Logic Basic Language Reference

Version 1.0.21

Abs Number

Description:

Returns the absolute value of a specified number or expression.

Arguments:

Number: Decimal number

Related commands:

Tan, Sin, Cos, Atn

Example:

```
Write Abs(3.14159)
```

AddIconSysTray

Description:

Add the Logic Basic icon to the systray bar.

Related commands:

MinimizeSysTray

Remarks:

This command is useful for programs that need to be minimized to systray (residents), because it allows the program to be maximized or closed by clicking the right mouse button on the icon.

AddRecord DatabaseName

Description:

Add a new record to the native database, with the data contained in the variables of the fields created in the **DB** command.

Arguments:

DatabaseName: Name of the database.

Related commands:

DB, OpenDB OrderDB, UpdateRecord, ReadRecord, TotalRecordsDB, DeleteRecord, FindDB, CompactDB

Example:

```
Agenda.Sequence = 1
Agenda.Name = "John Robinson"
Agenda.Phone = "1234-5678"
Agenda.Address = "First Avenue"
Agenda.Salary = 1500.37

AddRecord Agenda
```

AddSysStartup ProgramName

Description:

Makes a Logic Basic executable program runs automatically whenever windows starter.

Arguments:

ProgramName: Program name to be included in the Windows registry. This name can be any sentence at the discretion of the developer to describe the program and does not need to provide the name of the excutable file and its path.

Remarks:

This command should be used in programs compiled whith the **.exe** extension.
To disable this feature, use the command **RemoveSysStartup**, passing as argument the name of the program that was set in command **AddSysStartup**.

Related commands:

RemoveSysStartup

And

Description:

Operator used to perform a logical conjunction on two expressions. If both expressions evaluate to **True**, result is **True**. If either expression evaluates to **False**, result is **False**.

Related commands:

Or, If, Else, While

Example:

```
Var X Integer, Y Integer, Z Integer

X = 1; Y = 20; Z = 100
If X = 1 And Y > 10 And Z <> 50
   Write "True"
Else
   Write "False"
EndIf
```

AnimateGif Name, Image, Type

Description:

Component to create animated gifs in active window or Windows desktop.

Arguments:

Name: Nickname to be assigned to the animated gif.
Image: File path of the animated gif with extension **.gif**.
Type: Type of animated gif: 0 – Create animated gif in the Logic Basic active window; 1 – Creates the animated gif on the Windows desktop (float).

Properties:

Lin, Col: Coordinates (line and column) to position the animated gif on the screen.
Image: Defines a new image for the animated gif.
Height: Height of animated gif in lines.
Width: Width of animated gif in columns.
Loop: Sets the number of times the animation will be repeated when using the method **Play**, if the value is equal to 0 (zero) the animation will be repeated indefinitely.
Visible: Sets a value (**True** or **False**) indicating whether an animated gif is visible or hidden.

Methods:

Play: Starts the gif animation.
Stop: Stops the gif animation.
NextFrame: Draw the next frame of the animated gif.
PreviousFrame: Draw the previous frame of the animated gif.
FrameNumber, X: Draws the frame number X of the animated gif.
Front: Move the animated gif to the front of the other animated gifs.
Back: Move the animated gif to the back of the other animated gifs.
Remove: Removes the animated gif from the Logic Basic active window.

Related commands:

ClearWindow, Cls, Picture, DrawPictures, Image, Transition

Example:

```
Var Y Integer, Inc Integer

Window BgImage="Space.jpg"

AnimatedGif Angel, "Angel.gif"

AnimatedGif Angel.Col = 40

Y = 5; Inc = 1
While 1 = 1  'infinite loop
  Cls
  AnimatedGif Angel.NextFrame
  AnimatedGif Angel.Lin = Y
  Y = Y + Inc
  If Y > 21; Inc = -1; EndIf
  If Y < 0; Inc = 1; EndIf
  Wait 50
Loop
```

API Action, Parameters

Description:

Executes a function of the Windows API (Application Programming Language) contained in a DLL.

Arguments:

Action: Keyword to define or execute a particular action, they are:
  Library**:** Library name (or DLL) of the Windows.
  FunctionName**:** API function name contained in the library.
  Execute**:** Executes the API function according to the parameters passed in this argument.
  Argument**:** The function can return multiple arguments, this parameter defines which of them will be placed in the reserved variable **RetStr**.
Arguments: Arguments of the API function, comma separated. The developer can pass the arguments HWND or HDC where necessary, these keywords correspond to the addresses of memory segment of Logic Basic windows.

Example:

```
Var WindowsDirectory String

API Library, "kernel32.dll"
API FunctionName, "GetWindowsDirectoryA"

WindowsDirectory = SeqStr(144, " ")

API Execute, WindowsDirectory, 144

API Argument, 0
Write RetStr

API Argument, 1
Write RetStr
```

Asc Character

Description:

Returns the ASCII code for the character passed as argument.

Arguments:

Character: Any character alphabetic or numeric.

Example:

```
Var X Integer

X = Asc("A")      'Returns 65
Write X
```

Related commands:

Chr

Atn Number

Description:

Returns a decimal specifying the arctangent of a number

Arguments:

Number: Decimal number

Related commands:

Tan, Sin, Cos

Example:

```
Var PI Decimal

PI = Atn(1) * 4
Write "Value of Pi = ", PI
```

BinToDec BinaryValue

Description:

Returns a integer value for the binary value passed as argument.

Arguments:

BinaryValue: String containing a binary value with up to 8 digits.

Related commands:

DecToBin

Example:

```
Var X Integer

X = BinToDec("11111111")
Write "Binary value 11111111 in decimal = ", X
```

Box Name, Line, Column, Height

Description:

Component for creating listbox in the active window.

Arguments:

Name: Name of the listbox attributed to the choice of the developer.

Line, Column: Line and column of the upper left corner of the listbox.
Height: Listbox height in lines.

Properties:

Enabled: Sets the listbox as active or inactive (**True** or **False**).
Tip: Text that appears in a small rectangle below the object when the user's cursor hovers over the listbox for about one second.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Txt: Array of type string that stores the data for each column of the selected row in the listbox, for example, BoxName.Txt(0), BoxName.Txt(1), ...
TotalItems: Returns the number of items in the listbox.
AlterCell: If set to **True**, enables the editing and modification of texts contained in the cells of the listbox, for that, simply double click on the cell. To check if a cell has changed, test the return value of the command **Event("Box")**.
ColBoxRef: Returns the column number of the listbox when clicking on it.
LinBoxRef: Returns the line number of the listbox when clicking on it.
LinListRef: Returns the item number of the selected row of the listbox (for all itens included in the listbox).
TotalLinBox: Returns the number of rows of listbox.
TotalColBox: Returns the number of columns of listbox.
GetText: Returns the text contained in the cell defined by row and column of listbox.
Visible: Sets the listbox as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

Title Description, Width, Alignment, Type: Creates a new column and its title in the listbox. Has the following parameters:
   Description: Column title.
   Width: Column width
   Alignment: Optional. Alignment of the text in the column: **Left**, **Right** or **Center**.
   Type: Optional. If the field is numeric, you should inform the word **Numeric**.
Activate: Activates the listbox.
Add: Text, BackColor, ForeColor, FontName, FontSize, Bold: Inserts a text in a column from the current row of listbox. You should perform this method for each column fo the listbox, for example, if the listbox has 5 columns, you must perform this method 5 times in sequence, in the order of the sequence defined in the method **Title**. Has the following parameters:
   Text: Text to be inserted in cell.
   BackColor: Optional. Sets the background color of the cell.
   CorLetra: Optional. Sets the color of cell text.
   FontName: Optional. Defines the font name of the cell.
   FontSize: Optional. Defines the font size of the cell.
   Bold: Optional. If set to **True**, sets the text of the cell in bold.
GetText, Lin, Col: Reads the text in the cell defined by **Lin** (line) and **Col** (column), the line is on all items included in the listbox.
New: Creates a new row in the listbox and restarts the current column number to zero, ie, the next call to the **Add** method will insert the text in the first column.
Select Lin: Selects the line number **Lin** in the listbox.
Cls: Clear all items from the listbox.
End: This method should be executed after insertion of all items in the listbox.
SetFocus: Moves the cursor focus to the listbox.
Remove: Removes the listbox of the active window.

Remarks:

To check which line was clicked or selected, you must run the command **Wait ClickBox**, so when a row is selected, the reserved variable **RetStr** will contain the text **CLICKBOX** and property **Txt** will be loaded with data from all the columns in the reference line; when you double-click or press **Enter** on the selected line, the reserved variable **RetStr** will contain the text **DOUBLECLICK** and reserved variable **ObjectValue** will contain the name of listbox, and property **Txt** of the listbox will be loaded with data from the selected row.

If a cell in the listbox is changed, the command **Event("Box")** returns the value **Altered**.

Related commands:

Rep

Example:

```
Box Test, 2, 39, 5
Box Test.Title, "Name", 30; Box Test.Title, "Age", 5, Center, Numeric
Box Test.Activate

Box Test.Add "John Robinson"; Box Test.Add "37"
Box Test.NewLine
Box Test.Add "Will Robinson"; Box Test.Add "14"
Box Test.NewLine
Box Test.Add "Zachary Smith"; Box Test.Add "65"
Box Test.NewLine
Box Test.Add "Judy Robinson"; Box Test.Add "18"
Box Test.NewLine
Box Test.Add "Penny Robinson"; Box Test.Add "13"
Box Test.NewLine
Box Test.End

Write "Click and DoubleClick on names!"

RetStr = "CLICKBOX"
While RetStr = "CLICKBOX"
  Wait ClickBox
  Write "Value returned: ", RetStr, " - ", Test.Txt(0), ", ", Test.Txt(1)
Loop

Write "Selected: ", ObjectValue, " - ", Test.Txt(0), ", ", Test.Txt(1)
```

Browser

Description:

Command that creates a web browser in active window.

Methods:

Activate: Line, Column, Height, Width**:** Creates and activates the browser at the position defined by **Line** and **Column**, with size defined by **Height** and **Width**.
URL: Directs the browser to a URL (page or link).
Back: Directs the browser to the previous URL.
Forward: Directs the browser to the next URL.
Stop: Breaks the connection to the URL.
Refresh: Refreshes the current page of the browser.
Home: Directs the browser to the URL of the homepage.

Remarks:

If an error occurs in any command the browser, the error code and the error description will be placed in the reserved variable **RetStr**.

Example:

```
Window Size = 35, 100, Pos=Center,Center

Browser Activate, 2, 2, 33, 95
Browser URL, "http://www.logicbasic.net"
```

Button Name, [Type], Line, Column, Height, Width, Text, Picture, Tip

Description:

Component to create command buttons on the active window.

Arguments:

Name: Name of the command button.
Type: Optional. Type of graphical button, which can vary from **Type1** to **Type8**. The argument **Type0** alows user customization of buttons, so you should use the command **ButtonStyle**.
Line, Column: Coordinates where the button will be created in the active window.
Height, Width: Height and width of the button in lines and columns, these arguments are valid only for the default style buttons.
Text: Optional. Text that will appear on the button.
Picture: Optional. File path of the picture to be displayed on the button. You can also assign to this argument the name of a default picture.
Tip: Optional. Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the button.

Properties:

Enabled: Sets the button as active or inactive (**True** or **False**).
Caption: Sets the text to be displayed on the button.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Picture: File path of the picture to be displayed on the button. You can also assign to this argument the name of a default picture.
BackColor: Background color of the button.
Forecolor: Color of the text to be displayed on the button.
FontName: Font name of the text.
FontSize: Font size of the text.
Bold: If set to **True**, sets the button text in bold.
Italic: If set to **True**, sets the button text in italic.
Visible: Sets the button as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the button.
Remove: Removes the button of the active window.

Remarks:

To check which button was pressed, you must run the command **Wait ClickButton**, that will be awaiting the push of a button, when the button is pressed his name will be placed in the reserved variable **RetStr**.

Related commands:

Wait, ButtonStyle

Example:

```
Button OK, Type1, 2, 2, "Click here!"
Button Exit, 5, 2, 2, 10, "Exit"

Wait ClickButton
Message RetStr
Button OK.Caption = " OK!"
Button Exit.Setfocus
Wait ClickButton
EndWindow
```

ButtonStyle ButtonState = Picture

Description:

Customizes the pictures of buttons of **Type0** (button up, focused, pressed and inactive).

Arguments:

ButtonState: Button state, defined by the following words:
- BtnUp: Picture of the button when it is loose and unfocused.
- BtnHigh: Picture of the button when the mouse hovers over the button.
- BtnFocus: Picture of the button when it is loose and focused.
- BtnDown: Picture of the button when it is pressed.
- BtnDisabled: Picture of the button when it is inactive.

Picture: File path of the picture that represents each button state.

Related commands:

Button

Example:

```
ButtonStyle BtnUp = "BtnUp.gif"
ButtonStyle BtnHigh = "BtnHigh.gif"
ButtonStyle BtnFocus = "BtnFocus.gif"
ButtonStyle BtnDown = "BtnDown.gif"
ButtonStyle BtnDisabled = "BtnDisabled.gif"

Button NewButton, Type0, 5, 30, ""
Wait ClickButton
Button NewButton.Enabled = False
```

Calendar Name, Line, Column, Width

Description:

Component to create calendars for selecting dates.

Arguments:

Name: Name of the calendar.
Line, Column: Coordinates where the calendar will be created in the active window.
Width: Width of the calendar in columns,

Properties:

Enabled: Sets the calendar as active or inactive (**True** or **False**).
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the calendar.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Txt: Returns or sets the calendar date.
Visible: Sets the calendar as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the calendar.
Remove: Removes the calendar of the active window.

Related commands:

Combo, Text, Mask

Example:

```
Calendar Cal, 3, 20

Button OK, 7, 20
Wait ClickButton

Write Cal.Txt
```

Call Program, Mode

Description:

Calls an executable program of the operating system Windows or MS-DOS.

Arguments:

Program: Path of the program to be executed.
Mode: Activation mode of the program window to be executed:
Hide - Run the program without activating window.
Normal - Run the program in a normal window, moving focus to it.
Minimized - Run the program, starting in a minimized window.
Maximized - Run the program, starting in a maximized window.
NormalNoFocus - Window is restored to its most recent size and position. The currently active window remains active.
MinimizedNoFocus - Window is displayed as an icon. The currently active window remains active.

Example:

```
Write Call("Calc.exe", "Normal")
```

CaptureWindow

Description:

From the time this command is executed, the active window can be dragged by clicking and holding the left mouse button.

Related command:

UnCaptureWindow

ChangeIcon IconName, ExecutableName

Description:

Change the icon of an executable program, including programs compiled by Logic Basic.

Arguments:

IconName: Path of the icon file (*.ico).
ExecutableName: Path of the executable program (*.exe).

Related commands:

Icon

Remarks:

This command supports icons with a maximum of 64x64 pixels, truecolor.

Example:

```
ChangeIcon "C:\Temp\Taz.ico", "C:\Temp\.Tazmania.exe"
```

ChangeWallpaper Filename

Description:

Change the wallpaper of the Windows desktop.

Arguments:

Filename: Name of the image file with extension .jpg, .bmp or .gif.

Example:

```
Var S String

S = ChangeWallpaper("Wallpaper.jpg")
If S = True
  Message "Wallpaper changed successfully"
Else
  Message "Error when trying to change wallpaper!"
EndIf
EndWindow
```

Check Name, Line, Column, Text

Description:

Component to create checkboxes in the active window.

Arguments:

Name: Name of the checkbox.
Line, Column: Coordinates where the checkbox will be created in the active window.
Text: Text that will appear to the right of the checkbox.

Properties:

Enabled: Sets the checkbox as active or inactive (**True** or **False**).
BackColor: Background color of the checkbox.
Forecolor: Text color of the checkbox.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the checkbox.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Value: Returns or sets the value of the checkbox: 0=unchecked, 1=checked.
Visible: Sets the checkbox as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the checkbox.
Remove: Removes the checkbox of the active window.

Related commands:

Radio, Frame

Example:

```
Check ChkTest1, 3, 15, "Test check"
Check ChkTest2, 3, 30, "Test check", 1

Button OK, 6, 20
Wait ClickButton

Write "Chk1 = ", ChkTest1.Value
Write "Chk2 = ", ChkTest2.Value
```

Chr Number

Description:

Returns the character corresponding to the ASCII code passed as argument.

Argument:

Number: Any value between 0 and 255.

Related commands:

Asc

Example:

```
Var C String

C = Chr(97)
Write "Character 97 = ", C, Chr(13), "Character 65 = ", Chr(65)
```

Circle Line, Column, Radius, Color

Description:

Draw a circle in the active window.

Arguments:

Line, Column: Coordinate of the center of the circle.
Radius: Radius of the circle.
Color: Color of the circle.

Related commands:

Rectangle, Straight, Point, ReadPoint, RGB

Example:

```
Circle 10, 10, 5, Red
```

ClearWindow

Description:

Clear all texts and drawings of the active window, except the background image.

Remarks:

Alternatively you can use the command **Cls**, which is identical to the command **ClearWindow**.

Related commands:

Window, WindowHeight, WindowWidth, EndWindow

CloseDB DBName

Description:

Closes a native database of the Logic Basic.

Arguments:

DBName: Database name.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, UpdateRecord, DeleteRecord, FindDB, CompactDB, TotalRecordsDB

Example:

CloseDB Customers

CloseFile NickName

Description:

Closes the file defined by nickname passed as argument.

Arguments:

NickName: Nickname of the file defined in command **OpenFile**.

Related commands:

OpenFile, ReadFile, WriteFile, SizeFile

Example:

CloseFile "Test"

Cls

Description:

Clear all texts and drawings of the active window, except the background image.

Remarks:

Alternatively you can use the command **ClearWindow**, which is identical to the command **Cls**.

Related commands:

Window, WindowHeight, WindowWidth, EndWindow

Combo Name, Line, Column, Width

Description:

Component to create combo boxes in the active window.

Arguments:

Name: Name of the combo box.
Line, Column: Coordinates where the combo box will be created in the active window.
Width: Optional. Width of the combo box in columns,

Properties:

Enabled: Sets the combo box as active or inactive (**True** or **False**).
BackColor: Background color of the combo box.
Forecolor: Text color of the combo box.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the component.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Lines: Define the number of lines of the combobox list.
Txt: Returns or sets the text of the combobox.
Visible: Sets the combo box as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

Add: Adds a new item (text) on the combo box list.
SetFocus: Moves the cursor focus to the combo box.
Remove: Removes the combo box of the active window.

Related commands:

Text, Radio, Check, Currency, Calendar

Example:

```
Combo Colors, 3, 15

Combo Colors.Txt = "Yellow"
Combo Colors.Add "Yellow"
Combo Colors.Add "Red"
Combo Colors.Add "Blue"
Combo Colors.Add "Green"
Combo Colors.Lines = 4

Button OK, 7, 20
Wait ClickButton

Message Colors.Txt
```

CompactDB DatabaseName

Description:

Clean records excluded from the native database.

Arguments:

DatabaseName: Database name.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, TotalRecordsDB, DeleteRecord, FindDB, UpdateRecord

Example:

```
CompactDB Customers
```

ConnectFTP FTPAddress, User, Password

Description:

Makes a connection to an FTP address.

Arguments:

FTPAddress: FTP Address to connect.
User: FTP user name.
Password: FTP password.

Related commands:

DownFileFTP, UpFileFTP

Example:

```
ConnectFTP "ftp://ftp.domain.com", "Username", "Password"
```

ConnectP2P IP, Port

Description:

Makes a connection to a remote computer that has a particular IP.

Arguments:

IP: IP number of the remote computer that will work as a server.
Port: Port number to be used by the connection.

Relate commands:

SendP2P, ListenP2P

Example:

```
ConnectP2P "201.24.56.16", 6669
```

ConvStr NumericValue

Description:

Converts a numeric value (integer or decimal) to type **String**.

Arguments:

NumericValue: Numeric value or variable of type integer or decimal.

Related commands:

StrVal

Example:

```
Var X Integer, S String

X = 50 * 3
S = ConvStr(X)
S = "S = " & S
Write "S = ", S
```

Copy SourceFile, DestinationFile

Description:

Copies the source file to the destination file, returns an empty string on success or an error message if it occurs.

Arguments:

SourceFile: Path of the source file.
DestinationFile: Path of the destination file.

Related commands:

MkDir, RmDir, DeleteFile, Directory, FindFile

Cos Number

Description:

Returns a number specifying the cosine of an angle.

Arguments:

Number: Integer or decimal.

Remarks:

To convert degrees to radians, multiply degrees by pi/180. To convert radians to degrees, multiply radians by 180/pi.

Comandos relacionados:

Sin, Tan, Atn

Example:

```
Var X Decimal

X = Cos(1.3)
Write "Cosine of the angle 1.3 = ", X
```

Currency Name, Line, Column, Width

Description:

Component to create text boxes for entering numeric values with decimal places.

Arguments:

Name: Name of the component.
Line, Column: Coordinates where the component will be created in the active window.
Width: Optional. Width of the component in columns.

Properties:

Enabled: Sets the component as active or inactive (**True** or **False**).
Height: Define the height of the component (in lines).
Places: Defines the number of decimal places after the point (or comma depending on locale).
BackColor: Background color of the component.
Forecolor: Text color of the component.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the component.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Value: Returns or sets the value of the numeric textbox.
Visible: Sets the component as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Métodos:

SetFocus: Moves the cursor focus to the numeric text box.
Remove: Removes the component of the active window.

Related commands:

Text, Combo, Mask, Calendar

Example:

```
Currency TxtNumber, 5, 15

Button OK, 9, 19
Wait ClickButton

Message TxtNumber.Value
```

Date

Description:

Returns the system date in the format defined in the Windows regional settings.

Related commands:

Day, Month, Year, DateDiff, DateAdd, RevDate, Weekday, Time

Example:

```
Var Today String

Today = Date()
Write "Today is ", Today
```

DateAdd Interval, Number, InitialDate

Description:

Adds a number of intervals to a date and returns the final date.

Arguments:

Interval:
Second: Number of seconds.
Minute: Number of minutes.
Hour: Number of hours.
Week: Number of weeks.
Weekday: Number of weekdays.
Day: Number of days.
DayYear: Number of days of the year.
Month: Number of months.
Year: Number of years.
Number: Number of intervals.
InitialDate: Initial date to which is added the number of intervals.

Related commands:

Date, Time, DateDiff, WeekDay, Day, Month, Year, RevDate

Example:

```
Var NewDay String

NewDay = DateAdd("Day", 365, "1/1/2016)
Write NewDay
```

DateDiff Interval, Start, End

Description:

Returns the difference between two dates or times.

Arguments:

Interval:
Second: Number of seconds.
Minute: Number of minutes.
Hour: Number of hours.
Week: Number of weeks.
Weekday: Number of weekdays.
Day: Number of days.
DayYear: Number of days of the year.
Month: Number of months.
Year: Number of years.
Start, End: Start and end date or start and end time.

Related commands:

Date, Time, DateAdd, RevDate, WeekDay, Day, Month, Year

Example:

Write DateDiff("Day", Date(), "1/1/2015")

Day Date

Description:

Returns the day of the date passed as argument.

Arguments:

Date: Any string (or function return) that can represent a date. If you enter an empty string as an argument, the command assumes the system date.

Related commands:

Date, Month, Year, DateDiff, DateAdd, Weekday, Time

Example:

Write Day(Date())

DB Name, Directory, Password

Description:

Creates and defines a native database. In the rows below of the **DB** command, it must be informed in each row the field name in quotation marks, the data type and field size (only for fields of type **String**), and the last row should put the command **EndDB**.

Arguments:

Name: Name of the database, which can be any name chosen by the user to characterize it. This name follows the same rule for variable names, ie, there can be no spaces between the letters.
Directory: Name of the directory where the database will be created. Not provide the name of the file, because it will be created automatically by Logic Basic (name of the database and then the extension **.Lbf**).
If the database is used on a network, instead of passing as argument the name of the directory, pass a file name containing the word "**Share"**, eg "Share.txt", then this file should contain the path of the database on the network, and should be placed in the same directory as the executable program of the Logic Basic (for all terminals that use the database on the network).
Password: Password of the database.

Remarks:

The data recorded in the database are encrypted whit an algorithm very safe, and the database can be accessed only with a password.

The data type "**AutoInc**" defines an integer variable that has the **AutoIncrement** property, ie, every time a new record is added to the database, the value of the variable will be incremented automatically. This type of variable can be set only in the first field of the database, and only one field of this type.

Related commands:

OpenDB, OrderDB, AddRecord, UpdateRecord, ReadRecord, TotalRecordsDB, DeleteRecord, FindDB, CompactDB

Example:

```
DB Customers, "C:\Customers", "abc123"
  "Sequence", AutoInc
  "Name", String, 20
  "Phone", String, 15
  "Address", String, 30
  "Salary", Decimal
EndDB
```

DecToBin Value

Description:

Returns the binary value for the integer value passed as argument.

Arguments:

Value: Numeric value that can vary 0-255.

Related commands:

BinToDec

Example:

```
Var S String

S = DecToBin(5)
Write "Value 5 in binary = ", S
```

DeleteFile FileName

Description:

Remove one or more files from the Windows directory.

Arguments:

FileName: Path and file name to remove. You can use the wildcards ***** and **?** to remove multiple files.

Remarks:

Returns the file path if one or more files exist, or an empty string if there is no file that matches the criteria passed as argument.

Related commands:

MkDir, RmDir, Copy, Directory, FindFile

Example:

```
Var S String

S = DeleteFile("C:\Temp\*.*")

If S = ""
  Message "File not found!"
Else
  Message "File deleted successfully!"
EndIf
```

DeleteRecord DatabaseName, RecordNumber

Description:

Delete a record from the native database, in the position defined by **RecordNumber**.

Arguments:

DatabaseName: Name of the database.
RecordNumber: Record number to delete.

Remarks:

Note that when reading the records from the database the program code should ignore deleted records.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, TotalRecordsDB, UpdateRecord, FindDB, CompactDB

Example:

```
DeleteRecord Customers, 3
```

Desktop

Description:

Reserved variable that specifies the directory of the Windows desktop.

Related variables:

ProgramDir, WindowsDir, WindowsPrograms, LBDir

DialogBox Títle, Description, FileTypes, Action

Description:

Opens a dialog box to open or save files, color selection, font selection and printer selection.

Arguments:

Títle: Title of the dialog box.
Description: Description of file types allowed.
FileTypes: Relationship of file types allowed.
Action: **Open:** Dialog box for opening files. **Save:** Dialog box for saving files. **Color**: Dialog box for color selection. **Font**: Dialog box for fonts selection. **Printer**: Dialog box for printer selection.

Return values:

Open and **Save**: Puts the memory reserved **RetStr** the full path of the selected file.
Color: Puts the memory reserved **RetValue** the value of the selected color.
Font: Puts the memory reserved **RetStr** a string with font attributes selected in the following positions: 1-30: Font name; 31-33: Font size; 34-43: Bold; 44-53: Italic.
Printer: Returns **OK** if the printer has been selected, and a empty string if there was cancellation.

Example:

```
DialogBox "Songs", "Sound files (*.mp3, *.wav )", "*.mp3;*.wav", "Open"
Message RetStr

DialogBox "Select color", "", "", "Color"
Window BgColor = RetValue

DialogBox "Select the font", "", "", "Font"
Message RetStr

DialogBox "Select the printer", "", "", "Printer"
If RetStr <> ""
  Printer.Position 0, 0
  Printer.Write "Logic Basic"
  Printer.End
EndIf
```

Directory Path

Description:

Returns a **String** representing the name of a file, directory, or folder that matches a specified pattern or file attribute, or the volume label of a drive.

Arguments:

Path: String or variable that specifies a file name - may include directory or folder, and drive. You can inform the wildcards ***** and **?**. When not informed any arguments, the command returns the next file in the file list (set on the first call to the command), or an empty string when it reaches the end of the list.

Related commands:

MkDir, DeleteFile, RmDir, Copy, FindFile

Example:

```
Var File String

Box Test, 1, 1, 10
Box Test.Title, "C:\Windows\Fonts\*.*", 30
Box Test.Activate

File = Directory("C:\Windows\Fonts\*.*")

While File <> ""
  Box Test.Add File
  Box Test.NewLine
  File = Directory()
Loop
Box Test.End
```

DownFileFTP FileFTP, LocalFile

Description:

Downloads a file from the remote computer.

Arguments:

FileFTP: Path of the file to download from the remote connection.
LocalFile: File path on the local computer where file will be downloaded.

Related commands:

ConnectFTP, UpFileFTP

Example:

```
Var Reply String

ConnectFTP "ftp.kernel.org", "", ""
Write "Wait, downloading the file..."
Reply = DownFileFTP("/pub/site/README", "D:\Temp\README")
Write "Reply: ", Reply
```

DrawPictures Parameter

Description:

Draws all pictures created by the command "**Picture**" in the active window.

Arguments:

Parameter: Optional. If you pass as argument the keyword "**Preserve**", the command will not clear the window before drawing the pictures.

Related commands:

Picute, AnimatedGif, Image, Transition

EndProgram

Description:

Terminates the program, but does not close the main window (parent).

Related commands:

EndWindow

EndWindow

Description:

Definitively ends the execution of the program and closes all windows.

Related commands:

EndProgram

Event Type

Description:

Returns the value of a particular event when it occurs.

Arguments:

Type: Specifies the type of event to be returned.
    EventKey: Returns the code of the pressed key. If the **Shift** or **Ctrl** keys are pressed together with other keys, it returns the key code an then the words **Shift** or **Ctrl**.
    EventClick: Returns the word **CLICK** when a click on the window, or **DOUBLECLICK** when a double click on the window.
    EventResize: Returns the word **RESIZE** when the window is resized.
    EventAnimatedGif: Returns the word **CLICK** when a click on the animated gif, or **DOUBLECLICK** when a double click on the animated gif.

Related commands:

Wait, KeyTest

Example:

```
Var S String

Write "Press any key!"

While 1 = 1
  While S = ""
    S = Event(EventKey)
    Wait Events
  Loop
  Write S
  S = ""
Loop
```

Exp Number

Description:

Returns a Double specifying **e** (the base of natural logarithms) raised to a power.

Arguments:

Number: Integer or floating (decimal).

Related commands:

Log

Example:

```
Var X Decimal

X = Exp(1.3)
Write "Antilogarithm of 1.3 = ", X
```

ExtractFile File1,File2, File3

Description:

This command allows the user to extract files contained in a file encapsulated (created by the command **InsertFile**).

Arguments:

File1: File name to extract.
File2: File path after extracting.
File3: Path of the encapsulated file.

Related commands:

InsertFile

Example:

```
ExtractFile "Picture.jpg", "C:\Temp\Picture.jpg" , "C:\Temp\Package.pack"
```

False, True

Description:

Reserved keywords that return **True** or **False** values respectively.

FindDB DatabaseName, Position, S1, S2, ...

Description:

This command searches for a record (or set of records) in native database, which is equal to the argument passed, or a combination of several arguments passed in sequence (S1, S2, ...). Returns the number of records found, or -1 if no record is found.

Note that the search uses the last ordering defined in command **OrderDB**, so isn't possible to search for data in files with sequential definition.

Arguments:

DatabaseName: Name of the database.
Position: Keyword that defines the position of the record to return:
  First**:** Returns the position of the first record in a found set of records.
  Last**:** Returns the position of the last record in a found set of records.
  Next**:** If not found any records, returns the position of the next record that has the field (or combination of fields) more similar to the argument passed (S1, S2, ...).
S1, S2, ...: String (or combination of strings) to search.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, TotalRecordsDB, DeleteRecord, UpdateRecord, CompactDB

Example:

```
P = FindDB(Customers, "First", "John Robinson", "First Avenue")
If P <> -1
  S = ReadRecord(Customers, P)
  Write Customers.Name
  Write Customers.Phone
  Write Customers.Address
  Write Customers.Salary
Else
  Write "Name not found!"
EndIf
```

FindFile Drive, FileName

Description:

Looks for a file in the specified drive. Returns the full path of the file, or an empty string if not found.

Arguments:

Drive: Letter corresponding to the disk or partition to find the file, eg "**C:**"
FileName: Name of the file to find.

Related commands:

Copy, MkDir, DeleteFile, RmDir, Directory

Example:

```
Var S String

Write "Wait, searching for file..."
Wait Events
S = FindFile("C:", "Win.ini")
Write S
```

Fix Number

Description:

Returns the integer portion of a number.

Arguments:

Number: Floating number (decimal) positive or negative.

Related commands:

Int, Abs, Sgn

Example:

```
Var X Integer

X = Fix(11.95)   'Returns 11
Message X
```

Font WindowName.Attribute = Value

Description:

Changes the character font of a particular window of the Logic Basic.

Type the name of the window to change the font, then type point, the attribute name and its value. Note that the name of the main window is **Parent**.

Attributes:

Name: Font name of the window text.
Size: Font size of the window text.
Bold: If set to **True**, sets the window text in bold.
Italic: If set to **True**, sets the window text in italic.
Color: Color of the window text

Related commands:

Window, Write

Example:

```
Font Parent.Name = "Arial"
Font Parent.Color = Blue
Font Parent.Size = 20
Font Parent.Bold = True
Font Parent.Italic = True
Position 5,10; Write "Logic Basic"
```

Fonts Number

Description:

Returns the name or the font identified by a number in the list of printer fonts (or screen fonts if no printer installed).

Arguments:

Number: Number of the font, which ranges from 0 to the value specified in the reserved variable **TotalFonts**.

Related commands:

Font, Window

Example:

```
Var Count Integer, S String

Count = 0
While Count < TotalFonts
  S = Fonts(Count)
  Write S
  Count++
Loop
```

Format Value, Mask

Description:

Returns a String containing an expression formatted according to instructions contained in a mask.

Arguments:

Value: Number, string, variable, expression, or command that returns value.
Mask: Mask for formatting, which can be written with the character **#** or the number **0** (zero). To format currency, the mask should be written in standard American, eg, point to separate the decimal place and comma to separate the places of hundreds, thousands, etc., but the result is formatted according to the regional options in Windows.

Remarks:

This command can be used to perform rounding of values, for example:

```
Var Value Decimal

Value = 3099.25 / 0.85
```

Value = Format(Value, "#####0.00")

In the example above, the result (value) equals 3646.17647058824, and after formatting the value is rounded to two decimal places resulting in 3646.18.

Example:

Below, some formatting masks and their results:

```
X = Format(1257523.27, "###,###,##0.00" )  'Returns 1,257,523.27

T = Time
Result = Format(T, "hh:mm:ss AMPM")   'Returns "11:08:19 AM"

T = Date
X = Format(T, "dddd, mmm d yyyy")     'Returns "Monday. April 22, 2103"

X = Format(97.5, "##0.00%")   'Returns 97.50%

X = Format("7", "00000")   'Returns 00007

X = Format("Logic Basic", "<")   'Returns "logic basic"

X = Format("Logic Basic", ">")   'Returns "LOGIC BASIC"
```

Pratical example:

```
Var V Decimal, Result String

V = 1500
Result = Format(V, "###,##0.00")  'Result = 1,500.00
Write Result
Result = Format(V, "00,000.00")  'Result = 01,500.00
Write Result
```

Frame Name, Text, Line, Column, Height, Width

Description:

Component to create frames for other components in the active window.

Arguments:

Name: Name of the frame.
Text: Texto a ser apresentado na linha superior da moldura.
Line, Column: Coordinates where the frame will be created in the active window.
Height, Width: Height and width of the frame in lines and columns.

Properties:

Enabled: Sets the frame as active or inactive (**True** or **False**).
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the frame.
Visible: Sets the frame as visible or invisible (**True** or **False**).

Methods:

Remove: Removes the frame of the active window.

Related commands:

Check, Radio, Label

Example:

```
Frame Opc, "Options", 13, 13, 5, 15
```

GetString Key

Description:

Reads a string or text from the Logic Basic registry.

Arguments:

Key: Key name to read the Logic Basic registry.

Related commands:

SaveString

Example:

```
Var S String

S = GetString("ForeColor")
```

GetText Label, PasswordChar

Description:

Displays a small window with a text box, and returns the text written on it.

Arguments:

Label: Text that will appear just above the text box.
PasswordChar: Optional. If passed a character to this argument, the letters of the text will be replaced by this character while typing text.

Remarks:

If the user presses the **Cancel** button, the command returns the word **CANCELED**.

Related commands:

Text, Combo, Mask, Currency

Example:

```
Var Password

Password = GetText("Enter your password:", "*")
Message Password
```

GetWinReg KeyType, Section, Key

Description:

Returns the value of a particular windows registry.

Arguments:

KeyType: Type of the registry key, which can be one of the following names:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

Section: Section where the key is stored.
Key: Name of the registry key.

Related commands:

SaveWinReg, GetString, SaveString

Example:

```
Var S String

S = GetWinReg("HKEY_CURRENT_USER",
"Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders", "Desktop")
Message S
```

GotFocus

Description:

Reserved variable that tells if a component (button, text, etc.) received the cursor focus.

Related variables:

LostFocus, MouseX, MouseY, MouseClick, TestButton, ObjectValue

Example:

```
Text Test1, 1, 2, 15
Text Test2, 3, 2, 15
Button BtnTest, 5, 2, 2, 13, "OK"

Sub Timer_Test(10)
  If GotFocus = "Test1"
    GotFocus = ""; Message "Test1 received focus!"
  EndIf
  If GotFocus = "Test2"
    GotFocus = ""; Message "Test2 received focus!"
  EndIf
  If GotFocus = "BtnTest"
    GotFocus = ""; Message "BtnTest received focus!"
  EndIf
EndSub
```

GoTo Label

Description:

Directs execution to one particular label in the program. To define a tag, you must write a word or number that is unique in the program with **:** (colon) at the end of the word.

Argumentos:

Label: Word or number to direct execution, finalized by character : (colon).

Related commands:

While, If

Example:

```
Var Count Integer

Count = 0
While Count < 20
  Write Count
  If Count = 10; GoTo ENDLOOP; EndIf
  Count++
Loop

ENDLOOP:
Message "Count was finalized with the value = ", Count
```

Gradient StartColor, EndColor

Description:

Paints the active window with a gradient fill. If any argument is not informed, the window will be filled with the default colors (light blue).

Argumentos:

StartColor: Optional. Starting color of the gradient fill.
EndColor: Optional. Final color of the gradient fill.

Related commands:

Window, NewWindos

Example:

```
Gradient
Wait Events
Wait 3000 'Wait 3 seconds
Gradient RGB(255, 200, 200), RGB( 255, 100, 100)
```

Hex Number

Description:

Returns the hexadecimal code of a certain number.

Argument:

Number: Integer number or integer variable.

Example:

```
Var X String

X = Hex(255)    'Returs FF
Write X
```

Icon FileName

Description:

Loads an icon to the top bar (title bar) of the active window.

Arguments:

FileName: Path of the icon file (**.ICO)**.

Related commands:

ChangeIcon

Example:

```
Icon "C:\Windows\Winupd.ico"
Write "Windows Upgrade", WindowTitle
```

If Condition

Description:

Conditionally executes a group of statements, depending on the value of an expression: if the expression is **True** it executes the group of statements; if the expression is **False**, it doesn't execute the group of statements, or execute other group of statements **(Else**).

Arguments:

Condition: One or more expressions with comparison operators, and in the case of more than one expression, logical operators (And/Or) between them.

Remarks:

Every command **If** must be terminated by a command **EndIf**.
If the expression is false, you can execute other instructions with the command **Else**.

Related commands:

Else, While, And, Or

Example:

```
Var X Integer

Position 1, 1; Write "Enter the value of X:"
Text Test, 1, 23
Button OK, 3, 23

While 1 = 1
  Wait ClickButton
  X = StrVal(Test.Txt)
  X = X % 2        'Returns the remainder of dividing the X to 2
  If X = 0
      Message "The value of X is even number"
  Else
     Message "The value of X is odd number"
  EndIf
  Text Test.SetFocus
  Text Test.Select
Loop
```

Image Name, Picture

Description:

Component that allows the creation, manipulation, reading and writing files, and various other features.

Arguments:

Name: Component name.
Picture: Optional. Path of an image file (.jpg, .bmp, .gif, .ico).

Properties:

Lin, Col: Coordinates (Line, Column) where the component will be created in the active window.
Height, Width: Height and width of the component in lines and columns.
Img: Path of an image file (.jpg, .bmp, .gif, .ico) to be loaded in the component.
PosX: Defines the column for positioning images and texts.
PosY: Defines the line for positioning images and texts.
FontName: Defines the name of the font to text.
FontSize: Defines the size of the font to text.
BackColor: Defines the back color of the image.
ForeColor: Defines the font color of the text.
Negrito: If set to **True**, sets the text in bold.
Ordem: If it's equal to **Back** the component placed behind all other components, if equal **Front** place the component in front of all components.
MouseIcon: Path of a file icon to define a new icon for the mouse cursor.
MouseX: Returns the column of the mouse cursor.
MouseY: Returns the line of the mouse cursor.
MouseButton: Returns **1** if the left mouse button is pressed, **2** if the right mouse button is pressed and **0** (zero) if no button is pressed.

PointSize: Define the size of the points for the method **Point**.
Visible: Sets the component as visible or invisible (**True** or **False**).
HWND: This property returns a pointer to the component (address memory segment), allowing the use of API's with this component.
HDC: This property returns a descriptor of the graphic image of the component, for use in API's.
Parent: Define an image as the parent of component.

Methods:

Point, X, Y, Color: Draws a point at the coordinates **X**, **Y**, in the specified color.
ReadPoint, X, Y: Reading the color of a point which is at the coordinates **X, Y**.
Straight, X1, Y1, X2, Y2, Color: Draws a straight line from the coordinate **X1, Y1** to coordinate **X2, Y2**, in the specified color.
Write Arg1, Arg2, ...: Writing texts in the position defined by the properties **PosX** and **PosY**.
Cls: Clears all text, images and drawings of the image component.
SavePicture Filename, Quality: Saves the picture and/or all the drawings of the image component in a file format **.bmp** or **.jpg**.
  FileName: Path of the file with extension **.bmp** or **.jpg**.
  Quality: Optional. Image quality in compressed jpg file, which can vary from 0 to 100%, if not reported, it will be assumed value of 90%.
AutoSize: Adjusts the height and width of the component to the height and width of the image.
Use: Sets the image component as a new active window, ie the result of all commands will be directed to this component.
Remove: Removes the component of the active window.

Remarks:

Coordinates **X, Y** (column, line) defined in this component have scale in pixels.

Related commands:

Picture, Point, ReadPoint, AnimatedGif, Circle, Rectangle, Gradient, Transition

Example:

```
Var LightYellow Integer, PointColor Integer, X Integer, Y Integer

Window Size=35,100, Pos=Center,Center

Image Test, "Ben10.jpg"

Image Test.MouseIcon="Pencil.ico"
Image Test.PointSize=5
Image Test.Lin=1; Image Test.Col=2
Image Test.Height=30; Image Test.Width=50

Button LoadPicture, Type2, 33, 20, "Load picture"
Wait ClickButton

Image Test.Cls
Image Test.PosX=50
LightYellow=RGB(250, 250, 150); Image Test.BackColor=LightYellow
Image Test.Img="SpiderMan.gif"
Image Test.PosX=70; Image Test.PosY=380
Image Test.FontName = "Arial"; Image Test.FontSize = 10
Image Test.ForeColor = Red; Image Test.Bold = True

Button LoadPicture.Remove
Image Test.Write "Draw over the figure with the mouse!"
Button SaveFile, Type1, 33, 20, "Save picture"
```

```
TestButton = ""
While 1 = 1
  X = Image(Test.MouseX) - 15
  Y = Image(Test.MouseY) + 10
  If Image(Test.MouseButton) = 1
    Image Test.Point, X, Y, Black
  EndIf
  If TestButton = "SaveFile"
    TestButton = ""
    Image Test.SavePicture, "D:\Temp\Test.jpg"
    Message "Drawing saved successfully!"
  EndIf
  Wait Events
Loop
```

Inport PortNumber

Description:

Reads the value in the port number **PortNumber**. Will return a value corresponding to a byte which can vary from 0 to 255. For the binary value returned, use the command **DecToBin**.

Arguments:

PortNumber: Port number to read. This command can be passed as an argument a decimal or hexadecimal, in the latter case you must inform the characters **&H** in front of the number, eg, **&H378**.

Remarks

This command works on programs compiled only if the file "inpout32.dll" is in the same folder of the executable program.

Related commands:

Outport

Example:

```
Var X Integer, Z String

X = Inport(100)
Z = DecToBin(X)

Write X, " - ", Z
```

InsertFile File1, File2, File3

Description:

This command allows the user to insert multiple files into a single encapsulated file. It is useful to group image files, text, or any other type of file into a unique file, so the user will not have access to the files, except through the program. To extract the files you should use the command **ExtractFile**.

Arguments:

File1: Name of the file inside the encapsulated, ie, a name that will be the keyword to search for the file within the file wrapper, and can have a maximum of 25 characters.
File2: Path of the file to be inserted in the file wrapper.
File3: Path of the file wrapper.

Related commands:

ExtractFile

Example:

```
InsertFile "Picture.jpg", "C:\Images\Picture.jpg" , "C:\Temp\Package.pack"
```

Int Number

Description:

Returns the integer value of a number.

Arguments:

Number: Floating number (decimal) positive or negative.

Related commands:

Abs, Fix, Sgn

Example:

```
Var X Integer

X = Int(3.141592) 'Returns 3
Message X
```

JoinStr String1, String2, ...

Description:

Joins two or more strings or variables into a single string.

Arguments:

String1, String2, ...: Strings ou variables containing text or characters.

Remarks:

You can alternatively use the **&** (union operator) to join strings.

Related commands:

SubStr, RepStr, StrLen, Trim, StrPos

Example:

```
Var Result String, S String

S = "1958"

Result = JoinStr("Michael Jackson ", "Was born in ", S)
Write Result
```

KeyTest Code

Description:

Command that verifies the keypresses, including the Shift, Ctrl, arrows, spacebar, and moreover it can verify if multiple keys are pressed at the same time.
To test if a key is pressed or not, you must pass the key code as argument and check the command return: if it returns the key code and then the character "**+**", the key is pressed; if it returns the key code and then the character "**-**", the button is not pressed.

Related commands:

TestMouse

Arguments:

Code:

Key code to test, for example, 65 = code of the key "A".

To test the Shift, Ctrl, arrows and space bar, pass as argument the characters below:

S - Shift
C - Ctrl
R - Right arrow
L - Left arrow
D - Down arrow
U - Up arrow
# - Space bar

Remarks:

If you pass as argument the character "**?**", the command returns the code of the pressed key.

Example:

```
Var Right String, Left String, Key String

While 1 = 1
  Cls
  Right = KeyTest("R") 'Right arrow
  Left = KeyTest("L") 'Left arrow
  Key = KeyTest(65)  'Code of the key "A"
  Position 0, 0; Write Right, " ", Left, " ", Key
  Wait 20
Loop
```

Label Name, Text, Line, Column, Height, Width

Description:

Component to create text labels in the active window.

Arguments:

Name: Name of the label.
Text: Text that will appear on the label.
Line, Column: Coordinates where the label will be placed in the active window.
Height, Width: Height and width of the label in lines and columns.

Properties:

Enabled: Sets the label as active or inactive (**True** or **False**).
BackColor: Background color of the label.
Forecolor: Color of the text to be displayed on the label.
FontName: Font name of the text.
FontSize: Font size of the text.
Bold: If set to **True**, sets the text in bold.
Italic: If set to **True**, sets the text in italic.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the label.
Alignment: Defines the text alignment: 0 - Left, 1 - Right, 2 - Center.
Txt: Returns or sets the text of the label.
Visible: Sets the label as visible or invisible (**True** or **False**).

Methods:

Remove: Removes the label of the active window.

Related commands:

Text, Combo, Mask, Currency, Calendar

Example:

```
Label Test, "Logic Basic", 10, 10, 25, 3
Label Test.FontName = "Arial"; Label Test.FontSize = 24
Label Test.BackColor = Yellow; Label Test.Alignment = 2
```

LBDir

Description:

Reserved variable that returns the path to the Logic Basic executable program.

Related variables:

WindowsPrograms, ProgramDir, WindowsDir, Desktop

Link Name, Line, Column, Text, Link

Description:

Component that allows you to create links to web pages or emails.

Arguments:

Name: Name of the link.
Text: Text to be placed in the active window and activates the link by clicking.
Line, Column: Coordinates where the link will be placed in the active window.
Height, Width: Height and width of the link in lines and columns.
Link: Any valid internet link. To set an email link, write the word **mailto:** before an email address.

Properties:

Enabled: Sets the link as active or inactive (**True** or **False**).
Visible: Sets the link as visible or invisible (**True** or **False**).
Txt: Sets or returns the text of the link.
Link: Link address or email address.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the link.
FontName: Font name of the text.
FontSize: Font size of the text.
BackColor: Background color of the link.
Forecolor: Color of the text to be displayed on the link.
Bold: If set to **True**, sets the text in bold.
Italic: If set to **True**, sets the text in italic.
Underline: If set to **True**, sets the text underlined.

Methods:

Remove: Removes the link of the active window.

Related commands:

Write, Browser

Example:

```
Link Page, 5, 5, "Visit the web page of Logic Basic", "http://www.logicbasic.net"
Link Email, 9, 5, "Click here to send an email", "mailto:support@logicbasic.net"

Link Page.Underline = True
Link Page.Bold = True
Link Page.ForeColor = Blue
Link Email.ForeColor = Red
Link Email.BackColor = Yellow
```

ListenP2P Port

Description:

Initiates a process of listening to receive data from a remote computer via P2P.

Arguments:

Port: Connection port number, which must be the same port that was reported in command **ConnectP2P**. When any data is received on the port in question, the **IP** address of the client is placed in the reserved variable **ReferenceValue** and the data sent by the client will be placed in the reserved variable **RetStr**.

Related commands:

ConnectP2P, SendP2P

Example:

```
ListenP2P 6669
```

Log Number

Description:

Returns a decimal specifying the natural logarithm of a number.

Arguments:

Number: Integer or floating (decimal) greater than zero.

Related commands:

Exp

Example:

```
Var X Decimal

X = Log(10)
Write "Logarithm of 10 = ", X
```

LostFocus

Description:

Reserved variable that tells you when a component (button, text, etc.) lost focus.

Related commands:

GotFocus, Button, Text, Currency, Mask

Example:

```
Write "Click on text boxes!"

Text Test1, 2, 2, 15
Text Test2, 4, 2, 15

Sub Timer_Test(10)
  If LostFocus = "Test1"
    LostFocus = ""; Message "Test1 lost focus!"
  EndIf
  If LostFocus = "Test2"
    LostFocus = ""; Message "Test2 lost focus!"
  EndIf
EndSub
```

LunarPhase Line, Column

Description:

Component that displays or returns the phase of the moon on a certain date.

Arguments:

Line, Column: Coordinates where the component will be placed in the active window.

Properties:

Date: Sets date moon phase.
Fase: Returns an integer value corresponding to the phase of the moon on the date set in the argument **Date**. The phases range from 0 to 23 (24 cycles).
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the component.
Visible: Sets the component as visible or invisible (**True** or **False**).

Methods:

Remove: Removes the componet of the active window.

Related commands:

Date, Day, Month, Year, DateDiff, DateAdd, Weekday

Example:

```
LunarPhase Moon, 10, 30
Write "Today’s lunar phase: ", Moon.Phase
Wait 1000
LunarPhase Moon.Date = "12/10/2016"
Write "Lunar phase in 12/10/2016: ", Moon.Phase
```

Mask Name, Mask, Line, Column, Width

Description:

Component to create text boxes with mask in the active window.

Arguments:

Name: Name of the component.
Mask: Mask for text formatting while typing. You must use the # character to define the positions of the letters or numbers, for example, for date you can create a mask in the following format: **##/##/####**.
Line, Column: Coordinates where the text box will be placed in the active window.
Width: Width of the text box in columns.

Properties:

Enabled: Sets the component as active or inactive (**True** or **False**).
BackColor: Background color of the component.
Forecolor: Color of the text to be displayed on the component.

Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the component.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Mask: Sets or changes the mask of the text box.
Txt: Returns os sets the text of the text box.
Visible: Sets the text box as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the text box.
Select: Selects the text in the text box.
Remove: Removes the component of the active window.

Related commands:

Text, Currency, Combo, Calendar

Example:

```
Button OK, 10, 20
Mask Test, "##/##/####", 7, 20, 11
Mask Test.SetFocus

Wait ClickButton

Message Test.Txt
```

Message Argument1, Argument2, ...

Description:

Write strings (text) and contents of variables or expressions in a pop-up that is activated in the center of the computer screen. This command is identical to the command **Write** except that it shows the result in a message box.

Arguments:

The number of arguments is undefined and may be strings, variables, expressions, in any order, which will be written in the same order.

If you enter the word **OptionYesNo** as argument (in any position), the message box will be displayed with two buttons: **Yes** and **No**, then the command returns one of the keywords: **True** or **False**, corresponding to the selected option.

Related commands:

Write

Example:

```
Var SName String, S String

SName = "Betty Jane"

S = Message(SName, Chr(13), Chr(13), "This is a compound name?", OptionYesNo)

If S = True
```

```
  Message "Congratulations, you're right!"
Else
  Message "I'm sorry, you were wrong!"
EndIf
```

MinimizeSysTray

Description:

Minimizes the program to the systray bar.

Related commands:

AddIconSysTray

MkDir DirectoryName

Description:

Creates a new directory and returns an empty string on success or an error message if it occurs.

Arguments:

DirectoryName: Directory path to create.

Related commands:

Copy, DeleteFile, RmDir, Directory, FindFile

Example:

```
S = MkDir("D:\Temp")
Message S
```

Month Date

Description:

Returns the month of the date passed as argument.

Arguments:

Date: Any string (or function return) that can represent a date. If you enter an empty string as an argument, the command assumes the system date.

Related commands:

Date, Day, Year, DateDiff, DateAdd, Weekday, Time

Example:

```
Write Month(Date())
```

MouseX, MouseY

Description:

Reserved variables that store the row and column current mouse within the active window. As they are automatically updated every mouse movement, any time they are read by the program, they contain the exact coordinates of the mouse on the active window.

Remarks:

Usually the row and column refer to the mouse position on the active window, according to the resolution setting in rows and columns defined in command **Window**. If you want to check the mouse position in the entire area of the video monitor should be placed in command **Window** the following assignment: **MouseGeneral = True**, thus the value of these variables will be returned in pixels.

Example:

```
While 1 = 1
  Position MouseY, MouseX
  Write "O"
  Wait 10
Loop
```

NewWindow Arguments

Description:

Component for creation and manipulation of new windows.

For each window you define, you must create a subroutine to run her code, the name of the subroutine should be the window name followed by _ (underline) and the word **Code()**.

Arguments:

Set, Name, Option: Defines a new window with a name chosen by you. If option (optional argument) equals **WithoutBox** window will not have the control box (minimize, maximize and close), if option equals **Fixed** window will contain only the close button, and in both cases the window can not be resized.

Properties:

Size = Height, Width: Sets the height and width of the window, where "**Height**" is the height of the window in rows and "**Width**" is the width of the window in columns. If you enter the word **Max** in the arguments, the window will be resized to the maximum size allowed by the video monitor.
Res = Lines, Columns: Defines the resolution of the window in rows and columns.
PointSize = Pixels: Defines the size of the points in pixels (see **Point** command).
Pos = Line, Column: Defines the starting row and starting column of the window, relative to the Windows desktop. Take as reference the upper left corner of the window. If you enter the word **Center** in the arguments row or column, the window will be centered horizontally and/or vertically.

Background = Image or Color, Line, Column: Defines a background image or background color for the window. If you enter the first argument the path of an image file, the picture will be displayed at the bottom of the window. The arguments **Line**, **Column** should be informed only when you want to position the image in a certain coordinate of the window. But in this case the image can be erased by the command **ClearWindow** (**Cls**), and also in case you assign a background color to the window. Note that you can assign a background image with a background color for the window, but in this case you must inform this argument twice, once for each case.
Border = Option: If Border = False, removes all borders of the window.
Visible: Sets the window as visible or invisible (**True** or **False**).

Methods:

Activate, Mode: Active the window in the following modes:
   Dynamic: The dynamic window allows you to work with other windows without having to close it.
   Static: Static window doesn't allow you to work with other windows while it is active.
Use: Sets the reference window as the active window. To use the main window, use the word **Parent** for the reference window.
Off: Disables the window and removes all components that were created on it.

Related commands:

Window, WindowHeight, WindowWidth, Font

Example:

```
NewWindow Set, Test, WithoutBox
NewWindow Test.Size = 5, 21; NewWindow Test.Border = True
NewWindow Test.Pos = Center, Center; NewWindow Test.BackGround = Yellow

Font Test.Name = "Times New Roman"; Font Test.Color = Blue; Font Test.Bold = True
Font Test.Italic = True; Font Test.Size = 14

NewWindow Test.Activate, Static

Font Parent.Name = "Arial"; Font Parent.Size = 20
Write RetStr

EndProgram

Sub Test_Code()
  Write "Test", WindowTitle
  Position 1, 3; Write "Hello, world!"
  Button OK, 3, 4
  Wait ClickButton
  Button OK.Enabled = False
  NewWindow Test.Off
EndSub
```

OpenDB DBName, Password

Description:

Opens a native database for reading and writing records.

Arguments:

DBName: Name of the database, which should be as declared in the command **DB**.
Password: Password of the database.

Related commands:

DB, OrderDB, AddRecord, UpdateRecord, ReadRecord, TotalRecordsDB, DeleteRecord, FindDB, CompactDB

Example:

```
OpenDB Customers, "abc123"
```

OpenFile NickName, Path, OpenType

Description:

Opens a file for read/write in binary or text mode.

Arguments:

NickName: Name of the file to identify it in the file manipulation commands.
Path: Path of the file to open.
OpenType:
- Binary: Opens the file for read/write in binary mode.
- Input: Opens the file for reading in text mode.
- Append: Opens the file for writing in text mode.

Related commands:

ReadFile, WriteFile, CloseFile, SizeFile

Example:

```
OpenFile "Test", "Test.txt", "Binary"
```

Or

Description:

Operator that performs a logical operation betwenn expressions, so that, if at least one of the expressions is true, it returns **True**, and returns **False** only if all expressions are false.

Related commands:

And, If, Else, While

Example:

```
Var X Integer, Y Integer, Z Integer

X = 1; Y = 20; Z = 100
If X = 1 Or Y = 10 Or Z = 50
   Write "True"
Else
   Write "False"
EndIf
```

OrderDB DBName, Field1, Field2, ...

Description:

Orders a native database for one or more fields.

Arguments:

DBName: Name of the database, which should be as declared in the command **DB**.
Field1, Field2, ...: Field (or sequence of fields) to be ordained.

Related commands:

DB, OpenDB, AddRecord, UpdateRecord, ReadRecord, TotalRecordsDB, DeleteRecord, FindFile, CompactDB

Example:

```
OrderDB Customers, "Name", "Address"
```

Outport PortNumber, Value

Description:

Sends a value for the port number **PortNumber**.

Arguments:

PortNumber: Número da porta a ler. This command can be passed as an argument a decimal or hexadecimal, in the latter case you must inform the characters **&H** in front of the number, eg, **&H378**.
Value: Value is one byte, that is, which can vary from 0 to 255.

Remarks

This command works on programs compiled only if the file "inpout32.dll" is in the same folder of the executable program.

Related commands:

Inport

Example:

```
Var BinaryValue String, DecimalValue Integer

BinaryValue = "01000001"
DecimalValue = BinToDec(BinaryValue)

Write DecimalValue
OutPort 100, DecimalValue
```

Paint Line, Column, Color

Description:

Paints a certain area of the active window, with a particular color from the point defined by **Line, Column**.

Arguments:

Line, Column: Coordinate of the starting point to paint.
Color: Color to fill the area.

Related commands:

Rectangle, Straight, Point, ReadPoint, RGB, Circle

Example:

```
Circle 10, 20, 10, White
Wait 500
Paint 11, 11, Blue
```

--

Picture Name, Image

Description:

Component to create pictures in the active window

Arguments:

Name: Picture name.
Image: File path of the picture. To create pictures with transparent areas, you should use images with extension **.Gif** with transparency color.

Properties:

Height, Width: Height and width of the picture in lines and columns.
Lin, Col: Coordinates (line, column) where the picture will be created in the active window.
Position: Position of the picture in relation to the others pictures.
   Back: The picture is drawn behind the other pictures.
   Front: The picture is drawn in front of the other figures.

Methods:

Off: Disables the picture, making it invisible. To activate the picture (visible), use the property **Position** (defined as Back or Front).
Remove: Removes the picture of the active window.

Related commands:

DrawPictures, AnimatedGif, Image, Transition

Example:

```
Picture Ufo, "Ufo.gif"
Picture Satellite, "Satellite.gif"
Picture Ufo.Position = Back

Picture Ufo.Lin = 10; Picture Ufo.Col = 15
Picture Satellite.Lin = 5; Picture Satellite.Col = 50
DrawPictures
```

Point Line, Column, Color

Description:

Draws a point at coordinates **Line, Column** in the active window.

Arguments:

Line, Column: Window coordinates to draw the point.
Color: Point color.

Related commands:

ReadPoint, Straight, Rectangle, Circke, RGB, Image

Example:

```
Window PointSize = 5
Point 11, 39, Red
```

Position Line, Column

Description:

Sets the position of text in coordinates **Line, Column** of the active window.

Argument2s:

Line, Column: Coordinates of the active window to position the text.

Related commands:

Write

Example:

```
Position 11, 33
Write "Hello, world!"
```

Printer

Description:

Componente to print text and images directly to the printer.

Methods:

Font: Sets the font of the text to be printed. Attributes can be placed all on the same line separated by a comma, which should be assigned a value for each:
  Name: Defines the character font name.
  Size: Defines the size of the character font.

Bold: If set to **True**, sets the text in bold.
Italic: If set to **True**, sets the text in italic.
Position Line, Column: Positions the text at coordinates **Line, Column** on the printer paper.
Write: Writes a text at the coordinates defined by the **Position** method on the printer paper.
Draw ImagePath: Draws a picture in the coordinates defined by the **Position** method.
NewPage: Forces the printer to feed the paper to the next page.
End: Closes an flushes all print buffers.

Related commands:

Rep, Write

Example:

```
Var S String

Printer.Font Name = "Ms Sans Serif", Size = 38, Bold = True, Italic =  True

Printer.Position 0, 0
Printer.Write "Logic Basic"
Printer.Position 30, 0
Printer.Draw "Chart.gif"

Printer.NewPage

S = "End of the report!"
Printer.Font Name = "Arial", Size = 20, Bold = False, Italic =  False
Printer.Position 0, 0
Printer.Write "Message: ", S

Printer.End
```

ProgramDir

Description:

Reserved variable that returns the directory where the executable of the current program.

Related variables:

WindowsPrograms, WindowsDir, Desktop, LBDir

Radio Name, Line, Column, Text

Description:

Component to create radio buttons in the active window.

Arguments:

Name: Name of the radio button.
Line, Column: Coordinates where the radio button will be placed in the active window.
Text: Text that will appear to the right of the radio button.

Properties:

Enabled: Sets the radio button as active or inactive (**True** or **False**).
BackColor: Background color of the radio button.
Forecolor: Text color of the radio button.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the radio button.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Value: Returns or sets the value of the radio button: 0=False, 1=True
Visible: Sets the radio button as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the radio button.
Remove: Removes the radio button of the active window.

Related commands:

Check, Frame, Combo

Example:

```
Radio Opt1, 3, 15, "Option 1"
Radio Opt2, 5, 15, "Option 2"
Radio Opt1.Value = 1

Button OK, 9, 20
Wait ClickButton

Write "Opt1 = ", Opt1.Value
Write "Opt2 = ", Opt2.Value
```

Random MaxValue

Description:

Returns a random number which can vary from 0 to the argument value.

Arguments:

MaxValue: Maximum value of the random number to be returned.

Example:

```
Var X Integer

X = 0
While X < 25
  Write Random(100)
  X++
Loop
```

ReadFile NickName, StartPos, NumBytes

Description:

Performs a read data in a file in the mode defined by the command **OpenFile** (binary or text).

Arguments:

NickName: Name of the file defined in command **OpenFile**.
StartPos: (Optional). Starting position in bytes to read in the file, this argument must be informed only in binary mode.
NumBytes: (Optional). Number of bytes to read in the file, this argument must be informed only in binary mode.

Related commands:

OpenFile, WriteFile, CloseFile, SizeFile, OrderDB

Example:

```
'Performs a reading in binary mode
S = ReadFile("Test", 1, 11)
Write S

'Reads a line in text mode
Write ReadFile("Test3")
```

ReadPoint Line, Column

Description:

Returns the numeric value of the color for the point that is at the coordinates **Line, Column** of the active window.

Arguments:

Line, Column: Coordinates of the active window to read the color.

Related commands:

Point, Rectangle, Straight, Circle, RGB, Image

Example:

```
Var X Integer

X = ReadPoint(10, 10)
Write X
```

ReadRecord DBName, RegNum

Description:

Reads the record from the native database, in the position defined by the argument **RegNum**. Returns a string with all fields of the record, or an empty string if the record is deleted.

Arguments:

DBName: Name of the database.

RegNum: Record number to read.

Remarks:

In the program code you should test which records are deleted when reading the database and ignore deleted records.

Related commands:

DB, OpenDB, AddRecord, OrderDB, UpdateRecord, TotalRecordsDB, DeleteRecord, FindDB, CompactDB

Example:

```
S = ReadRecord(Customers, 3)

If S <> ""
  Write "S = ", S
  Write Customers.Sequence
  Write Customers.Name
  Write Customers.Phone
  Write Customers.Address
  Write Customers.Salary
EndIf
```

RecordSound Option

Description:

Command to make sound recordings, stop recording, or save the recording to a file.

Arguments:

Option:
  Start**:** Start a sound recording.
  Stop**:** Stops a sound recording.
  Save, FileName: Saves a sound recording in the file **FileName** (only in **Wav** format).

Remarks:

This command allows you to record sounds of Logic Basic or any sound device of the Windows, but recording should always be started before activating the sound (play).

Related commands

Sound

Example:

```
RecordSound Start
Wait 30000   'Record sound for 30 seconds
RecordSound Stop
RecordSound Save, "Sound.wav"
```

Rectangle Line1, Column1, Line2, Column2, Color

Description:

Draws a rectangle in the active window.

Arguments:

Line1, Column1: Upper left coordinates of the rectangle.
Line2, Column2: Lower right coordinates of the rectangle.
Color: Color of the rectangle.

Comandos relacionados:

Straight, Circle, Point, ReadPoint, RGB

Example:

```
Rectangle 1, 2, 5, 30, Red
```

RemoveSysStartup ProgramName

Description:

Makes an Logic Basic executable program no longer runs automatically when Windows starts.

Arguments:

ProgramName: Program name to be deleted from the Windows registry. This name must be same name that was passed as argument in the command **AddSysStartup**.

Related commands:

AddSysStartup

Rep

Description:

Component for creating an printing reports in printer with preview video. Allows you to insert tables, images and various fonts of characters in the same report, plus several unique features.

Properties:

Config Argument: Allows you to configure the report output for video or for the printer, and the top and left margin of the report. These settings must be passed as an argument into a string as shown below:

"Output:VIDEO LeftMargin:100 TopMargin:100"

Wherein:

Output: VIDEO or PRINTER
LeftMargin: Width of left margin in **twips**.
TopMargin: Height of the top edge in **twips**.

* **twips** is a standard measure high-resolution graphic.

PaperHeight: Sets the height of printer paper in twips. A sheet of A4 paper is approximately 15000 twips in height.

LineHeight: The height of a line for a character font size equal to 10, is around 250 twips.

Col ColumnNumber, Settings: Define the settings for each report column:
ColumnNumber: Column number to set, which can range from 0 (zero) to total columns of the report.
Settings: String containing the settings of the columns as shown below:

"Width:92 FontName:Tahoma FontSize:10 Bold:True HorTab:False VerTab:False Align:Center TabColor:255"

Wherein:

Width: Column width in characters.
FontName: Name of the character font.
FontSize: Size of the character font.
Bold: True = Bold Text, False = Normal text.
HorTab: True = Prints horizontal dividing line, False = Not print horizontal dividing line.
VerTab: True = Prints vertical dividing line, False = Not print vertical dividing line.
Align: Alignment of text in the column: Center, Right, Left.
TabColor: A color value (RGB) of the horizontal and vertical dividing lines.

Methods:

Add Text, ForeColor, BackColor: Add a text on the (**Current row, Current column**), where:

Text: Text to be printed on the cell defined by (**Current row, Current column**).
ForeColor: Font color of the text.
BackColor: Background color of the cell.

You must call the method **Add** for each column of the report in accordance with the sequence of creation of the columns with the property **Col**, ie, when a new row is started, the first call to the method **Add** will print the text in the first column, the second call will print the text in the second column, and so on, until you call the method **NewLine**.

Img OptInc, Margin: Prints a image on the current line of the report, where:

OptInc: Optional, if equal to **True** increments the current line with the image height.
Margin: Optional, distance from the left bank to print the image (in twips).

Line X1, X2, Color, Thickness: Draws a horizontal line in the current row, between columns **X1** and **X2** (in twips), where:

X1: Initial column in twips.
X2: Final column in twips.
Color: Line color.
Thickness: Line thickness, in twips.

Print: Text, Col: Print the text on the current row, from the initial column **Col** (in twips).

NewLine: Starts a new row in the report.

NewPage: Starts a new page in the report.

End: Finalizes the report, flushes all print buffers and load the report window, if the user has pressed the **Print** button, the report window is closed and the reserved variable **RetStr** contains the word "PRINTER", this allows the run the code of the report again to have it printed on the printer.

Related commands:

Printer

RepStr String1, String2, InitialPos, Length

Description:

Replaces the text contained in **String1** by text contained in **String2**, from the position **InitialPos** of **String1**, with the maximum of **Length** characters.

Arguments:

String1: String or text to be modified.
String2: String or text that will overlay **String1**.
InitialPos: Initial position to replace characters in **String1**.
Length: Length of the stretch to replace in **String1**, in characters.

Related commands:

SubStr, StrLen, JoinStr, Trim, StrPos

Example:

```
Var S1 String, S2 String

S1 = "Hello, !!!!!"
S2 = "world"

RepStr S1, S2, 8, 5

Message S1
```

RetStr

Description:

Reserved variable of Logic Basic used to store strings returned from various commands and events.

RetValue

Description:

Reserved variable of Logic Basic used to store values returned from various commands and events.

RevDate Date

Description:

Reverses a date format for the format **yyyymmdd** (year, month, day).

Arguments:

Date: Date format to reverse.

Related commands:

Date, DateAdd, DateDiff, Day, Month, Year

Example:

```
Write RevDate("7/15/2013")
```

RGB Red, Green, Blue

Description:

Command that returns an integer value corresponding to the color resulting from the combination of the colors red, green and blue.

Arguments:

Red: Integer value corresponding to the intensity of the red color which may vary from 0 to 255.
Green: Integer value corresponding to the intensity of the green color which may vary from 0 to 255.
Blue: Integer value corresponding to the intensity of the blue color which may vary from 0 to 255.

Example:

```
Var Color Integer

Color = RGB(150, 200, 255)
Window BackColor = Color
```

RmDir Path

Description:

Removes an existing directory.

Arguments:

Path: Directory path to remove.

Remarks:

The directory can not be removed if it contains files. To remove all files in the directory use the command **DeleteFile Path*.***.

Related commands:

MkDir, DeleteFile, Copy, Directory, FindFile

Example:

```
Var S String

S = RmDir("C:\Trash")
If S = ""
  Message "Directory successfully removed!"
Else
  Message S
EndIf
```

Round Value, X

Description:

Returns a number rounded to a specified number of decimal places.

Arguments:

Value: Decimal value.
X: Decimal places

Related commands:

Int, Abs, Fix, StrVal

Example:

```
Var X Decimal

X = 12345.84721
X = Round(X, 2)
Write X
```

SaveString Key, String

Description:

Saves a string or text in the Logic Basic registry.

Argumento:

Key: Name of the key in the Logic Basic registry.
String: String or text to save in Logic Basic registry.

Related commands:

GetString

Example:

```
SaveString "ForeColor", "Blue"
```

SaveWinReg KeyType, Section, Key, Value

Description:

Save a value in Windows registry.

Arguments:

KeyType: Type of the registry key, which can be one of the following names:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

Section: Section where the key is stored.
Key: Name of the registry key.
Value: Value to be assigned to the key.

Related commands:

GetWinReg, GetString, SaveString

Example:

SaveWinReg "HKEY_LOCAL_MACHINE", "Software\LogicBasic", "Test", "Value"

ScrollBar Name, Max, Line, Column, Orientation, Size

Description:

Component to create vertical or horizontal scrollbars.

Arguments:

Name: Name of the scroll bar.
Max: Maximum value of the slider of the scroll bar.
Line, Column: Coordinates where the scroll bar will be placed in the active window.
Orientation: Orientation of the scroll bar: 0 - vertical, 1 - horizontal.
Size: Length or height of the scroll bar.

Properties:

Enabled: Sets the scroll bar as active or inactive (**True** or **False**).
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Max: Maximum value of the slider of the scroll bar.
SmallStep: Defines the increment value of the slider of the scroll bar when they are used arrow keys to move it.
LargeStep: Defines the increment value of the slider of the scroll bar when the keys **PageUp** o **PageDown** are used to move it.
Value: Sets or returns the value of the cursor position of the scroll bar.
Visible: Sets the scroll bar as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the scroll bar.
Remove: Removes the scroll bar of the active window.

Related commands:

Box

Example:

```
Position 1, 3; Write "Cursor value:"
Text Test, 1, 20, 5
ScrollBar Bar, 20, 1, 30, 0, 20
While 1 = 1
  Text Test.Txt = Bar.Value
  Wait Events
Loop
```

SendP2P Data

Description:

Sends data via P2P connection to a remote computer.

Arguments:

Data: Data to send to the remote computer.

Related commands:

ConnectP2P, ListenP2P

Example:

```
SendP2P "Hello, how are you?"
```

SeqStr Number, Character

Description:

Returns a string containing a specified number of repeated characters.

Arguments:

Number: Number of characters to be repeated in sequence.
Character: Character to be repeated in the string.

Related commands:

Asc, Chr, Format, Trim, SubStr, JoinStr

Example:

```
Var S String

S = SeqStr(10, "X")
Write "[", S, "]"
```

Sgn Number

Description:

Returns the sign of a number: **1** (positive), **-1** (negative).

Arguments:

Number: Integer or floating (decimal), positive or negative.

Related commands:

Int, Abs, Fix

Example:

```
Var X Integer

X = Sgn(-20)
Message X
```

Sin Number

Description:

Returns the sine of an angle.

Arguments:

Number: Integer of floating (decimal).

Remarks:

To convert degrees to radians, multiply degrees by pi/180. To convert radians to degrees, multiply radians by 180/pi.

Related commands:

Cos, Tan, Atn

Example:

```
Var X Decimal

X = Sin(1.3)
Write "Sine of the angle 1.3 = ", X
```

SizeFile NickName

Description:

Returns the number of bytes in the file defined by the alias passed as argument.

Arguments:

NickName: Name of the file defined in command **OpenFile**.

Related commands:

OpenFile, ReadFile, WriteFile, CloseFile

Example:

```
Write "File size: ", SizeFile("Test"),  " bytes"
```

Sound Name, SoundFile

Description:

Component to play sounds and music in formats wav, wma, mp3, mid, etc.

Arguments:

Name: Name of the sound component.
SoundFile: Path of the sound file.

Properties:

FileName: Update the path and name of the sound file.
Enabled: Sets the component as active or inactive (**True** or **False**).
Visible: Sets the component as visible or invisible (**True** or **False**).
Line, Column: Coordinates where the component will be placed in the active window.
Height, Width: Height and width of the component in lines and columns.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Volume: Sound volume, which can vary from 0 to 100.
Balance: Sets the balance between left and right channels of the sound, the value can range from -100 to 100 (zero is balanced).
Loop: If set = **True** repeats the sound when finished, indefinitely.
Time: Returns the total time of the sound in seconds.
Position: Returns the current position (elapsed time) of the sound in seconds.

Methods:

Play: Plays the sound.
Pause: Break the sound at the current position.
Continue: Restarts the sound from the position where it was paused.
End: Closes the sound.

Related commands:

RecordSound

Example:

```
Sound Phone, "Phone.wav"
Sound Phone.Play
```

SQL

Description:

Component for managing databases, using SQL language instructions. The connection to database is via ODBC.

First you must assign aliases for the database and the tables to be accessed. These nicknames are words chosen by you, for example, to name a database or customers, you can create a reference as follows:

SQL DBCustomers

To create an alias for a sales table that is inside the database in question, you must reference this table by writing the name of the database, then point and the table alias, for example:

SQL DBCustomers.Sales

Methods for the database:

Open, ConnectionString, CursorType: Opens a database in accordance with the connection string passed as argument. The type of cursor is optional and may be defined by the following words: **UseClient** or **UseServer**.
Execute: Executes a SQL clause.
Close: Closes the database.

Methods related to the tables of the database:

Open, SQLClause, Cursor, Lock: Opens a table contained in the database through a SQL clause. The cursor type and lock type are optional, and can be defined by the following words:
  Cursor: ForwardOnly, KeySet, Dynamic, Static.
  Lock: ReadOnly, Pessimistic, Optimistic, BatchOptimistic.
Add: Adds a new record in the table.
Update: Updates (writes) the record at the current record pointer.
Edit: Edit the record that is at the current record pointer.
Find: Finds a record in the table.
Delete: Deletes the record that is at the current record pointer.
MoveFirst: Moves the record pointer to the first record of the table.
MoveNext: Moves the record pointer to the next record of the table.
MovePrevious: Moves the record pointer to the previous record of the table.
MoveLast: Moves the record pointer to the last record of the table.
TotalRecords: Returns the total records in the table.
Close: Closes the table.

Properties of tables:

Pointer: Returns **EOF** if the record pointer reaches the end of the file, and **BOF** if the pointer reaches the top of he file.

Remarks:

To assign values to the fields in a table, you should write the name of the database, the table name, the field name and the value to assign to the field, as follows:

SQL DBCustomers.Sales.Description, "TV Philips"
SQL DBCustomers.Sales.Price, 500.00

Instead of comma, you can also use the equal sign:

SQL DBCustomers.Sales.Description = "TV Philips"
SQL DBCustomers.Sales.Price = 500.00

The criterion to read values is the same, but the name of the database, table an field should be enclosed in parentheses, for example:

```
Description = SQL(DBCustomers.Sales.Description)
Price = SQL(DBCustomers.Sales.Price)
```

Sqr Number

Description:

Returns a numeric value corresponding to the square root of a number.

Arguments:

Number: Number which you want to take the square root.

Example:

```
Var X Decimal

X = Sqr(49)
Write "Square root of 49 = ", X
```

Straight Line1, Column1, Line2, Column2, Color

Description:

Draws a straight line in the active window.

Arguments:

Line1, Column1: Coordinates of the initial point.
Line2, Column2: Coordinates of the final point.
Color: Line color.

Related commands:

Rectangle, Circle, Point, ReadPoint, RGB

Example:

```
Straight 10, 10, 20, 20, Red
```

StrLen String

Description:

Returns the length in characters of a string or text.

Arguments:

String: String containing text or characters.

Related commands:

SubStr, RepStr, JoinStr, Trim

Example:

```
Write StrLen("Logic Basic")
```

StrPos String, SubStringToFind, Reverse, IniPos

Description:

This command searches for a substring (or character) within a string. Returns the position of the first occurrence of the substring within the string. If within the string is not found no instance of the substring, returns a null string ("").

Arguments:

String: String where the search is performed.
SubStringToFind: Substring or character to search in the string.
Reverse: Optional - If you pass this keyword as an argument, the command will search in reverse order, ie, the end to the beginning of the string.
IniPos: Optional - Starting position in the string (default = 1, for reverse order default = length of string).

Related commands:

SubStr, RepStr, Trim, StrLen, JoinStr

Example:

```
Var S1 String, S2 String, P Integer

S1 = "Logic Basic, the world language"
S2 = "ic"

Write "First occurrence of ", Chr(34), S2, Chr(34), " in ", Chr(34),  S1, Chr(34)
Write ""

P = StrPos(S1, S2)
Write "Searching from start to end = ", P

P = StrPos(S1, S2, "Reverse")
Write "Searching from end to start = ", P
```

StrVal String

Description:

Returns the numeric value of a string containing a numeric text.

Arguments:

String: String containing a integer or decimal numeric text.

Related commands:

ConvStr

Example:

```
Var X Decimal, S String

S = "10.5"

X = StrVal(S)
Write X
```

SubStr String, StartPos, Length

Description:

Returns a part of a string.

Arguments:

Text: Specifies the string to return a par of.
StartPos: Starting position in the string, which ranges from 1 to the size of the string in characters (including spaces).
Length: The length (in characters) to be returned from the start argument.

Related commands:

RepStr, StrLen, JoinStr, Trim, StrPos

Example:

```
Var S String

S = SubStr("Logic Basic, the world language!", 7, 5)
Write S
```

SystemMenu MainOption, Number, Description1, Option, Description2

Description:

Creates a system menu in the active window.

Arguments:

MainOption, Number. Description1: Create a top bar on the active window with one or more options, you should inform the option number that begins with the value 0 to the total number of options (less one), and then the description of the option.
Option, Description2: Creates a new item on the menu of a main option. You must enter the number of the main option, then the keyword **Option** and the description of the option.

Remarks:

To verify that the selected option in the menu, you should execute the command **Wait ClickButton**, and check the contents of memory **TestButton**.

To create a separator line in the menu, you must enter the description of the option the character "**-**" (dash).

To create a shortcut with the underline letter, you must enter the character **&** before the letter shortcut.

Related commands:

API, Button

Example:

```
SystemMenu MainOption, 0, "File"
  SystemMenu MainOption, 0, Option, "New"
  SystemMenu MainOption, 0, Option, "Open"
  SystemMenu MainOption, 0, Option, "Save File          CTRL+S"
  SystemMenu MainOption, 0, Option, "-"
  SystemMenu MainOption, 0, Option, "Exit               ESC"
SystemMenu MainOption, 1, "Help"
  SystemMenu MainOption, 1, Option, "Help topics"
  SystemMenu MainOption, 1, Option, "About"

TestButton = ""
While SubStr(TestButton, 1, 4) <> "Exit"
  Wait ClickButton
  Message TestButton
Loop
EndWindow
```

Tan Number

Description:

Returns the tangent of an angle.

Arguments:

Number: Integer or floating (decimal).

Remarks:

To convert degrees to radians, multiply degrees by pi/180. To convert radians to degrees, multiply radians by 180/pi.

Related commands:

Atn, Sin, Cos

Example:

```
Var X Decimal

X = Tan(1.3)
Write "Tangent of angle 1.3 = ", X
```

TestButton

Description:

Reserved variable that returns the name of the last button pressed, including options of the system menu.

Related commands:

Button, SystemMenu, Wait

Example:

```
Write "Click on the buttons!"

Button Button1, 10, 20, "Button 1"
Button Button2, 10, 35, "Button 2"
Text Test, 5, 20

While 1 = 1
  Text Test.Txt = TestButton
  Wait Events
Loop
```

--

TestDate Expression

Description:

Returns whether the expression passed as argument is a valid date or not (**True** or **False**).

Arguments:

Expression: Expression containing a date.

Related commands:

TestEmail

Example:

```
Var Test String

Test = TestDate("7/31/2013")
Write "Date is ", Test
```

--

TestEmail Email

Description:

Tests whether the email passed as an argument is valid or invalid, and returns **True** of **False**.

Arguments:

Email: String containing an email address.

Remarks:

This command only tests the syntax of the email, but not if it is active on the internet.

Related commands:

TestDate

Example:

```
Var Test String

Test = TestEmail("support@logicbasic.net")
Write "Email address is ", Test
```

TestMouse

Description:

This command tests whether the left and right buttons of the mouse are pressed. Returns a string with two characters, the first indicates whether the right button is pressed, and the second indicates if the left button is pressed. If the character is 0 (zero) means that the button is not pressed, if the character is equal to 1 means that the button is pressed.

Related command and variables:

MouseX, MouseY, ClickMouse, KeyTest

Example:

```
Var T String

While 1 = 1
  Cls
  T = TestMouse()
  Position 0,0; Write T
  Wait 10
Loop
```

Text Name, Line, Column, Width, Height

Description:

Component to create text boxes on the active window.

Arguments:

Name: Name of the text box.
Line, Column: Coordinates where the text box will be created in the active window.
Width, Height: Width and height of the text box in lines and columns.

Properties:

Enabled: Sets the text box as active or inactive (**True** or **False**).
BackColor: Background color of the text box.
Forecolor: Color of the text.
FontName: Font name of the text.
FontSize: Font size of the text.
Bold: If set to **True**, sets the text in bold.

Italic: If set to **True**, sets the text in italic.
Tip: Text that will appear in a small pop-up window when the mouse cursor stay more than a second on the text box.
TabIndex: Sets the priority of cursor focus when the tab key is pressed in the active window.
Alignment: Defines the text alignment: 0 - Left, 1 - Right, 2 - Center.
Txt: Returns or sets the text (string) of the text box.
TextRFT: Returns or sets the text of the text box in RTF format.
Max: Sets the maximum amount of characters in the text box, if this value is zero, there is no limit.
RightMargin: Sets the right margin of the text box in columns (only for multiline text).
CursorPos: Sets the position of the cursor in the text box, in characters.
Visible: Sets the text box as visible or invisible (**True** or **False**).
Parent: Define an image as the parent of component.

Methods:

SetFocus: Moves the cursor focus to the text box.
Remove: Removes the text box of the active window.
Select Start, Length, FontName, FontSize, Bold, Color: If you execute the method **Select** with no parameters, the entire text of the text box is selected. If you pass parameters, some text is selected with the following attributes:
  Start: Starting position in the string of text to select.
  Length: Length in characters of the text to select.
  FontName: Font name of the selected text.
  FontSize: Font size of the selected text.
  Bold: If you pass this keyword as an parameter, the selected text will be bold.
  Color: Color of the selected text.

Related commands:

Combo, Mask, Currency, Calendar

Example:

```
Text Test, 5, 15

Text Test.Txt = "Enter text here!"
Text Test.Tip = "This is a test!"
Text Test.Alignment = 2
Text Test.Select

Button OK, 9, 20
Wait ClickButton

Message Test.Txt

Text Test.Enabled = False
```

Time

Description:

Returns a string indicating the current system time.

Related commands:

Date, Day, Month, Year, DateDiff, DateAdd, Weekday

Example:

```
Var Now String

Now = Time()
Write "Now is ", Now
```

TotalRecordsDB DBName

Description:

Returns the number of records in a native database.

Arguments:

DBName: Name of the database.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, UpdateRecord, DeleteRecord, FindDB, CompactDB

Example:

```
Write "Total records from the database of customers: ", TotalRecordsDB(Customers)
```

Transition Keyword = Attribute

Description:

Command that makes a transition between two pictures, where the result is placed in the background of the active window.

Arguments:

You should write the keyword, and then the equal sign and the attribute:

InitialPicture: File path from the initial picture of transition. if the image file is in the same directory of the program, type only the file name.
FinalPicture: File path from the final picture of transition.
Position: Sets the starting position at coordinates **Line, Column** in the active window of the picture resulting from the transition. You can change the position of the picture during the transition.
Value: The transition value, which can vary from 0 to 255. For example, the value 0 corresponds to picture 1, the value 255 corresponds to picture 2, and the value 128 corresponds to the result of the combination of the two pictures of 50% each.

Remarks:

You should execute the command **DrawPictures** before each call to the command **Trasition Value**.

Related commands:

DrawPictures, AnimatedGif, Window

Example:

```
Var Counter Integer

Transition InitialPicture = "Picture1.jpg"
Transition FinalPicture = "Picture2.jpg"

Transition Position = 0,0

Counter = 0
While Counter < 256
  DrawPictures
  Transition Value = Counter
  Wait 30
  Counter++
Loop
```

Trim String

Description:

Removes spaces left and right of a string or text.

Arguments:

String: String to trim.

Related commands:

Asc, Chr, Format, SubStr, Sequence, JoinStr

Example:

```
Var S1 String, S2 String

S1 = "   Logic Basic   "
S2 = Trim(S1)

Write "[", S1, "]", Chr(13), "[", S2, "]"
```

UnCaptureWindow

Description:

Disable the command **CaptureWindow**.

Related command:

CaptureWindow

UpdateRecord DatabaseName, RegNum

Description:

Writes a record in the native database, in the position set by **RegNum**.

Arguments:

DabaseName: Name of the database.
RegNum: Number of the record where the new data is overwritten.

Related commands:

DB, OpenDB, AddRecord, OrderDB, ReadRecord, TotalRecordsDB, DeleteRecord, FindDB, CompactDB

Example:

```
S = ReadReCord(Customers, 15)

Customers.Name = "John Robinson"
Customers.Phone = "4567-8910"
Customers.Address = "Second Avenue, 123"
Customers.Salary = 2500.00

UpdateRecord Customers, 15
```

UpFileFTP LocalFile, FileFTP

Description:

Sends a file to a remote computer.

Arguments:

LocalFile: File path on the local computer to be sent a remote computer.
FileFTP: File path on the remote computer.

Related commands:

ConnectFTP, DownFileFTP

Example:

```
UpFileFTP "C:\LogicBasic.exe", "/download/LogicBasic.exe"
```

Var, Variable Variable1 Type, Variable2 Type, ...

Description:

Command that declares variables that can be recognized by Logic Basic. There are three types of variables in Logic Basic: **String**, **Integer** and Decimal. The type **String** should be used to declared variables or text string; The type **Integer** must be used to declare variables of type integer numeric to vary -2.147.483.648 to 2.147.483.647; The type **Decimal** should be used to declare variables of decimal number that can range from -1,79769313486232E308 to -4,94065645841247E-324 for negative values, and the 4,94065645841247E-324 to 1,79769313486232E308 for positive values. Simplifying, variables of type **Decimal** can be used to store both integer values as fractions, and are used primarily for working with currency values.

To declare variables you should use the keyword **Variable** or simply **Var** and then the variable name followed by its type. If not informed the variable type, the Logic Basic will assume to be of type **String**. You can declare multiple variables on the same line.

Variables can be declared as **Global** or **Local**. Global variables can be read at any point in the program, they retain their value within the main code, extensions, subroutines and functions. Local variables must be declared only within functions and subroutines, and they retain value only within the function (or subroutine) in which it is declared, and after the execution of the function, they are destroyed by Logic Basic. Local variables can have the same name in different functions or subroutines without causing conflicts.

Global variables can be declared in the first lines of the program (recommended) or through code. Local variables can only be declared in the first lines of a function or subroutine, if a variable is declared in the middle of the code of a function or subroutine, it will be considered as Global.

Remarks:

Every variable to be used in the program must be declared, otherwise it will not be recognized by Logic Basic. The number of variables from the command line is undefined.

Example:

```
Var Name String, Counter Integer, Salary Decimal
```

Wait Parameter

Description:

Causes program execution wait a certain time or the occurrence of an event in the line where this command is placed.

Arguments:

Parameter:
- Number: If a number is passed as an argument, the program will wait for that time (in milliseconds).
- ClickButton: The program will wait for a click on a button or in the menu system, and returns the result in the reserved variable **RetStr**.
- ClickBox: The program will wait for a click ou double-click on a list box, and returns the result in the reserved variable **RetStr**.
- Events: Wait of force the execution of other events in the active windows or the execution of other tasks by the operating system.

Example:

```
Write Time()
Wait 3000
Write Time()
```

WebCam Arguments, Parameters

Description:

Command that allow you to read images from a webcam connected to your computeer, and record the frames (images) into images files.

Arguments:

Config: This argument has the following parameters

> ***Line, Column:*** Coordinates where the component will be placed in the active window.
> ***Height, Width***: Height and width of the webcam screen in lines and columns.
> ***Time:*** Time interval between frames, the default value is 3, which corresponds to 300 milliseconds.
> ***Directory***: File path which will be recorded frames (images). The file name is a base for the files that are numbered, so don't put the extension name.

Start: Starts capturing images from webcam.
Stop: Finishes capturing images from webcam.

Related commands:

Sound

Example:

```
WebCam Config, 1, 1, 20, 50, 10, "C:\Frames\Frame"

Button Start, 10, 60, "Start"
Button Stop, 15, 60, "Stop"

Wait ClickButton

WebCam Start

Wait ClickButton

WebCam Stop

Message "OK"
```

WeekDay Date, Language

Description:

Returns the number of the weekday (0-6) or in full.

Arguments:

Date: Reference date.
Language: Optional. If you pass as argument the language code, the weekday will be returned in full. Below the codes of language:
En: English
PT: Portuguese
Gr: German
Sp: Spanish
It: Italian
Fr: French

Related commands:

Date, Day, Month, Year, Time, DateDiff, DateAdd

Example:

```
Var X Integer, S String

X = WeekDay(Date()) 'Day number (0-6)
Write X

S = WeekDay(Date(), "En") 'English
Write S
```

While Condition ... Loop

Description:

Repeats the execution of a routine (commands, functions, subroutines, etc.) as the result of an expression is **True**. You should put the command **Loop** after the last line of the routine to be executed.

Arguments:

Condition: One or more expressions with comparison operators, and if more than one expression, logical operators (and/or) between them.

Related commands:

And, Or, If, Else

Example:

```
Var X Integer, Y Integer

X = 0
While X < 3 And Int(9.5) = 9
  Y = 0
  While Y < 2
    Write "Y = ", Y
    Y = Y + 1
  Loop
  Write "X = ", X
  X = X + 1
Loop
```

Window Arguments

Description:

Sets the main window, called the **Parent**.

Arguments:

Arguments can be entered on the same line and in any order, separated by commas.

Res = Lines, Columns: Changes the resolution of the window (default 27 x 79), then if the values are increased, the height of lines and the width of columns will decrease.
Size = Height, Width: Changes the height of lines and the width of columns of the window. If you write the word **Max** in the arguments, the width and height will adjust the maximum size of the video monitor.
Pos = Line, Column: Defines the starting row and column of the window relative to the Windows desktop. To refer to the upper left corner of the window. If you write the word **Center** in the arguments, the window will be centered horizontally and/or vertically.
Background = Image or Color, Line, Column: Defines a background image or background color for the window. If you enter the first argument the path of an image file, the picture will be displayed at the bottom of the window. The arguments **Line**, **Column** should be informed only when you want to position the image in a certain coordinate of the window. But in this case the image can be erased by the command **ClearWindow** (**Cls**), and also in case you assign a background color to the window. Note that you can assign a background image with a background color for the window, but in this case you must inform this argument twice, once for each case.
Fixed = Option: If set to **True** window will contain only the close button.
Box = Option: If set to true, all the control buttons of the window will be removed.
Border = Option: If Border = False, removes all borders of the window.
Enter = Option: If set to **False**, the change of focus from one component to another should be done by pressing **Tab**, if set to **True**, the change of focus can also be done by pressing **Enter**.
Cursor = Option: If set to **False**, the mouse cursor will be disabled, if set to **True**, the mouse cursor will be activated.
MouseGeneral = Option: If Option = **True**, the value of the reserved variables **MouseX** and **MouseY** contain the coordinates relative to the total area of the video monitor in pixels, if Option = **False**, the coordinates will refer only to the area of the active window in lines and columns.
MouseIcon = IconFile: Defines a new icon for the mouse cursor, for this you must enter the path of the icon file.
VideoRes = Width, Height: Defines the resolution of the video monitor (width an height in pixels).
PointSize = Pixels: Defines the size of the points in pixels (see **Point** command).
Visible: Sets the window as visible or invisible (**True** or **False**).

Related commands:

WindowHeight, WindowWidth, CaptureWindow, EndWindow, Font, MinimizeSysTray

Example:

```
Window Res=50,150, Size=10,17, Pos=Center,Center
Window Background="BobEsponja.gif", Background=White, Fixed=True
```

WindowsDir

Description:

Reserved variable that returns the path to the directory where the Windows is installed.

Related variables:

ProgramDir, ProgramsDir, Desktop, LBDir

WindowHeight

Description:

Reserved variable that returns the height of the active window.

Example:

```
Write WindowHeight(), ", ", WindowWidth()
```

WindowWidth

Description:

Command that returns the width of the active window.

Example:

```
Write WindowHeight(), ", ", WindowWidth()
```

Write Argument1, Argument2, ..., WindowTitle

Description:

Command that writes strings, variable contents, results of expressions, function and command results, in the active window to the position set by the command **Position**.

Arguments:

Argument1, Argument2, ...: The number of arguments is undefined, and can be variables, strings, expressions, functions, commands, in any order (separated by commas), where the result will be written on one line. If you put the command **Chr(13)** as an argument, the next arguments will be written in the next line.
WindowTitle: If you write this word as an argument, the result is written to the header of the active window.

Related commands:

Position, Font, JoinFile, Message

Example:

```
Var Age Integer

Age = 12
Position 5, 10
Write "Penny Robinson has ", Age, " years old."
```

WriteFile NickName, StartPos, String (binary mode)

WriteFile NickName, String (text mode)

Description:

Makes writing data to a file, in the mode set by the command **OpenFile** (binary or text).

Arguments:

NickName: Name of the file defined in command **OpenFile**.
StartPos: Starting position in bytes to write on the file, this argument must be informed only in binary mode.
String: String to write to the file, this argument must be informed only in text mode. The number of bytes written is the string length.

Related commands:

OpenFile, ReadFile, CloseFile, SizeFile

Example:

```
'Performs a write in binary mode
WriteFile "Test1", 1, "Logic Basic"

'Performs a write in text mode
WriteFile "Test2", "Programming language"
```

Year Date

Description:

Returns the year of the date passed as argument.

Arguments:

Date: Any string (or function return) that can represent a date. If you enter an empty string as an argument, the command assumes the system date.

Related commands:

Date, Day, Month, DateDiff, DateAdd, Weekday, Time

Example:

```
Write Year(Date())
```