OpenOffice.org BASIC

Simplified Reference

Author : **Malcolm Ripley**
Issue : **1**
Date : **11th September 2007**

Table of Contents

Introduction

This document is intended to satisfy the requirement for a concise OpenOffice.org BASIC reference. It is written for developers who already have a knowledge of other languages and merely want a reminder as to the specific flavours of flow control and functions that this language employs.

The first few sections describe where the macros are stored and organised followed by an alphabetic single line per item list of all statements and functions. A complete description of all these statements and functions, with examples, follows the simple list.

The vast majority of this document has been lifted from the OpenOffice.org application help files although it has been re-formatted to make it clearer.

General Use

Locating Macros

Macros are arranged in a hierarchy:

Location
    Library
        Module

Macros i.e. Functions and procedures (Sub), are grouped together in a module. A module is limited to 64K characters.

Modules are grouped together in libraries. A library can contain 16,000 modules.

Libraries can be attached to the openOffice.org application or a document.

When you access the macros the macro filer will show at least 3 top level **location**s:

MyMacros
These are user macros stored in the users own directory and are available to all the documents for the user.

OpenOffice.org.Macros
These are macros included with the current distribution of OpenOffice.org and are available to all documents of all users.

<current document name>
Macros stored here are accessible to the named document only.

Code Syntax

This has the following attributes:

- A single line of code flows over multiple lines by having a "_" character at the end
- It is case insensitive although there does seem to be an (unwritten ?) convention:
    - All keywords are capital headed, some are double capital headed, see list below.
    - All variables start with a single lower case letter indicating the data type followed by a Capital headed value e.g. smyFirstName, iAge. This ties in with use of the Defxxx declarations.
- Comments can follow the REM keyword or the single quote character '

Symbols

The usual math symbols "+ - * / " apply.

The "=" is used to assign values or equate values in logical expressions.

The "^" is used as the power symbol.

The "\" is used to perform integer division.

All boolean expressions use the words AND, EQV, OR, NOT, XOR, IMP. Where EQV is an exclusive NOR function and XOR is an exclusive OR function. The IMP function has, in reality, a bitwise scope and is easier to think of as:

A IMP B

returns the same result as:

A AND (NOT B)

Datatypes

Bool
Boolean variable (True, False)

Currency
Currency-Variable (Currency with 4 Decimal places)

Date
Date variable

Double
Double-precision floating-point variable (1,79769313486232 x 10E308 - 4,94065645841247 x 10E-324)

Integer
Integer variable (-32768 - 32767)

Long
Long integer variable (-2.147.483.648 - 2.147.483.647)

Object
Object variable (Note: this variable can only subsequently be defined with Set!)

Single
Single-precision floating-point variable (3,402823 x 10E38 - 1,401298 x 10E-45).

String
String variable consisting of a maximum of 64,000 ASCII characters (sic). although I suspect that the real number is 65535! Strings can be concatenated with a “&” or “+”.

[Variant]
Variant variable type (contains all types, specified by definition). If a key word is not specified, variables are automatically defined as Variant Type, unless a statement from DefBool to DefVar is used.

The following types are assigned to an integer datatype but the explicit representation is recognised:

Octal
These are represented by &O, for example &O77 (decimal value 63)

Hexadecimal
These are represented by &H, for example &HFF (decimal value 255)

Declarations

Declare {Sub | Function} Name Lib "Libname" [Alias "Aliasname"] [Parameter] [As Type]
Declares and defines a subroutine in a DLL file that you want to execute from OpenOffice.org Basic.

Defxxx Characterrange1[, Characterrange2[,...]]
If no type-declaration character or keyword is specified, the Defxxx statement sets the default data type for variables, according to a letter range. Valid values of **xxx** are **Bool, Cur, Date, Dbl, Int, Lng, Obj, Sng, Str, Var**

Option Base {0 | 1}
Defines the default lower boundary for arrays as 0 or 1

Option Explicit
Specifies that every variable in the program code must be explicitly declared with the Dim statement.

Public <VarName>[(<start> To <end>)] [As <VarType>][, <VarName2>[(<start> To <end>)] [As <VarType>][,...]]
Dimensions a variable or an array at the module level (that is, not within a subroutine or function), so that the variable and the array are valid in all libraries and modules.

Constants

I have discovered a few constants:

Empty
Subtly different from Null. In this case a value of empty means the variable has not had anything assigned to it yet.

False
Boolean value of false or 0 if assigned to an integer or float variable or “False” if assigned to a string.

Null
Note that this can only be assigned to a variant datatype but it can be compared to a variable of any datatype.

Nothing
A non existant object. This value can be assigned to an object datatype.

PI
3.1415....

True
Boolean value of true or -1 if assigned to an integer or float variable or “True” if assigned to a string.

Flow Control Statements

These are the statements that control how the code flows.

[Call] Name [Parameter]

Do [{While | Until} <true condition>]
[Exit Do]
Loop

Do
[Exit Do]
Loop [{While | Until} <true condition>]

For <counter>=<start> To <end> [Step <step>]
[Exit For]
Next [<counter>]

Function Name[(VarName1 [As Type][, [Optional] VarName2 [As Type][,...]]]) [As Type]
End Function

GoSub <label>
<label>:
Return

GoTo <label>
<Label>:

If <true condition> Then
[ElseIf <true condition> Then]
[Else]
End If

Select Case <cond>
Case <match>
Case Else
End Select

While <true condition>
Wend

Assigning Values

When assigning an object to a variable the set statement is optional as opposed to Excel where it is mandatory:

Set oDocument = ThisComponent

oDocument = ThisComponent

When assigning values to an array the brackets are optional:

alist = oCols.getElementNames

alist() = oCols.getElementNames

Language Summary

This is a very very quick reference!

Abs(<number>)
Returns the absolute value of a numeric expression.

Array(<Argument List>)
Returns the type Variant with a data field.

Asc(<string>)
Returns the ASCII (American Standard Code for Information Interchange) value of the first character in a string expression.

Atn(<number>)
Trigonometric function that returns the arctangent of a numeric expression. The return value is in the range -Pi/2 to +Pi/2.

Beep
Plays a tone through the computer's speaker. The tone is system-dependent and you cannot modify its volume or pitch.

Blue(<Color As Long>)
Returns the blue component of the specified color code.

[Call] Name [parameter]
Transfers the control of the program to a subroutine, a function, or a DLL procedure.

Cbool(<expression>)
Converts a string comparison or numeric comparison to a Boolean expression, or converts a single numeric expression to a Boolean expression.

Cbyte(<expression>)
Converts a string or a numeric expression to the type Byte.

CCur(<expression>)
Converts a string expression or numeric expression to a currency expression. The locale settings are used for decimal separators and currency symbols.

CDate(<expression>)
Converts any string or numeric expression to a date value.

CDateFromIso(<string>)
Returns the internal date number from a string that contains a date in ISO format.

CDateToIso(<number>)
Returns the date in ISO format from a serial date number that is generated by the DateSerial or the DateValue function.

CDbl(<expression>)
Converts any numerical expression or string expression to a double type.

CDec(<expression>)
Converts a string expression or numeric expression to a decimal expression.

ChDir <String>
Changes the current directory or drive.

ChDrive <string>

Changes the current drive.

Choose(<Index>,<s1>,<s2>...)
Returns a selected value from a list of arguments.

Chr(<Integer>)
Returns the character that corresponds to the specified character code.

CInt(<expression>)
Converts any string or numeric expression to an integer.

CLng(<expression>)
Converts any string or numeric expression to a long integer.

Close #<Filenum>[,#<Filenum2>...]
Closes a specified file that was opened with the Open statement.

Const <var> = <expression>
Defines a string as a constant.

ConvertFromURL(<filename>)
Converts a file URL to a system file name.

ConvertToURL(<filename>)
Converts a system file name to a file URL.

Cos(<number>)
Calculates the cosine of an angle. The angle is specified in radians. The result lies between -1 and 1.

CreateObject(<type>)
Creates a UNO object.

CreateUnoDialog(<Dialog object>)
Creates a Basic Uno object that represents a Uno dialog control during Basic runtime.

CreateUnoListener(<Prefixname>, <ListenerInterfaceName>)
Creates a Listener instance.

CreateUnoService(<Uno service name>)
Instantiates a Uno service with the ProcessServiceManager.

CreateUnoStruct(<Uno type name>)
Creates an instance of a Uno structure type

CreateUnoValue("[]byte", <MyBasicValue>)
Returns an object that represents a strictly typed value referring to the Uno type system.

Csng(<expression>)
Converts any string or numeric expression to data type Single.

Cstr(<number>)
Converts any numeric expression to a string expression.

CurDir(<string>)
Returns a variant string that represents the current path of the specified drive.

Cvar(<expression>)
Converts a string expression or numeric expression to a variant expression.

CVErr(<expression>)
Converts a string expression or numeric expression to a variant expression of the sub type "Error".

Date

Date = <string>
Returns the current system date as a string, or resets the date. The date format depends on your local system settings.

DateAdd(<Add>, <Count>, <Date>)
Adds a date interval to a given date a number of times and returns the resulting date.

DateDiff (<Add>, <Date1>, <Date2> [, <Week_start> [, <Year_start>]])
Returns the number of date intervals between two given date values.

DatePart (<Add>, <Date> [, <Week_start> [, <Year_start>]])
Returns a part of a date.

DateSerial (<year>, <month>, <day>)
Returns a Date value for a specified year, month, or day.

DateValue [(<date>)]
Returns a number from a date string. The date string is a complete date in a single numeric value. You can also use this serial number to determine the difference between two dates.

Day(<number>)
Returns a value that represents the day of the month based on a serial date number generated by DateSerial or DateValue.

Dim VarName [(<start> [To <end>])] [As VarType][, <VarName2> ....]
Declares a variable or an array.

DimArray(<Argument list>)
Returns a Variant array.

Dir [(<file or path>) [, <Attrib>]]
Returns the name of a file, a directory, or all of the files and the directories on a drive or in a directory that match the specified search path.

Environ(<environment>)
Returns the value of an environment variable as a string. Environment variables are dependent on the type of operating system that you have.

Eof(<filenumber>)
Determines if the file pointer has reached the end of a file.

EqualUnoObjects(<object1>, <object2>)
Returns True if the two specified Basic Uno objects represent the same Uno object instance.

Erase <arraylist>
Erases the contents of array elements of fixed size arrays, and releases the memory used by arrays of variable size.

Erl
Returns the line number where an error occurred during program execution.

Err
Returns an error code that identifies the error that occurred during program execution.

Error(<expression>)
Returns the error message that corresponds to a given error code.

Exp(<number>)
Returns the base of the natural logarithm (e = 2.718282) raised to a power.

FileAttr (<fileNumber>, <attribute>)
Returns the access mode or the file access number of a file that was opened with the Open statement. The file access number is dependent on the operating system (OSH = Operating System Handle).

FileCopy <sourcefilename>, <destinationfilename>
Copies a file.

FileDateTime(<filename>)
Returns a string that contains the date and the time that a file was created or last modified.

FileExists(<filename>)
Determines if a file or a directory is available on the data medium.

FileLen(<filename>)
Returns the length of a file in bytes.

FindObject(<objectname>)
Enables an object to be addressed at run-time as a string parameter through the object name.

FindPropertyObject(<object> , <propertyname>)
Enables objects to be addressed at run-time as a string parameter using the object name.

Fix(<number>)
Returns the integer value of a numeric expression by removing the fractional part of the number.

Format(<number>, <formatstring>)

Converts a number to a string, and then formats it according to the format that you specify.

FreeFileReturns
the next available file number for opening a file. Use this function to open a file using a file number that is not already in use by a currently open file.

FreeLibrary(<libraryname>)
Releases DLLs that were loaded by a Declare statement. A released DLL is automatically reloaded if one of its functions is called. See also: Declare

Get #<filenumber> , [<position>], <variable>
Reads a record from a relative file, or a sequence of bytes from a binary file, into a variable.

GetAttr(<filename>)
Returns a bit pattern that identifies the file type or the name of a volume or a directory.

GetDefaultContext
Returns the default context of the process service factory, if existent, else returns a null reference.

GetGuiType
Returns a numerical value that specifies the graphical user interface.

GetProcessServiceManager()
Returns the ProcessServiceManager (central Uno ServiceManager).

GetSolarVersion
Returns the internal number of the current OpenOffice.org version.

GetSystemTicks()
Returns the number of system ticks provided by the operating system. You can use this function to optimize certain processes.

Global <VarName> [(<start> To <end>)] [As VarType][, <VarName2>....]
Dimensions a variable or an array at the global level (that is, not within a subroutine or function), so that the variable and the array are valid in all libraries and modules for the current session.

Green(<integer>)
Returns the Green component of the given color code.

HasUnoInterfaces(<object> , <interfacename> [,<interfacename2>...])
Tests if a Basic Uno object supports certain Uno interfaces.

Hex(<number>)
Returns a string that represents the hexadecimal value of a number.

Hour(<number>)
Returns the hour from a time value that is generated by the TimeSerial or the TimeValue function.

Iif(<evaluate>,<truevalue>,<falsevalue>)
Returns one of two possible function results, depending on the logical value of the evaluated expression.

Input #<filenumber>; <var1> [, <var2>....]
Reads data from an open sequential file.

InputBox (<message> [, <title>[, <default_input>[, <x_pos> , <y_pos>]]]])
Displays a prompt in a dialog at which the user can input text. The input is assigned to a variable.

InStr([<start_position> ,] <searched_string> , <substring> [, <compare_type>])
Returns the position of a string within another string.

Int(<number>)
Returns the integer portion of a number.

IsArray(<variable>)
Determines if a variable is a data field in an array.

IsDate(<expression>)
Tests if a numeric or string expression can be converted to a Date variable.

IsEmpty(<variable>)
Tests if a Variant variable contains the Empty value. The Empty value indicates that the variable

is not initialized.

IsError(<variable>)
Tests if a variable contains an error value.

IsMissing(argumentname)
Tests if a function is called with an optional parameter.

IsNull(<variable>)
Tests if a Variant contains the special Null value, indicating that the variable does not contains data.

IsNumeric(<variable>)
Tests if an expression is a number. If the expression is a number, the function returns True, otherwise the function returns False.

IsObject(<variable>)
Tests if an object variable is an OLE object. The function returns True if the variable is an OLE object, otherwise it returns False.

IsUnoStruct(object)
Returns True if the given object is a Uno struct.

Join(<string>)_array,delimiter)
Returns a string from a number of substrings in a string array.

Kill <filename>
Deletes a file from a disk.

Lbound(<arrayname> [, <dimension>])
Returns the lower boundary of an array.

Lcase(<string>)
Converts all uppercase letters in a string to lowercase.

Left(<string>),<characters>)
Returns the number of leftmost characters that you specify of a string expression.

Len(<string>)
Returns the number of characters in a string, or the number of bytes that are required to store a variable.

Let
Assigns a value to a variable.

Line Input #filenumber , variable
Reads strings from a sequential file into a variable.

Loc(<filenumber>)
Returns the current position in an open file.

Lof(<filenumber>)
Returns the size of an open file in bytes.

Log(<number>)
Returns the natural logarithm of a number.

Lset <variable> = <string literal>

Lset <variable> = <variable>
Aligns a string to the left of a string variable, or copies a variable of a user-defined type to another variable of a different user-defined type.

Ltrim(<string>)
Removes all leading spaces at the start of a string expression

Mid(<string>, <start_position>[,<length>])

Mid(<string>, <start_position>, <length>, <substring>)
Returns the specified portion of a string expression (Mid function), or replaces the portion of a string expression with another string (Mid statement).

Minute(<number>)
Returns the minute of the hour that corresponds to the serial time value that is generated by the TimeSerial or the TimeValue function.

MkDir <directoryspec>

Creates a new directory on a data medium.

<number> Mod <number>
Returns the integer remainder of a division.

Month(serial_date_number)
Returns the month of a year from a serial date that is generated by the DateSerial or the DateValue function

MsgBox(<message> [,<dialog_type> [,<dialog_title>]])
Displays a dialog box containing a message and returns a value.

MsgBox <message> [,<dialog_type> [,<dialog_title>]]
Displays a dialog box containing a message.

Name <old_filename> As <New_filename>
Renames an existing file or directory.

Now
Returns the current system date and time as a Date value.

Oct(<number>)
Returns the octal value of a number.

Open <filename> [For <Mode>] [Access <IOMode>] [Protected] As [#]<filenumber> As Integer [Len = <datasetlength>]
Opens a data channel.

Print <expression1>[{;|,} [Spc(<integer>);] [Tab(<integer>);] [<Expression2>[...]]
Prints the specified strings or numeric expressions in a dialog.

Put [#]<filenumber> , [<position>] , <variable>
Writes a record to a relative file or a sequence of bytes to a binary file.

Randomize [<integer>]
Initializes the random-number generator.

Red(<integer>)
Returns the Red component of the specified color code.

ReDim <VarName> [(<start> To <end>)] [As <VarType>][, <VarName2> [(<start> To <end>)] [As <VarType>][,...]]
Declares a variable or an array.

Reset
Closes all open files and writes the contents of all file buffers to the harddisk.

Right(<string>,<integer>)
Returns the rightmost "n" characters of a string expression.

RmDir <directory name>
Deletes an existing directory from a data medium.

Rnd(<number>)
Returns a random number between 0 and 1

Rset <variable> = <string literal>

Rset <variable> = <variable>
Right-aligns a string within a string variable, or copies a user-defined variable type into another.

Rtrim(<string>)
Deletes the spaces at the end of a string expression.

Second(<serial time>)
Returns an integer that represents the seconds of the serial time number that is generated by the TimeSerial or the TimeValue function.

Seek(<filenumber>)
Determines the position for the next writing or reading in a file that was opened with the open statement. The return value corresponds to the value determined by the Seek statement:

Seek [#]<filenumber> , <position>
Determines the position for the next writing or reading in a file that was opened with the Open statement.

Set <variable or property> = <object>
Sets an object reference on a variable or a Property.

SetAttr <filename> , <attribute>
Sets the attribute information for a specified file.

Sgn(<number>)
Returns an integer number between -1 and 1 that indicates if the number that is passed to the function is positive, negative, or zero.

Shell(<pathname> [, <windowstyle>] [,<parameter>] [,<sync flag>])
Starts another application and defines the respective window style, if necessary.

Sin(<number>)
Returns the sine of an angle. The angle is specified in radians. The result lies between -1 and 1.

Space(<long integer>)
Returns a string that consists of a specified amount of spaces.

Split(<string> , <delimiter> , <number>)
Returns an array of substrings from a string expression.

Sqr(<number>)
Calculates the square root of a numeric expression.

Static <VarName>[(<start> To <end>)] [As <VarType>], <VarName2>.....
Declares a variable or an array at the procedure level within a subroutine or a function, so that the values of the variable or the array are retained after exiting the subroutine or function. Dim statement conventions are also valid.

Stop
Stops the execution of the Basic program.

Str(<number>)
Converts a numeric expression into a string.

StrComp(<string1>, <string2> , <compare_flag>))
Compares two strings and returns an integer value that represents the result of the comparison.

String(<long> , <ascii_number>|<string>)
Creates a string according to the specified character, or the first character of a string expression that is passed to the function.

Switch(<expression1>,<value1> [,<expression2>,<value2> [,....]])
Evaluates a list of arguments, consisting of an expression followed by a value. The Switch function returns a value that is associated with the expression that is passed by this function.

Tan(<number>)
Determines the tangent of an angle. The angle is returned in radians.

Time
This function returns the current system time as a string in the format "HH:MM:SS".

Timer
Returns a value that specifies the number of seconds that have elapsed since midnight.

TimeSerial(<hour>, <minute>, <second>)
Calculates a serial time value for the specified hour, minute, and second parameters that are passed as numeric value. You can then use this value to calculate the difference between times.

TimeValue(<time string>)
Calculates a serial time value from the specified hour, minute, and second - parameters passed as strings - that represents the time in a single numeric value. This value can be used to calculate the difference between times.

Trim(<string>)
Removes all leading and trailing spaces from a string expression.

TwipsPerPixelX
Returns the number of twips that represent the width of a pixel.

TwipsPerPixelY
Returns the number of twips that represent the height of a pixel.

TypeName(<variable name>)
Returns a string that contains information for a variable.

Ubound(<arrayname [, <dimension>])
Returns the upper boundary of an array.

Ucase(<string>)
Converts lowercase characters in a string to uppercase.

Val(<string>)
Converts a string to a numeric expression.

VarType(<variable name>)
Returns a numeric value that contains information for a variable.

Wait <milliseconds>
Interrupts the program execution for the amount of time that you specify in milliseconds.

WeekDay(<integer>)
Returns the number corresponding to the weekday represented by a serial date number that is generated by the DateSerial or the DateValue function.

With <object> <statement block> End With
Sets an object as the default object. Unless another object name is declared, all properties and methods refer to the default object until the End With statement is reached.

Write [#]<filenumber> , [expression1 [,<expression2 [,.....]]]
Writes data to a sequential file.

Year(<integer>)
Returns the year from a serial date number that is generated by the DateSerial or the DateValue function.

Language Details

ABS()

Returns the absolute value of a numeric expression.

Syntax

Abs (Number)

Return value

Double

Parameters

Number:
Any numeric expression that you want to return the absolute value for. Positive numbers, including 0, are returned unchanged, whereas negative numbers are converted to positive numbers.

The following example uses the Abs function to calculate the difference between two values. It does not matter which value you enter first.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleDifference
      Dim siW1 As Single
      Dim siW2 As Single
      siW1 = Int(InputBox$ ("Please enter the first amount","Value input"))
      siW2 = Int(InputBox$ ("Please enter the second amount","Value input"))
      Print "The difference is "; Abs(siW1 - siW2)
End Sub
```

AND

Logically combines two expressions.

Syntax

Result = Expression1 And Expression2

Parameters

Result
Any numeric variable that records the result of the combination.

Expression1, Expression2
Any expressions that you want to combine.

Boolean expressions combined with AND only return the value **True** if both expressions evaluate to **True**:

True AND **True** returns **True**; for all other combinations the result is **False**.

The AND operator also performs a bitwise comparison of identically positioned bits in two numeric expressions.

Example

```
Sub ExampleAnd
      Dim A as Variant, B as Variant, C as Variant, D as Variant
      Dim vVarOut as Variant
      A = 10: B = 8: C = 6: D = Null
      vVarOut = A > B And B > C REM returns -1
      vVarOut = B > A And B > C REM returns 0
      vVarOut = A > B And B > D REM returns 0
      vVarOut = (B > D And B > A) REM returns 0
      vVarOut = B And A REM returns 8 due to the bitwise AND combination of
both arguments
End Sub
```

Array()

Returns the type Variant with a data field.

Syntax

Array (Argument list)

See also DimArray.

Parameters:

Argument list
A list of any number of arguments that are separated by commas.

Example:

```
Dim A As Variant
A = Array("Fred","Tom","Bill")
Msgbox A(2)
```

Asc()

Returns the ASCII (American Standard Code for Information Interchange) value of the first character in a string expression.

Syntax

Asc (Text As String)

Return value

Integer

Parameters

Text
Any valid string expression. Only the first character in the string is relevant.

Use the Asc function to replace keys with values. If the Asc function encounters a blank string, OpenOffice.org Basic reports a run-time error. In addition to 7 bit ASCII characters (Codes 0-127), the ASCII function can also detect non-printable key codes in ASCII code. This function can also handle 16 bit unicode characters.

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleASC
      Print ASC("A") REM returns 65
      Print ASC("Z") REM returns 90
      Print ASC("Las Vegas") REM returns 76, since only the first character is
taken into account
End Sub
```

Related Topics

CHR

Atn()

Trigonometric function that returns the arctangent of a numeric expression. The return value is in the range -Pi/2 to +Pi/2.

The arctangent is the inverse of the tangent function. The Atn Function returns the angle "Alpha", expressed in radians, using the tangent of this angle. The function can also return the angle "Alpha" by comparing the ratio of the length of the side that is opposite of the angle to the length of the side that is adjacent to the angle in a right-angled triangle.

Atn(side opposite the angle/side adjacent to angle)= Alpha

Syntax

Atn (Number)

Return value

Double

Parameters

Number
Any numerical expression that represents the ratio of two sides of a right triangle. The Atn function returns the corresponding angle in radians (arctangent).

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

degree=(radian*180)/pi

radian=(degree*pi)/180

Pi is here the fixed circle constant with the rounded value 3.14159.

Error Codes

5 Invalid procedure call

Example:

```
REM The following example calculates for a right-angled triangle
REM the angle Alpha from the tangent of the angle Alpha:
Sub ExampleATN
      REM rounded Pi = 3.14159 is a predefined constant
      Dim d1 As Double
      Dim d2 As Double
      d1 = InputBox$ ("Enter the length of the side adjacent to the angle:
","Adjacent")
      d2 = InputBox$ ("Enter the length of the side opposite the angle:
","Opposite")
      Print "The Alpha angle is"; (atn (d2/d1) * 180 / Pi); " degrees"
End Sub
```

Beep

Plays a tone through the computer's speaker. The tone is system-dependent and you cannot modify its volume or pitch.

Syntax

Beep

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleBeep
        beep
        beep
        beep
end sub
```

Blue()

Returns the blue component of the specified color code.

Syntax

```
Blue (Color As Long)
```

Return value

Integer

Parameter

Color
Long integer expression that specifies any color code for which to return the blue component.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleColor
      Dim lVar As Long
      lVar = rgb(128,0,200)
      MsgBox "The color " & lVar & " consists of:" & Chr(13) &_
      "red= " & Red(lVar) & Chr(13)&_
      "green= " & Green(lVar) & Chr(13)&_
      "blue= " & Blue(lVar) & Chr(13) , 64,"colors"
End Sub
```

Call

Transfers the control of the program to a subroutine, a function, or a DLL procedure.

Syntax

[Call] Name [Parameter]

Parameters

Name

Name of the subroutine, the function, or the DLL that you want to call

Parameter

Parameters to pass to the procedure. The type and number of parameters is dependent on the routine that is executing.

A keyword is optional when you call a procedure. If a function is executed as an expression, the parameters must be enclosed by brackets in the statement. If a DLL is called, it must first be specified in the **Declare-Statement**.

Example:

```
Sub ExampleCall
      Dim sVar As String
      sVar = "Office"
      Call f_callFun sVar
      end Sub
      Sub f_callFun (sText as String)
      Msgbox sText
end sub
```

Cbool()

Converts a string comparison or numeric comparison to a Boolean expression, or converts a single numeric expression to a Boolean expression.

Syntax

CBool (Expression1 {= | <> | < | > | <= | >=} Expression2) or CBool (Number)

Return value

Bool

Parameters

Expression1, Expression2
Any string or numeric expressions that you want to compare. If the expressions match, the **CBool** function returns **True**, otherwise **False** is returned.

Number
Any numeric expression that you want to convert. If the expression equals 0, **False** is returned, otherwise **True** is returned.

The following example uses the **CBool** function to evaluate the value that is returned by the **Instr** function. The function checks if the word "and" is found in the sentence that was entered by the user.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCBool
      Dim sText As String
      sText = InputBox("Please enter a short sentence:")
      REM Proof if the word »and« appears in the sentence.
      REM Instead of the command line
      REM If Instr(Input, "and")<>0 Then...
      REM the CBool function is applied as follows:
      If CBool(Instr(sText, "and")) Then
            MsgBox "The word »and« appears in the sentence you entered!"
      EndIf
End Sub
```

Cbyte()

Converts a string or a numeric expression to the type Byte.

Syntax

Cbyte(expression)

Return value

Byte

Parameters

Expression
A string or a numeric expression.

Error Codes

5 Invalid procedure call

CCur()

Converts a string expression or numeric expression to a currency expression. The locale settings are used for decimal separators and currency symbols.

Syntax

CCur(Expression)

Return value

Currency

Parameter

Expression: Any string or numeric expression that you want to convert.

CDate()

Converts any string or numeric expression to a date value.

Syntax

CDate (Expression)

Return value

Date

Parameters

Expression
Any string or numeric expression that you want to convert.

When you convert a string expression, the date and time must be entered in the format MM.DD.YYYY HH.MM.SS, as defined by the **DateValue** and **TimeValue** function conventions. In numeric expressions, values to the left of the decimal represent the date, beginning from December 31, 1899. Values to the right of the decimal represent the time.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCDate
      MsgBox cDate(1000.25) REM 09.26.1902 06:00:00
      MsgBox cDate(1001.26) REM 09.27.1902 06:14:24
End Sub
```

CDateFromIso()

Returns the internal date number from a string that contains a date in ISO format.

Syntax

CDateFromIso(String)

Return value

Internal date number

Parameters

String
A string that contains a date in ISO format. The year may have two or four digits.

Error Codes

5 Invalid procedure call

Example

```
dateval = CdateFromIso("20021231")
```

returns 12/31/2002 in the date format of your system

CDateToIso()

Returns the date in ISO format from a serial date number that is generated by the DateSerial or the DateValue function.

Syntax

CDateToIso(Number)

Return value

String

Parameters

Number
Integer that contains the serial date number.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCDateToIso
      MsgBox "" & CDateToIso(Now) ,64,"ISO Date"
End Sub
```

CDbl()

Converts any numerical expression or string expression to a double type.

Syntax

CDbl (Expression)

Return value

Double

Parameters

Expression

Any string or numeric expression that you want to convert. To convert a string expression, the number must be entered as normal text ("123.5") using the default number format of your operating system.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCountryConvert
      Msgbox CDbl(1234.5678)
      Msgbox CInt(1234.5678)
      Msgbox CLng(1234.5678)
End Sub
```

CDec()

Converts a string expression or numeric expression to a decimal expression.

Syntax

CDec(Expression)

Return value

Decimal number.

Parameter

Expression
Any string or numeric expression that you want to convert.

ChDir

Changes the current directory or drive.

Syntax

```
ChDir Text As String
```

Parameters

Text
Any string expression that specifies the directory path or drive.

If you only want to change the current drive, enter the drive letter followed by a colon.

Error Codes

5 Invalid procedure call

76 Path not found

Example

```
Sub ExampleChDir
      Dim sDir1 as String , sDir2 as String
      sDir1 = "c:\Test"
      sDir2 = "d:\private"
      ChDir( sDir1 )
      MsgBox CurDir
      ChDir( sDir2 )
      MsgBox CurDir
End Sub
```

ChDrive

Changes the current drive.

Syntax

```
ChDrive Text As String
```

Parameters

Text
Any string expression that contains the drive letter of the new drive. If you want, you can use URL notation.

The drive must be assigned a capital letter. Under Windows, the letter that you assign the drive is restricted by the settings in LASTDRV. If the drive argument is a multiple-character string, only the first letter is relevant. If you attempt to access a non-existent drive, an error occurs that you can respond to with the OnError statement.

Error Codes

5 Invalid procedure call

68 Device not available

76 Path not found

Example

```
Sub ExampleCHDrive
      ChDrive "D" REM Only possible if a drive 'D' exists.
End Sub
```

Choose()

Returns a selected value from a list of arguments.

Syntax

Choose (Index, Selection1[, Selection2, ... [,Selection_n]])

Parameters

Index
A numeric expression that specifies the value to return.

Selection1
Any expression that contains one of the possible choices.

The **Choose** function returns a value from the list of expressions based on the index value. If Index = 1, the function returns the first expression in the list, if index i= 2, it returns the second expression, and so on.

If the index value is less than 1 or greater than the number of expressions listed, the function returns a Null value.

The following example uses the **Choose** function to select a string from several strings that form a menu:

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleChoose
      Dim sReturn As String
      sReturn = ChooseMenu(2)
      Print sReturn
      End Sub

Function ChooseMenu(Index As Integer)
      ChooseMenu = Choose(Index, "Quick Format", "Save Format", _
            "System Format")
End Function
```

Chr()

Returns the character that corresponds to the specified character code.

Syntax

Chr(Expression As Integer)

Return value

String

Parameters

Expression
Numeric variables that represent a valid 8 bit ASCII value (0-255) or a 16 bit Unicode value.

Use the **Chr$** function to send special control sequences to a printer or to another output source. You can also use it to insert quotation marks in a string expression.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleChr
      REM This example inserts quotation marks (ASCII value 34) in a string.
      MsgBox "A "+ Chr$(34)+"short" + Chr$(34)+" trip."
      REM The printout appears in the dialog as: A "short" trip.
End Sub
```

Related Topics

ASC

CInt()

Converts any string or numeric expression to an integer.

Syntax

CInt (Expression)

Return value

Integer

Parameters

Expression

Any numeric expression that you want to convert. If the **Expression** exceeds the value range between -32768 and 32767, OpenOffice.org Basic reports an overflow error. To convert a string expression, the number must be entered as normal text ("123.5") using the default number format of your operating system.

This function always rounds the fractional part of a number to the nearest integer.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCountryConvert
      MsgBox CDbl(1234.5678)
      MsgBox CInt(1234.5678)
      MsgBox CLng(1234.5678)
End Sub
```

CLng()

Converts any string or numeric expression to a long integer.

Syntax

CLng (Expression)

Return value

Long

Parameters

Expression

Any numerical expression that you want to convert. If the **Expression** lies outside the valid long integer range between -2.147.483.648 and 2.147.483.647, OpenOffice.org Basic returns an overflow error. To convert a string expression, the number must be entered as normal text ("123.5") using the default number format of your operating system.

This function always rounds the fractional part of a number to the nearest integer.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCountryConvert
      MsgBox CDbl(1234.5678)
      MsgBox CInt(1234.5678)
      MsgBox CLng(1234.5678)
End Sub
```

Close

Closes a specified file that was opened with the Open statement.

Syntax

```
Close FileNumber As Integer[, FileNumber2 As Integer[,...]]
```

Parameters

FileNumber

Any integer expression that specifies the number of the data channel that was opened with the **Open** statement.

Example

```
Sub ExampleworkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      sMsg = ""
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "First line of text"
      Print #iNumber, "Another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      While Not Eof(iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" then
                  sMsg = sMsg & sLine & chr(13)
            End If
      Wend
      Close #iNumber
      MsgBox sMsg
End Sub
```

Const

Defines a string as a constant.

Syntax

Const Text = Expression

Parameters

Text

Any constant name that follows the standard variable naming conventions.

A constant is a variable that helps to improve the readability of a program. Constants are not defined as a specific type of variable, but rather are used as placeholders in the code. You can only define a constant once and it cannot be modified.

The type of expression is irrelevant. If a program is started, OpenOffice.org Basic converts the program code internally so that each time a constant is used, the defined expression replaces it.

Example

```
Sub ExampleConst
      Const iVar = 1964
      MsgBox iVar
      Const sVar = "Program", dVar As Double = 1.00
      MsgBox sVar & " " & dVar
End Sub
```

ConvertFromURL()

Converts a file URL to a system file name.

Syntax

ConvertFromURL(filename)

Return value:

String

Parameters

Filename
A file name as a string.

Error Codes

5 Invalid procedure call

Example

```
systemFile$ = "c:\folder\mytext.txt"
url$ = ConvertToURL( systemFile$ )
Print url$
systemFileAgain$ = ConvertFromURL( url$ )
Print systemFileAgain$
```

ConvertToURL()

Converts a system file name to a file URL.

Syntax

ConvertToURL(filename)

Return value

String

Parameters

Filename
A file name as string.

Error Codes

5 Invalid procedure call

Example

```
systemFile$ = "c:\folder\mytext.txt"
url$ = ConvertToURL( systemFile$ )
Print url$
systemFileAgain$ = ConvertFromURL( url$ )
Print systemFileAgain$
```

Cos()

Calculates the cosine of an angle. The angle is specified in radians. The result lies between -1 and 1.

Using the angle Alpha, the Cos-Function calculates the ratio of the length of the side that is adjacent to the angle, divided by the length of the hypotenuse in a right-angled triangle.

Cos(Alpha) = Adjacent/Hypotenuse

Syntax

Cos (Number)

Return value

Double

Parameters

Number
Numeric expression that specifies an angle in radians that you want to calculate the cosine for.

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

degree=(radian*180)/pi

radian=(degree*pi)/180

Pi is here the fixed circle constant with the rounded value 3.14159...

Error Codes

5 Invalid procedure call

Example

```
REM The following example allows for a right-angled triangle the input of
REM secant and angle (in degrees) and calculates the length of the hypotenuse:
Sub ExampleCosinus
      REM rounded Pi = 3.14159
      Dim d1 as Double, dAngle as Double
      d1 = InputBox$ (""Enter the length of the adjacent side: ","Adjacent")
      dAngle = InputBox$ ("Enter the angle Alpha (in degrees): ","Alpha")
      Print "The length of the hypothenuse is"; (d1 / cos (dAngle * Pi / 180))
End Sub
```

CreateObject()

Creates a UNO object.

This method creates instances of the type that is passed as parameter.

Syntax

oObj = CreateObject(type)

Example

```
Type address
      Name1 as String
      City as String
End Type

Sub main
      myaddress = CreateObject("address")
      MsgBox IsObject(myaddress)
End Sub
```

CreateUnoDialog()

Creates a Basic Uno object that represents a Uno dialog control during Basic runtime.

Dialogs are defined in the dialog libraries. To display a dialog, a "live" dialog must be created from the library.

See Examples.

Syntax

CreateUnoDialog(oDlgDesc)

Example

```
' Get dialog description from the dialog library
oDlgDesc = DialogLibraries.Standard.Dialog1
' generate "live" dialog
oDlgControl = CreateUnoDialog( oDlgDesc )
' display "live" dialog
oDlgControl.execute
```

CreateUnoListener()

Creates a Listener instance.

Many Uno interfaces let you register listeners on a special listener interface. This allows you to listen for specific events and call up the appropriate listener method. The CreateUnoListener function waits for the called listener interface and then passes the interface an object that the interface supports. This object is then passed to the method to register the listener.

Syntax

oListener = CreateUnoListener(Prefixname, ListenerInterfaceName)

Example

The following example is based on a Basic library object.

```
Dim oListener
oListener =
CreateUnoListener("ContListener_","com.sun.star.container.XContainerListener" )
```

The CreateUnoListener method requires two parameters. The first is a prefix and is explained in detail below. The second parameter is the fully qualified name of the Listener interface that you want to use.

The Listener must then be added to the Broadcaster Object. This is done by calling the appropriate method for adding a Listener. These methods always follow the pattern "addFooListener", where "Foo" is the Listener Interface Type, without the 'X'. In this example, the addContainerListener method is called to register the XContainerListener:

```
Dim oLib
oLib = BasicLibraries.Library1 ' Library1 must exist!
oLib.addContainerListener( oListener ) ' Register the listener
```

The Listener is now registered. When an event occurs, the corresponding Listener calls the appropriate method from the com.sun.star.container.XContainerListener Interface.

The prefix calls registered Listeners from Basic-subroutines. The Basic run-time system searches for Basic-subroutines or functions that have the name "PrefixListenerMethode" and calls them when found. Otherwise, a run-time error occurs.

In this example, the Listener-Interface uses the following methods:

- disposing:
- Listener base interface (com.sun.star.lang.XEventListener): base interface for all Listener Interfaces
- elementInserted:
- Method of the com.sun.star.container.XContainerListener interface
- elementRemoved:
- Method of the com.sun.star.container.XContainerListener interface
- elementReplaced:
- Method of the com.sun.star.container.XContainerListener interface

In this example, the prefix is ContListener_. The following subroutines must therefore be implemented in Basic:

- ContListener_disposing
- ContListener_elementInserted

- ContListener_elementRemoved
- ContListener_elementReplaced

An event structure type that contains information about an event exists for every Listener type. When a Listener method is called, an instance of this event is passed to the method as a parameter. Basic Listener methods can also call these event objects, so long as the appropriate parameter is passed in the Sub declaration. For example:

```
Sub ContListener_disposing( oEvent )
      MsgBox "disposing"
      MsgBox oEvent.Dbg_Properties
End Sub

Sub ContListener_elementInserted( oEvent )
      MsgBox "elementInserted"
      MsgBox oEvent.Dbg_Properties
End Sub

Sub ContListener_elementRemoved( oEvent )
      MsgBox "elementRemoved"
      MsgBox oEvent.Dbg_Properties
End Sub

Sub ContListener_elementReplaced( oEvent )
      MsgBox "elementReplaced"
      MsgBox oEvent.Dbg_Properties
End Sub
```

You do not need to include the parameter of an event object if the object is not used:

```
' Minimal implementation of Sub disposing
Sub ContListener_disposing
End Sub
```

Listener methods must **always** be implemented to avoid Basic run-time errors.

CreateUnoService()

Instantiates a Uno service with the ProcessServiceManager.

Syntax

oService = CreateUnoService(Uno service name)

For a list of available services, go to:
http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html

Examples

```
oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
```

The following code uses a service to open a file open dialog:

```
Sub Main
      fName = FileOpenDialog ("Please select a file")
      Print "file chosen: "+fName
End Sub

Function FileOpenDialog(title as String) as String
      filepicker = createUnoService("com.sun.star.ui.dialogs.FilePicker")
      filepicker.Title = title
      filepicker.execute()
      files = filepicker.getFiles()
      FileOpenDialog=files(0)
End function
```

CreateUnoStruct()

Creates an instance of a Uno structure type.

Use the following structure for your statement:

Dim oStruct as new com.sun.star.beans.Property

Syntax

oStruct = CreateUnoStruct(Uno type name)

Example

```
oStruct = CreateUnoStruct( "com.sun.star.beans.Property" )
```

CreateUnoValue()

Returns an object that represents a strictly typed value referring to the Uno type system.

This object is automatically converted to an Any of the corresponding type when passed to Uno. The type must be specified by its fully qualified Uno type name.

The OpenOffice.org API frequently uses the Any type. It is the counterpart of the Variant type known from other environments. The Any type holds one arbitrary Uno type and is used in generic Uno interfaces.

Syntax

oUnoValue = CreateUnoValue("[]byte", MyBasicValue) to get a byte sequence.

If CreateUnoValue cannot be converted to the specified Uno type, and error occurs. For the conversion, the TypeConverter service is used.

This function is intended for use in situations where the default Basic to Uno type converting mechanism is insufficient. This can happen when you try to access generic Any based interfaces, such as XPropertySet::setPropertyValue(Name, Value) or X???Container::insertBy???(???, Value), from OpenOffice.org Basic. The Basic runtime does not recognize these types as they are only defined in the corresponding service.

In this type of situation, OpenOffice.org Basic chooses the best matching type for the Basic type that you want to convert. However, if the wrong type is selected, an error occurs. You there use the CreateUnoValue() function to create a value for the unknown Uno type.

You can also use this function to pass non-Any values, but this is not recommend. If Basic already knows the target type, using the CreateUnoValue() function will only lead to additional converting operations that slow down the Basic execution.

Csng()

Converts any string or numeric expression to data type Single.

Syntax

CSng (Expression)

Return value

Single

Parameters

Expression
Any string or numeric expression that you want to convert. To convert a string expression, the number must be entered as normal text ("123.5") using the default number format of your operating system.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCSNG
      MsgBox CDbl(1234.5678)
      MsgBox CInt(1234.5678)
      MsgBox CLng(1234.5678)
      MsgBox CSng(1234.5678)
End Sub
```

Cstr()

Converts any numeric expression to a string expression.

Syntax

CStr (Expression)

Return value

String

Parameters

Expression
Any valid string or numeric expression that you want to convert.

Expression Types and Conversion Returns

Boolean :	String that evaluates to either **True** or **False**.
Date :	String that contains the date and time.
Null :	Run-time error.
Empty :	String without any characters.
Any :	Corresponding number as string.

Zeros at the end of a floating-point number are not included in the returned string.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleCSTR
      Dim sVar As String
      MsgBox CDbl(1234.5678)
      MsgBox CInt(1234.5678)
      MsgBox CLng(1234.5678)
      MsgBox CSng(1234.5678)
      sVar = CStr(1234.5678)
      MsgBox sVar
End Sub
```

CurDir()

Returns a variant string that represents the current path of the specified drive.

Syntax

```
CurDir [(Text As String)]
```

Return value

String

Parameters

Text

Any string expression that specifies an existing drive (for example, "C" for the first partition of the first hard drive).

If no drive is specified or if the drive is a zero-length string (""), CurDir returns the path for the current drive. OpenOffice.org Basic reports an error if the syntax of the drive description is incorrect, the drive does not exist, or if the drive letter occurs after the letter defined in the CONFIG.SYS with the Lastdrive statement.

This function is not case-sensitive.

Error Codes

5 Invalid procedure call

68 Device not available

7 Out of memory

51 Internal error

Example

```
Sub ExampleCurDir
      Dim sDir1 as String , sDir2 as String
      sDir1 = "c:\Test"
      sDir2 = "d:\private"
      ChDir( sDir1 )
      MsgBox CurDir
      ChDir( sDir2 )
      MsgBox CurDir
End Sub
```

Cvar()

Converts a string expression or numeric expression to a variant expression.

Syntax

CVar(Expression)

Return value

Variant.

Parameter

Expression
Any string or numeric expression that you want to convert.

CVErr()

Converts a string expression or numeric expression to a variant expression of the sub type "Error".

Syntax

CVErr(Expression)

Return value

Variant.

Parameter

Expression
Any string or numeric expression that you want to convert.

Date

Returns the current system date as a string, or resets the date. The date format depends on your local system settings.

Syntax

Date ; Date = Text As String

Parameters

Text
Only required in order to reset the system date. In this case, the string expression must correspond to the date format defined in your local settings.

Example

```
Sub ExampleDate
      MsgBox "The date is " & Date
End Sub
```

DateAdd()

Adds a date interval to a given date a number of times and returns the resulting date.

Syntax

DateAdd (Add, Count, Date)

Return value

A Variant containing a date.

Parameters

Add

A string expression from the following table, specifying the date interval.

Add (string value)	Explanation
yyyy	Year
q	Quarter
m	Month
y	Day of year
w	Weekday
ww	Week of year
d	Day
h	Hour
n	Minute
s	Second

Count

A numerical expression specifying how often the Add interval will be added (Count is positive) or subtracted (Count is negative).

Date

A given date or the name of a Variant variable containing a date. The Add value will be added Count times to this value.

Example

```
Sub example_dateadd
      MsgBox DateAdd("m", 1, "1/31/2004") &" - "& DateAdd("m", 1, "1/31/2005")
End Sub
```

DateDiff()

Returns the number of date intervals between two given date values.

Syntax

DateDiff (Add, Date1, Date2 [, Week_start [, Year_start]])

Return value

A number.

Parameters

Add

A string expression from the following table, specifying the date interval.

Add (string value)	Explanation
yyyy	Year
q	Quarter
m	Month
y	Day of year
w	Weekday
ww	Week of year
d	Day
h	Hour
n	Minute
s	Second

Date1, Date2

The two date values to be compared.

Week_start

An optional parameter that specifies the starting day of a week.

Week_start value	Explanation
0	Use system default value
1	Sunday (default)
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

Year_start

An optional parameter that specifies the starting week of a year.

Year_start value	Explanation
0	Use system default value
1	Week 1 is the week with January, 1st (default)
2	Week 1 is the first week containing four or more days of that year
3	Week 1 is the first week containing only days of the new year

Example

```
Sub example_datediff
      MsgBox DateDiff("d", "1/1/2005", "12/31/2005")
End Sub
```

DatePart()

Returns a part of a date.

Syntax

DatePart (Add, Date [, Week_start [, Year_start]])

Return value

A date value.

Parameters

Add

A string expression from the following table, specifying the date interval.

Add (string value)	Explanation
yyyy	Year
q	Quarter
m	Month
y	Day of year
w	Weekday
ww	Week of year
d	Day
h	Hour
n	Minute
s	Second

Date

The date from which the result is calculated.

Week_start

An optional parameter that specifies the starting day of a week.

Week_start value	Explanation
0	Use system default value
1	Sunday (default)
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

Year_start
An optional parameter that specifies the starting week of a year.

Year_start value	Explanation
0	Use system default value
1	Week 1 is the week with January, 1st (default)
2	Week 1 is the first week containing four or more days of that year
3	Week 1 is the first week containing only days of the new year

Example

```
Sub example_datepart
      MsgBox DatePart("ww", "12/31/2005")
End Sub
```

DateSerial()

Returns a **Date** value for a specified year, month, or day.

Syntax

DateSerial (year, month, day)

Return value

Date

Parameters

Year
Integer expression that indicates a year. All values between 0 and 99 are interpreted as the years 1900-1999. For years that fall outside this range, you must enter all four digits.

Month
Integer expression that indicates the month of the specified year. The accepted range is from 1-12.

Day
Integer expression that indicates the day of the specified month. The accepted range is from 1-31, depending on the month.

The **DateSerial function** returns the number of days between December 30,1899 and the given date. You can use this function to calculate the difference between two dates.

The **DateSerial function** returns the data type Variant with VarType 7 (Date). Internally, this value is stored as a Double value, so that when the given date is 1.1.1900, the returned value is 2. Negative values correspond to dates before December 30, 1899 (not inclusive).

If a date is defined that lies outside of the accepted range, OpenOffice.org Basic returns an error message.

Whereas you define the **DateValue function** as a string that contains the date, the **DateSerial function** evaluates each of the parameters (year, month, day) as separate numeric expressions.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleDateSerial
      Dim lDate as Long
      Dim sDate as String
      lDate = DateSerial(1964, 4, 9)
      sDate = DateSerial(1964, 4, 9)
      MsgBox lDate REM returns 23476
      MsgBox sDate REM returns 09.04.1964 00:00:00
End Sub
```

DateValue()

Returns a number from a date string. The date string is a complete date in a single numeric value. You can also use this serial number to determine the difference between two dates.

Syntax

DateValue [(date)]

Return value

Long

Parameters

Date

String expression that contains the date that you want to calculate. The date can be specified in almost any format.

You can use this function to convert a date that occurs between December 1, 1582 and December 31, 9999 into a single integer value. You can then use this value to calculate the difference between two dates. If the date argument lies outside the acceptable range, OpenOffice.org Basic returns an error message.

In contrast to the DateSerial function that passes years, months, and days as separate numeric values, the DateValue function passes the date using the format "month.[,]day.[,]year".

You can set the locale used for controlling the formatting numbers, dates and currencies in OpenOffice.org Basic in **Tools** - **Options** - **Language Settings** - **Languages**. In Basic format codes, the decimal point (.) is always used as **placeholder** for the decimal separator defined in your locale and will be replaced by the corresponding character.

The same applies to the locale settings for date, time and currency formats. The Basic format code will be interpreted and displayed according to your locale setting.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleDateValue
      MsgBox DateValue("12/02/1997")
End Sub
```

Day()

Returns a value that represents the day of the month based on a serial date number generated by **DateSerial** or **DateValue**.

Syntax

Day (Number)

Return value

Integer

Parameters

Number
A numeric expression that contains a serial date number from which you can determine the day of the month.

This function is basically the opposite of the DateSerial function, returning the day of the month from a serial date number generated by the **DateSerial** or the **DateValue** function. For example, the expression

Print Day (DateSerial(1994, 12, 20))

returns the value 20.

Error Codes

5 Invalid procedure call

Example

```
sub ExampleDay
      Print "Day " & Day(DateSerial(1994, 12, 20)) & " of the month"
End Sub
```

Declare

Declares and defines a subroutine in a DLL file that you want to execute from OpenOffice.org Basic.

See also: FreeLibrary

Syntax

Declare {Sub | Function} Name Lib "Libname" [Alias "Aliasname"] [Parameter] [As Type]

Parameters

Name
A different name than defined in the DLL, to call the subroutine from OpenOffice.org Basic.

Aliasname
Name of the subroutine as defined in the DLL.

Libname
File or system name of the DLL. This library is automatically loaded the first time the function is used.

Argumentlist
List of parameters representing arguments that are passed to the procedure when it is called. The type and number of parameters is dependent on the executed procedure.

Type
Defines the data type of the value that is returned by a function procedure. You can exclude this parameter if a type-declaration character is entered after the name.

To pass a parameter to a subroutine as a value instead of as a reference, the parameter must be indicated by the keyword **ByVal**.

Example

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )

Sub ExampleDeclare
      Dim lValue As Long
      lValue = 5000
      MyMessageBeep( lValue )
      FreeLibrary("user32.dll" )
End Sub
```

Defxxx

If no type-declaration character or keyword is specified, the Defxxx statement sets the default data type for variables, according to a letter range.

Syntax

Defxxx Characterrange1[, Characterrange2[,...]]

Parameters

Characterrange: Letters that specify the range of variables that you want to set the default data type for.

xxx: Keyword that defines the default variable type:

Keyword: Default variable type, which are:

DefBool: Boolean

DefCur: Currency

DefDate: Date

DefDbl: Double

DefInt: Integer

DefLng: Long

DefObj: Object

DefSng: Single

DefStr: String

DefVar: Variant

Example

```
REM Prefix definition for variable types:

DefBool b
DefDate t
DefDbL d
DefInt i
DefLng l
DefObj o
DefVar v

Sub ExampleDefBool
      bOK=TRUE    REM bOK is an implicit Boolean variable
      tStart=NOW  REM tStart is an implicit date variable
end sub
```

Dim

Declares a variable or an array.

If the variables are separated by commas (for example, DIM sPar1, sPar2, sPar3 AS STRING), only Variant variables can be defined. Use a separate definition line for each variable.

```
DIM sPar1 AS STRING
DIM sPar2 AS STRING
DIM sPar3 AS STRING
```

Dim declares local variables within subroutines. Global variables are declared with the PUBLIC or the PRIVATE statement.

Syntax:

[ReDim]Dim VarName [(start To end)] [As VarType][, VarName2 [(start To end)] [As VarType][,...]]

Parameters:

VarName:
Any variable or array name.

Start, End:
Numerical values or constants that define the number of elements (NumberElements=(end-start)+1) and the index range.
Start and End can be numerical expressions if ReDim is applied at the procedure level.

VarType:
Key word that declares the data type of a variable. Keywords are: Bool, Currency, Date, Double, Integer, Long, Object, Single, String, Variant.

In OpenOffice.org Basic, you do not need to declare variables explicitly. However, you need to declare an array before you can use them. You can declare a variable with the Dim statement, using commas to separate multiple declarations. To declare a variable type, enter a type-declaration character following the name or use a corresponding key word.

OpenOffice.org Basic supports single or multi-dimensional arrays that are defined by a specified variable type. Arrays are suitable if the program contains lists or tables that you want to edit. The advantage of arrays is that it is possible to address individual elements according to indexes, which can be formulated as numeric expressions or variables.

Arrays are declared with the Dim statement. There are two methods to define the index range:

DIM text(20) as String REM 21 elements numbered from 0 to 20

DIM text(5 to 25) as String REM 21 elements numbered from 5 to 25

DIM text(-15 to 5) as String REM 21 elements (including 0)

REM numbered from -15 to 5

Two-dimensional data field

DIM text(20,2) as String REM 63 elements; form 0 to 20 level 1, from 0 to 20 level 2 and from 0 to 20 level 3.

You can declare an array types as dynamic if a ReDim statement defines the number of dimensions in the subroutine or the function that contains the array. Generally, you can only define an array dimension once, and you cannot modify it. Within a subroutine, you can declare an array with ReDim. You can only define dimensions with numeric expressions. This ensures that the fields are only as large as necessary.

Example:

```
Sub ExampleDim1
      Dim sVar As String
      Dim iVar As Integer
      sVar = "Office"
End Sub


Sub ExampleDim2
      REM Two-dimensional data field
      Dim stext(20,2) as String
      Const sDim as String = " Dimension:"
      For i = 0 to 20
            For ii = 0 to 2
                  stext(i,ii) = Str(i) & sDim & Str(ii)
            Next ii
      Next i
      For i = 0 to 20
            For ii = 0 to 2
                  MsgBox stext(i,ii)
            Next ii
      Next i
End Sub
```

DimArray()

Returns a Variant array.

Syntax

DimArray (Argument list)

See also Array

If no parameters are passed, an empty array is created (like Dim A() that is the same as a sequence of length 0 in Uno). If parameters are specified, a dimension is created for each parameter.

Parameters

Argument list
A list of any number of arguments that are separated by commas.

Error Codes

9 Subscript out of range

Example

```
DimArray( 2, 2, 4 ) is the same as DIM a( 2, 2, 4 )
```

Dir()

Returns the name of a file, a directory, or all of the files and the directories on a drive or in a directory that match the specified search path.

Syntax:

```
Dir [(Text As String) [, Attrib As Integer]]
```

Return value:

String

Parameters:

Text
Any string expression that specifies the search path, directory or file. This argument can only be specified the first time that you call the Dir function. If you want, you can enter the path in URL notation.

Attrib
Any integer expression that specifies bitwise file attributes. The Dir function only returns files or directories that match the specified attributes. You can combine several attributes by adding the attribute values:
0 : Normal files.
16 : Returns the name of the directory only.

Use this attribute to check if a file or directory exists, or to determine all files and folders in a specific directory.

To check if a file exists, enter the complete path and name of the file. If the file or directory name does not exist, the Dir function returns a zero-length string ("").

To generate a list of all existing files in a specific directory, proceed as follows: The first time you call the Dir function, specify the complete search path for the files, for example, "D:\Files*.sxw". If the path is correct and the search finds at least one file, the Dir function returns the name of the first file that matches the search path. To return additional file names that match the path, call Dir again, but with no arguments.

To return directories only, use the attribute parameter. The same applies if you want to determine the name of a volume (for example, a hard drive partition)

Error Codes

5 Invalid procedure call

53 File not found

Example

```
Sub ExampleDir
   REM Displays all files and directories
   Dim sPath As String
   Dim sDir as String, sValue as String
   sDir="Directories:"
   sPath = CurDir
   sValue = Dir$(sPath + getPathSeparator + "*",16)
   Do
      If sValue <> "." and sValue <> ".." Then
         If (GetAttr( sPath + getPathSeparator + sValue) AND 16) >0 Then
            REM get the directories
            sDir = sDir & Chr(13) & sValue
```

```
      End If
    End If
    sValue = Dir$
  Loop Until sValue = ""
  MsgBox sDir,0,sPath
End sub
```

Do

Repeats the statements between the Do and the Loop statement while the condition is True or until the condition becomes True.

Syntax

Do [{While | Until} condition = True]

statement block

[Exit Do]

statement block

Loop

or

Do

statement block

[Exit Do]

statement block

Loop [{While | Until} condition = True]

Parameters/Elements

Condition
A comparison, numeric or string expression, that evaluates either True or False.

Statement block
Statements that you want to repeat while or until the condition is True.

The **Do...Loop** statement executes a loop as long as, or until, a certain condition is True. The condition for exiting the loop must be entered following either the **Do** or the **Loop** statement. The following examples are valid combinations:

Syntax

Do While condition = True

...statement block

Loop

The statement block between the Do While and the Loop statements is repeated so long as the condition is true.

Do Until condition = True

...statement block

Loop

The statement block between the Do Until and the Loop statements is repeated if the condition so long as the condition is false.

Do

...statement block

Loop While condition = True

The statement block between the Do and the Loop statements repeats so long as the condition is true.

Do

...statement block

Loop Until condition = True

The statement block between the Do and the Loop statements repeats until the condition is true.

Use the **Exit Do** statement to unconditionally end the loop. You can add this statement anywhere in a **Do**...**Loop** statement. You can also define an exit condition using the **If...Then** structure as follows:

Do...

statements

If condition = True Then Exit Do

statements

Loop...

Example

```
Sub ExampleDoLoop
      Dim sFile As String
      Dim sPath As String
      sPath = "c:\"
      sFile = Dir$( sPath ,22)
      If sFile <> "" Then
            Do
                  MsgBox sFile
                  sFile = Dir$
            Loop Until sFile = ""
      End If
End Sub
```

End

Ends a procedure or block.

Syntax

End, End Function, End If, End Select, End Sub

Parameters

Use the End statement as follows:

Statement

End: Is not required, but can be entered anywhere within a procedure to end the program execution.

End Function: Ends a **Function** statement.

End If: Marks the end of a **If…Then…Else** block.

End Select: Marks the end of a **Select Case** block.

End Sub: Ends a **Sub** statement.

Example

```
Sub ExampleRandomSelect
      Dim iVar As Integer
      iVar = Int((15 * Rnd) -2)
      Select Case iVar
            Case 1 To 5
                  Print "Number from 1 to 5"
            Case 6, 7, 8
                  Print "Number from 6 to 8"
            Case Is > 8 And iVar < 11
                  Print "Greater than 8"
            Case Else
                  Print "Outside range 1 to 10"
      End Select
End Sub
```

Environ()

Returns the value of an environment variable as a string. Environment variables are dependent on the type of operating system that you have.

Syntax

```
Environ (Environment As String)
```

Return value

String

Parameters

Environment
Environment variable that you want to return the value for.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleEnviron
      Dim sTemp As String
      sTemp=Environ ("TEMP")
      If sTemp = "" Then sTemp=Environ("TMP")
      MsgBox "'" & sTemp & "'" ,64,"Directory of temporary files:"
End Sub
```

Eof()

Determines if the file pointer has reached the end of a file.

Syntax

```
Eof (intexpression As Integer)
```

Return value

Bool

Parameters

Intexpression
Any integer expression that evaluates to the number of an open file.

Use EOF to avoid errors when you attempt to get input past the end of a file. When you use the Input or Get statement to read from a file, the file pointer is advanced by the number of bytes read. When the end of a file is reached, EOF returns the value "True" (-1).

Error Codes

5 Invalid procedure call

52 Bad file name or number

Example:

```
Sub ExampleWorkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "First line of text"
      Print #iNumber, "Another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      While not eof(iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" then
                  sMsg = sMsg & sLine & Chr(13)
            End If
      Wend
      Close #iNumber
      Msgbox sMsg
End Sub
```

EqualUnoObjects()

Returns True if the two specified Basic Uno objects represent the same Uno object instance.

Syntax

EqualUnoObjects(oObj1, oObj2)

Return value

Bool

Example

```
// Copy of objects -> same instance
oIntrospection = CreateUnoService( "com.sun.star.beans.Introspection" )
oIntro2 = oIntrospection
Print EqualUnoObjects( oIntrospection, oIntro2 )
// Copy of structs as value -> new instance
Dim Struct1 as new com.sun.star.beans.Property
Struct2 = Struct1
Print EqualUnoObjects( Struct1, Struct2 )
```

Eqv

Calculates the logical equivalence of two expressions.

Syntax:

Result = Expression1 Eqv Expression2

Parameters:

Result
Any numeric variable that contains the result of the comparison.

Expression1, Expression2
Any expressions that you want to compare.

When testing for equivalence between Boolean expressions, the result is **True** if both expressions are either **True** or **False**.

In a bit-wise comparison, the Eqv operator only sets the corresponding bit in the result if a bit is set in both expressions, or in neither expression.

Example:

```
Sub ExampleEqv
      Dim A as Variant, B as Variant, C as Variant, D as Variant
      Dim vOut as Variant
      A = 10: B = 8: C = 6: D = Null
      vOut = A > B Eqv B > C REM returns -1
      vOut = B > A Eqv B > C REM returns 0
      vOut = A > B Eqv B > D REM returns 0
      vOut = (B > D Eqv B > A) REM returns -1
      vOut = B Eqv A REM returns -3
End Sub
```

Erase

Erases the contents of array elements of fixed size arrays, and releases the memory used by arrays of variable size.

Syntax

Erase Arraylist

Parameters

Arraylist
The list of arrays to be erased.

Erl

Returns the line number where an error occurred during program execution.

Syntax

Erl

Return value

Integer

Parameters

The Erl function only returns a line number, and not a line label.

Example

```
sub ExampleError
      On Error Goto ErrorHandler REM Set up error handler
      Dim iVar as Integer
      Dim sVar As String
      REM Error caused by non-existent file
      iVar = Freefile
      Open "\file9879.txt" for Input as #iVar
      Line Input #iVar, sVar
      Close #iVar
      Exit Sub
      ErrorHandler:
      MsgBox "Error " & err & ": " & error$ + chr(13) + "In line : " + Erl + _
            chr(13) + Now , 16 ,"An error occurred"
End Sub
```

Err

Returns an error code that identifies the error that occurred during program execution.

Syntax:

Err

Return value

Integer

Parameters

The Err function is used in error-handling routines to determine the error and the corrective action.

Example

```
sub ExampleError
      On Error Goto ErrorHandler REM Set up error handler
      Dim iVar as Integer
      Dim sVar As String
      REM Error occurs due to non-existent file
      iVar = Freefile
      Open "\file9879.txt" for Input as #iVar
      Line Input #iVar, sVar
      Close #iVar
      Exit Sub
      ErrorHandler:
      MsgBox "Error " & Err & ": " & Error$ + chr(13) + "At line : " + Erl + _
            chr(13) + Now , 16 ,"an error occurred"
End Sub
```

Error()

Returns the error message that corresponds to a given error code.

Syntax

Error (Expression)

Return value

String

Parameters

Expression
Any numeric expression that contains the error code of the error message that you want to return.

If no parameters are passed, the Error function returns the error message of the most recent error that occurred during program execution.

Error Codes

51 Internal error

13 Type mismatch

Example

```
Sub ExampleError
      On Error Goto ErrorHandler REM Set up error handler
      Dim iVar as Integer
      Dim sVar As String
      REM An error occurs due to non-existent file
      iVar = Freefile
      Open "\file9879.txt" for Input as #iVar
      Line Input #iVar, sVar
      Close #iVar
      Exit Sub
      ErrorHandler:
      MsgBox "Error " & err & ": " & error$ + chr(13) + "In line : " + Erl + _
            chr(13) + Now , 16 ,"error occurred"
End Sub
```

Exit

Exits a **Do...Loop**, **For...Next**, a function, or a subroutine.

Syntax

see Parameters

Parameters

Exit Do
Only valid within a **Do...Loop** statement to exit the loop. Program execution continues with the statement that follows the Loop statement. If **Do...Loop** statements are nested, the control is transferred to the loop in the next higher level.

Exit For
Only valid within a **For...Next** loop to exit the loop. Program execution continues with the first statement that follows the **Next** statement. In nested statements, the control is transferred to the loop in the next higher level.

Exit Function
Exits the **Function** procedure immediately. Program execution continues with the statement that follows the **Function** call.

Exit Sub
Exits the subroutine immediately. Program execution continues with the statement that follows the **Sub** call.

The Exit statement does not define the end of a structure, and must not be confused with the End statement.

Example

```
Sub ExampleExit
      Dim sReturn As String
      Dim sListArray(10) as String
      Dim siStep as Single
      For siStep = 0 to 10 REM Fill array with test data
            sListArray(siStep) = chr(siStep + 65)
            msgbox sListArray(siStep)
      Next siStep
      sReturn = LinSearch(sListArray(), "B")
      Print sReturn
End Sub

Function LinSearch( sList(), sItem As String ) as integer
      Dim iCount as Integer
      REM LinSearch searches a TextArray:sList() for a TextEntry:
      REM Returns the index of the entry or 0 ( Null)
      For iCount=1 to Ubound( sList() )
            if sList( iCount ) = sItem Then
                  Exit For REM sItem found
            end if
      Next iCount
      If iCount = Ubound( sList() ) Then iCount = 0
      LinSearch = iCount
End Function
```

Exp()

Returns the base of the natural logarithm (e = 2.718282) raised to a power.

Syntax

Exp (Number)

Return value

Double

Parameters

Number

Any numeric expression that specifies the power that you want to raise "e" to (the base of natural logarithms). The power must be for both single-precision numbers less than or equal to 88.02969 and double-precision numbers less than or equal to 709.782712893, since OpenOffice.org Basic returns an Overflow error for numbers exceeding these values.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleLogExp
      Dim dValue as Double
      const b1=12.345e12
      const b2=1.345e34
      dValue=Exp( Log(b1)+Log(b2) )
      MsgBox "" & dValue & chr(13) & (b1*b2) ,0,"Multiplication by logarithm"
end sub
```

FileAttr()

Returns the access mode or the file access number of a file that was opened with the Open statement. The file access number is dependent on the operating system (OSH = Operating System Handle).

If you use a 32-Bit operating system, you cannot use the FileAttr-Function to determine the file access number.

See also: Open

Syntax

```
FileAttr (FileNumber As Integer, Attribute As Integer)
```

Return value

Integer

Parameters

FileNumber
The number of the file that was opened with the Open statement.

Attribute
Integer expression that indicates the type of file information that you want to return. The following values are possible:

1: The FileAttr-Function indicates the access mode of the file.
2: The FileAttr-Function returns the file access number of the operating system.

If you specify a parameter attribute with a value of 1, the following return values apply:

1 - INPUT (file open for input)

2 - OUTPUT (file open for output)

4 - RANDOM (file open for random access)

8 - APPEND (file open for appending)

32 - BINARY (file open in binary mode).

Error Codes

5 Invalid procedure call

52 Bad file name or number

Example

```
Sub ExampleFileAttr
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "This is a line of text"
      MsgBox FileAttr(#iNumber, 1 ),0,"Access mode"
      MsgBox FileAttr(#iNumber, 2 ),0,"File attribute"
```

```
    Close #iNumber
End Sub
```

FileCopy

Copies a file.

Syntax

```
FileCopy TextFrom As String, TextTo As String
```

Parameters

TextFrom
Any string expression that specifies the name of the file that you want to copy. The expression can contain optional path and drive information. If you want, you can enter a path in URL notation.

TextTo
Any string expression that specifies where you want to copy the source file to. The expression can contain the destination drive, the path, and file name, or the path in URL notation.

You can only use the FileCopy statement to copy files that are not opened.

Error Codes

5 Invalid procedure call

76 Path not found

Example

```
Sub ExampleFilecopy
      Filecopy "c:\autoexec.bat", "c:\Temp\Autoexec.sav"
End Sub
```

FileDateTime()

Returns a string that contains the date and the time that a file was created or last modified.

Syntax

```
FileDateTime (Text As String)
```

Parameters

Text:

Any string expression that contains an unambiguous (no wildcards) file specification. You can also use URL notation.

This function determines the exact time of creation or last modification of a file, returned in the format "MM.DD.YYYY HH.MM.SS".

You can set the locale used for controlling the formatting numbers, dates and currencies in OpenOffice.org Basic in **Tools** - **Options** - **Language Settings** - **Languages**. In Basic format codes, the decimal point (**.**) is always used as **placeholder** for the decimal separator defined in your locale and will be replaced by the corresponding character.

The same applies to the locale settings for date, time and currency formats. The Basic format code will be interpreted and displayed according to your locale setting.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleFileDateTime
      MsgBox FileDateTime("C:\autoexec.bat")
End Sub
```

FileExists()

Determines if a file or a directory is available on the data medium.

Syntax

```
FileExists(FileName As String | DirectoryName As String)
```

Return value

Bool

Parameters

FileName | DirectoryName
Any string expression that contains an unambiguous file specification. You can also use URL notation.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleFileExists
      MsgBox FileExists("C:\autoexec.bat")
      MsgBox FileExists("file:///d|/bookmark.htm")
      MsgBox FileExists("file:///d|/private")
End Sub
```

FileLen()

Returns the length of a file in bytes.

Syntax

```
FileLen (Text As String)
```

Return value

Long

Parameters

Text
Any string expression that contains an unambiguous file specification. You can also use URL notation.

This function determines the length of a file. If the FileLen function is called for an open file, it returns the file length before it was opened. To determine the current file length of an open file, use the Lof function.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleFileLen
      MsgBox FileLen("C:\autoexec.bat")
End Sub
```

FindObject()

Enables an object to be addressed at run-time as a string parameter through the object name.

Syntax

FindObject(ObjName As String)

Parameters

ObjName
String that specifies the name of the object that you want to address at run-time.

Error Codes

5 Invalid procedure call

12 Variable undefined

Example

For example, the following command:

```
MyObj.Prop1.Command = 5
```

corresponds to the command block:

```
Dim ObjVar as Object
Dim ObjProp as Object
ObjName As String = "MyObj"
ObjVar = FindObject( ObjName As String )
PropName As String = "Prop1"
ObjProp = FindPropertyObject( ObjVar, PropName As String )
ObjProp.Command = 5
```

This allows names to be dynamically created at run-time. For example:

"TextEdit1" to TextEdit5" in a loop to create five control names.

See also: FindPropertyObject

FindPropertyObject()

Enables objects to be addressed at run-time as a string parameter using the object name.

Syntax

FindPropertyObject(ObjVar, PropName As String)

Parameters

ObjVar
Object variable that you want to dynamically define at run-time.

PropName
String that specifies the name of the property that you want to address at run-time.

Error Codes

5 Invalid procedure call

12 Variable undefined

14 Invalid parameter

52 Bad file name or number

57 Device I/O error

Example

For instance, the command:

```
MyObj.Prop1.Command = 5
```

corresponds to the following command block:

```
Dim ObjVar as Object
Dim ObjProp as Object
ObjName As String = "MyObj"
ObjVar = FindObject( ObjName As String )
PropName As String = "Prop1"
ObjProp = FindPropertyObject( ObjVar, PropName As String )
ObjProp.Command = 5
```

To dynamically create Names at run-time, use:

"TextEdit1" to TextEdit5" in a loop to create five names.

See also: FindObject

Fix()

Returns the integer value of a numeric expression by removing the fractional part of the number.

Syntax

Fix (Expression)

Return value

Double

Parameters

Expression
Numeric expression that you want to return the integer value for.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleFix
      Print Fix(3.14159) REM returns 3.
      Print Fix(0) REM returns 0.
      Print Fix(-3.14159) REM returns -3.
End Sub
```

For

Repeats the statements between the For...Next block a specified number of times.

Syntax:

For counter=start To end [Step step]

statement block

[Exit For]

statement block

Next [counter]

Variables:

Counter
Loop counter initially assigned the value to the right of the equal sign (start). Only numeric variables are valid. The loop counter increases or decreases according to the variable Step until End is passed.

Start
Numeric variable that defines the initial value at the beginning of the loop.

End
Numeric variable that defines the final value at the end of the loop.

Step
Sets the value by which to increase or decrease the loop counter. If Step is not specified, the loop counter is incremented by 1. In this case, End must be greater than Start. If you want to decrease Counter, End must be less than Start, and Step must be assigned a negative value.

The **For...Next** loop repeats all of the statements in the loop for the number of times that is specified by the parameters.

As the counter variable is decreased, OpenOffice.org Basic checks if the end value has been reached. As soon as the counter passes the end value, the loop automatically ends.

It is possible to nest **For...Next** statements. If you do not specify a variable following the **Next** statement, **Next** automatically refers to the most recent **For** statement.

If you specify an increment of 0, the statements between **For** and **Next** are repeated continuously.

When counting down the counter variable, OpenOffice.org Basic checks for overflow or underflow. The loop ends when Counter exceeds End (positive Step value) or is less than End (negative Step value).

Use the **Exit For** statement to exit the loop unconditionally. This statement must be within a **For...Next** loop. Use the **If...Then** statement to test the exit condition as follows:

For...

statements

If condition = True Then Exit For

statements

Next

Note: In nested **For...Next** loops, if you exit a loop unconditionally with **Exit For**, only one loop is exited.

Example

The following example uses two nested loops to sort a string array with 10 elements (sEntry()), that are first filled with various contents:

```
Sub ExampleSort
      Dim sEntry(9) As String
      Dim iCount As Integer
      Dim iCount2 As Integer
      Dim sTemp As String
      sEntry(0) = "Jerry"
      sEntry(1) = "Patty"
      sEntry(2) = "Kurt"
      sEntry(3) = "Thomas"
      sEntry(4) = "Michael"
      sEntry(5) = "David"
      sEntry(6) = "Cathy"
      sEntry(7) = "Susie"
      sEntry(8) = "Edward"
      sEntry(9) = "Christine"
      For iCount = 0 To 9
            For iCount2 = iCount + 1 To 9
                  If sEntry(iCount) > sEntry(iCount2) Then
                        sTemp = sEntry(iCount)
                        sEntry(iCount) = sEntry(iCount2)
                        sEntry(iCount2) = sTemp
                  End If
            Next iCount2
      Next iCount
      For iCount = 0 To 9
            Print sEntry(iCount)
      Next iCount
End Sub
```

Format()

Converts a number to a string, and then formats it according to the format that you specify.

Syntax

Format (Number [, Format As String])

Return value

String

Parameters

Number
Numeric expression that you want to convert to a formatted string.

Format
String that specifies the format code for the number. If **Format** is omitted, the Format function works like the **Str** function.

Formatting Codes

The following list describes the codes that you can use for formatting a number:

0
If **Number** has a digit at the position of the 0 in the format code, the digit is displayed, otherwise a zero is displayed.

If **Number** has fewer digits than the number of zeros in the format code, (on either side of the decimal), leading or trailing zeros are displayed. If the number has more digits to the left of the decimal separator than the amount of zeros in the format code, the additional digits are displayed without formatting.

Decimal places in the number are rounded according to the number of zeros that appear after the decimal separator in the **Format** code.

#
If **Number** contains a digit at the position of the # placeholder in the **Format** code, the digit is displayed, otherwise nothing is displayed at this position.

This symbol works like the 0, except that leading or trailing zeroes are not displayed if there are more # characters in the format code than digits in the number. Only the relevant digits of the number are displayed.

.
The decimal placeholder determines the number of decimal places to the left and right of the decimal separator.

If the format code contains only # placeholders to the left of this symbol, numbers less than 1 begin with a decimal separator. To always display a leading zero with fractional numbers, use 0 as a placeholder for the first digit to the left of the decimal separator.

%
Multiplies the number by 100 and inserts the percent sign (%) where the number appears in the format code.

E- E+ e- e+
If the format code contains at least one digit placeholder (0 or #) to the right of the symbol E-, E+, e-, or e+, the number is formatted in the scientific or exponential format. The letter E or e is inserted between the number and the exponent. The number of placeholders for digits to the right of the symbol determines the number of digits in the exponent.

If the exponent is negative, a minus sign is displayed directly before an exponent with E-, E+, e-, e+. If the exponent is positive, a plus sign is only displayed before exponents with E+ or e+.

thousands
The thousands delimiter is displayed if the format code contains the delimiter enclosed by digit placeholders (0 or #).

The use of a period as a thousands and decimal separator is dependent on the regional setting. When you enter a number directly in Basic source code, always use a period as decimal delimiter. The actual character displayed as a decimal separator depends on the number format in your system settings.

+ - $ ( ) space
A plus (+), minus (-), dollar ($), space, or brackets entered directly in the format code is displayed as a literal character.

To display characters other than the ones listed here, you must precede it by a backslash (\), or enclose it in quotation marks (" ").

\
The backslash displays the next character in the format code.

Characters in the format code that have a special meaning can only be displayed as literal characters if they are preceded by a backslash. The backslash itself is not displayed, unless you enter a double backslash (\\) in the format code.

Characters that must be preceded by a backslash in the format code in order to be displayed as literal characters are date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, /, :), numeric-formatting characters (#, 0, %, E, e, comma, period), and string-formatting characters (@, &, <, >, !).

You can also use the following predefined number formats. Except for "General Number", all of the predefined format codes return the number as a decimal number with two decimal places.

If you use predefined formats, the name of the format must be enclosed in quotation marks.

Predefined format

General Number
Numbers are displayed as entered.

Currency
Inserts a dollar sign in front of the number and encloses negative numbers in brackets.

Fixed
Displays at least one digit in front of the decimal separator.

Standard
Displays numbers with a thousands separator.

Percent
Multiplies the number by 100 and appends a percent sign to the number.

Scientific
Displays numbers in scientific format (for example, 1.00E+03 for 1000).

A format code can be divided into three sections that are separated by semicolons. The first part defines the format for positive values, the second part for negative values, and the third part for zero. If you only specify one format code, it applies to all numbers.

You can set the locale used for controlling the formatting numbers, dates and currencies in OpenOffice.org Basic in **Tools** - **Options** - **Language Settings** - **Languages**. In Basic format codes, the decimal point (**.**) is always used as **placeholder** for the decimal separator defined in your locale and will be replaced by the corresponding character.

The same applies to the locale settings for date, time and currency formats. The Basic format code will be interpreted and displayed according to your locale setting.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleFormat
      MsgBox Format(6328.2, "##,##0.00")
      REM always use a period as decimal delimiter when you enter numbers in
      REM Basic source code.
      REM displays for example 6,328.20 in English locale, 6.328,20 in
      REM German locale.
End Sub
```

FreeFile

Returns the next available file number for opening a file. Use this function to open a file using a file number that is not already in use by a currently open file.

Syntax:

`FreeFile`

Return value

Integer

Parameters

This function can only be used immediately in front of an Open statement. FreeFile returns the next available file number, but does not reserve it.

Error Codes

5 Invalid procedure call

67 Too many files

Example

```
Sub ExampleWorkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      sMsg = ""
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "First line of text"
      Print #iNumber, "Another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As #iNumber
      While not eof(#iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" then
                  sMsg = sMsg & sLine & chr(13)
            End If
      Wend
      Close #iNumber
      Msgbox sMsg
End Sub
```

FreeLibrary()

Releases DLLs that were loaded by a Declare statement. A released DLL is automatically reloaded if one of its functions is called. See also: Declare

Syntax

FreeLibrary (LibName As String)

Parameters

LibName
String expression that specifies the name of the DLL.

FreeLibrary can only release DLLs that are loaded during Basic runtime.

Error Codes

5 Invalid procedure call

Example:

```
Declare Sub MyMessageBeep Lib "user32.dll" Alias "MessageBeep" ( long )
Sub ExampleDeclare
      Dim lValue As Long
      lValue = 5000
      MyMessageBeep( lValue )
      FreeLibrary("user32.dll" )
End Sub
```

Function()

Defines a subroutine that can be used as an expression to determine a return type.

Syntax

Function Name[(VarName1 [As Type][, VarName2 [As Type][,...]]]) [As Type]

statement block

[Exit Function]

statement block

End Function

Parameters

Name
Name of the subroutine to contain the value returned by the function.

VarName
Parameter to be passed to the subroutine.

Type
Type-declaration keyword.

Example

```
Sub ExampleExit
      Dim sReturn As String
      Dim sListArray(10) as String
      Dim siStep as Single
      For siStep = 0 to 10 REM Fill array with test data
            sListArray(siStep) = chr$(siStep + 65)
            MsgBox sListArray(siStep)
      Next siStep
      sReturn = LinSearch(sListArray(), "B")
      Print sReturn
End Sub

Function LinSearch( sList(), sItem As String ) as integer
      Dim iCount as Integer
      REM Linsearch searches a TextArray:sList() for a TextEntry:
      REM Return value is the index of the entry or 0 (Null)
      For iCount=1 to Ubound( sList() )
            If sList( iCount ) = sItem then
                  Exit For REM sItem found
            End If
      Next iCount
      if iCount = Ubound( sList() ) then iCount = 0
      LinSearch = iCount
End Function
```

Get

Reads a record from a relative file, or a sequence of bytes from a binary file, into a variable.

See also: PUT Statement

Syntax

```
Get [#] FileNumber As Integer, [Position], Variable
```

Parameters

FileNumber
Any integer expression that determines the file number.

Position
For files opened in Random mode, **Position** is the number of the record that you want to read.
For files opened in Binary mode, **Position** is the byte position in the file where the reading starts.
If **Position** is omitted, the current position or the current data record of the file is used.

Variable
Name of the variable to be read. With the exception of object variables, you can use any variable type.

Example:

```
Sub ExampleRandomAccess
      Dim iNumber As Integer
      Dim sText As Variant REM Must be a variant
      Dim aFile As String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Seek #iNumber,1 REM Position at beginning
      Put #iNumber,, "This is the first line of text" REM Fill line with text
      Put #iNumber,, "This is the second line of text"
      Put #iNumber,, "This is the third line of text"
      Seek #iNumber,2
      Get #iNumber,,sText
      Print sText
      Close #iNumber
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Get #iNumber,2,sText
      Put #iNumber,,"This is a new text"
      Get #iNumber,1,sText
      Get #iNumber,2,sText
      Put #iNumber,20,"This is the text in record 20"
      Print Lof(#iNumber)
      Close #iNumber
End Sub
```

GetAttr()

Returns a bit pattern that identifies the file type or the name of a volume or a directory.

Syntax:

```
GetAttr (Text As String)
```

Return value

Integer, bit position values:

0 : Normal files.

1 : Read-only files.

8 : Returns the name of the volume

16 : Returns the name of the directory only.

32 : File was changed since last backup (Archive bit).

Parameters

Text

Any string expression that contains an unambiguous file specification. You can also use URL notation.

This function determines the attributes for a specified file and returns the bit pattern that can help you to identify the following file attributes:

Error Codes

5 Invalid procedure call

53 File not found

Example:

```
Sub ExampleSetGetAttr
      On Error Goto ErrorHandler REM Define target for error-handler
      If Dir("C:\test",16)="" Then MkDir "C:\test"
      If Dir("C:\test\autoexec.sav")="" THEN Filecopy "c:\autoexec.bat",
      "c:\test\autoexec.sav"
      SetAttr "c:\test\autoexec.sav" ,0
      Filecopy "c:\autoexec.bat", "c:\test\autoexec.sav"
      SetAttr "c:\test\autoexec.sav" ,1
      Print GetAttr( "c:\test\autoexec.sav" )
      End


      ErrorHandler:
      Print Error
      End
End Sub
```

GetDefaultContext

Returns the default context of the process service factory, if existent, else returns a null reference.

This runtime function returns the default component context to be used, if instantiating services via XmultiServiceFactory. See the `Professional UNO` chapter in the `Developer's Guide` on api.openoffice.org for more information.

GetGuiType

Returns a numerical value that specifies the graphical user interface.

This runtime function is only provided for downward compatibility to previous versions. The return value is not defined in client-server environments.

Syntax

GetGUIType

Return value

Integer

1: Windows

3: Mac OS

4: UNIX

Example

```
Sub ExampleEnvironment
      MsgBox GetGUIType
End Sub
```

GetProcessServiceManager()

Returns the ProcessServiceManager (central Uno ServiceManager).

This function is required when you want to instantiate a service using CreateInstanceWithArguments.

Syntax

oServiceManager = GetProcessServiceManager()

Example

```
oServiceManager = GetProcessServiceManager()
oIntrospection =
oServiceManager.createInstance("com.sun.star.beans.Introspection");
```

this is the same as the following statement:

```
oIntrospection = CreateUnoService("com.sun.star.beans.Introspection")
```

GetSolarVersion

Returns the internal number of the current OpenOffice.org version.

Syntax

s = GetSolarVersion

Return value

String

Example

```
Sub ExampleGetSolarVersion
      Dim sSep As String
      sSep = GetSolarVersion
      MsgBox sSep,64,"Version number of the solar technology"
End Sub
```

GetSystemTicks()

Returns the number of system ticks provided by the operating system. You can use this function to optimize certain processes.

Syntax

GetSystemTicks()

Return value

Long

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleWait
      Dim lTick As Long
      lTick = GetSystemTicks()
      Wait 2000
      lTick = (GetSystemTicks() - lTick)
      MsgBox "" & lTick & " Ticks" ,0,"The pause lasted"
End Sub
```

Global

Dimensions a variable or an array at the global level (that is, not within a subroutine or function), so that the variable and the array are valid in all libraries and modules for the current session.

Syntax

Global VarName[(start To end)] [As VarType][, VarName2[(start To end)] [As VarType][,...]]

Example

```
Global iGlobalVar As Integer
Sub ExampleGlobal
      iGlobalVar = iGlobalVar + 1
      MsgBox iGlobalVar
End sub
```

GlobalScope

Basic source code and dialogs are organized in a library system.

- The LibraryContainer contains libraries
- Libraries can contain modules and dialogs

In Basic

- The LibraryContainer is called **BasicLibraries**.

In dialogs

- The LibraryContainer is called **DialogLibraries**.

Both LibraryContainers exist in an application level and within every document. In the document Basic, the document's LibraryContainers are called automatically. If you want to call the global LibraryContainers from within a document, you must use the keyword **GlobalScope**.

Syntax

GlobalScope

Example

Example in the document Basic

```
' calling Dialog1 in the document library Standard
oDlgDesc = DialogLibraries.Standard.Dialog1
' calling Dialog2 in the application library Library1
oDlgDesc = GlobalScope.DialogLibraries.Library1.Dialog2
```

GoSub

Calls a subroutine that is indicated by a label from a subroutine or a function. The statements following the label are executed until the next Return statement. Afterwards, the program continues with the statement that follows the **GoSub** statement.

Syntax

see Parameters

Parameters

Sub/Function

statement block

Label

statement block

GoSub Label

Exit Sub/Function

Label:

statement block

Return

End Sub/Function

The **GoSub** statement calls a local subroutine indicated by a label from within a subroutine or a function. The name of the label must end with a colon (":").

If the program encounters a Return statement not preceded by **GoSub**, OpenOffice.org Basic returns an error message. Use **Exit Sub** or **Exit Function** to ensure that the program leaves a Sub or Function before reaching the next Return statement.

The following example demonstrates the use of **GoSub** and **Return**. By executing a program section twice, the program calculates the square root of two numbers that are entered by the user.

Example:

```
Sub ExampleGoSub
      Dim iInputa as Single
      Dim iInputb as Single
      Dim iInputc as Single
      iInputa = Int(InputBox$ "Enter the first number: ","NumberInput"))
      iInputb = Int(InputBox$ "Enter the second number: ","NumberInput"))
      iInputc=iInputa
      GoSub SquareRoot
      Print "The square root of";iInputa;" is";iInputc
      iInputc=iInputb
      GoSub SquareRoot
      Print "The square root of";iInputb;" is";iInputc
      Exit Sub

SquareRoot:
      iInputc=sqr(iInputc)
      Return
End Sub
```

GoTo

Continues program execution within a Sub or Function at the procedure line indicated by a label.

Syntax

see Parameters

Parameters

Sub/Function

statement block

Label1

Label2:

statement block

Exit Sub

Label1:

statement block

GoTo Label2

End Sub/Function

Use the GoTo statement to instruct OpenOffice.org Basic to continue program execution at another place within the procedure. The position must be indicated by a label. To set a label, assign a name, and then and end it with a colon (":").

You cannot use the GoTo statement to jump out of a Sub or Function.

Example:

```
Sub ExampleOnGosub
      Dim iVar As Integer
      Dim sVar As String
      iVar = 2
      sVar =""
      On iVar GoSub Sub1, Sub2
      On iVar GoTo Line1, Line2
      Exit Sub
Sub1:
      sVar =sVar & " From Sub 1 to" : Return
Sub2:
      sVar =sVar & " From Sub 2 to" : Return
Line1:
      sVar =sVar & " Label 1"
Line2:
      sVar =sVar & " Label 2"
      Print sVar
End Sub
```

Green()

Returns the Green component of the given color code.

Syntax

```
Green (Color As Long)
```

Return value

Integer

Parameter

Color
Long integer expression that specifies a color code for which to return the Green component.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleColor
      Dim lVar As Long
      lVar = rgb(128,0,200)
      MsgBox "The color " & lVar & " contains the components:" & Chr(13) &_
      "red = " & red(lVar) & Chr(13)&_
      "green = " & green(lVar) & Chr(13)&_
      "blue = " & blue(lVar) & Chr(13) , 64,"colors"
End Sub
```

HasUnoInterfaces()

Tests if a Basic Uno object supports certain Uno interfaces.

Returns True, if **all** stated Uno interfaces are supported, otherwise False is returned.

Syntax

HasUnoInterfaces(oTest, Uno-Interface-Name 1 [, Uno-Interface-Name 2, ...])

Return value

Bool

Parameters

Otest
the Basic Uno object that you want to test.

Uno-Interface-Name
list of Uno interface names.

Example

```
bHas = HasUnoInterfaces( oTest, "com.sun.star.beans.XIntrospection" )
```

Hex()

Returns a string that represents the hexadecimal value of a number.

Syntax

Hex (Number)

Return value

String

Parameters

Number
Any numeric expression that you want to convert to a hexadecimal number.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleHex
      REM uses BasicFormulas in OpenOffice.org Calc
      Dim a2, b2, c2 as String
      a2 = "&H3E8"
      b2 = Hex2Int(a2)
      MsgBox b2
      c2 = Int2Hex(b2)
      MsgBox c2
End Sub

Function Hex2Int( sHex As String ) As Long
      REM Returns a Long-Integer from a hexadecimal value.
      Hex2Int = clng( sHex )
End Function

Function Int2Hex( iLong As Long) As String
      REM Calculates a hexadecimal value in Integer.
      Int2Hex = "&H" & Hex( iLong )
End Function
```

Hour()

Returns the hour from a time value that is generated by the TimeSerial or the TimeValue function.

Syntax:

Hour (Number)

Return value

Integer

Parameters

Number
Numeric expression that contains the serial time value that is used to return the hour value.

This function is the opposite of the **TimeSerial** function. It returns an integer value that represents the hour from a time value that is generated by the **TimeSerial** or the **TimeValue** function. For example, the expression

Print Hour(TimeSerial(12:30:41))

returns the value 12.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleHour
      Print "The current hour is " & Hour( Now )
End Sub
```

If...Then...Else

Defines one or more statement blocks that you only want to execute if a given condition is True.

Syntax

If condition=true Then Statement block [ElseIf condition=true Then] Statement block [Else] Statement block End If

Parameters

The **If...Then** statement executes program blocks depending on given conditions. When OpenOffice.org Basic encounters an **If** statement, the condition is tested. If the condition is True, all subsequent statements up to the next **Else** or **ElseIf** statement are executed. If the condition is False, and an **ElseIf** statement follows, OpenOffice.org Basic tests the next condition and executes the following statements if the condition is True. If False, the program continues either with the next **ElseIf** or **Else** statement. Statements following **Else** are executed only if none of the previously tested conditions were True. After all conditions are evaluated, and the corresponding statements executed, the program continues with the statement following **EndIf**.

You can nest multiple **If...Then** statements.

Else and **ElseIf** statements are optional.

You can use **GoTo** and **GoSub** to jump out of an **If...Then** block, but not to jump into an **If...Then** structure.

The following example enables you to enter the expiration date of a product, and determines if the expiration date has passed.

Example

```
Sub ExampleIfThenDate
      Dim sDate as String
      Dim sToday as String
      sDate = InputBox("Enter the expiration date (MM.DD.YYYY)")
      sDate = Right$(sDate, 4) + Mid$(sDate, 4, 2) + Left$(sDate, 2)
      sToday = Date$
      sToday = Right$(sToday, 4)+ Mid$(sToday, 4, 2) + Left$(sToday, 2)
      If sDate < sToday Then
            MsgBox "The expiration date has passed"
      ElseIf sDate > sToday Then
            MsgBox "The expiration date has not yet passed"
      Else
            MsgBox "The expiration date is today"
      End If
End Sub
```

Iif()

Returns one of two possible function results, depending on the logical value of the evaluated expression.

Syntax:

IIf (Expression, ExpressionTrue, ExpressionFalse)

Parameters:

Expression
Any expression that you want to evaluate. If the expression evaluates to **True**, the function returns the result of ExpressionTrue, otherwise it returns the result of ExpressionFalse.

ExpressionTrue, ExpressionFalse
Any expression, one of which will be returned as the function result, depending on the logical evaluation.

Error Codes

5 Invalid procedure call

Imp

Performs a logical implication on two expressions.

Syntax

Result = Expression1 Imp Expression2

Parameters

Result
Any numeric variable that contains the result of the implication.

Expression1, Expression2
Any expressions that you want to evaluate with the Imp operator.

If you use the Imp operator in Boolean expressions, False is only returned if the first expression evaluates to True and the second expression to False.

If you use the Imp operator in bit expressions, a bit is deleted from the result if the corresponding bit is set in the first expression and the corresponding bit is deleted in the second expression.

Example

```
Sub ExampleImp
      Dim A as Variant, B as Variant, C as Variant, D as Variant
      Dim vOut as Variant
      A = 10: B = 8: C = 6: D = Null
      vOut = A > B Imp B > C REM returns -1
      vOut = B > A Imp B > C REM returns -1
      vOut = A > B Imp B > D REM returns 0
      vOut = (B > D Imp B > A) REM returns -1
      vOut = B Imp A REM returns -1
End Sub
```

Input#

Reads data from an open sequential file.

Syntax

```
Input #FileNumber As Integer; var1[, var2[, var3[,...]]]
```

Parameters

FileNumber
Number of the file that contains the data that you want to read. The file must be opened with the Open statement using the key word INPUT.

Var
A numeric or string variable that you assign the values read from the opened file to.

The **Input#** statement reads numeric values or strings from an open file and assigns the data to one or more variables. A numeric variable is read up to the first carriage return (Asc=13), line feed (Asc=10), space, or comma. String variables are read to up to the first carriage return (Asc=13), line feed (Asc=10), or comma.

Data and data types in the opened file must appear in the same order as the variables that are passed in the "var" parameter. If you assign non-numeric values to a numeric variable, "var" is assigned a value of "0".

Records that are separated by commas, commas cannot be assigned to a string variable. Quotation marks (") in the file are disregarded as well. If you want to read these characters from the file, use the **Line Input#** statement to read pure text files (files containing only printable characters) line by line.

If the end of the file is reached while reading a data element, an error occurs and the process is aborted.

Example

```
Sub ExampleWorkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "This is a line of text"
      Print #iNumber, "This is another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      While not eof(iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" Then
                  sMsg = sMsg & sLine & Chr(13)
            End If
      Wend
      Close #iNumber
      Msgbox sMsg
End Sub
```

InputBox()

Displays a prompt in a dialog at which the user can input text. The input is assigned to a variable.

The **InputBox** statement is a convenient method of entering text through a dialog. Confirm the input by clicking OK or pressing Return. The input is returned as the function return value. If you close the dialog with Cancel, **InputBox** returns a zero-length string ("").

Syntax

```
InputBox (Msg As String[, Title As String[, Default As String[, x_pos As
Integer, y_pos As Integer]]]])
```

Return value

String

Parameter

Msg
String expression displayed as the message in the dialog box.

Title
String expression displayed in the title bar of the dialog box.

Default
String expression displayed in the text box as default if no other input is given.

x_pos
Integer expression that specifies the horizontal position of the dialog. The position is an absolute coordinate and does not refer to the window of the office application.

y_pos
Integer expression that specifies the vertical position of the dialog. The position is an absolute coordinate and does not refer to the window of the office application.

If **x_pos** and **y_pos** are omitted, the dialog is centered on the screen. The position is specified in twips.

Example

```
Sub ExampleInputBox
      Dim sText As String
      sText = InputBox ("Please enter a phrase:","Dear User")
      MsgBox ( sText , 64, "Confirmation of phrase")
End Sub
```

InStr()

Returns the position of a string within another string.

The Instr function returns the position at which the match was found. If the string was not found, the function returns 0.

Syntax

```
InStr ([Start As Long,] Text1 As String, Text2 As String[, Compare])
```

Return value

Integer

Parameters

Start
A numeric expression that marks the position in a string where the search for the specified substring starts. If you omit this parameter, the search starts at the first character of the string. The maximum allowed value is 65535.

Text1
The string expression that you want to search.

Text2
The string expression that you want to search for.

Compare
Optional numeric expression that defines the type of comparison. The value of this parameter can be 0 or 1. The default value of 1 specifies a text comparison that is not case-sensitive. The value of 0 specifies a binary comparison that is case-sensitive.

To avoid a run-time error, do not set the Compare parameter if the first return parameter is omitted.

Error Codes

5 Invalid procedure call

Example

```
Sub ExamplePosition
      Dim sInput As String
      Dim iPos as Integer
      sInput = "Office"
      iPos = Instr(sInput,"c")
      Print iPos
End Sub
```

Int()

Returns the integer portion of a number.

Syntax

Int (Number)

Return value

Double

Parameters

Number
Any valid numeric expression. Rounds the fractional part of the number and returns the integer value.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleINT
      Print Int(3.14159) REM returns the value 3.
      Print Int(0) REM returns the value 0.
      Print Int(-3.14159) REM returns the value -4.
End Sub
```

IsArray()

Determines if a variable is a data field in an array.

Syntax

IsArray (Var)

Return value

Bool

Parameters

Var

Any variable that you want to test if it is declared as an array. If the variable is an array, then the function returns **True**, otherwise **False** is returned.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleIsArray
      Dim sDatf(10) as String
      Print isarray(sdatf())
End Sub
```

IsDate()

Tests if a numeric or string expression can be converted to a **Date** variable.

Syntax

IsDate (Expression)

Return value

Bool

Parameters

Expression
Any numeric or string expression that you want to test. If the expression can be converted to a date, the function returns **True**, otherwise the function returns **False**.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleIsDate
      Dim sDateVar as String
      sDateVar = "12.12.1997"
      Print IsDate(sDateVar) REM Returns True
      sDateVar = "12121997"
      Print IsDate(sDateVar) REM Returns False
End Sub
```

IsEmpty()

Tests if a Variant variable contains the Empty value. The Empty value indicates that the variable is not initialized.

Syntax

IsEmpty (Var)

Return value

Bool

Parameters

Var

Any variable that you want to test. If the Variant contains the Empty value, the function returns True, otherwise the function returns False.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleIsEmpty
      Dim sVar as Variant
      sVar = Empty
      Print IsEmpty(sVar) REM Returns True
End Sub
```

IsError()

Tests if a variable contains an error value.

Syntax

IsError (Var)

Return value

Bool

Parameters

Var

Any variable that you want to test. If the variable contains an error value, the function returns True, otherwise the function returns False.

IsMissing()

Tests if a function is called with an optional parameter.

See also: Optional

Syntax

IsMissing(ArgumentName)

Parameters

ArgumentName
the name of an optional argument.

If the IsMissing function is called by the ArgumentName, then True is returned.

See also Examples.

Error Codes

5 Invalid procedure call

IsNull()

Tests if a Variant contains the special Null value, indicating that the variable does not contains data.

Syntax

IsNull (Var)

Return value

Bool

Parameters

Var
Any variable that you want to test. This function returns True if the Variant contains the Null value, or False if the Variant does not contain the Null value.

Null
This value is used for a variant data sub type without valid contents.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleIsNull
      Dim vVar As Variant
      MsgBox IsNull(vVar)
End Sub
```

IsNumeric()

Tests if an expression is a number. If the expression is a number, the function returns True, otherwise the function returns False.

Syntax:

IsNumeric (Var)

Return value

Bool

Parameters

Var
Any expression that you want to test.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleIsNumeric
      Dim vVar as variant
      vVar = "ABC"
      Print IsNumeric(vVar) REM Returns False
      vVar = "123"
      Print IsNumeric(vVar) REM Returns True
End Sub
```

IsObject()

Tests if an object variable is an OLE object. The function returns True if the variable is an OLE object, otherwise it returns False.

Syntax

IsObject (ObjectVar)

Return value

Bool

Parameters

ObjectVar
Any variable that you want to test. If the Object variable contains an OLE object, the function returns True.

Error Codes

5 Invalid procedure call

IsUnoStruct()

Returns True if the given object is a Uno struct.

Syntax

IsUnoStruct(Uno type)

Return value

Bool

Parameters

Uno type
A UnoObject

Example

```
Sub Main
	Dim bIsStruct
	' Instantiate a service
	Dim oSimpleFileAccess
	oSimpleFileAccess = CreateUnoService( "com.sun.star.ucb.SimpleFileAccess"
)
	bIsStruct = IsUnoStruct( oSimpleFileAccess )
	MsgBox bIsStruct ' Displays False because oSimpleFileAccess is NO struct
	' Instantiate a Property struct
	Dim aProperty As New com.sun.star.beans.Property
	bIsStruct = IsUnoStruct( aProperty )
	MsgBox bIsStruct ' Displays True because aProperty is a struct
	bIsStruct = IsUnoStruct( 42 )
	MsgBox bIsStruct ' Displays False because 42 is NO struct
End Sub
```

Join()

Returns a string from a number of substrings in a string array.

Syntax

Join (Text As String Array, delimiter)

Return value

String

Parameters

Text
A string array.

delimiter (optional)
A string character that is used to separate the substrings in the resulting string. The default delimiter is the space character. If delimiter is a string of length zero "", the substrings are joined without separator.

Example

```
Dim a(3)
Sub main()
      a(0) = "ABCDE"
      a(1) = 42
      a(2) = "MN"
      a(3) = "X Y Z"
      JStr = Join1()
      Call Show(JStr, Split1(JStr))
      JStr = Join2()
      Call Show(JStr, Split1(JStr))
      JStr = Join3()
      Call Show(JStr, Split1(JStr))
End Sub

Function Join1()
      Join1 = Join(a(), "abc")
End Function

Function Join2()
      Join2 = Join(a(), ",")
End Function

Function Join3()
      Join3 = Join(a())
End Function

Function Split1(aStr)
      Split1 = Split(aStr, "D")
End Function

Sub Show(JoinStr, TheArray)
      l = LBound(TheArray)
      u = UBound(TheArray)
      total$ = "============================" + Chr$(13) + JoinStr + _
            Chr$(13) + Chr$(13)
      For i = l To u
            total$ = total$ + TheArray(i) + Str(Len(TheArray(i))) + Chr$(13)
      Next i
      MsgBox total$
End Sub
```

Kill

Deletes a file from a disk.

Syntax

```
Kill File As String
```

Parameters

File
Any string expression that contains an unambiguous file specification. You can also use URL notation.

Error Codes

5 Invalid procedure call

76 Path not found

Example

```
Sub ExampleKill
      Kill "C:\datafile.dat" REM File must be created in advance
End Sub
```

Lbound()

Returns the lower boundary of an array.

Syntax

LBound (ArrayName [, Dimension])

Return value

Integer

Parameters

ArrayName
Name of the array for which you want to return the upper (**Ubound**) or the lower (**LBound**) boundary of the array dimension.

[Dimension]
Integer that specifies which dimension to return the upper (**Ubound**) or the lower (**LBound**) boundary for. If a value is not specified, the first dimension is assumed.

Error Codes

5 Invalid procedure call

9 Subscript out of range

Example

```
Sub ExampleUboundLbound
      Dim sVar(10 to 20) As String
      Print LBound(sVar())
      Print UBound(sVar())
End Sub

Sub ExampleUboundLbound2
      Dim sVar(10 to 20,5 To 70) As String
      Print LBound(sVar()) REM Returns 10
      Print UBound(sVar()) REM Returns 20
      Print LBound(sVar(),2) REM Returns 5
      Print UBound(sVar(),2) REM Returns 70
Snd Sub
```

Lcase()

Converts all uppercase letters in a string to lowercase.

See also: UCase Function

Syntax

LCase (Text As String)

Return value

String

Parameters

Text
Any string expression that you want to convert.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleLUCase
      Dim sVar As String
      sVar = "Las Vegas"
      Print LCase(sVar) REM Returns "las vegas"
      Print UCase(sVar) REM Returns "LAS VEGAS"
End Sub
```

Left()

Returns the number of leftmost characters that you specify of a string expression.

Syntax

Left (Text As String, n As Long)

Return value

String

Parameters

Text
Any string expression that you want to return the leftmost characters from.

n
Numeric expression that specifies the number of characters that you want to return. If **n** = 0, a zero-length string is returned. The maximum allowed value is 65535.

The following example converts a date in YYYY.MM.DD format to MM/DD/YYYY format.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleUSDate
      Dim sInput As String
      Dim sUS_date As String
      sInput = InputBox("Please input a date in the international format “&_
            “'YYYY-MM-DD'")
      sUS_date = Mid(sInput, 6, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Right(sInput, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Left(sInput, 4)
      MsgBox sUS_date
End Sub
```

Len()

Returns the number of characters in a string, or the number of bytes that are required to store a variable.

Syntax:

Len (Text As String)

Return value

Long

Parameters

Text
Any string expression or a variable of another type.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleLen
      Dim sText as String
      sText = "Las Vegas"
      MsgBox Len(sText) REM Returns 9
End Sub
```

Let

Assigns a value to a variable.

Syntax

[Let] VarName=Expression

Parameters

VarName
Variable that you want to assign a value to. Value and variable type must be compatible.

As in most BASIC dialects, the key word **Let** is optional.

Example

```
Sub ExampleLen
      Dim stext as String
      Let sText = "Las Vegas"
      msgbox Len(sText) REM returns 9
End Sub
```

Line Input

Reads strings from a sequential file into a variable.

Syntax

```
Line Input #FileNumber As Integer, Var As String
```

Parameters

FileNumber
Number of the file that contains the data that you want to read. The file must have been opened in advance with the Open statement using the key word READ.

Var
The name of the variable that stores the result.

With the **Line Input#** statement, you can read strings from an open file into a variable. String variables are read line-by-line up to the first carriage return (Asc=13) or linefeed (Asc=10). Line end marks are not included in the resulting string.

Example

```
Sub ExampleWorkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "This is a line of text"
      Print #iNumber, "This is another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      While not eof(iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" Then
                  sMsg = sMsg & sLine & chr(13)
            End if
      Wend
      Close #iNumber
      Msgbox sMsg
End Sub
```

Loc()

Returns the current position in an open file.

Syntax

```
Loc(FileNumber)
```

Return value

Long

Parameters

FileNumber
Any numeric expression that contains the file number that is set by the Open statement for the respective file.

If the Loc function is used for an open random access file, it returns the number of the last record that was last read or written.

For a sequential file, the Loc function returns the position in a file divided by 128. For binary files, the position of the last read or written byte is returned.

Error Codes

5 Invalid procedure call

52 Bad file name or number

Lof()

Returns the size of an open file in bytes.

Syntax

```
Lof (FileNumber)
```

Return value

Long

Parameters

FileNumber
Any numeric expression that contains the file number that is specified in the Open statement.

To obtain the length of a file that is not open, use the **FileLen** function.

Error Codes

5 Invalid procedure call

52 Bad file name or number

Example

```
Sub ExampleRandomAccess
      Dim iNumber As Integer
      Dim sText As Variant REM must be a Variant
      Dim aFile As String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Seek #iNumber,1 REM Position at start
      Put #iNumber,, "This is the first line of text" REM Fill with text
      Put #iNumber,, "This is the second line of text"
      Put #iNumber,, "This is the third line of text"
      Seek #iNumber,2
      Get #iNumber,,sText
      Print sText
      Close #iNumber
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Get #iNumber,2,sText
      Put #iNumber,,"This is a new line of text"
      Get #iNumber,1,sText
      Get #iNumber,2,sText
      Put #iNumber,20,"This is the text in record 20"
      Print Lof(#iNumber)
      Close #iNumber
End Sub
```

Log()

Returns the natural logarithm of a number.

Syntax

Log (Number)

Return value

Double

Parameters

Number
Any numeric expression that you want to calculate the natural logarithm for.

The natural logarithm is the logarithm to the base e. Base e is a constant with an approximate value of 2.718282...

You can calculate logarithms to any base (n) for any number (x) by dividing the natural logarithm of x by the natural logarithm of n, as follows:

Log n(x) = Log(x) / Log(n)

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleLogExp
      Dim a as Double
      Dim const b1=12.345e12
      Dim const b2=1.345e34
      a=Exp( Log(b1)+Log(b2) )
      MsgBox "" & a & chr(13) & (b1*b2) ,0,_
            "Multiplication by logarithm function"
End Sub
```

Lset

Aligns a string to the left of a string variable, or copies a variable of a user-defined type to another variable of a different user-defined type.

Syntax

LSet Var As String = Text or LSet Var1 = Var2

Parameters

Var
Any String variable that contains the string that you want align to the left.

Text
String that you want to align to the left of the string variable.

Var1
Name of the user-defined type variable that you want to copy to.

Var2
Name of the user-defined type variable that you want to copy from.

If the string is shorter than the string variable, **LSet** left-aligns the string within the string variable. Any remaining positions in the string variable are replaced by spaces. If the string is longer than the string variable, only the leftmost characters up to the length of the string variable are copied. With the **LSet** statement, you can also copy a user-defined type variable to another variable of the same type.

Example

```
Sub ExampleRLSet
      Dim sVar As String
      Dim sExpr As String
      sVar = String(40,"*")
      sExpr = "SBX"
      REM Align "SBX" within the 40-character reference string
      REM Replace asterisks with spaces
      RSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(5,"*")
      sExpr = "123457896"
      RSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(40,"*")
      sExpr = "SBX"
      REM Left-align "SBX" within the 40-character reference string
      LSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(5,"*")
      sExpr = "123456789"
      LSet sVar = sExpr
      Print ">"; sVar; "<"
End Sub
```

Ltrim()

Removes all leading spaces at the start of a string expression.

Syntax

LTrim (Text As String)

Return value

String

Parameters

Text
Any string expression.

Use this function to remove spaces at the beginning of a string expression.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSpaces
      Dim sText2 As String,sText As String,sOut As String
      sText2 = " <*Las Vegas*> "
      sOut = "'"+sText2 +"'"+ Chr(13)
      sText = Ltrim(sText2) REM sText = "<*Las Vegas*> "
      sOut = sOut + "'"+sText +"'" + Chr(13)
      sText = Rtrim(sText2) REM sText = " <*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'" + Chr(13)
      sText = Trim(sText2) REM sText = "<*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'"
      MsgBox sOut
End Sub
```

Mid()

Returns the specified portion of a string expression (**Mid function**), or replaces the portion of a string expression with another string (**Mid statement**).

Syntax

Mid (Text As String, Start As Long [, Length As Long]) or Mid (Text As String, Start As Long , Length As Long, Text As String)

Return value

String (only by Function)

Parameters

Text
Any string expression that you want to modify.

Start
Numeric expression that indicates the character position within the string where the string portion that you want to replace or to return begins. The maximum allowed value is 65535.

Length
Numeric expression that returns the number of characters that you want to replace or return. The maximum allowed value is 65535.

If the Length parameter in the **Mid function** is omitted, all characters in the string expression from the start position to the end of the string are returned.

If the Length parameter in the **Mid statement** is less than the length of the text that you want to replace, the text is reduced to the specified length.

Text: The string to replace the string expression (**Mid statement**).

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleUSDate
      Dim sInput As String
      Dim sUS_date As String
      sInput = InputBox("Please input a date in the international format “&_
            “'YYYY-MM-DD'")
      sUS_date = Mid(sInput, 6, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Right(sInput, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Left(sInput, 4)
      MsgBox sUS_date
End Sub
```

Minute()

Returns the minute of the hour that corresponds to the serial time value that is generated by the TimeSerial or the TimeValue function.

Syntax

Minute (Number)

Return value

Integer

Parameters

Number
Numeric expression that contains the serial time value that is used to return the minute value.

This function is the opposite of the **TimeSerial** function. It returns the minute of the serial time value that is generated by the **TimeSerial** or the **TimeValue** function. For example, the expression:

Print Minute(TimeSerial(12:30:41))

returns the value 30.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleMinute
      MsgBox "The current minute is "& Minute(Now)& "."
End Sub
```

MkDir

Creates a new directory on a data medium.

Syntax

```
MkDir Text As String
```

Parameters

Text

Any string expression that specifies the name and path of the directory to be created. You can also use URL notation.

If the path is not determined, the directory is created in the current directory.

Error Codes

5 Invalid procedure call

76 Path not found

Example

```
Sub ExampleFileIO
	' Example for functions of the file organisation
	Const sFile1 as String = "file://c|/autoexec.bat"
	Const sDir1 as String = "file://c|/Temp"
	Const sSubDir1 as String ="Test"
	Const sFile2 as String = "Copied.tmp"
	Const sFile3 as String = "Renamed.tmp"
	Dim sFile as String
	sFile = sDir1 + "/" + sSubDir1
	ChDir( sDir1 )
	If Dir(sSubDir1,16)="" then ' Does the directory exist ?
		MkDir sSubDir1
		MsgBox sFile,0,"Create directory"
	End If
	sFile = sFile + "/" + sFile2
	FileCopy sFile1 , sFile
	MsgBox fSysURL(CurDir()),0,"Current directory"
	MsgBox sFile & Chr(13) & FileDateTime( sFile ),0,"Creation time"
	MsgBox sFile & Chr(13)& FileLen( sFile ),0,"File length"
	MsgBox sFile & Chr(13)& GetAttr( sFile ),0,"File attributes"
	Name sFile as sDir1 + "/" + sSubDir1 + "/" + sFile3
	' Rename in the same directory
	sFile = sDir1 + "/" + sSubDir1 + "/" + sFile3
	SetAttr( sFile, 0 ) 'Delete all attributes
	MsgBox sFile & Chr(13) & GetAttr( sFile ),0,"New file attributes"
	Kill sFile
	RmDir sDir1 + "/" + sSubDir1
End Sub

' Converts a system path in URL
Function fSysURL( fSysFp as String ) as String
	Dim iPos As String
	iPos = 1
	iPos = Instr(iPos,fSysFp, getPathSeparator())
	Do While iPos > 0
		Mid( fSysFp, iPos , 1,"/")
		iPos = Instr(iPos+1,fSysFp, getPathSeparator())
	Loop
	' the colon with DOS
	iPos = Instr(1,fSysFp,":")
```

```
      If iPos > 0 Then Mid( fSysFp, iPos , 1,"|")
      fSysURL = "file://" & fSysFp
End Function
```

Mod

Returns the integer remainder of a division.

Syntax:

Result = Expression1 MOD Expression2

Return value

Integer

Parameters

Result
Any numeric variable that contains the result of the MOD operation.

Expression1, Expression2
Any numeric expressions that you want to divide.

Example

```
Sub ExampleMod
      Print 10 mod 2.5 REM returns 0
      Print 10 / 2.5 REM returns 4
      Print 10 mod 5 REM returns 0
      Print 10 / 5 REM returns 2
      Print 5 mod 10 REM returns 5
      Print 5 / 10 REM returns 0.5
End Sub
```

Month()

Returns the month of a year from a serial date that is generated by the DateSerial or the DateValue function.

Syntax

Month (Number)

Return value

Integer

Parameters

Number
Numeric expression that contains the serial date number that is used to determine the month of the year.

This function is the opposite of the **DateSerial** function. It returns the month in the year that corresponds to the serial date that is generated by **DateSerial** or **DateValue**. For example, the expression

Print Month(DateSerial(1994, 12, 20))

returns the value 12.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleMonth
      MsgBox "" & Month(Now) ,64,"The current month"
End sub
```

MsgBox()

Displays a dialog box containing a message and returns a value.

Syntax

```
MsgBox (Text As String [,Type As Integer [,Dialogtitle As String]])
```

Return value

Integer

1 : OK

2 : Cancel

3 : Abort

4 : Retry

5 : Ignore

6 : Yes

7 : No

Parameter

Text
String expression displayed as a message in the dialog box. Line breaks can be inserted with Chr$(13).

DialogTitle
String expression displayed in the title bar of the dialog. If omitted, the name of the respective application is displayed.

Type
Any integer expression that specifies the dialog type and defines the number and type of buttons or icons displayed. **Type** represents a combination of bit patterns (dialog elements defined by adding the respective values):
0 : Display OK button only.
1 : Display OK and Cancel buttons.
2 : Display Abort, Retry, and Ignore buttons.
3 : Display Yes, No, and Cancel buttons.
4 : Display Yes and No buttons.
5 : Display Retry and Cancel buttons.
16 : Add the Stop icon to the dialog.
32 : Add the Question icon to the dialog.
48 : Add the Exclamation Point icon to the dialog.
64 : Add the Information icon to the dialog.
128 : First button in the dialog as default button.
256 : Second button in the dialog as default button.
512 : Third button in the dialog as default button.

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleMsgBox
      Dim sVar as Integer
      sVar = MsgBox("Las Vegas")
```

```
      sVar = MsgBox("Las Vegas",1)
      sVar = MsgBox( "Las Vegas",256 + 16 + 2,"Dialog title")
End Sub
```

MsgBox

Displays a dialog box containing a message.

Syntax:

```
MsgBox Text As String [,Type As Integer [,Dialogtitle As String]] (As
Statement) or MsgBox (Text As String [,Type As Integer [,Dialogtitle As
String]]) (As Function)
```

Parameter:

Text
String expression displayed as a message in the dialog box. Line breaks can be inserted with Chr$(13).

DialogTitle
String expression displayed in the title bar of the dialog. If omitted, the title bar displays the name of the respective application.

Type
Any integer expression that specifies the dialog type, as well as the number and type of buttons to display, and the icon type. **Type** represents a combination of bit patterns, that is, a combination of elements can be defined by adding their respective values:
0 : Display OK button only.
1 : Display OK and Cancel buttons.
2 : Display Abort, Retry, and Ignore buttons.
3 : Display Yes, No and Cancel buttons.
4 : Display Yes and No buttons.
5 : Display Retry and Cancel buttons.
16 : Add the Stop icon to the dialog.
32 : Add the Question icon to the dialog.
48 : Add the Exclamation icon to the dialog.
64 : Add the Information icon to the dialog.
128 : First button in the dialog as default button.
256 : Second button in the dialog as default button.
512 : Third button in the dialog as default button.

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleMsgBox
      Const sText1 = "An unexpected error occurred."
      Const sText2 = "The program execution will continue, however."
      Const sText3 = "Error"
      MsgBox(sText1 + Chr(13) + sText2,16,sText3)
End sub
```

Name

Renames an existing file or directory.

Syntax

```
Name OldName As String As NewName As String
```

Parameters

OldName, NewName
Any string expression that specifies the file name, including the path. You can also use URL notation.

If the path is not determined, the directory is created in the current directory.

Example

```
Sub ExampleReName
      On Error Goto Error
      Filecopy "c:\autoexec.bat", "c:\temp\autoexec.sav"
      Name "c:\test\autoexec.sav" as "c:\temp\autoexec.bat"
End

Error:
      If Err = 76 Then
            MsgBox "File already exists"
      End If
      End
End Sub
```

Not

Negates an expression by inverting the bit values.

Syntax

Result = Not Expression

Parameters

Result
Any numeric variable that contains the result of the negation.

Expression
Any expression that you want to negate.

When a Boolean expression is negated, the value True changes to False, and the value False changes to True.

In a bitwise negation each individual bit is inverted.

Example

```
Sub ExampleNot
      Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
      Dim vOut as Variant
      vA = 10: vB = 8: vC = 6: vD = Null
      vOut = Not vA REM Returns -11
      vOut = Not(vC > vD) REM Returns -1
      vOut = Not(vB > vA) REM Returns -1
      vOut = Not(vA > vB) REM Returns 0
End Sub
```

Now

Returns the current system date and time as a **Date** value.

Syntax

Now

Return value

Date

Example

```
Sub ExampleNow
      MsgBox "It is now " & Now
End Sub
```

Oct()

Returns the octal value of a number.

Syntax

Oct (Number)

Return value

String

Parameters

Number
Any numeric expression that you want to convert to an octal value.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleOkt
      Msgbox Oct(255)
End Sub
```

On Error GoTo ... Resume

Enables an error-handling routine after an error occurs, or resumes program execution.

Syntax

On {Error GoTo Labelname | GoTo 0 | Resume Next}

Parameters

GoTo Labelname
If an error occurs, enables the error-handling routine that starts at the line "Labelname".

Resume Next
If an error occurs, program execution continues with the statement that follows the statement in which the error occurred.

GoTo 0
Disables the error handler in the current procedure.

The On Error GoTo statement is used to react to errors that occur in a macro. The statement must be inserted at the start of a procedure (in a local error-handling routine) or at the start of a module.

Example

```
Sub ExampleReset
	On Error Goto ErrorHandler
	Dim iNumber As Integer
	Dim iCount As Integer
	Dim sLine As String
	Dim aFile As String
	aFile = "c:\data.txt"
	iNumber = Freefile
	Open aFile For Output As #iNumber
	Print #iNumber, "This is a line of text"
	Close #iNumber
	iNumber = Freefile
	Open aFile For Input As iNumber
	For iCount = 1 to 5
		Line Input #iNumber, sLine
		If sLine <>"" then
			rem
		end if
	Next iCount
	Close #iNumber
	Exit Sub

ErrorHandler:
	Reset
	MsgBox "All files will be closed",0,"Error"
End Sub
```

On...GoSub

On...GoTo

Branches to one of several specified lines in the program code, depending on the value of a numeric expression.

Syntax:

On N GoSub Label1[, Label2[, Label3[,...]]]

On NumExpression GoTo Label1[, Label2[, Label3[,...]]]

Parameters:

NumExpression
Any numeric expression between 0 and 255 that determines which of the lines the program branches to. If NumExpression is 0, the statement is not executed. If NumExpression is greater than 0, the program jumps to the label that has a position number that corresponds to the expression (1 = First label; 2 = Second label)

Label
Target line according to **GoTo** or **GoSub** structure.

The **GoTo** or **GoSub** conventions are valid.

Example:

```
Sub ExampleOnGosub
      Dim iVar As Integer
      Dim sVar As String
      iVar = 2
      sVar =""
      On iVar GoSub Sub1, Sub2
      On iVar GoTo Line1, Line2
      Exit Sub

Sub1:
      sVar =sVar & " From Sub 1 to" : Return

Sub2:
      sVar =sVar & " From Sub 2 to" : Return

Line1:
      sVar =sVar & " Label 1" : GoTo Ende

Line2:
      sVar =sVar & " Label 2"

Ende:
      MsgBox sVar,0,"On...Gosub"
End Sub
```

Open

Opens a data channel.

Syntax

```
Open FileName As String [For Mode] [Access IOMode] [Protected] As [#]FileNumber
As Integer [Len = DatasetLength]
```

Parameters

FileName
Name and path of the file that you wan to open. If you try to read a file that does not exist (Access = Read), an error message appears. If you try to write to a file that does exist (Access = Write), a new file is created.

Mode
Keyword that specifies the file mode. Valid values: Append (append to sequential file), binary (data can be accessed by bytes using Get and Put), Input (opens data channel for reading), Output (opens data channel for writing), and Random (edits relative files).

IOMode
Keyword that defines the access type. Valid values: Read (read-only), Write (write-only), Read Write (both).

Protected
Keyword that defines the security status of a file after opening. Valid values: Shared (file may be opened by other applications), Lock Read (file is protected against reading), Lock Write (file is protected against writing), Lock Read Write (denies file access).

FileNumber
Any integer expression from 0 to 511 to indicate the number of a free data channel. You can then pass commands through the data channel to access the file. The file number must be determined by the FreeFile function immediately before the Open statement.

DatasetLength
For relative files, returns the length of a certain record. This parameter is only necessary if the file was opened for Random access.

You can only modify the contents of a file that was opened with the Open statement. If you try to open a file that is already open, an error message appears.

Example

```
Sub ExampleWorkWithAFile
      Dim iNumber As Integer
      Dim sLine As String
      Dim aFile As String
      Dim sMsg as String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "This is a line of text"
      Print #iNumber, "This is another line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      While not eof(iNumber)
            Line Input #iNumber, sLine
            If sLine <>"" Then
                  sMsg = sMsg & sLine & chr(13)
```

```
            End If
        Wend
        Close #iNumber
        Msgbox sMsg
End Sub
```

Option Base

Defines the default lower boundary for arrays as 0 or 1.

Syntax

Option Base { 0 | 1}

Parameters

	This statement must be added before the executable program code in a module.

Example

```
option Base 1
Sub ExampleOptionBase
      Dim sVar(20) As String
      msgbox LBound(sVar())
End Sub
```

Option Explicit

Specifies that every variable in the program code must be explicitly declared with the Dim statement.

Syntax

Option Explicit

Parameters

	This statement must be added before the executable program code in a module.

Example

```
Option Explicit
Sub ExampleExplicit
      Dim sVar As String
      sVar = "Las Vegas"
      For i% = 1 to 10 REM This results in a run-time error
            REM
      Next i%
End Sub
```

Optional

Allows you to define parameters that are passed to a function as optional.

See also: IsMissing

Syntax

Function MyFunction(Text1 As String, Optional Arg2, Optional Arg3)

Examples

```
Result = MyFunction("Here", 1, "There") ' all arguments are passed.
Result = MyFunction("Test", ,1) ' second argument is missing.
```

See also Examples.

Or

Performs a logical OR disjunction on two expressions.

Syntax

Result = Expression1 Or Expression2

Parameters

Result
Any numeric variable that contains the result of the disjunction.

Expression1, Expression2
Any numeric expressions that you want to compare.

A logical OR disjunction of two Boolean expressions returns the value True if at least one comparison expression is True.

A bit-wise comparison sets a bit in the result if the corresponding bit is set in at least one of the two expressions.

Example

```
Sub ExampleOr
      Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
      Dim vOut as Variant
      vA = 10: vB = 8: vC = 6: vD = Null
      vOut = vA > vB Or vB > vC REM -1
      vOut = vB > vA Or vB > vC REM -1
      vOut = vA > vB Or vB > vD REM -1
      vOut = (vB > vD Or vB > vA) REM 0
      vOut = vB Or vA REM 10
End Sub
```

Print

Prints the specified strings or numeric expressions in a dialog.

Syntax

```
Print Expression1[{;|,} [Spc(Number As Integer);] [Tab(pos As Integer);]
[Expression2[...]]
```

Parameter

Expression
Any numeric or string expression to be printed. Multiple expressions can be separated by a semicolon. If separated by a comma, the expressions are indented to the next tab stop. The tab stops cannot be adjusted.

Number
Number of spaces to be inserted by the **Spc** function.

Pos
Spaces are inserted until the specified position.

If a semicolon or comma appears after the last expression to be printed, OpenOffice.org Basic stores the text in an internal buffer and continues program execution without printing. When another Print statement without a semicolon or comma at the end is encountered, all text to be printed is printed at once.

Positive numeric expressions are printed with a leading space. Negative expressions are printed with a leading minus sign. If a certain range is exceeded for floating-point values, the respective numeric expression is printed in exponential notation.

If the expression to be printed exceeds a certain length, the display will automatically wrap to the next line.

You can insert the Tab function, enclosed by semicolons, between arguments to indent the output to a specific position, or you can use the **Spc** function to insert a specified number of spaces.

Example

```
Sub ExamplePrint
      Print "ABC"
      Print "ABC","123"
End Sub
```

Public

Dimensions a variable or an array at the module level (that is, not within a subroutine or function), so that the variable and the array are valid in all libraries and modules.

Syntax

Public VarName[(start To end)] [As VarType][, VarName2[(start To end)] [As VarType][,...]]

Example

```
Public iPublicVar As Integer
Sub ExamplePublic
      iPublicVar = iPublicVar + 1
      MsgBox iPublicVar
End sub
```

Put

Writes a record to a relative file or a sequence of bytes to a binary file.

See also: Get statement

Syntax

```
Put [#] FileNumber As Integer, [position], Variable
```

Parameters

FileNumber
Any integer expression that defines the file that you want to write to.

Position
For relative files (random access files), the number of the record that you want to write.
For binary files (binary access), the position of the byte in the file where you want to start writing.

Variable
Name of the variable that you want to write to the file.

Note for relative files: If the contents of this variable does not match the length of the record that is specified in the **Len** clause of the **Open** statement, the space between the end of the newly written record and the next record is padded with existing data from the file that you are writing to.

Note for binary files: The contents of the variables are written to the specified position, and the file pointer is inserted directly after the last byte. No space is left between the records.

Example

```
Sub ExampleRandomAccess
      Dim iNumber As Integer
      Dim sText As Variant REM Must be a variant type
      Dim aFile As String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Seek #iNumber,1 REM Position to start writing
      Put #iNumber,, "This is the first line of text" REM Fill line with text
      Put #iNumber,, "This is the second line of text"
      Put #iNumber,, "This is the third line of text"
      Seek #iNumber,2
      Get #iNumber,,sText
      Print sText
      Close #iNumber
      iNumber = Freefile
      Open aFile For Random As #iNumber Len=32
      Get #iNumber,2,sText
      Put #iNumber,,"This is new text"
      Get #iNumber,1,sText
      Get #iNumber,2,sText
      Put #iNumber,20,"This is the text in record 20"
      Print Lof(#iNumber)
      Close #iNumber
End Sub
```

Randomize

Initializes the random-number generator.

Syntax:

Randomize [Number]

Parameters:

Number
Any integer value that initializes the random-number generator. If Number is omitted, the generator uses the current value of the system timer.

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleRandomize
      Dim iVar As Integer, sText As String
      Dim iSpectral(10) As Integer
      Randomize 2^14-1
      For iCount = 1 To 1000
      iVar = Int((10 * Rnd) ) REM Range from 0 to 9
      iSpectral(iVar) = iSpectral(iVar) +1
      Next iCount
      sText = " | "
      For iCount = 0 To 9
            sText = sText & iSpectral(iCount) & " | "
      Next iCount
      MsgBox sText,0,"Spectral Distribution"
end sub
```

Red()

Returns the Red component of the specified color code.

Syntax

```
Red (ColorNumber As Long)
```

Return value

Integer

Parameter

ColorNumber
Long integer expression that specifies any color code for which to return the Red component.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleColor
      Dim lVar As Long
      lVar = rgb(128,0,200)
      MsgBox "The color " & lVar & " consists of:" & Chr(13) &_
      "red= " & red(lVar) & Chr(13)&_
      "green= " & green(lVar) & Chr(13)&_
      "blue= " & blue(lVar) & Chr(13) , 64,"colors"
End Sub
```

ReDim

Declares a variable or an array.

Syntax:

[ReDim]Dim VarName [(start To end)] [As VarType][, VarName2 [(start To end)] [As VarType][,...]]

Optionally, you can add the **Preserve** keyword as a parameter to preserve the contents of the array that is redimensioned.

Parameters:

VarName
Any variable or array name.

Start, End
Numerical values or constants that define the number of elements (NumberElements=(end-start)+1) and the index range.
Start and End can be numeric expressions if ReDim is used at the procedure level.

VarType
Keyword that declares the data type of a variable.

Keyword
Variable type which are: Bool, Date, Double, Integer, Long, Object, Single, String, Variant

In OpenOffice.org Basic, you do not need to declare variables explicitly. However, you need to declare an array before you can use them. You can declare a variable with the Dim statement, using commas to separate multiple declarations. To declare a variable type, enter a type-declaration character following the name or use a corresponding key word.

OpenOffice.org Basic supports single or multi-dimensional arrays that are defined by a specified variable type. Arrays are suitable if the program contains lists or tables that you want to edit. The advantage of arrays is that it is possible to address individual elements according to indexes, which can be formulated as numeric expressions or variables.

There are two ways to set the range of indices for arrays declared with the Dim statement:

DIM text(20) As String REM 21 elements numbered from 0 to 20

DIM text(5 to 25) As String REM 21 elements numbered from 5 to 25

DIM text$(-15 to 5) As String REM 21 elements (0 inclusive),

rem numbered from -15 to 5

Variable fields, regardless of type, can be made dynamic if they are dimensioned by ReDim at the procedure level in subroutines or functions. Normally, you can only set the range of an array once and you cannot modify it. Within a procedure, you can declare an array using the ReDim statement with numeric expressions to define the range of the field sizes.

Example:

```
Sub ExampleRedim
      Dim iVar() As Integer, iCount As Integer
      ReDim iVar(5) As integer
      For iCount = 1 To 5
            iVar(iCount) = iCount
      Next iCount
      ReDim iVar(10) As integer
      For iCount = 1 To 10
            iVar(iCount) = iCount
      Next iCount
end sub
```

Rem

Specifies that a program line is a comment.

Syntax

Rem Text

Parameters

Text
Any text that serves as a comment.

You can use the single quotation mark instead of the Rem keyword to indicate that the text on a line is comments. This symbol can be inserted directly to the right of the program code, followed by a comment.

Example

```
Sub ExampleMid
      DIM sVar As String
      sVar = "Las Vegas"
      Print Mid(sVar,3,5) REM Returns "s Veg"
      REM Nothing occurs here
End Sub
```

Reset

Closes all open files and writes the contents of all file buffers to the harddisk.

Syntax

```
Reset
```

Example

```
Sub ExampleReset
      On Error Goto ErrorHandler
      Dim iNumber As Integer
      Dim iCount As Integer
      Dim sLine As String
      Dim aFile As String
      aFile = "c:\data.txt"
      iNumber = Freefile
      Open aFile For Output As #iNumber
      Print #iNumber, "This is a new line of text"
      Close #iNumber
      iNumber = Freefile
      Open aFile For Input As iNumber
      For iCount = 1 to 5
            Line Input #iNumber, sLine
            If sLine <>"" Then
                  rem
            end if
      Next iCount
      Close #iNumber
      Exit Sub

ErrorHandler:
      Reset
      MsgBox "All files will be closed",0,"Error"
End Sub
```

Right()

Returns the rightmost "n" characters of a string expression.

See also: Left Function.

Syntax

Right (Text As String, n As Long)

Return value

String

Parameters

Text
Any string expression that you want to return the rightmost characters of.

n
Numeric expression that defines the number of characters that you want to return. If **n** = 0, a zero-length string is returned. The maximum allowed value is 65535.

Error Codes

5 Invalid procedure call

Example

The following example converts a date in YYYY-MM-DD format to the US date format (MM/DD/YYYY).

```
Sub ExampleUSDate
      Dim sInput As String
      Dim sUS_date As String
      sInput = InputBox("Please input a date in the international format “&_
            “'YYYY-MM-DD'")
      sUS_date = Mid(sInput, 6, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Right(sInput, 2)
      sUS_date = sUS_date & "/"
      sUS_date = sUS_date & Left(sInput, 4)
      MsgBox sUS_date
End Sub
```

RmDir

Deletes an existing directory from a data medium.

Syntax

```
RmDir Text As String
```

Parameters

Text

Any string expression that specifies the name and path of the directory that you want to delete. You can also use URL notation.

If the path is not determined, the **RmDir Statement** searches for the directory that you want to delete in the current path. If it is not found there, an error message appears.

Error Codes

5 Invalid procedure call

76 Path not found

Example

```
Sub ExampleRmDir
      MkDir "C:\Test2"
      ChDir "C:\test2"
      MsgBox Curdir
      ChDir "\"
      RmDir "C:\test2"
End Sub
```

Rnd()

Returns a random number between 0 and 1.

Syntax

Rnd [(Expression)]

Return value

Double

Parameters

Expression
Any numeric expression that defines how to generate random numbers.

Less than zero
Always returns the same random number.

Greater than zero
Returns the next random number in the sequence.

Zero
Returns the random number that was last generated.

Omitted
Returns the next random number in the sequence.

If the same number is passed for each successive call to the Rnd function, the same random-number sequence is generated. This is because the Expression parameter is used as a starting point for the next number.

The **Rnd** function only returns values ranging from 0 to 1. To generate random integers in a given range, use the formula in the following example:

Error Codes

5 Invalid procedure call

Example:

```
Sub ExampleRandomSelect
      Dim iVar As Integer
      iVar = Int((15 * Rnd) -2)
      Select Case iVar
      Case 1 To 5
            Print "Number from 1 to 5"
      Case 6, 7, 8
            Print "Number from 6 to 8"
      Case Is > 8 And iVar < 11
            Print "Greater than 8"
      Case Else
            Print "Outside range 1 to 10"
      End Select
End Sub
```

Rset

Right-aligns a string within a string variable, or copies a user-defined variable type into another.

Syntax

RSet Text As String = Text or RSet Variable1 = Variable2

Parameters

Text
Any string variable.

Text
String that you want to right-align in the string variable.

Variable1
User-defined variable that is the target for the copied variable.

Variable2
User-defined variable that you want to copy to another variable.

If the string is shorter than the string variable, **RSet** aligns the string to the right within the string variable. Any remaining characters in the string variable are replaced with spaces. If the string is longer than the string variable, characters exceeding the length of the variable are truncated, and only the remaining characters are right-aligned within the string variable.

You can also use the **RSet statement** to assign variables of one user-defined type to another.

The following example uses the **RSet** and **LSet** statements to modify the left and right alignment of a string.

Example

```
Sub ExampleRLSet
      Dim sVar as string
      Dim sExpr as string
      sVar = String(40,"*")
      sExpr = "SBX"
      REM Right-align "SBX" in a 40-character string
      REM Replace asterisks with spaces
      RSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(5,"*")
      sExpr = "123457896"
      RSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(40,"*")
      sExpr = "SBX"
      REM Left-align "SBX" in a 40-character string
      LSet sVar = sExpr
      Print ">"; sVar; "<"
      sVar = String(5,"*")
      sExpr = "123456789"
      LSet sVar = sExpr
      Print ">"; sVar; "<"
End Sub
```

Rtrim()

Deletes the spaces at the end of a string expression.

See also: LTrim Function

Syntax

RTrim (Text As String)

Return value

String

Parameters

Text
Any string expression.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSpaces
      Dim sText2 as String,sText as String,sOut as String
      sText2 = " <*Las Vegas*> "
      sOut = "'"+sText2 +"'"+ Chr(13)
      sText = Ltrim(sText2) REM sText = "<*Las Vegas*> "
      sOut = sOut + "'"+sText +"'" + Chr(13)
      sText = Rtrim(sText2) REM sText = " <*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'" + Chr(13)
      sText = Trim(sText2) REM sText = "<*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'"
      MsgBox sOut
End Sub
```

Second()

Returns an integer that represents the seconds of the serial time number that is generated by the TimeSerial or the TimeValue function.

Syntax

Second (Number)

Return value

Integer

Parameters

Number
Numeric expression that contains the serial time number that is used to calculate the number of seconds.

This function is the opposite of the **TimeSerial** function. It returns the seconds of a serial time value that is generated by the **TimeSerial** or **TimeValue** functions. For example, the expression:

Print Second(TimeSerial(12,30,41))

returns the value 41.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSecond
      MsgBox "The exact second of the current time is "& Second( Now )
End sub
```

Seek()

Determines the position for the next writing or reading in a file that was opened with the open statement. The return value corresponds to the value determined by the Seek statement:

For random access files, the Seek function returns the number of the next record to be read.

For all other files, the function returns the byte position at which the next operation is to occur.

See also: Open, Seek.

Syntax

```
Seek (FileNumber)
```

Return value

Long

Parameters

FileNumber
Returns the file number determined by the Open statement.

Example

```
Sub ExampleSeek
      Dim i As Integer
      Dim iNumber As Integer
      iNumber = Freefile
      Open "c:\Data.txt" For Random As #iNumber
      For i = 1 To 10
            PUT #iNumber, , i
      Next i
      Seek #iNumber, 2
      Get #1, , i
      Print "Date: "; i;" Next Record: "; Seek(iNumber)
End Sub
```

Seek

Determines the position for the next writing or reading in a file that was opened with the Open statement.

For random access files, the Seek function returns the number of the next record to be written.

For all other files, the function returns the byte position at which the next operation is to occur.

See also: Open, Seek.

Syntax

```
Seek[#]FileNumber, Position (As Long)
```

Parameters

FileNumber
Returns the file number determined by the Open statement.

Position
Position for the next writing or reading. Position can be a number between 1 and 2,147,483,647. According to the file type, the position indicates the number of the record (files in the Random mode) or the byte position (files in the Binary, Output, Append or Input mode). The first byte in a file is position 1, the second byte is position 2, and so on.

Error Codes

5 Invalid procedure call

52 Bad file name or number

Select...Case

Defines one or more statement blocks depending on the value of an expression.

Syntax

Select Case condition Case expression Statement Block [Case expression2 Statement Block][Case Else] Statement Block End Select

Parameters

Condition
Any expression that controls if the statement block that follows the respective Case clause is executed.

Expression
Any expression that is compatible with the Condition type expression. The statement block that follows the Case clause is executed if **Condition** matches **Expression**.

Example

```
Sub ExampleRandomSelect
      Dim iVar As Integer
      iVar = Int((15 * Rnd) -2)
      Select Case iVar
            Case 1 To 5
                  Print "Number from 1 to 5"
            Case 6, 7, 8
                  Print "Number from 6 to 8"
            Case 8 To 10
                  Print "Greater than 8"
            Case Else
                  Print "Out of range 1 to 10"
      End Select
End Sub
```

Set

Sets an object reference on a variable or a Property.

Syntax

Set ObjectVar = Object

Parameters

ObjectVar
a variable or a property that requires an object reference.

Object
Object that the variable or the property refers to.

Nothing
Assign the **Nothing** object to a variable to remove a previous assignment.

Example

```
Sub ExampleSet
Dim oDoc As Object
Set oDoc = ActiveWindow
Print oDoc.Name
End Sub
```

SetAttr

Sets the attribute information for a specified file.

Syntax

```
SetAttr FileName As String, Attribute As Integer
```

Parameters

FileName

Name of the file, including the path, that you want to test attributes of. If you do not enter a path, **SetAttr** searches for the file in the current directory. You can also use URL notation.

Attribute

Bit pattern defining the attributes that you want to set or to clear:
0 : Normal files.
1 : Read-only files.
32 : File was changed since last backup (Archive bit).
You can set multiple attributes by combining the respective values with a logic OR statement.

Error Codes

5 Invalid procedure call

53 File not found

70 Permission denied

Example

```
Sub ExampleSetGetAttr
      On Error Goto ErrorHandler REM Define target for error-handler
      If Dir("C:\test",16)="" Then MkDir "C:\test"
      If Dir("C:\test\autoexec.sav")="" THEN Filecopy "c:\autoexec.bat",
      "c:\test\autoexec.sav"
      SetAttr "c:\test\autoexec.sav" ,0
      Filecopy "c:\autoexec.bat", "c:\test\autoexec.sav"
      SetAttr "c:\test\autoexec.sav" ,1
      Print GetAttr( "c:\test\autoexec.sav" )
      End

ErrorHandler:
      Print Error
      End
End Sub
```

Sgn()

Returns an integer number between -1 and 1 that indicates if the number that is passed to the function is positive, negative, or zero.

Syntax

Sgn (Number)

Return value

Integer

Parameters

Number
Numeric expression that determines the value that is returned by the function.

NumExpression	Return value
negative	Sgn returns -1.
0	Sgn returns 0.
positive	Sgn returns 1.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSgn
      Print sgn(-10) REM returns -1
      Print sgn(0) REM returns 0
      Print sgn(10) REM returns 1
End Sub
```

Shell()

Starts another application and defines the respective window style, if necessary.

Syntax

Shell (Pathname As String[, Windowstyle As Integer][, Param As String][, bSync])

Parameter

Pathname
Complete path and program name of the program that you want to start.

Windowstyle
Optional integer expression that specifies the style of the window that the program is executed in. The following values are possible:

0	The focus is on the hidden program window.
1	The focus is on the program window in standard size.
2	The focus is on the minimized program window.
3	focus is on the maximized program window.
4	Standard size program window, without focus.
6	Minimized program window, focus remains on the active window.
10	Full-screen display.

Param
Any string expression that specifies the command line that want to pass.

bSync
If this value is set to **true**, the **Shell** command and all OpenOffice.org tasks wait until the shell process completes. If the value is set to **false**, the shell returns directly. The default value is **false**.

Error Codes

5 Invalid procedure call

53 File not found

73 Feature not implemented

Example

```
Sub ExampleShellForWin
      Shell("c:\windows\calc.exe",2)
End Sub
```

Sin()

Returns the sine of an angle. The angle is specified in radians. The result lies between -1 and 1.

Using the angle Alpha, the Sin Function returns the ratio of the length of the opposite side of an angle to the length of the hypotenuse in a right-angled triangle.

Sin(Alpha) = side opposite the angle/hypotenuse

Syntax

Sin (Number)

Return value

Double

Parameters

Number
Numeric expression that defines the angle in radians that you want to calculate the sine for.

To convert degrees to radians, multiply degrees by Pi/180, and to convert radians to degrees, multiply radians by 180/Pi.

grad=(radiant*180)/pi

radiant=(grad*pi)/180

Pi is approximately 3.141593.

Error Codes

5 Invalid procedure call

Example

```
REM In this example, the following entry is possible for a right-angled
triangle:
REM The side opposite the angle and the angle (in degrees) to calculate the
length of the hypotenuse:
Sub ExampleSine
      REM Pi = 3.1415926 is a predefined variable
      Dim d1 as Double
      Dim dAlpha as Double
      d1 = InputBox$ ("Enter the length of the opposite side: ",_
            "Opposite   Side")
      dAlpha = InputBox$ ("Enter the angle Alpha (in degrees): ","Alpha")
      Print "The length of the hypotenuse is"; (d1 / sin (dAlpha * Pi / 180))
End Sub
```

Space()

Returns a string that consists of a specified amount of spaces.

Syntax

Space (n As Long)

Return value

String

Parameters

n

Numeric expression that defines the number of spaces in the string. The maximum allowed value of n is 65535.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSpace
      Dim sText As String,sOut As String
      DIm iLen As Integer
      iLen = 10
      sText = "Las Vegas"
      sOut = sText & Space(iLen) & sText & Chr(13) &_
      sText & Space(iLen*2) & sText & Chr(13) &_
      sText & Space(iLen*4) & sText & Chr(13)
      MsgBox sOut,0,"Info:"
End Sub
```

Split()

Returns an array of substrings from a string expression.

Syntax

Split (Text As String, delimiter, number)

Return value

String

Parameters

Text
Any string expression.

delimiter (optional)
A string of one or more characters length that is used to delimit the Text. The default is the space character.

number (optional)
The number of substrings that you want to return.

Example

```
Dim a(3)
Sub main()
      a(0) = "ABCDE"
      a(1) = 42
      a(2) = "MN"
      a(3) = "X Y Z"
      JStr = Join1()
      Call Show(JStr, Split1(JStr))
      JStr = Join2()
      Call Show(JStr, Split1(JStr))
      JStr = Join3()
      Call Show(JStr, Split1(JStr))
End Sub

Function Join1()
      Join1 = Join(a(), "abc")
End Function

Function Join2()
      Join2 = Join(a(), ",")
End Function

Function Join3()
      Join3 = Join(a())
End Function

Function Split1(aStr)
      Split1 = Split(aStr, "D")
End Function

Sub Show(JoinStr, TheArray)
      l = LBound(TheArray)
      u = UBound(TheArray)
      total$ = "============================" + Chr$(13) + JoinStr + _
            Chr$(13)    + Chr$(13)
      For i = l To u
            total$ = total$ + TheArray(i) + Str(Len(TheArray(i))) + Chr$(13)
      Next i
MsgBox total$
```

```
End Sub
```

Sqr()

Calculates the square root of a numeric expression.

Syntax

Sqr (Number)

Return value

Double

Parameters

Number
Any numeric expression that you want to calculate the square root for.

A square root is the number that you multiply by itself to produce another number, for example, the square root of 36 is 6.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSqr
      Dim iVar As Single
      iVar = 36
      MsgBox Sqr(iVar)
End Sub
```

Static

Declares a variable or an array at the procedure level within a subroutine or a function, so that the values of the variable or the array are retained after exiting the subroutine or function. Dim statement conventions are also valid.

The **Static statement** cannot be used to define variable arrays. Arrays must be specified according to a fixed size.

Syntax

Static VarName[(start To end)] [As VarType], VarName2[(start To end)] [As VarType], ...

Example

```
Sub ExampleStatic
      Static iInit As Integer
      if iInit = 0 then iInit = InitVar() REM Test if variable is initialized
      iInit = iInit + 1
      MsgBox iInit,0,"The answer is"
End Sub

REM Function for initialization of the static variable
Function InitVar() As Integer
      InitVar = 40 REM any value for initialization
End Function
```

Stop

Stops the execution of the Basic program.

Syntax

Stop

Example

```
Sub ExampleStop
      Dim iVar As Single
      iVar = 36
      Stop
      Msgbox Sqr(iVar)
End Sub
```

Str()

Converts a numeric expression into a string.

Syntax

Str (Expression)

Return value

String

Parameters

Expression
Any numeric expression.

The **Str** function converts a numeric variable, or the result of a calculation into a string. Negative numbers are preceded by a minus sign. Positive numbers are preceded by a space (instead of the plus sign).

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleStr
      Dim iVar As Single
      Dim sVar As String
      iVar = 123.123
      sVar = LTrim(Str(iVar))
      Msgbox sVar & chr(13) & Str(iVar)
End Sub
```

StrComp()

Compares two strings and returns an integer value that represents the result of the comparison.

Syntax

StrComp (Text1 As String, Text2 As String[, Compare])

Return value

Integer

- If Text1 < Text2 the function returns -1
- If Text1 = Text2 the function returns 0
- If Text1 > Text2 the function returns 1

Parameter

Text1
Any string expression

Text2
Any string expression

Compare
This optional parameter sets the comparison method. If Compare = 1, the string comparison is case-sensitive. If Compare = 0, no distinction is made between uppercase and lowercase letters.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleStrComp
      Dim iVar As Single
      Dim sVar As String
      iVar = 123.123
      sVar = Str$(iVar)
      MsgBox strcomp(sVar , Str$(iVar),1)
End Sub
```

String()

Creates a string according to the specified character, or the first character of a string expression that is passed to the function.

Syntax

String (n As Long, {expression As Integer | character As String})

Return value

String

Parameters

n
Numeric expression that indicates the number of characters to return in the string. The maximum allowed value of n is 65535.

Expression
Numeric expression that defines the ASCII code for the character.

Character
Any single character used to build the return string, or any string of which only the first character will be used.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleString
      Dim sText as String
      sText = String(10,"A")
      Msgbox sText
      sText = String(10,65)
      Msgbox sText
End Sub
```

Sub

Defines a subroutine.

Syntax

```
Sub Name[(VarName1 [As Type][, VarName2 [As Type][,...]])]
statement block
End Sub
```

Parameters

Name
Name of the subroutine .

VarName
Parameter that you want to pass to the subroutine.

Type
Type-declaration key word.

Example

```
Sub Example
      REM some statements
End Sub
```

Switch()

Evaluates a list of arguments, consisting of an expression followed by a value. The Switch function returns a value that is associated with the expression that is passed by this function.

Syntax

Switch (Expression1, Value1[, Expression2, Value2[..., Expression_n, Value_n]])

Parameters

The **Switch** function evaluates the expressions from left to right, and then returns the value that is assigned to the function expression. If expression and value are not given as a pair, a runtime error occurs.

Expression
The expression that you want to evaluate.

Value
The value that you want to return if the expression is True.

In the following example, the **Switch** function assigns the appropriate gender to the name that is passed to the function:

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSwitch
      Dim sGender As String
      sGender = GetGenderIndex( "John" )
      MsgBox sGender
End Sub

Function GetGenderIndex (sName As String) As String
      GetGenderIndex = Switch(sName = "Jane", "female", sName = "John", "male")
End Function
```

Tan()

Determines the tangent of an angle. The angle is returned in radians.

Using the angle Alpha, the Tan Function calculates the ratio of the length of the side opposite the angle to the length of the side adjacent to the angle in a right-angled triangle.

Tan(Alpha) = side opposite the angle/side adjacent to angle

Syntax

Tan (Number)

Return value

Double

Parameters:

Number
Any numeric expression that you want to calculate the tangent for (in radians).

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

grad=(radiant*180)/pi

radiant=(grad*pi)/180

Pi is approximately 3.141593.

Error Codes

5 Invalid procedure call

Example

```
REM In this example, the following entry is possible for a right-angled
REM triangle:
REM The side opposite the angle and the angle (in degrees) to calculate the
REM length of the side adjacent to the angle:

Sub ExampleTangens
      REM Pi = 3.1415926 is a pre-defined variable
      Dim d1 as Double
      Dim dAlpha as Double
      d1 = InputBox$ ("Enter the length of the side opposite the angle: ",_
            "opposite")
      dAlpha = InputBox$ ("Enter the Alpha angle (in degrees): ","Alpha")
      Print "the length of the side adjacent the angle is"; _
            (d1 / tan (dAlpha * Pi / 180))
End Sub
```

Time

This function returns the current system time as a string in the format "HH:MM:SS".

Syntax

Time

Parameters

Text
Any string expression that specifies the new time in the format "HH:MM:SS".

Example

```
Sub ExampleTime
      MsgBox Time,0,"The time is"
End Sub
```

Timer

Returns a value that specifies the number of seconds that have elapsed since midnight.

You must first declare a variable to call the Timer function and assign it the "Long " data type, otherwise a Date value is returned.

Syntax

Timer

Return value

Date

Example

```
Sub ExampleTimer
      Dim lSec as long,lMin as long,lHour as long
      lSec = Timer
      MsgBox lSec,0,"Seconds since midnight"
      lMin = lSec / 60
      lSec = lSec Mod 60
      lHour = lMin / 60
      lMin = lMin Mod 60
      MsgBox Right("00" & lHour , 2) & ":"& Right("00" & lMin , 2) & ":" &_
             Right("00" & lSec , 2) ,0,"The time is"
End Sub
```

TimeSerial()

Calculates a serial time value for the specified hour, minute, and second parameters that are passed as numeric value. You can then use this value to calculate the difference between times.

Syntax

TimeSerial (hour, minute, second)

Return value

Date

Parameters

hour
Any integer expression that indicates the hour of the time that is used to determine the serial time value. Valid values: 0-23.

minute
Any integer expression that indicates the minute of the time that is used to determine the serial time value. In general, use values between 0 and 59. However, you can also use values that lie outside of this range, where the number of minutes influence the hour value.

second
Any integer expression that indicates the second of the time that is used to determine the serial time value. In general, you can use values between 0 and 59. However, you can also use values that lie outside of this range, where the number seconds influences the minute value.

Examples
12, -5, 45 corresponds to 11, 55, 45

12, 61, 45 corresponds to 13, 2, 45

12, 20, -2 corresponds to 12, 19, 58

12, 20, 63 corresponds to 12, 21, 4

You can use the TimeSerial function to convert any time into a single value that you can use to calculate time differences.

The TimeSerial function returns the type Variant with VarType 7 (Date). This value is stored internally as a double-precision number between 0 and 0.9999999999. As opposed to the DateSerial or DateValue function, where the serial date values are calculated as days relative to a fixed date, you can calculate with values returned by the TimeSerial function, but you cannot evaluate them.

In the TimeValue function, you can pass a string as a parameter containing the time. For the TimeSerial function, however, you can pass the individual parameters (hour, minute, second) as separate numeric expressions.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleTimeSerial
      Dim dDate As Double, sDate As String
      dDate = TimeSerial(8,30,15)
      sDate = TimeSerial(8,30,15)
      MsgBox dDate,64,"Time as a number"
```

```
      MsgBox sDate,64,"Formatted time"
End Sub
```

TimeValue()

Calculates a serial time value from the specified hour, minute, and second - parameters passed as strings - that represents the time in a single numeric value. This value can be used to calculate the difference between times.

Syntax

TimeValue (Text As String)

Return value

Date

Parameters

Text
Any string expression that contains the time that you want to calculate in the format "HH:MM:SS".

Use the TimeValue function to convert any time into a single value, so that you can calculate time differences.

This TimeValue function returns the type Variant with VarType 7 (Date), and stores this value internally as a double-precision number between 0 and 0.9999999999.

As opposed to the DateSerial or the DateValue function, where serial date values result in days relative to a fixed date, you can calculate with the values that are returned by the TimeValue function, but you cannot evaluate them.

In the TimeSerial function, you can pass individual parameters (hour, minute, second) as separate numeric expressions. For the TimeValue function, however, you can pass a string as a parameter containing the time.

Error Codes

5 Invalid procedure call

13 Type mismatch

Example

```
Sub ExampleTimerValue
      Dim daDT as Date
      Dim a1, b1, c1, a2, b2, c2 as String
      a1 = "start time"
      b1 = "end time"
      c1 = "total time"
      a2 = "8:34"
      b2 = "18:12"
      daDT = TimeValue(b2) - TimeValue(a2)
      c2 = a1 & ": " & a2 & chr(13)
      c2 = c2 & b1 & ": " & b2 & chr(13)
      c2 = c2 & c1 & ": " & trim(Str(Hour(daDT))) & ":" &_
            trim(Str(Minute(daDT))) & ":" & trim(Str(Second(daDT)))
      Msgbox c2
End Sub
```

Trim()

Removes all leading and trailing spaces from a string expression.

Syntax

Trim(Text As String)

Return value

String

Parameters

Text
Any string expression.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleSpaces
      Dim sText2 as String,sText as String,sOut as String
      sText2 = " <*Las Vegas*> "
      sOut = "'"+sText2 +"'"+ Chr(13)
      sText = Ltrim(sText2) REM sText = "<*Las Vegas*> "
      sOut = sOut + "'"+sText +"'" + Chr(13)
      sText = Rtrim(sText2) REM sText = " <*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'" + Chr(13)
      sText = Trim(sText2) REM sText = "<*Las Vegas*>"
      sOut = sOut +"'"+ sText +"'"
      MsgBox sOut
End Sub
```

TwipsPerPixelX

Returns the number of twips that represent the width of a pixel.

Syntax

n = TwipsPerPixelX

Return value

Integer

Example

```
Sub ExamplePixelTwips
      MsgBox "" & TwipsPerPixelX() & " Twips * " & TwipsPerPixelY() & _
            " Twips",0,"Pixel size"
End Sub
```

TwipsPerPixelY

Returns the number of twips that represent the height of a pixel.

Syntax

n = TwipsPerPixelY

Return value

Integer

Example

```
Sub ExamplePixelTwips
      MsgBox "" & TwipsPerPixelX() & " Twips * " & TwipsPerPixelY() & _
            " Twips",0,"Pixel size"
End Sub
```

TypeName()

Returns a string that contains information for a variable. See VarType() function for numeric version.

Syntax

TypeName (Variable)

Return value

String

Parameters

Variable

The variable that you want to determine the type of. You can use the following values:

key word	VarType	Typename
Boolean	11	Boolean variable
Date	7	Date variable
Double	5	Double floating point variable
Integer	2	Integer variable
Long	3	Long integer variable
Object	9	Object variable
Single	4	Single floating-point variable
String	8	String variable
Variant	12	Variant variable (can contain all types specified by the definition)
Empty	0	Variable is not initialized
Null	1	No valid data

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleType
      Dim iVar As Integer
      Dim sVar As String
      Dim siVar As Single
      Dim dVar As Double
      Dim bVar As Boolean
      Dim lVar As Long
      Msgbox TypeName(iVar) & " " & VarType(iVar) & Chr(13) &_
      TypeName(sVar) & " " & VarType(sVar) & Chr(13) &_
      TypeName(siVar) & " " & VarType(siVar) & Chr(13) &_
      TypeName(dVar) & " " & VarType(dVar) & Chr(13) &_
      TypeName(bVar) & " " & VarType(bVar) & Chr(13) &_
      TypeName(lVar) & " " & VarType(lVar),0,_
            "Some types in OpenOffice.org Basic"
End Sub
```

Ubound()

Returns the upper boundary of an array.

Syntax

UBound (ArrayName [, Dimension])

Return value

Integer

Parameters

ArrayName
Name of the array for which you want to determine the upper (**Ubound**) or the lower (**LBound**) boundary.

[Dimension]
Integer that specifies which dimension to return the upper(**Ubound**) or lower (**LBound**) boundary for. If no value is specified, the boundary of the first dimension is returned.

Error Codes

5 Invalid procedure call

9 Subscript out of range

Example

```
Sub ExampleUboundLbound
      Dim sVar(10 to 20) As String
      Print LBound(sVar())
      Print UBound(sVar())
End Sub

Sub ExampleUboundLbound2
      Dim sVar(10 to 20,5 To 70) As String
      Print LBound(sVar()) REM Returns 10
      Print UBound(sVar()) REM Returns 20
      Print LBound(sVar(),2) REM Returns 5
      Print UBound(sVar(),2) REM Returns 70
End Sub
```

Ucase()

Converts lowercase characters in a string to uppercase.

See also: LCase Function

Syntax:

UCase (Text As String)

Return value:

String

Parameters

Text
Any string expression that you want to convert.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleLUCase
      Dim sVar As String
      sVar = "Las Vegas"
      Print LCase(sVar) REM returns "las vegas"
      Print UCase(sVar) REM returns "LAS VEGAS"
End Sub
```

Val()

Converts a string to a numeric expression.

Syntax

Val (Text As String)

Return value

Double

Parameters

Text
String that represents a number.

Using the Val function, you can convert a string that represents numbers into numeric expressions. This is the inverse of the **Str** function. If only part of the string contains numbers, only the first appropriate characters of the string are converted. If the string does not contain any numbers, the **Val** function returns the value 0.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleVal
      MsgBox Val("123.123")
      MsgBox Val("A123.123")
End Sub
```

VarType()

Returns a numeric value that contains information for a variable. See the Typename() function for a string version.

Syntax

VarType (Variable)

Return value

Integer

Parameters

Variable

The variable that you want to determine the type of. You can use the following values:

key word	VarType	Typename
Boolean	11	Boolean variable
Date	7	Date variable
Double	5	Double floating point variable
Integer	2	Integer variable
Long	3	Long integer variable
Object	9	Object variable
Single	4	Single floating-point variable
String	8	String variable
Variant	12	Variant variable (can contain all types specified by the definition)
Empty	0	Variable is not initialized
Null	1	No valid data

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleType
      Dim iVar As Integer
      Dim sVar As String
      Dim siVar As Single
      Dim dVar As Double
      Dim bVar As Boolean
      Dim lVar As Long
      Msgbox TypeName(iVar) & " " & VarType(iVar) & Chr(13) &_
      TypeName(sVar) & " " & VarType(sVar) & Chr(13) &_
      TypeName(siVar) & " " & VarType(siVar) & Chr(13) &_
      TypeName(dVar) & " " & VarType(dVar) & Chr(13) &_
      TypeName(bVar) & " " & VarType(bVar) & Chr(13) &_
      TypeName(lVar) & " " & VarType(lVar),0,_
            "Some types in OpenOffice.org Basic"
```

```
End Sub
```

Wait

Interrupts the program execution for the amount of time that you specify in milliseconds.

Syntax

Wait millisec

Parameters

millisec
Numeric expression that contains the amount of time (in milliseconds) to wait before the program is executed.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleWait
      Dim lTick As Long
      lTick = GetSystemTicks()
      Wait 2000
      lTick = (GetSystemTicks() - lTick)
      MsgBox "" & lTick & " Ticks" ,0,"The pause lasted"
End Sub
```

WeekDay()

Returns the number corresponding to the weekday represented by a serial date number that is generated by the DateSerial or the DateValue function.

Syntax

WeekDay (Number)

Return value

Integer

Parameters:

Number

Integer expression that contains the serial date number that is used to calculate the day of the week (1-7).

The following example determines the day of the week using the WeekDay function when you enter a date.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleWeekDay
      Dim sDay As String
      REM Return and display the day of the week
      Select Case WeekDay( Now )
            Case 1
                  sDay="Sunday"
            Case 2
                  sDay="Monday"
            Case 3
                  sDay="Tuesday"
            Case 4
                  sDay="Wednesday"
            Case 5
                  sDay="Thursday"
            Case 6
                  sDay="Friday"
            Case 7
                  sDay="Saturday"
      End Select
MsgBox "" + sDay,64,"Today is"
End Sub
```

While...Wend

When a program encounters a While statement, it tests the condition. If the condition is False, the program continues directly following the Wend statement. If the condition is True, the loop is executed until the program finds Wend and then jumps back to the **While** statement. If the condition is still True, the loop is executed again.

Unlike the Do...Loop statement, you cannot cancel a **While...Wend** loop with Exit. Never exit a While...Wend loop with GoTo, since this can cause a run-time error.

A Do...Loop is more flexible than a While...Wend.

Syntax

While Condition [Statement] Wend

Example

```
Sub ExampleWhileWend
      Dim stext As String
      Dim iRun As Integer
      sText ="This is a short text"
      iRun = 1
      While iRun < Len(sText)
      If Mid(sText,iRun,1 )<> " " Then
            Mid( sText ,iRun, 1, Chr( 1 + Asc( Mid(sText,iRun,1 )) )
      End If
      iRun = iRun + 1
      Wend
      MsgBox sText,0,"Text encoded"
End Sub
```

With

Sets an object as the default object. Unless another object name is declared, all properties and methods refer to the default object until the End With statement is reached.

Syntax

With Object Statement block End With

Parameters

Use **With** and **End With** if you have several properties or methods for a single object.

Write

Writes data to a sequential file.

Syntax

```
Write [#]FileName, [Expressionlist]
```

Parameters

FileName
Any numeric expression that contains the file number that was set by the Open statement for the respective file.

Expressionlist
Variables or expressions that you want to enter in a file, separated by commas.

If the expression list is omitted, the **Write#** statement appends an empty line to the file.

To add an expression list to a new or an existing file, the file must be opened in the **Output** or **Append** mode.

The **Write#** statement enters data that is enclosed by quotation marks and separated by commas into a file. You do not need to use delimiters in the list. The end of a file created with the **Write#** statement is indicated by a line end symbol.

Example

```
Sub ExampleWrite
      Dim iCount As Integer
      Dim sValue As String
      iCount = Freefile
      Open "C:\data.txt" for OutPut as iCount
      sValue = "Hamburg"
      Write #iCount,sValue,200
      sValue = "New York"
      Write #iCount,sValue,300
      sValue = "Miami"
      Write #iCount,sValue,450
      Close #iCount
End Sub
```

Xor

Performs a logical Exclusive-Or combination of two expressions.

Syntax

Result = Expression1 Xor Expression2

Parameters

Result
Any numeric variable that contains the result of the combination.

Expression1, Expression2
Any numeric expressions that you want to combine.

A logical Exclusive-Or conjunction of two Boolean expressions returns the value True only if both expressions are different from each other.

A bitwise Exclusive-Or conjunction returns a bit if the corresponding bit is set in only one of the two expressions.

Example

```
Sub ExampleXor
      Dim vA as Variant, vB as Variant, vC as Variant, vD as Variant
      Dim vOut as Variant
      vA = 10: vB = 8: vC = 6: vD = Null
      vOut = vA > vB Xor vB > vC REM returns 0
      vOut = vB > vA Xor vB > vC REM returns -1
      vOut = vA > vB Xor vB > vD REM returns -1
      vOut = (vB > vD Xor vB > vA) REM returns 0
      vOut = vB Xor vA REM returns 2
End Sub
```

Year()

Returns the year from a serial date number that is generated by the DateSerial or the DateValue function.

Syntax

Year (Number)

Return value

Integer

Parameters

Number
Integer expression that contains the serial date number that is used to calculate the year.

This function is the opposite of the **DateSerial** function, and returns the year of a serial date. For example, the expression:

Print Year(DateSerial(1994, 12, 20))

returns the value 1994.

Error Codes

5 Invalid procedure call

Example

```
Sub ExampleYear
      MsgBox "" & Year(Now) ,64,"Current year"
End sub
```