties**

POINTER DialogHandle

INTEGER Kind

INTEGER ID

INTEGER X

INTEGER Y

INTEGER Width

INTEGER Height

INTEGER Attributes

INTEGER ForeColor

INTEGER BackColor

CSTRING FontName

CSTRING FontSize

CSTRING FontSels

CSTRING Anchor

CSTRING ToolTip

INTEGER Tab

INTEGER VALUE

CONTROL Ctrl

POINTER CtrlHandle

INTEGER ParentID

INTEGER Picture

INT(4) MaxLen

CSTRING Characters

CSTRING Format

CSTRING TEXT

CSTRING SEPARATORS

CSTRING TABULATIONS

INT(2) TabStopInterval

INT(1) NBLINES

INTEGER GROUPID

CSTRING Released

CSTRING Pressed

CSTRING Disabled

INTEGER SubKind

INTEGER style

INTEGER exstyle

INT(1) Version

INT(2) Internal%

INTEGER Internal2%

POINTER Data

CSTRING Paramstr

ENDSEGMENT

7-44 NSDYNCTR Library

Fields	<i>DialogHandle</i>	POINTER	NSHandle of the parentWindow (self%)
	<i>Kind</i>	INTEGER	a DI_* constant (NsMisc library)
	<i>ID</i>	INTEGER	id of the control
	<i>X</i>	INTEGER	X coordinate
	<i>Y</i>	INTEGER	Y coordinate
	<i>Width</i>	INTEGER	width (except for icons and vertical scrolling)
	<i>Height</i>	INTEGER	height for Push Buttons, ListBox, ComboBox, Combo Entry, MLE, GroupBox, Static Text, Vertical Scrollbar, Bitmaps
	<i>Attributes</i>	INTEGER	a DCA_* constant Disabled Autosize etc
	<i>ForeColor</i>	INTEGER	foreground colour (doesn't work with Icon and Bitmap controls)
	<i>BackColor</i>	INTEGER	background colour (doesn't work with Icon and Bitmap controls)
	<i>FontName</i>	CSTRING	name of the font (doesn't work with Icon and Bitmap controls)
	<i>FontSize</i>	CSTRING	size of the font (doesn't work with Icon and Bitmap controls)
	<i>FontSels</i>	CSTRING	GFS_%% constant (NSGraph library). (doesn't work with Icon and Bitmap controls)
	<i>Anchor</i>	CSTRING	a NS_AS_* constant (NSMisc library)
	<i>ToolTip</i>	CSTRING	tooltip (doesn't work with Icon controls)
	<i>Tab</i>	INTEGER	last control in the tabulation sequence or the ID of the control to which the Static text or Group Box is connected. If you refer to a control in this parameter, make sure it really exists.
	<i>VALUE</i>	INTEGER	value of the RadioButton, CheckBox, Icon. Select a SPTR_%% constant

<i>Ctrl</i>	CONTROL	control created
<i>CtrlHandle</i>	POINTER	handle of the created control
<i>ParentID</i>	INTEGER	Id of the MenuItem Parent, MenuItem or 0
<i>Picture</i>	INTEGER	a SMIP_ *% constant (NSMisc library)
<i>MaxLen</i>	INT(4)	maximum width of an Entry Field control (by default, its maximum width is 31) or the maximum number of characters in an MLE control.
<i>Characters</i>	CSTRING	characters authorized
<i>Format</i>	CSTRING	format to display
<i>TEXT</i>	CSTRING	text of the control
<i>SEPARATORS</i>	CSTRING	character string with commas separating each character. Eg: " , #7"
<i>TABULATIONS</i>	CSTRING	comma separating the columns. Eg. : "0, 15,100, 100"
<i>TabStopInterval</i>	INT(2)	tab stop interval in an MLE control.
<i>NBLINES</i>	INT(1)	size of a dropdown list in terms of the number of lines for a ComboBox control.
<i>GROUPID</i>	INTEGER	Group control id for Radio Buttons.
<i>Released</i>	CSTRING	full path of a bitmap. Environment variables can be used.
<i>Pressed</i>	CSTRING	full path of a bitmap corresponding to the mouse button sunken. Environment variables can be used.
<i>Disabled</i>	CSTRING	full path of a bitmap corresponding to the mouse button deactivated. Environment variables can be used.
<i>SubKind</i>	INTEGER	a DI_WCC_* constant
<i>Style</i>	INTEGER	a XXS_* constant. The X charcaters correspond to the initials of the control

Exstyle

INTEGER

a WCC_* constant

Example

```
; Création d'un contrôle ComboBox
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY
new @Properties
Properties.Kind=DI_COMBBOX%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=4
Properties.Y=getClientheight% - y%
Properties.Width=CtrlWidth%
Properties.Height=24
y%=y% + Properties.Height + 4
Properties.TAB=102
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.FontName="MS Sans Serif"
Properties.FontSize=8
Properties.FontSels=GFS_BOLD%
Properties.ToolTip="Dynamic Combobox"
Properties.Attributes = DCA_CB_BELOW
Properties.SEPARATORS = " '|', #7 "
Properties.TABULATIONS ="0,100,100"
j% =DC_CreateControl% (Properties)
insert at END "Jean|Dupont" to Properties.Ctrl

dispose @Properties
```


ItExists% Function

Lets you identify the presence of a *MenuItem* control.

Syntax **ItExists%** (*Handle*, *ID*)

Parameters	<i>Handle</i>	POINTER	control's handle
	<i>ID</i>	INTEGER	id

Return value INTEGER

zero, if the control is a *MenuItem*, invalid or does not exist, otherwise returns a value other than zero.

GetTotalDynamicControls% Function

Returns the total number of dynamic controls.

Syntax **GetTotalDynamicControls%**

Return value INTEGER

Example

```
Local DC_ControlProperties@ Properties
LOCAL i%, j%, h%, h1%, h2%
SEND EXECUTED TO PB_DESTROY

new @Properties
Properties.Kind=DI_ICON%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.GroupID =Properties.ID
Properties.X=4
Properties.Y=getClientheight% - y%
Properties.Height=24
Properties.Width=40
y%=y% + Properties.Height + 4
Properties.ForeColor=COL_NEUTRAL%
Properties.BackColor=COL_BACKGROUND%
Properties.Anchor=NS_AS_NONE
Properties.ToolTip="Dynamic Icon"
Properties.Value = SPTR_ARROW%
; be carefull if tab <> 0: the next control in the tab sequence
; should be created otherwise it'll hangs
Properties.Tab = 102
j% =DC_CreateControl% (Properties)
Properties.Kind=DI_ICON%
Properties.DialogHandle=self%
Properties.ID =Firstid% + GetTotalDynamicControls%
Properties.X=Properties.X+4+Properties.Width
Properties.Width=40
Properties.Anchor=NS_AS_NONE
Properties.ToolTip="Dynamic Icon"
Properties.Value = SPTR_ILLEGAL%
Properties.Tab = 103
j% =DC_CreateControl% (Properties)

dispose @Properties
```

DC_CreateControl% Function

Creates a dynamic control that returns the total number of dynamic controls.

Syntax	DC_CreateControl% (<i>DC_ControlProperties</i>)		
Parameter	<i>DC_ControlProperties</i>	SEGMENT	DC_ControlProperties segment
Return value	INTEGER		
See also	DC_ControlProperties segment		

DC_DestroyControl% Function

Deletes a dynamic control that returns the total number of dynamic controls.

Syntax **DC_DestroyControl%** (*HWnd*, *Id%*)

Parameters	<i>HWnd</i>	POINTER	window's handle
	<i>Id%</i>	INT	id of the control to delete

Return value INTEGER

See also DC_ControlProperties segment

DC_GetControlPropertiesFromID Instruction

Lets you retrieve a control's properties.

Syntax **DC_GetControlPropertiesFromID** *ControlProperties*

Parameter *ControlProperties* SEGMENT DC_ControlProperties segment

Note

1. Before calling the instruction DC_GetControlPropertiesFromID, the fields DialogHandle and ID in the DC_ControlProperties segment need to be filled.

Example

```
Global POINTER HCALLBACK
Global Firstid%
SEGMENT Exchange
    INTEGER MSG
    POINTER PARM1
    POINTER PARM2
    DC_ControlProperties@ Properties
ENDSEGMENT

Function OldWndProc(POINTER HWND,INTEGER ID,INTEGER MSG,POINTER
PARM1,POINTER PARM2) RETURN INTEGER DYNAMIC

Function NewWndProc(POINTER HWND,INTEGER ID,INTEGER MSG,POINTER
PARM1,POINTER PARM2) RETURN INTEGER
local DC_ControlProperties@ Properties
local Exchange@ pExchange
    EVALUATE MSG
        WHERE EVENT_EXECUTED%,EVENT_SELECTED%
            if ID >= FIRSTID%
                new @Properties
                Properties.DialogHandle =HWND
                Properties.ID =ID
                DC_GetControlPropertiesFromID Properties
                new @pExchange
                pExchange.MSG = MSG
                pExchange.PARM1 = PARM1
                pExchange.PARM2 = PARM2
                @pExchange.Properties = Properties
                Send USER1, @pExchange, MSG to HWND
                dispose @Properties
                dispose @pExchange
            else
                RETURN CALL_PREVCALLBACK(hCallback, HWND,\
id, msg, parm1, parm2)
            endif
        ENDWHERE
    ELSE
```

7-52 *NSDYNCTR Library*

```
RETURN CALL_PREV CALLBACK(hCallback, HWND, id, msg,  
parm1, parm2)  
ENDEVALUATE  
EndFunction  
Firstid% = 1000
```

See also DC_ControlProperties segment

DC_GetControlProperties Instruction

Allows you to retrieve the control properties.

Syntax **DC_GetControlProperties** *Handle, ControlProperties*

Parameters	<i>Handle</i>	POINTER	control's handle
	<i>ControlProperties</i>	SEGMENT	DC_ControlProperties segment

See also DC_ControlProperties segment

Appendix A

Display Formats



This appendix describes the formats that you can use with the `STRING$` and `ESTRING$` functions in the `NSMisc` library.

Contents

About formats for STRING\$ and ESTRING\$.....	A-3
Date formats A-3	
Time formats A-3	
Formats and meanings	A-4
Examples	A-7

About formats for *STRING\$* and *ESTRING\$*

You can select a format from the pre-defined list, modify a format or create one using the characters and symbols shown below.

Checks on the characters entered and format conversions are performed when the focus leaves the editable field.

Date formats

The date formats are based on a numeric value corresponding to the number of days since January 1, 1900. For dates prior to January 1, 1900, the numeric value is negative and it is possible to go back as far as the beginning of the Gregorian calendar, i.e. January 1, 1583.

E.g.	:	01.1.1900	0
		01.2.1900	1
		01.23.1992	33624
		12.31.1899	-1
		01.1.1583	-115782

This method of conversion is the basis for all date calculations, including differences between two dates. In fact, all dates displayed can be converted into numbers of days calculated from January 1, 1900, be used in a calculation and be reconverted into date format.

You can enter the date in any format that conforms to the defined rules. The conversion will be performed according to the specified format.

For example, if you have defined the format d/m/yy and enter 12041965, 12/4/65 will be displayed.

Time formats

Similarly, time formats are based on a numeric value corresponding to the number of seconds since midnight.

E.g.:	0:00:00	0
	12:00:00	43200
	23:59:59	86399

The table on the next page lists the characters that can be specified in the format description together with their meanings.

The last page of this appendix provides some typical example formats and the resulting value displayed.

Formats and meanings

Format	Meaning
0	<p>Indicates a digit.</p> <p>If the number entered contains less digits than allowed for by the format, zeros are displayed.</p> <p>If the number entered contains more digits after the decimal point than the format allows for, the number will be truncated in line with the specified format, i.e. rounded to the nearest figure.</p> <p>If the number you enter contains more digits before the decimal point than the format allows for, the number will still be displayed in full.</p>
#	<p>Represents a digit and removes any insignificant zeros.</p> <p>If leading zeros appear in the portion before the decimal point, they will be removed.</p> <p>Similarly, if trailing zeros appear in the portion after the decimal point, they will be removed.</p>
.	<p>Character representing the decimal point.</p> <p>Include at least one 0 format after the decimal point to avoid the number ending in a period.</p> <p>Include at least one 0 format before the decimal point to avoid numbers less than 1 beginning with a period.</p> <p>If the format used after the separator is smaller than the number entered, the number will be rounded.</p> <p>If the format used before the separator is smaller than the number entered, the number will be displayed in full.</p>
%	<p>Percentage symbol and operator.</p> <p>Multiplies the number entered by 100 and adds the % sign.</p>
space	<p>Used to separate thousands if the format contains a # or 0 character before and after the space.</p> <p>If the format only contains one space for the first thousand and the number exceeds four digits, a single space will be inserted in line with the format.</p>

Format	Meaning
;	<p>If the format is to differ according to whether the number is positive, negative or equal to 0, separate the formats with a ";".</p> <p>The first format will apply to positive numbers, the second to negative numbers and the third to 0.</p> <p>If a format is followed by a single semi-colon, negative numbers will be ignored.</p> <p>If a format is followed by two semi-colons, negative numbers and zeros will be ignored.</p> <p>If a ";" is placed after positive and negative number formats, null values will be ignored.</p> <p>Two ";" used as a format will ignore all numbers.</p>
[Red]	Displays the text that follows in red.
E-E+e-e+	<p>Scientific notation format.</p> <p>If a 0 or # is placed to the right of the decimal point, an E or e will be displayed representing the exponent.</p> <p>The number of format characters placed to the right of the decimal separator determines the number of digits for the exponent.</p> <p>Place a + or - sign just after the E or e to see the sign displayed with the number.</p>
:eE-+() space	To display an exponent character other than those previously mentioned, use brackets followed by the required character.
"text"	The text appearing between quotes will be displayed, regardless of the characters typed.
M	Displays the month as a number without a leading zero for the months 1 to 9.
MM	Displays the month as a number with a leading zero for the months 1 to 9.
mmm or MMM	Displays the month in abbreviated form (jan, feb, mar...).
mmmm or MMMM	Displays the month in its full form (January, February...).

Format	Meaning
d	Displays the day of the week as a number without a leading zero.
dd	Displays the day of the week as a number with a leading zero.
ddd	Displays the day of the week in abbreviated form (Mon, Tue...).
dddd	Displays the day of the week in full (Monday, Tuesday...).
yy	Displays the year in the form of a two-digit number from 00 à 99.
yyyy	Displays the year in full (4 digits -1900...2040)./Character recognized by the date format as separator. If the date is entered in date format, it will be displayed accordingly. If the date is entered as a number, it will be considered as being the number of days since January 1, 1900 and will be converted.
h	Displays the time without leading zeros for 0 to 9.
hh	Displays the time with a leading zero for 0 to 9.
m	Displays minutes without leading zeros for 0 to 9.
mm	Displays minutes with a leading zero before the figures 0 to 9.
s	Displays seconds without a leading zero before the figures 0 to 9.
ss	Displays seconds with a leading zero before the figures 0 to 9.

Examples

Format	Input	Result
0	23	23
00.000	2.5	02.500
0	-3	-3
0.##	34	34.
	5.1236	5.12
	1.168	1.17
0.0	4.58	4.6
##.##	2.5	2.5
##.##	0.1	.1
##.##	34.1254	34.13
#,000.##	1,321.147	1,321.15
\$#,##0;(\$#,##0)	3,125	\$3,125
	-3,125	(\$3,125)
# ###.00	4567	4 567.00
# ##0 \$	3254	3 254 \$
::	4587.12	no number displayed
00-00-00	123456	12-34-56
0%	0.1234	12%
0.00%	0.1234	12.34%
0.00E+00	1234.568	1.23E+03
###.#	12.5	\$12.5
d-MM-yy	31291	1-09-85

A-8 *Display Formats*

Format	Input	Result
d-mmm-yy	2/12/91	2-dec-91
dd-MM-yyyy	2-12-91	02/12/1991
d mmm yyyy	12/07/91	12 jul 1991
hh:mm	13:25	1:25

Appendix B

NSMisc Error Codes

This appendix lists the various error codes that can be returned by the `MISCERROR%`, `T_ERROR%` and `F_ERROR%` functions in the NSMisc Library. The error codes appear as they are declared internally.

Contents

NSMisc Error Codes B-3

NSMisc Error Codes

CONST NO_ERROR%	0
CONST ERROR_INVALID_FUNCTION%	1
CONST ERROR_INVALID_HANDLE%	2
CONST ERROR_NOT_ENOUGH_MEMORY%	3
CONST ERROR_CANNOT_ALLOC_MEMORY%	4
CONST ERROR_CANNOT_FREE_MEMORY%	5
CONST ERROR_OUT_OF_PAPER%	6
CONST ERROR_INVALID_PARAMETER%	7
CONST ERROR_INVALID_DATA%	8
CONST ERROR_INVALID_NAME%	9
CONST ERROR_DISK_FULL%	10
CONST ERROR_WRITE_PROTECT%	11
CONST ERROR_HARDWARE_FAULT%	12
CONST ERROR_SHARING_VIOLATION%	13
CONST ERROR_LOCK_VIOLATION%	14
CONST ERROR_INVALID_FILE_SPECIFICATION%	15
CONST ERROR_TOO_MANY_OPENED_FILES%	16
CONST ERROR_ACCESS_DENIED%	17
CONST ERROR_FILE_EXISTS%	18
CONST ERROR_FILE_NOT_FOUND%	19
CONST ERROR_FILE_NOT_OPENED%	20
CONST ERROR_BEYOND_END_OF_FILE%	21
CONST ERROR_CANNOT_CREATE_FILE%	22
CONST ERROR_CANNOT_ACCESS_TO_FILE%	23
CONST ERROR_CANNOT_OPEN_FILE%	24
CONST ERROR_CANNOT_READ_FILE%	25
CONST ERROR_CANNOT_WRITE_ON_FILE%	26
CONST ERROR_CANNOT_CLOSE_FILE%	27
CONST ERROR_NO_FILE_FOUND%	28
CONST ERROR_CANNOT_UNLOCK_FILE%	29
CONST ERROR_LOCK_FAILED%	30
CONST ERROR_NOT_LOCKED%	31
CONST ERROR_BAD_FORMAT%	32
CONST ERROR_QUEUE_EMPTY%	33
CONST ERROR_CANNOT_EXEC_PROG%	34
CONST ERROR_CANNOT_CHANGE_DIR%	35
CONST ERROR_CANNOT_MAKE_DIR%	36
CONST ERROR_CANNOT_REM_DIR%	37
CONST ERROR_CANNOT_GET_DIR%	38
CONST ERROR_CANNOT_CHANGE_DISK%	39
CONST ERROR_CANNOT_GET_DISK%	40
CONST ERROR_CANNOT_REN_FILE%	41
CONST ERROR_CANNOT_REM_FILE%	42
CONST ERROR_NO_MORE_CONTROL%	43
CONST ERROR_CANNOT_ACCESS_TO_QUEUE%	44
CONST ERROR_CANNOT_CREATE_QUEUE%	45

B-4 *NSMisc Error Codes*

CONST ERROR_INVALID_UNIT%	46
CONST ERROR_CANNOT_LOAD_MODULE%	47
CONST ERROR_CANNOT_GET_PROC_ADDRESS%	48
CONST ERROR_FAILED_TO_COPY_FILE%	49