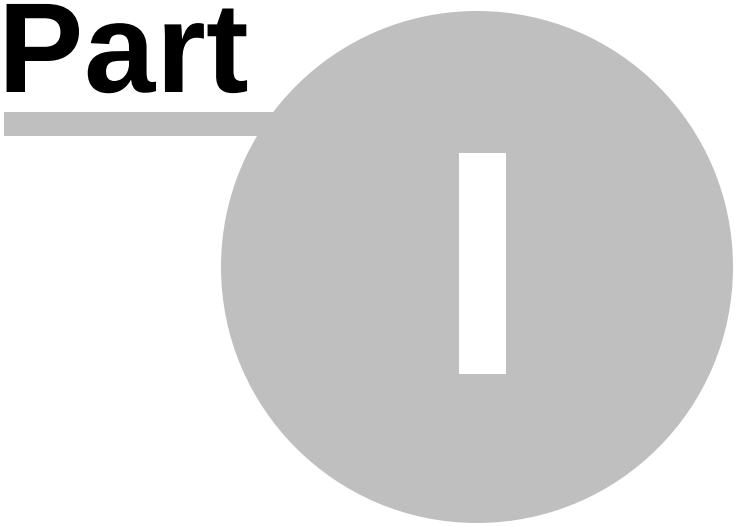
Sax Basic Help

Table of Contents

Top Level Intro

This page is printed before a new top-level chapter starts



1 Sax Basic Language

1.1 Sax Basic Language

The Sax Basic Language provides the core language definition. It is Visual Basic for Applications(TM) compatible.

Language reference by group:

- Declaration, Data Type, Assignment
- Flow Control, Error Handling
- Conversion, Variable Info
- Constant
- Math, String, Object, Time/Date
- File
- User Input, User Dialog, Dialog Function
- DDE
- Settings
- Miscellaneous
- Operator

Copyright Trademarks

1.2 Functions By Groups

Declaration:
#Reference, #Uses, Attribute, Class Module, Code Module, Const, Declare, Deftype, Dim, Enum...End Enum, Function...End Function, Object Module, Option, Private, Property...End Property, Public, ReDim , Static, Sub...End Sub, Type...End Type. WithEvents

Data Type:
Any, Boolean, Byte, Currency, Date, Decimal, Double, Integer, Long, Object, PortInt, Single, String, String*n, Variant, obj type, user enum, user type.

Assignment:
Erase, Let, LSet, RSet, Set.

Flow Control:
Call, CallByName, Do...Loop, End, Exit, For...Next, For Each...Next, GoTo, If...ElseIf...Else...End If, MacroRun, MacroRunThis, Select Case...End Select, Stop, While...Wend.

Error Handling:
Err, Error, On Error, Resume.

Conversion:
Array, CBool, CByte, CCur, CDate, CDec, CDbl, CInt, CLng, CSng, CStr, CVar, CVDate, CVErr, Val.

Variable Info:
IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject, LBound, TypeName, UBound, VarType.

Constant:

Empty, False, Nothing, Null, True, Win16, Win32.

Math:
Abs, Atn, Cos, Exp, Fix, Int, Log, Randomize, Rnd, Round, Sgn, Sin, Sqr, Tan.

String:
Asc, AscB, AscW, Chr, ChrB, ChrW, Format, Hex, InStr, InStrB, InStrRev, Join, LCase, Left, LeftB, Len , LenB, LTrim, Mid, MidB, Oct, Replace, Right, RightB, RTrim, Space, Split, String, Str, StrComp, StrConv, StrReverse, Trim, UCase.

Object:
CreateObject, GetObject, Me, With...End With.

Time/Date:
Date, DateAdd, DateDiff, DatePart, DateSerial, DateValue, Day, Hour, Minute, Month, MonthName, Now , Second, Time, Timer, TimeSerial, TimeValue, Weekday, WeekdayName, Year.

File:
ChDir, ChDrive, Close, CurDir, Dir, EOF, FileAttr, FileCopy, FileDateTime, FileLen, FreeFile, Get, GetAttr, Input, Input, Kill, Line Input, Loc, Lock, LOF, MkDir, Name, Open, Print, Put, Reset, RmDir, Seek, Seek, SetAttr, Unlock, Write.

User Input:
Dialog, GetFilePath, InputBox, MsgBox. ShowPopupMenu

User Dialog:
Begin Dialog...End Dialog, CancelButton, CheckBox, ComboBox, DropListBox, GroupBox, ListBox, MultiListBox, OKButton, OptionButton, OptionGroup, Picture, PushButton, Text, TextBox.

Dialog Function:
Dialog Func, DlgControlId, DlgCount, DlgEnable, DlgEnd, DlgFocus, DlgListBoxArray, DlgName, DlgNumber, DlgSetPicture, DlgText, DlgType, DlgValue, DlgVisible.

DDE:
DDEExecute, DDEInitiate, DDEPoke, DDERequest, DDETerminate, DDETerminateAll.

Settings:
DeleteSetting, GetAllSettings, GetSetting, SaveSetting

Miscellaneous:
AboutWinWrapBasic, AppActivate, Attribute, Beep, CallersLine, Choose, Clipboard, Command, Debug. Print, DoEvents, Environ, Eval, Iif, KeyName, MacroDir, QBColor, Rem, RGB, SendKeys, Shell, Wait.

Operator:
Operators: +, -, ^, *, /, \, Mod, +, -, &, =, <>, <, >, <=, >=, Like. Not, And, Or, Xor, Eqv, Imp, Is.

1.3 Language

1.3.1 Modules

1.3.1.1 Class

Group: Declaration

Description:
A class module implements an ActiveX Automation object.

- Has a set of Public procedures accessible from other macros and modules.
- These public symbols are accessed via an object variable.
- Public Consts, Types, arrays, fixed length strings are not allowed.

- A class module is similar to a object module except that no instance is automatically created.
- To create an instance use:

```
Dim Obj As classname
Set Obj = New classname
```

See Also: Code Module, Object Module, Uses.

Example:

```
'A.BAS
'#Uses "File.CLS"
Sub Main
    Dim File As New File
    File.Attach "C:\AUTOEXEC.BAT"
    Debug.Print File.ReadLine
End Sub

'File.CLS
'File|New Module|Class Module
'Edit|Properties|Name=File
Option Explicit
Dim FN As Integer
Public Sub Attach(FileName As String)
    FN = FreeFile
    Open FileName For Input As #FN
End Sub
Public Sub Detach()
    If FN <> 0 Then Close #FN
    FN = 0
End Sub
Public Function ReadLine() As String
    Line Input #FN,ReadLine
End Function

Private Sub Class_Initialize()
    Debug.Print "Class_Initialize"
End Sub

Private Sub Class_Terminate()
    Debug.Print "Class_Terminate"
    Detach
End Sub
```

1.3.1.2 Code

Group: Declaration

Description:
A Code module implements a code library.

- Has a set of Public procedures accessible from other macros and modules.
- The public symbols are accessed directly.

See Also: Class Module, Object Module, Uses.

Example:

```
'A.BAS
'#Uses "Module1.BAS"
Sub Main
    Debug.Print Value "'Hello"
End Sub

'Module1.BAS
'File|New Module|Code Module
'Edit|Properties|Name=Module1
Option Explicit
Private mValue As String
Property Get Value() As String
    Value = mValue
End Property
'this sub is called when the module is first loaded
Private Sub Main
    mValue = "Hello"
End Sub
```

1.3.1.3 Object

Group: Declaration

Description:
An object module implements an ActiveX Automation object.

- It has a set of Public procedures accessible from other macros and modules.
- These public symbols are accessed via the name of the object module or an object variable.
- Public Consts, Types, arrays, fixed length strings are not allowed.
- An object module is similar to a class module except that one instance is automatically created. That instance has the same name as the object module's name.
- To create additional instances use:

```
Dim Obj As objectname
Set Obj = New objectname
```

See Also: Class Module, Code Module, Uses.

Example:

```
'A.BAS
'#Uses "System.OBM"
Sub Main
    Debug.Print Hex(System.Version)
End Sub

'System.OBM
'File|New Module|Object Module
'Edit|Properties|Name=System
Option Explicit
Declare Function GetVersion16 Lib "Kernel" _
    Alias "GetVersion" () As Long
Declare Function GetVersion32 Lib "Kernel32" _
    Alias "GetVersion" () As Long

Public Function Version() As Long
    If Win16 Then
        Version = GetVersion16
    Else
        Version = GetVersion32
    End If
End Function
```

1.3.2 Data Types

1.3.2.1 Any

Group: Data Type

Description:
Any variable expression (Declare only).

1.3.2.2 Boolean

Group: Data Type

Description:
A True or False value.

1.3.2.3 Byte

Group: Data Type

Description:
An 8 bit unsigned integer value.

1.3.2.4 Currency

Group: Data Type

Description:
A 64 bit fixed point real. (A twos complement binary value scaled by 10000.)

1.3.2.5 Date

Group: Data Type

Description:
A 64 bit real value. The whole part represents the date, while the fractional part is the time of day. (December 30, 1899 = 0.) Use #date# as a literal date value in an expression.

1.3.2.6 Decimal

Group: Data Type

Description:
Win32 only. A 96 bit scaled real value. Decimal is not a valid variable type, but Variant variables can contain decimal values (see CDec). A decimal number is of the form: s*m*10^-p where

- s - sign (+1 or -1)
- m - mantissa, unsigned binary value of 96 bits (0 to 79,228,162,514,264,337,593,543,950,335)
- p - scaling power (0 to +28)

1.3.2.7 Double

Group: Data Type

Description:
A 64 bit real value.

1.3.2.8 Integer

Group: Data Type

Description:
A 16 bit integer value.

1.3.2.9 Long

Group: Data Type

Description:
A 32 bit integer value.

1.3.2.10 Object

Group: Data Type

Description:
An object reference value. (see Objects)

1.3.2.11 PortInt

Group: Data Type

Description:
A portable integer value.

- For Win16: A 16 bit integer value.
- For Win32: A 32 bit integer value.

1.3.2.12 Single

Group: Data Type

Description:
A 32 bit real value.

1.3.2.13 String

Group: Data Type

Description:
An arbitrary length string value. Some useful string constants are predefined:

- vbNullChar - same as Chr(0)
- vbCrLf - same as Chr(13) & Chr(10)
- vbCr - same as Chr(13)
- vbLf - same as Chr(10)
- vbBack - same as Chr(8)
- vbFormFeed - same as Chr(12)
- vbTab - same as Chr(9)
- vbVerticalTab - same as Chr(11)

1.3.2.14 String*n

Group: Data Type

Description:
A fixed length (n) string value.

1.3.2.15 UserDialog

Group: Data Type

Description:
A usertype defined by Begin Dialog UserDialog.

1.3.2.16 Variant

Group: Data Type

Description:
An empty, numeric, currency, date, string, object, error code, null or array value.

1.3.2.17 usertype

usertype definition

User defined types are defined with Type.

1.3.3 Constants

1.3.3.1 Empty

Group: Constant

Description:
A variantvar that does not have any value.

1.3.3.2 False

Group: Constant

Description:

A condexpr is false when its value is zero. A function that returns False returns the value 0.

1.3.3.3 Nothing

Group: Constant

Description:
An objexpr that does not refer to any object.

1.3.3.4 Null

Group: Constant

Description:
A variant expression that is null. A null value propagates through an expression causing the entire expression to be Null. Attempting to use a Null value as a string or numeric argument causes a run-time error. A Null value prints as "#NULL#".

Example:

```
Sub Main
    X = Null
    Debug.Print X = Null  '#NULL#
    Debug.Print IsNull(X) 'True
End Sub
```

1.3.3.5 True

Group: Constant

Description:
A conditional expression is True when its value is non-zero. A function that returns True returns the value -1.

1.3.3.6 Win16

Group: Constant

Description:
True if running in 16 bits. False if running in 32 bits.

1.3.3.7 Win32

Group: Constant

Description:
True if running in 32 bits. False if running in 16 bits.

1.3.4 Keywords

1.3.4.1 Friend

Group: Declaration

Description:
Friend Functions, Propertys and Subs in a module are available in all other macros/modules that access it. Friends are not accessible via Object variables.

1.3.4.2 New

Syntax:
Set objvar = objexpr
-or-
Set objvar = New objtype

Group: Assignment

Description:
Form 1: Set objvar's object reference to the object reference of objexpr.

Form 2: Set objvar's object reference to the a new instance of objtype.

The Set instruction is how object references are assigned.

Example:

```
Sub Main
   Dim App As Object
   Set App = CreateObject("WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
   Set App = Nothing
   App.Quit       ' run-time error (no object)
End Sub
```

1.3.4.3 Private

Group: Declaration

Description:
Private Consts, Declares, Functions, Propertys, Subs and Types are only available in the current macro/ module.

1.3.4.4 Public

Group: Declaration

Description:
Public Consts, Declares, Functions, Propertys, Subs and Types in a module are available in all other macros/modules that access it.

1.3.5 Functions and Instructions

1.3.5.1 AboutWinWrapBasic

Syntax:
AboutWinWrapBasic [Timeout]

Group: Miscellaneous

Description:
Show the WinWrap Basic about box.

Parameter	Description
Timeout	This numeric value is the maximum number of seconds to show the about box. A value less than or equal to zero displays the about box until the user closes it. If this value is omitted then a three second timeout is used.

Example:

```
Sub Main
    AboutWinWrapBasic
End Sub
```

1.3.5.2 Abs

Syntax:
Abs(Num)

Group: Math

Description:
Return the absolute value.

Parameter	Description
Num	Return the absolute value of this numeric value. If this value is Null then Null is returned.

See Also: Sgn.

Example:

```
Sub Main
    Debug.Print Abs(9)  ' 9
    Debug.Print Abs(0)  ' 0
    Debug.Print Abs(-9) ' 9
End Sub
```

1.3.5.3 AppActivate

Syntax:
AppActivate Title$
-or-
AppActivate TaskID

Group: Miscellaneous

Description:
Form 1: Activate the application top-level window titled Title$. If no window by that title exists then the first window with at title that starts with Title$ is activated. If no window matches then an error occurs.

Form 2: Activate the application top-level window for task TaskID. If no window for that task exists then an error occurs.

Parameter	Description
Title$	The name shown in the title bar of the window.
TaskID	This numeric value is the task identifier.

See Also: SendKeys, Shell( ).

Example:

```
Sub Main
    ' make ProgMan the active application
    AppActivate "Program Manager"
End Sub
```

1.3.5.4 Array

Syntax:
Array([expr[, ...]])

Group: Conversion

Description:
Return a variant value array containing the exprs.

Example:

```
Sub Main
    X = Array(0,1,4,9)
    Debug.Print X(2) ' 4
End Sub
```

1.3.5.5 Asc

Syntax:
Asc(S$)

Group: String

Description:
Return the ASCII value.

Note: A similar function, AscB, returns the first byte in S$. Another similar function, AscW, returns the Unicode number.

Parameter	Description
S$	Return the ASCII value of the first char in this string value.

See Also: Chr$( ).

Example:

```
Sub Main
    Debug.Print Asc("A") ' 65
End Sub
```

1.3.5.6 Atn

Syntax:
Atn(Num)

Group: Math

Description:
Return the arc tangent. This is the number of radians. There are 2*Pi radians in a full circle.

Parameter	Description
Num	Return the arc tangent of this numeric value.

See Also: Cos, Sin, Tan.

Example:

```
Sub Main
    Debug.Print Atn(1)*4 ' 3.1415926535898
End Sub
```

1.3.5.7 Attribute

Syntax:
Attribute attributename = value
Attribute varname.attributename = value
Attribute procname.attributename = value

Group: Declaration

Description:
All attribute definitions and statements are ignored except for:

- Form 1: Module level attribute

  Attribute VB_Name = "name"
  Attribute VB_GlobalNameSpace = bool
  Attribute VB_Creatable = bool
  Attribute VB_PredeclaredId = bool
  Attribute VB_Exposed = bool
  Attribute VB_HelpID = int
  Attribute VB_Description = "text"

  VB_Name - Declares the name of the class module or object module.
  VB_GlobalNameSpace - Declares the class module as a global class. (ignored)
  VB_Creatable - Declares the module as creatable (True), non-creatable (False). (ignored)
  VB_PredeclaredId - Declares the module as a predeclared identifier (True). (ignored)
  VB_Exposed - Declares the module as public (True). (ignored)
  VB_HelpID - Declares the module's help context displayed by the object browser.
  VB_Description - Declares the module's help text displayed by the object browser.

- Form 2: Macro/Module level variable attribute

  Public varname As Type
  Attribute varname.VB_VarUserMemId = 0
  Attribute varname.VB_VarHelpID = int
  Attribute varname.VB_VarDescription = "text"

  VB_VarUserMemID - Declares Public varname as the default property for a class module or object module.
  VB_VarHelpID - Declares the variable's help context displayed by the object browser.
  VB_VarDescription - Declares the variable's help text displayed by the object browser.

- Form 3: User defined procedure attribute

  [Sub | Function | Property [Get|Let|Set]] procname ...
  Attribute procname.VB_UserMemId = 0
  Attribute procname.VB_HelpID = int
  Attribute procname.VB_Description = "text"
  ...
  End [Sub | Function | Property]

  VB_UserMemID - Declares Property procname as the default property for a class module or object module.
  VB_HelpID - Declares the procedure's help context displayed by the object browser.
  VB_Description - Declares the procedure's help text displayed by the object browser.

HelpFile:
Each macro/module can define the HelpFile for the object browser:

'#HelpFile "helpfile"

where "helpfile" is a full path to the help file associated with the help text and help context.

1.3.5.8 Beep

Syntax:
Beep

Group: Miscellaneous

Description:
Sound the bell.

Example:

```
Sub Main
   Beep ' beep the bell
End Sub
```

1.3.5.9 Begin Dialog

Syntax:

```
Begin Dialog UserDialog [X, Y,] DX, DY[, Title$] _
        [, .dialogfunc]
   User Dialog Item
   [User Dialog Item]...
End Dialog
```

Group: User Dialog

Description:
Define a UserDialog type to be used later in a Dim As UserDialog statement.

Parameter	Description
X	This numeric value is the distance from the left edge of the screen to the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.
Y	This numeric value is the distance from the top edge of the screen to the top edge of the dialog box. It is measured in 1/12 ths of the average character width for the dialog's font. If this is omitted then the dialog will be centered.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Title$	This string value is the title of the user dialog. If this is omitted then there is no title.
dialogfunc	This is the function name that implements the DialogFunc for this UserDialog. If this is omitted then the UserDialog doesn't have a dialogfunc.
User Dialog Item	One of: CancelButton, CheckBox, ComboBox, DropListBox, GroupBox, ListBox, MultiListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox.

See Also: Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.10 Call

Syntax:
Call name[(arglist)]
-or-
name [arglist]

Group: Flow Control

Description:
Evaluate the arglist and call subroutine (or function) name with those values. Sub (or function) name must be previously defined by either a Sub, Function or Property definition. If name is a function then the result is discarded. If Call is omitted and name is a subroutine then the arglist must not be enclosed in parens.

See Also: Declare, Sub.

Example:

```
Sub Show(Title$,Value)
   Debug.Print Title$;"=";Value
End Sub

Sub Main
   Call Show("2000/9",2000/9) ' 222.2222222222
   Show "1<2",1<2             'True
End Sub
```

1.3.5.11 CallByName

Syntax:
CallByName(Obj,ProcName,CallType,[expr[, ...]])

Group: Flow Control

Description:
Call an Obj's method/property, ProcName, by name. Pass the exprs to the method/property.

Parameter	Description
Obj	Call the method/property for this object reference.
ProcName	This string value is the name of the method/property to be called.
CallType	Type of method/property call. See table below.
expr	These expressions are passed to the obj's method/property.

CallType	Value	Effect
vbMethod	1	Call or evaluate the method.
vbGet	2	Evaluate the property's value.

vbLet	4	Assign the property's value.
vbSet	8	Set the property's reference.

Example:

```
Sub Main
   On Error Resume Next
   CallByName Err, "Raise", vbMethod, 1
   Debug.Print CallByName(Err, "Number", vbGet) ' 1
End Sub
```

1.3.5.12 CallersLine

Syntax:
CallersLine[(Depth)]

Group: Miscellaneous

Description:
Return the caller's line as a text string.

The text format is: "[macroname|subname#linenum] linetext".

Parameter	Description
Depth	This integer value indicates how deep into the stack to get the caller's line. If Depth = -1 then return the current line. If Depth = 0 then return the calling subroutine's current line, etc.. If Depth is greater than or equal to the call stack depth then a null string is returned. If this value is omitted then the depth is 0.

Example:

```
Sub Main
   A
End Sub
Sub A
   Debug.Print CallersLine "'[(untitled 1)|Main#  2] A"
End Sub
```

1.3.5.13 CancelButton

Syntax:
CancelButton X, Y, DX, DY[, .Field]

Group: User Dialog

Description:
Define a cancel button item. Pressing the Cancel button from a Dialog instruction causes a run-time error. (Dialog( ) function call returns 0.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character

height for the dialog's font.

Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this is omitted then the field name is "Cancel".

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,30,"Please push the Cancel button"
      OKButton 40,90,40,20
      CancelButton 110,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for cancel)
   Debug.Print "Cancel was not pressed"
End Sub
```

1.3.5.14 CBool

Syntax:
CBool(Num|$)

Group: Conversion

Description:
Convert to a boolean value. Zero converts to False, while all other values convert to True.

Parameter	Description
Num|$	Convert a number or string value to a boolean value.

Example:

```
Sub Main
   Debug.Print CBool(-1) 'True
   Debug.Print CBool(0)  'False
   Debug.Print CBool(1)  'True
End Sub
```

1.3.5.15 CByte

Syntax:
CByte(Num|$)

Group: Conversion

Description:
Convert to a byte value.

Parameter	Description
Num|$	Convert a number or string value to a byte value.

Example:

```
Sub Main
   Debug.Print CByte(1.6) ' 2
End Sub
```

1.3.5.16 CCur

Syntax:
CCur(Num|$)

Group: Conversion

Description:
Convert to a currency value.

Parameter	Description
Num|$	Convert a number or string value to a currency value.

Example:

```
Sub Main
    Debug.Print CCur("1E6") ' 1000000
End Sub
```

1.3.5.17 CDate

Syntax:
CDate(Num|$)
-or-
CVDate(Num|$)

Group: Conversion

Description:
Convert to a date value.

Parameter	Description
Num|$	Convert a number or string value to a date value.

Example:

```
Sub Main
    Debug.Print CDate(2) ' 1/1/00
End Sub
```

1.3.5.18 CDbl

Syntax:
CDbl(Num|$)

Group: Conversion

Description:
Convert to a double precision real.

Parameter	Description
Num|$	Convert a number or string value to a double precision real.

Example:

```
Sub Main
    Debug.Print CDbl("1E6") ' 1000000
End Sub
```

1.3.5.19 CDec

Syntax:
CDec(Num|$)

Group: Conversion

Description:
Win32 only. Convert to a decimal (96 bit scaled real).

Parameter	Description
Num|$	Convert a number or string value to a 96 bit scaled real.

Example:

```
Sub Main
    Debug.Print CDec("1E16")+0.1 ' 10000000000000000.1
End Sub
```

1.3.5.20 ChDir

Syntax:
ChDir Dir$

Group: File

Description:
Change the current directory to Dir$.

Parameter	Description
Dir$	This string value is the path and name of the directory.

See Also: ChDrive, CurDir$( ).

Example:

```
Sub Main
    ChDir "C:\"
    Debug.Print CurDir$() "'C:\"
End Sub
```

1.3.5.21 ChDrive

Syntax:
ChDrive Drive$

Group: File

Description:
Change the current drive to Drive$.

Parameter	Description
Drive$	This string value is the drive letter.

See Also: ChDir, CurDir$( ).

Example:

```
Sub Main
    ChDrive "B"
    Debug.Print CurDir$() "'B:\"
End Sub
```

1.3.5.22 CheckBox

Syntax:
CheckBox X, Y, DX, DY, Title$, .Field[, Options]

Group: User Dialog

Description:
Define a checkbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	The value of the check box is accessed via this field. Unchecked is 0, checked is 1 and grayed is 2.
Options	This numeric value controls the type of check box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Check box is either check or unchecked.
1	Check box is either check, unchecked or grayed, and it switches between checked and unchecked when clicked.
2	Check box is either check, unchecked or grayed, and it cycles through all three states as the button is clicked.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      CheckBox 10,25,180,15,"&Check box",.Check
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.Check = 1
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.Check
End Sub
```

1.3.5.23 Choose

Syntax:
Choose(Index, expr[, ...])

Group: Flow Control

Description:

Return the value of the expr indicated by Index.

Parameter	Description
Index	The numeric value indicates which expr to return. If this value is less than one or greater than the number of exprs then Null is returned.
expr	All expressions are evaluated.

See Also: If, Select Case, Iif( ).

Example:

```
Sub Main
    Debug.Print Choose(2,"Hi","there") "there"
End Sub
```

1.3.5.24 Chr

Syntax:
Chr[$](Num)

Group: String

Description:
Return a one char string for the ASCII value.

Note: A similar function, ChrB, returns a single byte ASCII string. Another similar function, ChrW, returns a single char Unicode string.

Parameter	Description
Num	Return one char string for this ASCII numeric value.

See Also: Asc( ).

Example:

```
Sub Main
    Debug.Print Chr$(48) "0"
End Sub
```

1.3.5.25 CInt

Syntax:
CInt(Num|$)

Group: Conversion

Description:
Convert to a 16 bit integer. If Num|$ is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
Num|$	Convert a number or string value to a 16 bit integer.

Example:

```
Sub Main
    Debug.Print CInt(1.6) ' 2
End Sub
```

1.3.5.26 Clipboard

Syntax:

Clipboard Text$
-or-
Clipboard[$][( )]

Group: Miscellaneous

Description:
Form 1: Set the clipboard to Text$. This is like the Edit|Copy menu command.

Form 2: Return the text in the clipboard.

Parameter	Description
Text$	Put this string value into the clipboard.

Example:

```
Sub Main
    Debug.Print Clipboard$()
    Clipboard "Hello"
    Debug.Print Clipboard$() "'Hello"
End Sub
```

1.3.5.27 CLng

Syntax:
CLng(Num|$)

Group: Conversion

Description:
Convert to a 32 bit long integer. If Num|$ is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
Num|$	Convert a number or string value to a 32 bit integer.

Example:

```
Sub Main
    Debug.Print CLng(1.6) ' 2
End Sub
```

1.3.5.28 Close

Syntax:
Close [[#]StreamNum][, ...]

Group: File

Description:
Close StreamNums.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros. If this is omitted then all open streams for the current macro/ module are closed.

See Also: Open, Reset.

Example:

```
Sub Main
   ' read the first line of XXX and print it
   Open "XXX" For Input As #1
   Line Input #1,L$
   Debug.Print L$
   Close #1
End Sub
```

1.3.5.29 ComboBox

Syntax:
ComboBox X, Y, DX, DY, StrArray$( ), .Field$[, Options]

Group: User Dialog

Description:
Define a combobox item. Combo boxes combine the functionality of an edit box and a list box.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field$	The value of the combo box is accessed via this field. This is the text in the edit box.
Options	This numeric value controls the type of combo box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	List is not sorted.
2	List is sorted.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Dim combos$(3)
   combos$(0) = "Combo 0"
   combos$(1) = "Combo 1"
   combos$(2) = "Combo 2"
   combos$(3) = "Combo 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      ComboBox 10,25,180,60,combos$(),.combo$
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.combo$ = "none"
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.combo$
End Sub
```

1.3.5.30 Command

Syntax:
Command[$]

Group: Miscellaneous

Description:
Contains the value of the MacroRun parameters.

See Also: MacroRun.

Example:

```
Sub Main
   Debug.Print "Command line parameter is: """;
   Debug.Print Command$;
   Debug.Print """"
End Sub
```

1.3.5.31 Const

Syntax:
[| Private | Public] _
Const name[type] [As Type] = expr[, ...]

Group: Declaration

Description:
Define name as the value of expr. The expr may be refer other constants or built-in functions. If the type of the constants is not specified, the type of expr is used. Constants defined outside a Sub, Function or Property block are available in the entire macro/module.

Private is assumed if neither Private or Public is specified.

Note: Const statement in a Sub, Function or Property block may not use Private or Public.

Example:

```
Sub Main
   Const Pi = 4*Atn(1), e = Exp(1)
   Debug.Print Pi ' 3.14159265358979
   Debug.Print e  ' 2.71828182845905
End Sub
```

1.3.5.32 Cos

Syntax:
Cos(Num)

Group: Math

Description:
Return the cosine.

Parameter	Description
Num	Return the cosine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also: Atn, Sin, Tan.

Example:

```
Sub Main
   Debug.Print Cos(1) ' 0.54030230586814
End Sub
```

1.3.5.33 CreateObject

Syntax:
CreateObject(Class$)

Group: Object

Description:
Create a new object of type Class$. Use Set to assign the returned object to an object variable.

Parameter	Description
Class$	This string value is the application's registered class name. If this application is not currently active it will be started.

See Also: Objects.

Example:

```
Sub Main
   Dim App As Object
   Set App = CreateObject("WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
   Set App = Nothing
   App.Quit       ' run-time error (no object)
End Sub
```

1.3.5.34 CSng

Syntax:
CSng(Num|$)

Group: Conversion

Description:
Convert to a single precision real. If Num|$ is too big (or too small) to fit then an overflow error occurs.

Parameter	Description
Num|$	Convert a number or string value to a single precision real.

Example:

```
Sub Main
    Debug.Print CSng(Sqr(2)) ' 1.4142135381699
End Sub
```

1.3.5.35 CStr

Syntax:
CStr(Num|$)

Group: Conversion

Description:
Convert to a string.

Parameter	Description
Num|$	Convert a number or string value to a string value.

Example:

```
Sub Main
    Debug.Print CStr(Sqr(2)) "'1.4142135623731"
End Sub
```

1.3.5.36 CurDir

Syntax:
CurDir[$]([Drive$])

Group: File

Description:
Return the current directory for Drive$.

Parameter	Description
Drive$	This string value is the drive letter. If this is omitted or null then return the current directory for the current drive.

See Also: ChDir, ChDrive.

Example:

```
Sub Main
    Debug.Print CurDir$()
End Sub
```

1.3.5.37 CVar

Syntax:
CVar(Num|$)

Group: Conversion

Description:

Convert to a variant value.

Parameter	Description
Num|$	Convert a number or string value (or object reference) to a variant value.

Example:

```
Sub Main
    Debug.Print CVar(Sqr(2)) ' 1.4142135623731
End Sub
```

1.3.5.38 CVErr

Syntax:
CVErr(Num|$)

Group: Conversion

Description:
Convert to a variant that contains an error code. An error code can't be used in expressions.

Parameter	Description
Num|$	Convert a number or string value to an error code.

See Also: IsError.

Example:

```
Sub Main
    Debug.Print CVErr(1) ' Error 1
End Sub
```

1.3.5.39 Date

Syntax:
Date[$]

Group: Time/Date

Description:
Return today's date as a date value.

See Also: Now, Time, Timer.

Example:

```
Sub Main
    Debug.Print Date ' example: 1/1/1995
End Sub
```

1.3.5.40 DateAdd

Syntax:
DateAdd(interval, number, dateexpr)

Group: Time/Date

Description:
Return a date value a number of intervals from another date.

Parameter	Description
interval	This string value indicates which kind of interval to add.

number	Add this many intervals. Use a negative value to get an earlier date.
dateexpr	Calculate the new date relative to this date value. If this value is Null then Null is returned.

Interval	Description
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

See Also: DateDiff, DatePart.

Example:

```
Sub Main
    Debug.Print DateAdd("yyyy",1,#1/1/2000#) '1/1/2001
End Sub
```

1.3.5.41 DateDiff

Syntax:
DateDiff(interval, dateexpr1, dateexpr2)

Group: Time/Date

Description:
Return the number of intervals between two dates.

Parameter	Description
interval	This string value indicates which kind of interval to subtract.
dateexpr1	Calculate the from this date value to dateexpr2. If this value is Null then Null is returned.
dateexpr2	Calculate the from dateexpr1 to this date value. If this value is Null then Null is returned.

Interval	Description
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute

s	Second

See Also: DateAdd, DatePart.

Example:

```
Sub Main
    Debug.Print DateDiff("yyyy",#1/1/1990#,#1/1/2000#) ' 10
End Sub
```

1.3.5.42 DatePart

Syntax:
DatePart(interval, dateexpr)

Group: Time/Date

Description:
Return the number from the date corresponding to the interval.

Parameter	Description
interval	This string value indicates which kind of interval to extract.
dateexpr	Get the interval from this date value. If this value is Null then Null is returned.

Interval	Description (return value range)
yyyy	Year (100-9999)
q	Quarter (1-4)
m	Month (1-12)
y	Day of year (1-366)
d	Day (1-31)
w	Weekday (1-7)
ww	Week (1-53)
h	Hour (0-23)
n	Minute (0-59)
s	Second (0-59)

See Also: DateAdd, DateDiff.

Example:

```
Sub Main
    Debug.Print DatePart("yyyy",#1/1/2000#) ' 2000
End Sub
```

1.3.5.43 DateSerial

Syntax:
DateSerial(Year, Month, Day)

Group: Time/Date

Description:
Return a date value.

Parameter	Description
Year	This numeric value is the year (0 to 9999). (0 to 99 are interpreted by the operating system.)

Month	This numeric value is the month (1 to 12).
Day	This numeric value is the day (1 to 31).

See Also: DateValue, TimeSerial, TimeValue.

Example:

```
Sub Main
    Debug.Print DateSerial(2000,7,4) '7/4/2000
End Sub
```

1.3.5.44 DateValue

Syntax:
DateValue(Date$)

Group: Time/Date

Description:
Return the day part of the date encoded as a string.

Parameter	Description
Date$	Convert this string value to the day part of date it represents.

See Also: DateSerial, TimeSerial, TimeValue.

Example:

```
Sub Main
    Debug.Print DateValue("1/1/2000 12:00:01 AM")
        '1/1/2000
End Sub
```

1.3.5.45 Day

Syntax:
Day(dateexpr)

Group: Time/Date

Description:
Return the day of the month (1 to 31).

Parameter	Description
dateexpr	Return the day of the month for this date value. If this value is Null then Null is returned.

See Also: Date( ), Month( ), Weekday( ), Year( ).

Example:

```
Sub Main
    Debug.Print Day(#1/1/1900#) ' 1
    Debug.Print Day(#1/2/1900#) ' 2
End Sub
```

1.3.5.46 DDEExecute

Syntax:
DDEExecute ChanNum, Command$[, Timeout]

Group: DDE

Description:
Send the DDE Execute Command$ string via DDE ChanNum.

Parameter	Description
ChanNum	This is the channel number returned by the DDEInitiate function. Up to 10 channels may be used at one time.
Command$	Send this command value to the server application. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

1.3.5.47 DDEInitiate

Syntax:
DDEInitiate(App$, Topic$)

Group: DDE

Description:
Initiate a DDE conversation with App$ using Topic$. If the conversation is successfully started then the return value is a channel number that can be used with other DDE instructions and functions.

Parameter	Description
App$	Locate this server application.
Topic$	This is the server application's topic. The interpretation of this value is defined by the server application.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

1.3.5.48 DDEPoke

Syntax:
DDEPoke ChanNum, Item$, Data$[, Timeout]

Group: DDE

Description:
Poke Data$ to the Item$ via DDE ChanNum.

Parameter	Description
ChanNum	This is the channel number returned by the DDEInitiate function. Up to 10 channels may be used at one time.

Item$	This is the server application's item. The interpretation of this value is defined by the server application.
Data$	Send this data value to the server application. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   DDEPoke ChanNum,"Group","XXX"
   DDETerminate ChanNum
End Sub
```

1.3.5.49 DDERequest

Syntax:
DDERequest[$](ChanNum, Item$[, Timeout])

Group: DDE

Description:
Request information for Item$. If the request is not satisfied then the return value will be a null string.

Parameter	Description
ChanNum	This is the channel number returned by the DDEInitiate function. Up to 10 channels may be used at one time.
Item$	This is the server application's item. The interpretation of this value is defined by the server application.
Timeout	The command will generate an error if the number of seconds specified by the timeout is exceeded before the command has completed. The default is five seconds.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   Debug.Print DDERequest$(ChanNum,"Groups")
   DDETerminate ChanNum
End Sub
```

1.3.5.50 DDETerminate

Syntax:
DDETerminate ChanNum

Group: DDE

Description:
Terminate DDE ChanNum.

Parameter	Description
ChanNum	This is the channel number returned by the DDEInitiate function. Up to 10 channels may be used at one time.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminate ChanNum
End Sub
```

1.3.5.51 DDETerminateAll

Syntax:
DDETerminateAll

Group: DDE

Description:
Terminate all open DDE channels.

Example:

```
Sub Main
   ChanNum = DDEInitiate("PROGMAN","PROGMAN")
   DDEExecute ChanNum,"[CreateGroup(XXX)]"
   DDETerminateAll
End Sub
```

1.3.5.52 Debug

Syntax:
Debug.Clear
-or-
Debug.Print [expr[; ...][;]]

Group: Miscellaneous

Description:
Form 1: Clear the output window.

Form 2: Print the expr(s) to the output window. Use ; to separate expressions. A num is it automatically converted to a string before printing (just like Str$( )). If the instruction does not end with a ; then a newline is printed at the end.

Example:

```
Sub Main
   X = 4
   Debug.Print "X/2=";X/2 ' 2
   Debug.Print "Start..."; ' don't print a newline
   Debug.Print "Finish" ' print a newline
End Sub
```

1.3.5.53 Declare

Syntax:

```
[ | Private | Public ] _
Declare Sub name Lib "dll name" _
   [Alias "module name"] [([param[, ...]])]
-or-
[ | Private | Public ] _
Declare Function name[type] Lib "dll name" _
```

[Alias "module name"] [([param[, ...]])] [As type[()]]

Group: Declaration

Description:
Interface to a DLL defined subroutine or function. The values of the calling arglist are assigned to the params.

Declare defaults to Public if neither Private or Public is specified.

WARNING! Be very careful when declaring DLL subroutines or functions. If you make a mistake and declare the parementers or result incorrectly then Windows might halt. Save any open documents before testing new DLL declarations.

Err.LastDLLError returns the error code for that last DLL call (Windows 32 bit versions only).

Parameter	Description
name	This is the name of the subroutine or function being defined. If Alias "module name" is omitted then this is the module name, too.
"dll name"	This is the DLL file where the module's code is.
"module name"	This is the name of the module in the DLL file. If this is #number then it is the ordinal number of the module. If it is omitted then name is the module name. The DLL is searched for the specified module name. If this module exists, it is used. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards. (Use "Unicode:module name" to prevent ASCII to Unicode conversion.) If the module does not exist, one or two other module names are tried: 1) For Windows NT only: The module name with a "W" appended is tried. All As String parameters are passed as Unicode to calling the DLL. 2) For Windows NT and Windows 95: The module name with an "A" appended is tried. All As String parameters are converted from Unicode to ASCII prior to calling the DLL and from ASCII to Unicode afterwards. If none of these module names is found a run-time error occurs.
params	A list of zero or more params that are used by the DLL subroutine or function. (Note: A ByVal string's value may be modified by the DLL.)

See Also: Function, Sub, Call.

Example:

```
Declare Function GetActiveWindow& Lib "user32" ()
Declare Function GetWindowTextLengthA& Lib "user32" _
   (ByVal hwnd&)
Declare Sub GetWindowTextA Lib "user32" _
   (ByVal hwnd&, ByVal lpsz$, ByVal cbMax&)

Function ActiveWindowTitle$()
   ActiveWindow = GetActiveWindow()
   TitleLen = GetWindowTextLengthA(ActiveWindow)
   Title$ = Space$(TitleLen)
   GetWindowTextA ActiveWindow,Title$,TitleLen+1
   ActiveWindowTitle$ = Title$
End Function

Sub Main
   Debug.Print ActiveWindowTitle$()
End Sub
```

1.3.5.54 Def

Syntax:

```
Def{Bool|Cur|Date|Dbl|Int|Lng|Obj|Sng|Str|Var} _
   letterrange[, ...]
```

Group: Declaration

Description:

Define untyped variables as:

- DefBool - Boolean
- DefByte - Byte
- DefCur - Currency
- DefDate - Date
- DefDbl - Double
- DefInt - Integer
- DefLng - Long
- DefObj - Object
- DefSng - Single
- DefStr - String
- DefVar - Variant

Parameter	Description
letterrange	letter, or letter-letter: A letter is one of A to Z. When letter-letter is used, the first letter must be alphabetically before the second letter. Variable names that begin with a letter in this range default to declared type. If a variable name begins with a letter not specific in any letterrange then the variable is a Variant. The letterranges are not allowed to overlap.

See Also: Option Explicit.

Example:

```
DefInt A,C-W,Y' integer
DefBool B     ' boolean
DefStr X      ' string
              ' all others are variant

Sub Main
   B = 1         ' B is an boolean
   Debug.Print B ' True
   X = "A"       ' X is a string
   Debug.Print X "A"
   Z = 1         ' Z is a variant (anything)
   Debug.Print Z ' 1
   Z = "Z"
   Debug.Print Z "Z"
End Sub
```

1.3.5.55 DeleteSetting

Syntax:
DeleteSetting AppName$, Section$[, Key$]

Group: Settings

Description:
Delete the settings for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings. If this is omitted then delete the entire section.

Example:

```
Sub Main
   SaveSetting "MyApp","Font","Size",10
   DeleteSetting "MyApp","Font","Size"
End Sub
```

1.3.5.56 Dialog

Syntax:
Dialog dialogvar[, default]
-or-
Dialog(dialogvar[, default])

Group: User Input

Description:
Display the dialog associated with dialogvar. The initial values of the dialog fields are provided by dialogvar. If the OK button or any push button is pressed then the fields in dialog are copied to the dialogvar. The Dialog( ) function returns a value indicating which button was pressed. (See the result table below.)

Parameter	Description
dlgvar	This variable that holds the values of the fields in a dialog. Use .field to access

individual fields in a dialog variable.

default — This numeric value indicates which button is the default button. (Pressing the Enter key on a non-button pushes the default button.) Use -2 to indicate that there is no default button. Other possible values are shown the result table below. If this value is omitted then the first PushButton, OKButton or CancelButton is the default button.

Result	Description
-1	OK button was pressed.
0	Cancel button was pressed.
>0	Nth push button was pressed.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.57 Dim

Syntax:
Dim [WithEvents] name[type][([dim[, ...]])][As [New] type][, ...]

Group: Declaration

Description:
Dimension var array(s) using the dims to establish the minimum and maximum index value for each dimension. If the dims are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any dims. It must be ReDimensioned before it can be used.

See Also: Begin Dialog, Dialog, Option Base, Private, Public, ReDim, Static, WithEvents.

Example:

```
Sub DoIt(Size)
   Dim C0,C1(),C2(2,3)
   ReDim C1(Size) ' dynamic array
   C0 = 1
   C1(0) = 2
   C2(0,0) = 3
   Debug.Print C0;C1(0);C2(0,0) ' 1 2 3
End Sub

Sub Main
   DoIt 1
End Sub
```

1.3.5.58 Dir

Syntax:
Dir[$]([Pattern$][, AttribMask])

Group: File

Description:
Scan a directory for the first file matching Pattern$.

Parameter	Description
Pattern$	This string value is the path and name of the file search pattern. If this is omitted then continue scanning with the previous pattern. Each macro has its own independent search. A path relative to the current directory can be used.
AttribMask	This numeric value controls which files are found. A file with an attribute that matches will be found.

See Also: GetAttr( ).

Example:

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$
      F$ = Dir$()
   Wend
End Sub
```

1.3.5.59 DlgControlId

Syntax:
DlgControlId(DlgItem|$)

Group: Dialog Function

Description:
Return the field's window id.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hello"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      If DlgItem$ = "Hello" Then
         DialogFunc% = True 'do not exit the dialog
      End If
   Case 4 ' Focus changed
      Debug.Print "DlgFocus=""";DlgFocus();"""
      Debug.Print "DlgControlId(";DlgItem$;")=";
      Debug.Print DlgControlId(DlgItem$)
   End Select
End Function
```

1.3.5.60 DlgCount

Syntax:
DlgCount()

Group: Dialog Function

Description:
Return the number of dialog items in the dialog.

This instruction/function must be called directly or indirectly from a dialogfunc.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
      Debug.Print "DlgCount=";DlgCount() ' 3
   End Select
End Function
```

1.3.5.61 DlgEnd

Syntax:
DlgEnd ReturnCode

Group: Dialog Function

Description:
Set the return code for the Dialog Function and close the user dialog.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
ReturnCode	Return this numeric value.

Example:

```
Sub Main
   Begin Dialog UserDialog 210,120,.DialogFunc
      Text 10,10,190,15,"Please push the Close button"
      OKButton 30,90,60,20
      CheckBox 120,90,60,20,"&Close",.CheckBox1
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "CheckBox1"
         DlgEnd 1000
      End Select
   End Select
End Function
```

1.3.5.62 DlgEnable

Syntax:
DlgEnable DlgItem|$[, Enable]
-or-
DlgEnable(DlgItem|$)

Group: Dialog Function

Description:
Instruction: Enable or disable DlgItem|$.

Function: Return True if DlgItem|$ is enabled.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name. **Note:** Use -1 to enable or disable all the dialog items at once.
Enable	It this numeric value is True then enable DlgItem|$. Otherwise, disable it. If this omitted then toggle it.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Disable"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "Disable"
         DlgText DlgItem$,"&Enable"
         DlgEnable "Text",False
         DialogFunc% = True 'do not exit the dialog
      Case "Enable"
         DlgText DlgItem$,"&Disable"
         DlgEnable "Text",True
         DialogFunc% = True 'do not exit the dialog
      End Select
   End Select
End Function
```

1.3.5.63 DlgFocus

Syntax:
DlgFocus DlgItem|$
-or-
DlgFocus[$]()

Group: Dialog Function

Description:
Instruction: Move the focus to this DlgItem|$.

Function: Return the field name which has the focus as a string.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hello"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      If DlgItem$ = "Hello" Then
         MsgBox "Hello"
         DialogFunc% = True 'do not exit the dialog
      End If
   Case 4 ' Focus changed
      Debug.Print "DlgFocus=""";DlgFocus();""""
   End Select
End Function
```

1.3.5.64 DlgListBoxArray

Syntax:
DlgListBoxArray DlgItem|$, StrArray$( )
-or-
DlgListBoxArray(DlgItem|$[, StrArray$( )])

Group: Dialog Function

Description:
Instruction: Set the list entries for DlgItem|$.

Function: Return the number entries in DlgItem|$'s list.

This instruction/function must be called directly or indirectly from a dialogfunc. The DlgItem|$ should refer to a ComboBox, DropListBox, ListBox or MultiListBox.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.
StrArray$( )	Set the list entries of DlgItem|$. This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.

Example:

```
Dim lists$()

Sub Main
   ReDim lists$(0)
   lists$(0) = "List 0"
   Begin Dialog UserDialog 200,119,.DialogFunc
      Text 10,7,180,14,"Please push the OK button"
      ListBox 10,21,180,63,lists(),.list
      OKButton 30,91,40,21
      PushButton 110,91,60,21,"&Change"
   End Dialog
   Dim dlg As UserDialog
   dlg.list = 2
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.list
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Select Case Action%
   Case 2 ' Value changing or button pressed
      If DlgItem$ = "Change" Then
         Dim N As Integer
         N = UBound(lists$)+1
         ReDim Preserve lists$(N)
         lists$(N) = "List " & N
         DlgListBoxArray "list",lists$()
         DialogFunc% = True 'do not exit the dialog
      End If
   End Select
End Function
```

1.3.5.65 DlgName

Syntax:
DlgName[$](DlgItem)

Group: Dialog Function

Description:
Return the field name of the DlgItem number.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem	This numeric value is the dialog item number. The first item is 0, second is 1, etc.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
      For I = 0 To DlgCount()-1
         Debug.Print I;DlgName(I)
      Next I
   End Select
End Function
```

1.3.5.66 DlgNumber

Syntax:
DlgNumber(DlgItem$)

Group: Dialog Function

Description:
Return the number of the DlgItem$. The first item is 0, second is 1, etc.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem$	This string value is the dialog item's field name.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 4 ' Focus changed
      Debug.Print DlgItem$;"=";DlgNumber(DlgItem$)
   End Select
End Function
```

1.3.5.67 DlgSetPicture

Syntax:
DlgSetPicture DlgItem|$, FileName, Type

Group: Dialog Function

Description:
Instruction: Set the file name for DlgItem|$.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.
FileName	Set the file name of DlgItem|$ to this string value.
Type	This numeric value indicates the type of bitmap used. See below.

Type	Effect
0	FileName is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is displayed. Not supported.
+16	Instead of displaying "(missing picture)" a run-time error occurs.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Picture 10,10,180,75,"",0,.Picture
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&View"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "View"
         FileName = GetFilePath("Bitmap","BMP")
         DlgSetPicture "Picture",FileName,0
         DialogFunc% = True 'do not exit the dialog
      End Select
   End Select
End Function
```

1.3.5.68 DlgText

Syntax:
DlgText DlgItem|$, Text
-or-
DlgText[$](DlgItem|$)

Group: Dialog Function

Description:
Instruction: Set the text for DlgItem|$.

Function: Return the text from DlgItem|$.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name. **Note:** Use -1 to access the dialog's title.
Text	Set the text of DlgItem|$ to this string value.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Now"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "Now"
         DlgText "Text",CStr(Now)
         DialogFunc% = True 'do not exit the dialog
      End Select
   End Select
End Function
```

1.3.5.69 DlgType

Syntax:
DlgType[$](DlgItem|$)

Group: Dialog Function

Description:
Return a string value indicating the type of the DlgItem|$. One of: "CancelButton", "CheckBox", "ComboBox", "DropListBox", "GroupBox", "ListBox", "MultiListBox", "OKButton", "OptionButton", "OptionGroup", "PushButton", "Text", "TextBox".

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
      For I = 0 To DlgCount()-1
         Debug.Print I;DlgType(I)
      Next I
   End Select
End Function
```

1.3.5.70 DlgValue

Syntax:
DlgValue DlgItem|$, Value
-or-
DlgValue(DlgItem|$)

Group: Dialog Function

Description:
Instruction: Set the numeric value(s) DlgItem|$.

Function: Return the numeric value(s) for DlgItem|$. (A MultiListBox user dialog item returns an array.)

This instruction/function must be called directly or indirectly from a dialogfunc. The DlgItem|$ should refer to a CheckBox, ComboBox, DropListBox, ListBox, MultiListBox or OptionGroup.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.
Value	Set the text of DlgItem|$ to this numeric value. (A MultiListBox user dialog item uses an array.)

Example:

```
Sub Main
   Begin Dialog UserDialog 150,147,.DialogFunc
      GroupBox 10,7,130,77,"Direction",.Field1
      PushButton 100,28,30,21,"&Up"
      PushButton 100,56,30,21,"&Dn"
      OptionGroup .Direction
         OptionButton 20,21,80,14,"&North",.North
         OptionButton 20,35,80,14,"&South",.South
         OptionButton 20,49,80,14,"&East",.East
         OptionButton 20,63,80,14,"&West",.West
      OKButton 10,91,130,21
      CancelButton 10,119,130,21
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg
   MsgBox "Direction=" & dlg.Direction
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "Up"
         DlgValue "Direction",0
         DialogFunc% = True 'do not exit the dialog
      Case "Dn"
         DlgValue "Direction",1
         DialogFunc% = True 'do not exit the dialog
      End Select
   End Select
End Function
```

1.3.5.71 DlgVisible

Syntax:
DlgVisible DlgItem|$[, Visible]
-or-
DlgVisible(DlgItem|$)

Group: Dialog Function

Description:
Instruction: Show or hide DlgItem|$.

Function: Return True if DlgItem|$ is visible.

This instruction/function must be called directly or indirectly from a dialogfunc.

Parameter	Description
DlgItem|$	If this is a numeric value then it is the dialog item number. The first item is 0, second is 1, etc. If this is a string value then it is the dialog item's field name.
Enable	It this numeric value is True then show DlgItem|$. Otherwise, hide it. If this

omitted then toggle it.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120,.DialogFunc
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,40,180,15,.Text
      OKButton 30,90,60,20
      PushButton 110,90,60,20,"&Hide"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub

Function DialogFunc%(DlgItem$, Action%, SuppValue%)
   Debug.Print "Action=";Action%
   Select Case Action%
   Case 1 ' Dialog box initialization
      Beep
   Case 2 ' Value changing or button pressed
      Select Case DlgItem$
      Case "Hide"
         DlgText DlgItem$,"&Show"
         DlgVisible "Text",False
         DialogFunc% = True 'do not exit the dialog
      Case "Show"
         DlgText DlgItem$,"&Hide"
         DlgVisible "Text",True
         DialogFunc% = True 'do not exit the dialog
      End Select
   End Select
End Function
```

1.3.5.72 Do

Syntax:

```
Do
   statements
Loop
-or-
Do {Until|While} condexpr
   statements
Loop
-or-
Do
   statements
Loop {Until|While} condexpr
```

Group: Flow Control

Description:

Form 1: Do statements forever. The loop can be exited by using Exit or Goto.

Form 2: Check for loop termination before executing the loop the first time.

Form 3: Execute the loop once and then check for loop termination.

Loop Termination:

- Until condexpr: Do statements until condexpr is True.
- While condexpr: Do statements while condexpr is True.

See Also: For, For Each, Exit Do, While.

Example:

```
Sub Main
   I = 2
   Do
      I = I*2
   Loop Until I > 10
   Debug.Print I ' 16
End Sub
```

1.3.5.73 DoEvents

Syntax:
DoEvents

Group: Miscellaneous

Description:
This instruction allows other applications to process events.

Example:

```
Sub Main
   DoEvents ' let other apps work
End Sub
```

1.3.5.74 DropListBox

Syntax:
DropListBox X, Y, DX, DY, StrArray$( ), .Field[, Options]

Group: User Dialog

Description:
Define a drop-down listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The value of the drop-down list box is accessed via this field. It is the index of

the StrArray$( ) var.

Options This numeric value controls the type of drop-down list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text box is not editable and list is not sorted.
1	Text box is editable and list is not sorted.
2	Text box is not editable and list is sorted.
3	Text box is editable and list is sorted.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Dim lists$(3)
   lists$(0) = "List 0"
   lists$(1) = "List 1"
   lists$(2) = "List 2"
   lists$(3) = "List 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      DropListBox 10,25,180,60,lists$(),.list1
      DropListBox 10,50,180,60,lists$(),.list2,1
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.list1 = 2       ' list1 is a numeric field
   dlg.list2 = "xxx"   ' list2 is a string field
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print lists$(dlg.list1)
   Debug.Print dlg.list2
End Sub
```

1.3.5.75 End

Syntax:
End

Group: Flow Control

Description:
The end instruction causes the macro to terminate immediately. If the macro was run by another macro using the MacroRun instruction then that macro continues on the instruction following the MacroRun.

Example:

```
Sub DoSub
   L$ = UCase$(InputBox$("Enter End:"))
   If L$ = "END" Then End
   Debug.Print "End was not entered."
End Sub

Sub Main
   Debug.Print "Before DoSub"
   DoSub
   Debug.Print "After DoSub"
End Sub
```

1.3.5.76 Enum

Syntax:

```
[ | Private | Public ] _
Enum name
   elem [ = value]
  [...]
End Enum
```

Group: Declaration

Description:
Define a new userenum. Each elem defines an element of the enum. If value is given then that is the element's value. The value can be any constant integer expression. If value is omitted then the element's value is one more than the previous element's value. If there is no previous element then zero is used.

Enum defaults to Public if neither Private or Public is specified.

Example:

```
Enum Days
   Monday
   Tuesday
   Wednesday
   Thursday
   Friday
   Saturday
   Sunday
End Enum

Sub Main
   Dim D As Days
   For D = Monday To Friday
      Debug.Print D ' 0 through 4
   Next D
End Sub
```

1.3.5.77 Environ

Syntax:

```
Environ[$](Index)
-or-
Environ[$](Name)
```

Group: Miscellaneous

Description:
Return an environment string.

Parameter	Description
Index	Return this environment string's value. If there is no environment string at this index a null string is returned. Indexes start at one.
Name	Return this environment string's value. If the environment string can't be found a null string is returned.

Example:

```
Sub Main
   Debug.Print Environ("Path")
End Sub
```

1.3.5.78 EOF

Syntax:
EOF(StreamNum)

Group: File

Description:
Return True if StreamNum is at the end of the file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example:

```
Sub Main
   Open "XXX" For Input As #1
   While Not EOF(1)
      Line Input #1,L$
      Debug.Print L$
   Wend
   Close #1
End Sub
```

1.3.5.79 Erase

Syntax:
Erase arrayvar[, ...]
-or-
Erase usertypevar.elem[, ...]

Group: Assignment

Description:
Reset arrayvar or user defined type array element to zero. (Dynamic arrays are reset to undimensioned arrays.) String arrays values are set to a null string. arrayvar must be declared as an array.

- Declare with Dim, Private, Public or Static.
- Declare as a parameter of Sub, Function or Property definition.

Example:

```
Sub Main
    Dim X%(2)
    X%(1) = 1
    Erase X%
    Debug.Print X%(1) ' 0
End Sub
```

1.3.5.80 Err

Syntax:
Err

Group: Error Handling

Description:
Set Err to zero to clear the last error event. Err in an expression returns the last error code. Add vbObjectError to your error number in ActiveX Automation objects. Use Err.Raise or Error to trigger an error event.

Err[.Number]

This is the error code for the last error event. Set it to zero (or use Err.Clear) to clear the last error condition. Use Error or Err.Raise to trigger an error event. This is the default property.

Err.Description

This string is the description of the last error event.

Err.Source

This string is the error source file name of the last error event.

Err.HelpFile

This string is the help file name of the last error event.

Err.HelpContext

This number is the help context id of the last error event.

Err.Clear

Clear the last error event.

```
Err.Raise [Number:=]errorcode _
    [, [Source:=]source] _
    [, [Description:=]errordesc] _
    [, [HelpFile:=]helpfile] _
    [, [HelpContext:=]context]
```

Raise an error event.

Err.LastDLLError

For 32 bit windows this returns the error code for the last DLL call (see Declare). For 16 bit windows this always returns 0.

Example:

```
Sub Main
   On Error GoTo Problem
   Err = 1 ' set to error #1 (handler not triggered)
   Exit Sub

   Problem: ' error handler
   Error Err ' halt macro with message
End Sub
```

1.3.5.81 Error

Syntax:
Error ErrorCode
-or-
Error[$]([ErrorCode])

Group: Error Handling

Description:
Instruction: Signal error ErrorCode. This triggers error handling just like a real error. The current procedure's error handler is activated, unless it is already active or there isn't one. In that case the calling procedure's error handler is tried. (Use Err.Raise to provide complete error information.)

Function: The Error( ) function returns the error text string.

Parameter	Description
ErrorCode	This is the error number.

Example:

```
Sub Main
   On Error GoTo Problem
   Err.Raise 1 ' simulate error #1
   Exit Sub

   Problem: ' error handler
   Debug.Print "Error$=";Error$
   Resume Next
End Sub
```

1.3.5.82 Eval

Syntax:
Eval(Expr[, Depth])

Group: Miscellaneous

Description:
Return the value of the string expression as evaluated.

Parameter	Description
Expr	Evaluate this string value.
Depth	This integer value indicates how deep into the stack to locate the local veriables. If Depth = 0 then use the current procedure. If this value is omitted then the depth is 0.

Example:

```
Sub Main
    Dim X As String
    X = "Hello"
    Debug.Print Eval("X") 'Hello
    A
End Sub
Sub A
    Dim X As String
    X = "Bye"
    Debug.Print Eval("X") 'Bye
    Debug.Print Eval("X",1) 'Hello
End Sub
```

1.3.5.83 Exit

Syntax:
Exit {All|Do|For|Function|Property|Sub|While}

Group: Flow Control

Description:
The exit instruction causes the macro to continue with out doing some or all of the remaining instructions.

Exit	Description
All	Exit all macros.
Do	Exit the Do loop.
For	Exit the For of For Each loop.
Function	Exit the Function block. Note: This instruction clears the Err and sets Error$ to null.
Property	Exit the Property block. Note: This instruction clears the Err and sets Error$ to null.
Sub	Exit the Sub block. Note: This instruction clears the Err and sets Error$ to null.
While	Exit the While loop.

Example:

```
Sub Main
   L$ = InputBox$("Enter Do, For, While, Sub or All:")
   Debug.Print "Before DoSub"
   DoSub UCase$(L$)
   Debug.Print "After DoSub"
End Sub

Sub DoSub(L$)
   Do
      If L$ = "DO" Then Exit Do
      I = I+1
   Loop While I < 10
   If I = 0 Then Debug.Print "Do was entered"

   For I = 1 To 10
      If L$ = "FOR" Then Exit For
   Next I
   If I = 1 Then Debug.Print "For was entered"

   I = 10
   While I > 0
      If L$ = "WHILE" Then Exit While
      I = I-1
   Wend
   If I = 10 Then Debug.Print "While was entered"

   If L$ = "SUB" Then Exit Sub
   Debug.Print "Sub was not entered."
   If L$ = "ALL" Then Exit All
   Debug.Print "All was not entered."
End Sub
```

1.3.5.84 Exp

Syntax:
Exp(Num)

Group: Math

Description:
Return the exponential.

Parameter	Description
Num	Return e raised to the power of this numeric value. The value e is approximately 2.718282.

See Also: Log.

Example:

```
Sub Main
   Debug.Print Exp(1) ' 2.718281828459
End Sub
```

1.3.5.85 FileAttr

Syntax:
FileAttr(StreamNum, ReturnValue)

Group: File

Description:
Return StreamNum's open mode or file handle.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
ReturnValue	1 - return the mode used to open the file: 1=Input, 2=Output, 4=Random, 8=Append, 32=Binary 2 - return the file handle

See Also: Open.

Example:

```
Sub Main
   Open "XXX" For Output As #1
   Debug.Print FileAttr(1,1) ' 2
   Close #1
End Sub
```

1.3.5.86 FileCopy

Syntax:
FileCopy FromName$, ToName$

Group: File

Description:
Copy a file.

Parameter	Description
FromName$	This string value is the path and name of the source file. A path relative to the current directory can be used.
ToName$	This string value is the path and name of the destination file. A path relative to the current directory can be used.

Example:

```
Sub Main
   FileCopy "C:\AUTOEXEC.BAT","C:\AUTOEXEC.BAK"
End Sub
```

1.3.5.87 FileDateTime

Syntax:
FileDateTime(Name$)

Group: File

Description:
Return the date and time file Name$ was last changed as a date value. If the file does not exist then a run-time error occurs.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Example:

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;" ";FileDateTime(F$)
      F$ = Dir$()
   Wend
End Sub
```

1.3.5.88 FileLen

Syntax:
FileLen(Name$)

Group: File

Description:
Return the length of file Name$. If the file does not exist then a run-time error occurs.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Example:

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;" ";FileLen(F$)
      F$ = Dir$()
   Wend
End Sub
```

1.3.5.89 Fix

Syntax:
Fix(Num)

Group: Math

Description:
Return the integer value.

Parameter	Description
Num	Return the integer portion of this numeric value. The number is truncated. Positive numbers return the next lower integer. Negative numbers return the next higher integer. If this value is Null then Null is returned.

See Also: Int.

Example:

```
Sub Main
   Debug.Print Fix(9.9)  ' 9
   Debug.Print Fix(0)    ' 0
   Debug.Print Fix(-9.9) '-9
End Sub
```

1.3.5.90 For

Syntax:

```
For Num = First To Last [Step Inc]
   statements
Next [Num]
```

Group: Flow Control

Description:

Execute statements while Num is in the range First to Last.

Parameter	Description
Num	This is the iteration variable.
First	Set Num to this value initially.
Last	Continue looping while Num is in the range. See Step below.
Step	If this numeric value is greater than zero then the for loop continues as long as Num is less than or equal to Last. If this numeric value is less than zero then the for loop continues as long as Num is greater than or equal to Last. If this is omitted then one is used.

See Also: Do, For Each, Exit For, While.

Example:

```
Sub Main
   For I = 1 To 2000 Step 100
      Debug.Print I;I+I;I*I
   Next I
End Sub
```

1.3.5.91 For Each

Syntax:

```
For Each var In items
   statements
Next [var]
```

Group: Flow Control

Description:

Execute statements for each item in items.

Parameter	Description
var	This is the iteration variable.
items	This is the collection of items to be done.

See Also: Do, For, Exit For, While.

Example:

```
Sub Main
   Dim Document As Object
   For Each Document In App.Documents
      Debug.Print Document.Title
   Next Document
End Sub
```

1.3.5.92 Format

Syntax:

```
Format[$](expr[, form$], [firstday], _
      [firstweek])
```

Group: String

Description:
Return the formatted string representation of expr.

Parameter	Description
expr	Return the formatted string representation of this numeric value.
form	Format expr using to this string value. If this is omitted then return the expr as a string.
firstday	Format using this day as the first day of the week. If this is omitted then the vbSunday is used. (Only supported for Win32.)
firstweek	Format using this week as the first week of the year. If this is omitted then the vbFirstJan1 is used. (Only supported for Win32.)

firstday	Value	Description
vbUseSystemFirstDay	0	Use the systems first day of the week.
vbSunday	1	Sunday (default)
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

firstweek	Value	Description
vbUseSystem	0	Use the systems first week of the year.
vbFirstJan1	1	The week that January 1 occurs in. This is the default value.
2	vbFirstFourDays	The first week that has at least four days in the year.
3	vbFirstFullWeek	The first week that entirely in the year.

See Also: Predefined Date Format, Predefined Number Format, User defined Date Format, User defined Number Format, User defined Text Format.

1.3.5.93 FreeFile

Syntax:

FreeFile[( )]

Group: File

Description:
Return the next unused shared stream number (greater than or equal to 256). Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example:

```
Sub Main
   Debug.Print FreeFile ' 256
   FN = FreeFile
   Open "XXX" For Output As #FN
   Debug.Print FreeFile ' 257
   Close #FN
   Debug.Print FreeFile ' 256
End Sub
```

1.3.5.94 Friend

Group: Declaration

Description:
Friend Functions, Propertys and Subs in a module are available in all other macros/modules that access it. Friends are not accessible via Object variables.

1.3.5.95 Function

Syntax:

```
[ | Private | Public | Friend ] _
[ Default ] _
Function name[type][([param[, ...]])] [As type[()]]
   statements
End Function
```

Group: Declaration

Description:
User defined function. The function defines a set of statements to be executed when it is called. The values of the calling arglist are assigned to the params. Assigning to name[type] sets the value of the function result.

Function defaults to Public if Private, Public or Friend are not is specified.

See Also: Declare, Property, Sub.

Example:

```
Function Power(X,Y)
   P = 1
   For I = 1 To Y
      P = P*X
   Next I
   Power = P
End Function

Sub Main
   Debug.Print Power(2,8) ' 256
End Sub
```

1.3.5.96 Get

Syntax:
Get StreamNum, [RecordNum], var

Group: File

Description:
Get a variable's value from StreamNum.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.
var	This variable value is read from the file. For a fixed length variable (like Long) the number of bytes required to restore the variable are read. For a Variant variable two bytes are read which describe its type and then the variable value is read accordingly. For a usertype variable each field is read in sequence. For an array variable each element is read in sequence. For a dynamic array variable the number of dimensions and range of each dimension is read prior to reading the array values. All binary data values are read from the file in little-endian format. Note: When reading a string (or a dynamic array) from a Binary mode file the length (or array dimension) information is not read. The current string length determines how much string data is read. The current array dimension determines how may array elements are read.

See Also: Open, Put.

Example:

```
Sub Main
   Dim V As Variant
   Open "SAVE_V.DAT" For Binary Access Read As #1
   Get #1, , V
   Close #1
End Sub
```

1.3.5.97 GetAllSettings

Syntax:
GetAllSettings(AppName$, Section$, Key$)

Group: Settings

Description:
Get all of Section's settings in project AppName. Settings are returned in a Variant. Empty is returned if there are no keys in the section. Otherwise, the Variant contains a two dimension array: (I,0) is the key and (I,1) is the setting. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.

Example:

```
Sub Main
   SaveSetting "MyApp","Font","Size",10
   SaveSetting "MyApp","Font","Name","Courier"
   Settings = GetAllSettings("MyApp","Font")
   For I = LBound(Settings) To UBound(Settings)
      Debug.Print Settings(I,0); "="; Settings(I,1)
   Next I
   DeleteSetting "MyApp","Font"
End Sub
```

1.3.5.98 GetAttr

Syntax:
GetAttr(Name$)

Group: File

Description:
Return the attributes for file Name$. If the file does not exist then a run-time error occurs.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Example:

```
Sub Main
   F$ = Dir$("*.*")
   While F$ <> ""
      Debug.Print F$;" ";GetAttr(F$)
      F$ = Dir$()
   Wend
End Sub
```

1.3.5.99 GetFilePath

Syntax:
GetFilePath[$]([DefName$], [DefExt$], [DefDir$], _
[Title$], [Option])

Group: User Input

Description:
Put up a dialog box and get a file path from the user. The returned string is a complete path and file

name. If the cancel button is pressed then a null string is returned.

Parameter	Description
DefName$	Set the initial File Name in the to this string value. If this is omitted then *. DefExt$ is used.
DefExt$	Initially show files whose extension matches this string value. (Multiple extensions can be specified by using ";" as the separator.) If this is omitted then * is used.
DefDir$	This string value is the initial directory. If this is omitted then the current directory is used.
Title$	This string value is the title of the dialog. If this is omitted then "Open" is used.
Option	This numeric value determines the file selection options. If this is omitted then zero is used. See table below.

Option	Effect
0	Only allow the user to select a file that exists.
1	Confirm creation when the user selects a file that does not exist.
2	Allow the user to select any file whether it exists or not.
3	Confirm overwrite when the user selects a file that exists.
+4	Selecting a different directory changes the application's current directory.

Example:

```
Sub Main
   Debug.Print GetFilePath$()
End Sub
```

1.3.5.100 GetObject

Syntax:
GetObject([File$][, Class$])

Group: Object

Description:
Get an existing object of type Class$ from File$. Use Set to assign the returned object to an object variable.

Parameter	Description
File$	This is the file where the object resides. If this is omitted then the currently active object for Class$ is returned.
Class$	This string value is the application's registered class name. If this application is not currently active it will be started. If this is omitted then the application associated with the file's extension will be started.

Example:

```
Sub Main
   Dim App As Object
   Set App = GetObject(,"WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
   Set App = Nothing
   App.Quit       ' run-time error (no object)
End Sub
```

1.3.5.101 GetSetting

Syntax:
GetSetting[$](AppName$, Section$, Key$[, Default$])

Group: Settings

Description:
Get the setting for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings.
Default$	Return this string value if no setting has been saved. If this is omitted then a null string is used.

Example:

```
Sub Main
    SaveSetting "MyApp","Font","Size",10
    Debug.Print GetSetting("MyApp","Font","Size") ' 10
End Sub
```

1.3.5.102 Goto

Syntax:
GoTo label

Group: Flow Control

Description:
Go to the label and continue execution from there. Only labels in the current user defined procedure are accessible.

Example:

```
Sub Main
    X = 2
Loop:
    X = X*X
    If X < 100 Then GoTo Loop
    Debug.Print X ' 256
End Sub
```

1.3.5.103 GroupBox

Syntax:
GroupBox X, Y, DX, DY, Title$[, .Field]

Group: User Dialog

Description:
Define a groupbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.

Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Title$	This string value is the title of the group box.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this identifer is omitted then the first two words of the title are used.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      GroupBox 10,25,180,60,"Group box"
      OKButton 80,90,40,20
      End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.104 Hex

Syntax:
Hex[$](Num)

Group: String

Description:
Return a hex string.

Parameter	Description
Num	Return a hex encoded string for this numeric value.

See Also: Oct$( ), Str$( ), Val( ).

Example:

```
Sub Main
   Debug.Print Hex$(15) 'F
End Sub
```

1.3.5.105 Hour

Syntax:
Hour(dateexpr)

Group: Time/Date

Description:
Return the hour of the day (0 to 23).

Parameter	Description
dateexpr	Return the hour of the day for this date value. If this value is Null then Null is returned.

See Also: Minute( ), Second( ), Time( ).

Example:

```
Sub Main
    Debug.Print Hour(#12:00:01 AM#) ' 0
End Sub
```

1.3.5.106 If

Syntax:

```
If condexpr Then [instruction] [Else instruction]
-or-
If condexpr Then
    statements
[ElseIf condexpr Then
    statements]...
[Else
    statements]
End If
-or-
If TypeOf objexpr Is objtype Then ...
```

Group: Flow Control

Description:

Form 1: Single line if statement. Execute the instruction following the Then if condexpr is True. Otherwise, execute the instruction following the Else. The Else portion is optional.

Form 2: The multiple line if is useful for complex ifs. Each if condexpr is checked in turn. The first True one causes the following statements to be executed. If all are False then the Else's statements are executed. The ElseIf and Else portions are optional.

Form 3: If objexpr's type is the same type or a type descended from objtype the Then portion is executed.

See Also: Select Case, Choose( ), IIf( ).

Example:

```
Sub Main
    S = InputBox("Enter hello, goodbye, dinner or sleep:")
    S = UCase(S)
    If S = "HELLO" Then Debug.Print "come in"
    If S = "GOODBYE" Then Debug.Print "see you later"
    If S = "DINNER" Then
        Debug.Print "Please come in."
        Debug.Print "Dinner will be ready soon."
    ElseIf S = "SLEEP" Then
        Debug.Print "Sorry."
        Debug.Print "We are full for the night"
    End If
End Sub
```

1.3.5.107 IIf

Syntax:

Iif(condexpr, TruePart, FalsePart)

Group: Miscellaneous

Description:
Return the value of the parameter indicated by condexpr. Both TruePart and FalsePart are evaluated.

Parameter	Description
condexpr	If this value is True then return TruePart. Otherwise, return FalsePart.
TruePart	Return this value if condexpr is True.
FalsePart	Return this value if condexpr is False.

See Also: If, Select Case, Choose( ).

Example:

```
Sub Main
   Debug.Print Iif(1 > 0,"True","False") "'True"
End Sub
```

1.3.5.108 Input

Syntax:
Input [#]StreamNum, var[, ...]

Group: File

Description:
Get input from StreamNum and assign it to vars. Input values are comma delimited. Leading and trailing spaces are ignored. If the first char (following the leading spaces) is a quote (") then the string is terminated by an ending quote. Special values #NULL#, #FALSE#, #TRUE#, #date# and #ERROR number# are converted to their appropriate value and data type.

See Also: Line Input, Print, Write.

Example:

```
Sub Main
   Open "XXX" For Input As #1
   Input #1,A,B,C$
   Debug.Print A;B;C$
   Close #1
End Sub
```

1.3.5.109 InputBox

Syntax:
InputBox[$](Prompt$[, Title$][, Default$][, XPos, YPos])

Group: User Input

Description:
Display an input box where the user can enter a line of text. Pressing the OK button returns the string entered. Pressing the Cancel button returns a null string.

Parameter	Description
Prompt$	Use this string value as the prompt in the input box.
Title$	Use this string value as the title of the input box. If this is omitted then the input box does not have a title.

Default$	Use this string value as the initial value in the input box. If this is omitted then the initial value is blank.
XPos	When the dialog is put up the left edge will be at this screen position. If this is omitted then the dialog will be centered.
YPos	When the dialog is put up the top edge will be at this screen position. If this is omitted then the dialog will be centered.

Example:

```
Sub Main
   L$ = InputBox$("Enter some text:", _
        "Input Box Example","asdf")
   Debug.Print L$
End Sub
```

1.3.5.110 InStr

Syntax:
InStr([Index,]S1$, S2$)

Group: String

Description:
Return the index where S2$ first matches S1$. If no match is found return 0.

Note: A similar function, InStrB, returns the byte index instead.

Parameter	Description
Index	Start searching for S2$ at this index in S1$. If this is omitted then start searching from the beginning of S1$.
S1$	Search for S2$ in this string value. If this value is Null then Null is returned.
S2$	Search S1$ for this string value. If this value is Null then Null is returned.

See Also: InStrRev( ), Left$( ), Len( ), Mid$( ), Replace$( ), Right$( ).

Example:

```
Sub Main
   Debug.Print InStr("Hello","l") ' 3
End Sub
```

1.3.5.111 InStrRev

Syntax:
InStrRev(S1$, S2$[, Index])

Group: String

Description:
Return the index where S2$ last matches S1$. If no match is found return 0.

Parameter	Description
S1$	Search for S2$ in this string value. If this value is Null then Null is returned.
S2$	Search S1$ for this string value. If this value is Null then Null is returned.
Index	Start searching for S2$ ending at this index in S1$. If this is omitted then start searching from the end of S1$.

See Also: Left$( ), Len( ), Mid$( ), Replace$( ), Right$( ).

Example:

```
Sub Main
   Debug.Print InStrRev("Hello","l") ' 4
End Sub
```

1.3.5.112 Int

Syntax:
Int(Num)

Group: Math

Description:
Return the integer value.

Parameter	Description
Num	Return the largest integer which is less than or equal to this numeric value. If this value is Null then Null is returned.

See Also: Fix.

Example:

```
Sub Main
   Debug.Print Int(9.9)  ' 9
   Debug.Print Int(0)    ' 0
   Debug.Print Int(-9.9) '-10
End Sub
```

1.3.5.113 Is

Syntax:
expr Is expr

Group: Operator

Description:
Return the True if both exprs refer to the same object.

See Also: Objects.

Example:

```
Sub Main
   Dim X As Object
   Dim Y As Object
   Debug.Print X Is Y ' True
End Sub
```

1.3.5.114 IsArray

Syntax:
IsArray(var)

Group: Variable Info

Description:
Return the True if var is an array of values.

Parameter	Description

var	A array variable or a variant var can contain multiple of values.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant, Y(2) As Integer
    Debug.Print IsArray(X) 'False
    X = Array(1,4,9)
    Debug.Print IsArray(X) 'True
    X = Y
    Debug.Print IsArray(X) 'True
End Sub
```

1.3.5.115 IsDate

Syntax:
IsDate(expr)

Group: Variable Info

Description:
Return the True if expr is a valid date.

Parameter	Description
expr	A variant expression to test for a valid date.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    X = 1
    Debug.Print IsDate(X) 'False
    X = Now
    Debug.Print IsDate(X) 'True
End Sub
```

1.3.5.116 IsEmpty

Syntax:
IsEmpty(variantvar)

Group: Variable Info

Description:
Return the True if variantvar is Empty.

Parameter	Description
variantvar	A variant var is Empty if it has never been assign a value.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    Debug.Print IsEmpty(X) 'True
    X = 0
    Debug.Print IsEmpty(X) 'False
    X = Empty
    Debug.Print IsEmpty(X) 'True
End Sub
```

1.3.5.117 IsError

Syntax:
IsError(expr)

Group: Variable Info

Description:
Return the True if expr is an error code.

Parameter	Description
expr	A variant expression to test for an error code value.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    Debug.Print IsError(X) 'False
    X = CVErr(1)
    Debug.Print IsError(X) 'True
End Sub
```

1.3.5.118 IsMissing

Syntax:
IsMissing(variantvar)

Group: Variable Info

Description:
Return the True if Optional parameter variantvar does not have a defaultvalue and it did not get a value. An Optional parameter may be omitted in the Sub, Function or Property call.

Parameter	Description
variantvar	Return True if this variant parameter's argument expression was not specified in the Sub, Function or Property call.

Example:

```
Sub Main
    Opt             'IsMissing(A)=True
    Opt "Hi"        'IsMissing(A)=False
    Many            'No args
    Many 1,"Hello"   'A(0)=1 A(1)=Hello
    OptBye          "'Bye"
    OptBye "No"     "'No"
End Sub

Sub Opt(Optional A)
    Debug.Print "IsMissing(A)=";IsMissing(A)
End Sub

Sub Many(ParamArray A())
    If LBound(A) > UBound(A) Then
        Debug.Print "No args"
    Else
        For I = LBound(A) To UBound(A)
            Debug.Print "A(" & I & ")=" & A(I) & " ";
        Next I
        Debug.Print
    End If
End Sub

Sub OptBye(Optional A As String = "Bye")
    Debug.Print A
End Sub
```

1.3.5.119 IsNumeric

Syntax:
IsNumeric(expr)

Group: Variable Info

Description:
Return the True if expr is a numeric value.

Parameter	Description
expr	A variant expression is a numeric value if it is numeric or string value that represents a number.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    X = 1
    Debug.Print IsNumeric(X) 'True
    X = "1"
    Debug.Print IsNumeric(X) 'True
    X = "A"
    Debug.Print IsNumeric(X) 'False
End Sub
```

1.3.5.120 IsNull

Syntax:
IsNull(expr)

Group: Variable Info

Description:
Return the True if expr is Null.

Parameter	Description
expr	A variant expression to test for Null.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    Debug.Print IsEmpty(X) 'True
    Debug.Print IsNull(X)  'False
    X = 1
    Debug.Print IsNull(X)  'False
    X = "1"
    Debug.Print IsNull(X)  'False
    X = Null
    Debug.Print IsNull(X)  'True
    X = X*2
    Debug.Print IsNull(X)  'True
End Sub
```

1.3.5.121 IsObject

Syntax:
IsObject(var)

Group: Variable Info

Description:
Return the True if var contains an object reference.

Parameter	Description
var	A var contains an object reference if it is objexpr reference.

See Also: TypeName, VarType.

Example:

```
Sub Main
    Dim X As Variant
    X = 1
    Debug.Print IsObject(X) 'False
    X = "1"
    Debug.Print IsObject(X) 'False
    Set X = Nothing
    Debug.Print IsObject(X) 'True
End Sub
```

1.3.5.122 Join

Syntax:
Join(StrArray, [Sep])

Group: Miscellaneous

Description:
Return a string by concatenating all the values in the array with Sep in between each one.

Parameter	Description
StrArray	Concatenate values from this array.
Sep	Use this string value to separate the values. (Default: " ")

See Also: Split( ).

Example:

```
Sub Main
    Debug.Print Join(Array(1,2,3)) "'1 2 3"
End Sub
```

1.3.5.123 KeyName

Syntax:
KeyName(Key)

Group: Miscellaneous

Description:
Return the key name for a key number. This is the name used by SendKeys.

Parameter	Description
Key	Key number.

See Also: SendKeys.

Example:

```
Sub Main
    Debug.Print KeyName(&H270)  "'^{F1}"
End Sub
```

1.3.5.124 Kill

Syntax:
Kill Name$

Group: File

Description:
Delete the file named by Name$.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.

Example:

```
Sub Main
    Kill "XXX"
End Sub
```

1.3.5.125 LBound

Syntax:
LBound(arrayvar[, dimension])

Group: Variable Info

Description:
Return the lowest index.

Parameter	Description
arrayvar	Return the lowest index for this array variable.
dimension	Return the lowest index for this dimension of arrayvar. If this is omitted then return the lowest index for the first dimension.

See Also: UBound( ).

Example:

```
Sub Main
    Dim A(-1 To 3,2 To 6)
    Debug.Print LBound(A)   '-1
    Debug.Print LBound(A,1) '-1
    Debug.Print LBound(A,2) ' 2
End Sub
```

1.3.5.126 LCase

Syntax:
LCase[$](S$)

Group: String

Description:
Return a string from S$ where all the uppercase letters have been lowercased.

Parameter	Description
S$	Return the string value of this after all chars have been converted to lowercase. If this value is Null then Null is returned.

See Also: StrComp( ), StrConv$( ), UCase$( ).

Example:

```
Sub Main
    Debug.Print LCase$("Hello") "'hello"
End Sub
```

1.3.5.127 Left

Syntax:
Left[$](S$, Len)

Group: String

Description:
Return a string from S$ with only the Len chars.

Note: A similar function, LeftB, returns the first Len bytes.

Parameter	Description

S$	Return the left portion of this string value. If this value is Null then Null is returned.
Len	Return this many chars. If S$ is shorter than that then just return S$.

See Also: InStr( ), InStrRev( ), Len( ), Mid$( ), Replace$( ), Right$( ).

Example:

```
Sub Main
    Debug.Print Left$("Hello",2) "'He"
End Sub
```

1.3.5.128 Len

Syntax:
Len(S$)
-or-
Len(usertypevar)

Group: String

Description:
Return the number of characters in S$.

Note: A similar function, LenB, returns the number of bytes in the string. For a usertypevar, LenB returns the number of bytes of memory occupied by the variable's data.

Parameter	Description
S$	Return the number of chars in this string value. If this value is Null then Null is returned.
usertypevar	Return the number of bytes required to store this user type variable. If the user type has any dynamic String and Variant elements the length returned may not be as big as the actual number of bytes required.

See Also: InStr( ), InStrRev( ), Left$( ), Mid$( ), Replace$( ), Right$( ).

Example:

```
Sub Main
    Debug.Print Len("Hello") ' 5
End Sub
```

1.3.5.129 Let

Syntax:
[Let] var = expr

Group: Assignment

Description:
Assign the value of expr to var. The keyword Let is optional.

Example:

```
Sub Main
    Let X = 1
    X = X*2
    Debug.Print X ' 2
End Sub
```

1.3.5.130 Like

Syntax:
str1 Like str2

Group: Operator

Description:
Return the True if str1 matches pattern str2. The pattern in str2 is one or more of the special character sequences shown in the following table.

Char(s)	Description
?	Match any single character.
*	Match zero or more characters.
#	Match a single digit (0-9).
[charlist]	Match any char in the list.
[!charlist]	Match any char not in the list.

Example:

```
Sub Main
    Debug.Print "abcdfgcdefg" Like "" ' False
    Debug.Print "abcdfgcdefg" Like "a*g" ' True
    Debug.Print "abcdfgcdefg" Like "a*cde*g" ' True
    Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
    Debug.Print "abcdfgcdefg" Like "a*cd*cd*g" ' True
    Debug.Print "00aa" Like "####" ' False
    Debug.Print "00aa" Like "????" ' True
    Debug.Print "00aa" Like "##??" ' True
    Debug.Print "00aa" Like "*##*" ' True
    Debug.Print "hk" Like "hk*" ' True
End Sub
```

1.3.5.131 Line Input

Syntax:
Line Input [#]StreamNum, S$

Group: File

Description:
Get a line of input from StreamNum and assign it to S$.

See Also: Input, Print, Write.

Example:

```
Sub Main
    Open "XXX" For Input As #1
    Line Input #1,S$
    Debug.Print S$
    Close #1
End Sub
```

1.3.5.132 ListBox

Syntax:

ListBox X, Y, DX, DY, StrArray$( ), .Field[, Options]

Group: User Dialog

Description:
Define a listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The value of the list box is accessed via this field. It is the index of the StrArray$( ) var.
Options	This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	List is not sorted.
1	List is not sorted and horizontally scrollable.
2	List is sorted.
3	List is sorted and horizontally scrollable.

See Also: Begin Dialog, Dim As UserDialog, MultiListBox.

Example:

```
Sub Main
   Dim lists$(3)
   lists$(0) = "List 0"
   lists$(1) = "List 1"
   lists$(2) = "List 2"
   lists$(3) = "List 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      ListBox 10,25,180,60,lists$(),.list
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.list = 2
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.list
End Sub
```

1.3.5.133 Loc

Syntax:
Loc(StreamNum)

Group: File

Description:
Return StreamNum file position. For Random mode files this is the current record number minus one. For Binary mode files it is the current byte position minus one. Otherwise, it is the current byte position minus one divided by 128. The first position in the file is 0.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example:

```
Sub Main
    Open "XXX" For Input As #1
    L = Loc(1)
    Close #1
    Debug.Print L ' 0
End Sub
```

1.3.5.134 Lock

Syntax:
Lock StreamNum
-or-
Lock StreamNum, RecordNum
-or-
Lock StreamNum, [start] To end

Group: File

Description:
Form 1: Lock all of StreamNum.

Form 2: Lock a record (or byte) of StreamNum.

Form 3: Lock a range of records (or bytes) of StreamNum. If start is omitted then lock starting at the first record (or byte).

Note: Be sure to Unlock for each Lock instruction.
Note: For sequential files (Input, Output and Append) lock always affects the entire file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

See Also: Open, Unlock.

Example:

```
Sub Main
    Dim V As Variant
    Open "SAVE_V.DAT" For Binary As #1
    Lock #1
    Get #1, 1, V
    V = "Hello"
    Put #1, 1, V
    Unlock #1
    Close #1
End Sub
```

1.3.5.135 LOF

Syntax:
LOF(StreamNum)

Group: File

Description:
Return StreamNum file length (in bytes).

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.

Example:

```
Sub Main
    Open "XXX" For Input As #1
    L = LOF(1)
    Close #1
    Debug.Print L
End Sub
```

1.3.5.136 Log

Syntax:
Log(Num)

Group: Math

Description:
Return the natural logarithm.

Parameter	Description
Num	Return the natural logarithm of this numeric value. The value e is approximately 2.718282.

See Also: Exp.

Example:

```
Sub Main
    Debug.Print Log(1) ' 0
End Sub
```

1.3.5.137 LSet

Syntax:

LSet strvar = str
-or-
LSet usertypevar1 = usertypevar2

Group: Assignment

Description:
Form 1: Assign the value of str to strvar. Shorten str by removing trailing chars (or extend with blanks). The previous length strvar is maintained.

Form 2: Assign the value of usertypevar2 to usertypevar1. If usertypevar2 is longer than usertypevar1 then only copy as much as usertypevar1 can handle.

See Also: RSet.

Example:

```
Sub Main
    S$ = "123"
    LSet S$ = "A"
    Debug.Print ".";S$;"." "'.A  ."
End Sub
```

1.3.5.138 LTrim

Syntax:
LTrim[$](S$)

Group: String

Description:
Return the string with S$'s leading spaces removed.

Parameter	Description
S$	Copy this string without the leading spaces. If this value is Null then Null is returned.

See Also: RTrim$( ), Trim$( ).

Example:

```
Sub Main
    Debug.Print ".";LTrim$("  x  ");"." "'.x  ."
End Sub
```

1.3.5.139 MacroDir

Syntax:
MacroDir[$]

Group: Flow Control

Description:
Return the directory of the current macro. A run-time error occurs if the current macro has never been saved.

See Also: MacroRun.

Example:

```
Sub Main
    ' open the file called Data that is in the
    ' same directory as the macro
    Open MacroDir & "\Data" For Input As #1
    Line Input #1, S$
    Debug.Print S$
    Close #1
End Sub
```

1.3.5.140 MacroRun

Syntax:
MacroRun MacroName$[, Command$]

Group: Flow Control

Description:
Play a macro. Execution will continue at the following statement after the macro has completed.

Parameter	Description
MacroName$	Run the macro named by this string value.
Command$	Pass this string value as the macro's Command$ value.

See Also: Command$, MacroDir$, MacroRunThis.

Example:

```
Sub Main
    Debug.Print "Before Demo"
    MacroRun "Demo"
    Debug.Print "After Demo"
End Sub
```

1.3.5.141 MacroRunThis

Syntax:
MacroRunThis MacroCode$

Group: Flow Control

Description:
Play the macro code. Execution will continue at the following statement after the macro code has completed. The macro code can be either a single line or a complete macro.

Parameter	Description
MacroName$	Run the macro named by this string value.

See Also: Command$, MacroDir$, MacroRun.

Example:

```
Sub Main
    Debug.Print "Before Demo"
    MacroRunThis "MsgBox ""Hello"""
    Debug.Print "After Demo"
End Sub
```

1.3.5.142 Me

Syntax:
Me

Group: Object

Description:
Me references the current macro/module. It can be used like any other object variable, except that it's reference can't be changed.

See Also: Set.

Example:

```
Sub Main
   DoIt
   Me.DoIt ' calls the same sub
End Sub
Sub DoIt
   MsgBox "Hello"
End Sub
```

1.3.5.143 Mid

Syntax:
Mid[$](S$, Index[, Len])
-or-
Mid[$](strvar, Index[, Len]) = S$

Group: String

Description:
Function: Return the substring of S$ starting at Index for Len chars.

Instruction: Assign S$ to the substring in strvar starting at Index for Len chars.

Note: A similar function, MidB, returns the Len bytes starting a byte Index.

Parameter	Description (Mid Function)
S$	Copy chars from this string value. If this value is Null then Null is returned.
Index	Start copying chars starting at this index value. If the string is not that long then return a null string.
Len	Copy this many chars. If the S$ does not have that many chars starting at Index then copy the remainder of S$.

Parameter	Description (Mid Assignment)
strvar	Change part of this string.
Index	Change strvar starting at this index value. If the string is not that long then it is not changed.
Len	The number of chars copied is smallest of: the value of Len, the length of S$ and the remaining length of strvar. (If this value is omitted then the number of chars copied is the smallest of: the length of S$ and the remaining length of strvar.)
S$	Copy chars from this string value.

See Also: InStr( ), Left$( ), Len( ), Replace$( ), Right$( ).

Example:

```
Sub Main
    S$ = "Hello There"
    Mid$(S$,7) = "??????????"
    Debug.Print S$ "'Hello ?????"
    Debug.Print Mid$("Hello",2,1) "'e"
End Sub
```

1.3.5.144 Minute

Syntax:
Minute(dateexpr)

Group: Time/Date

Description:
Return the minute of the hour (0 to 59).

Parameter	Description
dateexpr	Return the minute of the hour for this date value. If this value is Null then Null is returned.

See Also: Hour( ), Second( ), Time( ).

Example:

```
Sub Main
    Debug.Print Minute(#12:00:01 AM#) ' 0
End Sub
```

1.3.5.145 MkDir

Syntax:
MkDir Name$

Group: File

Description:
Make directory Name$.

Parameter	Description
Name$	This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also: RmDir.

Example:

```
Sub Main
    MkDir "C:\WWTEMP"
End Sub
```

1.3.5.146 Month

Syntax:
Month(dateexpr)

Group: Time/Date

Description:

Return the month of the year (1 to 12).

Parameter	Description
dateexpr	Return the month of the year for this date value. If this value is Null then Null is returned.

See Also: Date( ), Day( ), MonthName( ), Weekday( ), Year( ).

Example:

```
Sub Main
    Debug.Print Month(#1/1/1900#) ' 1
    Debug.Print Month(#2/1/1900#) ' 2
End Sub
```

1.3.5.147 MonthName

Syntax:
MonthName(NumZ{month}[, CondZ{abbrev}])

Group: Time/Date

Description:
Return the localized name of the month.

Parameter	Description
month	Return the localized name of this month. (1-12)
abbrev	If this conditional value is True then return the abbreviated form of the month name.

See Also: Month( ).

Example:

```
Sub Main
    Debug.Print MonthName(1) 'January
    Debug.Print MonthName(Month(Now))
End Sub
```

1.3.5.148 MsgBox

Syntax:
MsgBox Message$[, Type][, Title$]
-or-
MsgBox(Message$[, Type][, Title$])

Group: User Input

Description:
Show a message box titled Title$. Type controls what the message box looks like (choose one value from each category). Use MsgBox( ) if you need to know what button was pressed. The result indicates which button was pressed.

Result	Value	Button Pressed
vbOK	1	OK button
vbCancel	2	Cancel button
vbAbort	3	Abort button
vbRetry	4	Retry button

vbIgnore	5	Ignore button
vbYes	6	Yes button
vbNo	7	No button

Parameter	Description
Message$	This string value is the text that is shown in the message box.
Type	This numeric value controls the type of message box. Choose one value from each of the following tables.
Title$	This string value is the title of the message box.

Button	Value	Effect
vbOkOnly	0	OK button
vbOkCancel	1	OK and Cancel buttons
vbAbortRetryIgnore	2	Abort, Retry, Ignore buttons
vbYesNoCancel	3	Yes, No, Cancel buttons
vbYesNo	4	Yes and No buttons
vbRetryCancel	5	Retry and Cancel buttons

Icon	Value	Effect
	0	No icon
vbCritical	16	Stop icon
vbQuestion	32	Question icon
vbExclamation	48	Attention icon
vbInformation	64	Information icon

Default	Value	Effect
vbDefaultButton1	0	First button
vbDefaultButton2	256	Second button
vbDefaultButton3	512	Third button

Mode	Value	Effect
vbApplicationModal	0	Application modal
vbSystemModal	4096	System modal
vbMsgBoxSetForeground	&h10000	System modal

Example:

```
Sub Main
   MsgBox "Please press OK button"
   If MsgBox("Please press OK button",vbOkCancel) = vbOK Then
      Debug.Print "OK was pressed"
   Else
   Debug.Print "Cancel was pressed"
   End If
End Sub
```

1.3.5.149 MultiListBox

Syntax:
MultiListBox X, Y, DX, DY, StrArray$( ), .Field[, Options]

Group: User Dialog

Description:
Define a multiple selection listbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
StrArray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
Field	The values of the list box are accessed via this field. It is the index of the StrArray$( ) var.
Options	This numeric value controls the type of list box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	List is not sorted.
1	List is not sorted and horizontally scrollable.
2	List is sorted.
3	List is sorted and horizontally scrollable.

See Also: Begin Dialog, Dim As UserDialog, ListBox.

Example:

```
Sub Main
   Dim lists$(3)
   lists$(0) = "List 0"
   lists$(1) = "List 1"
   lists$(2) = "List 2"
   lists$(3) = "List 3"
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      MultiListBox 10,25,180,60,lists$(),.list
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.list = Array(2)
   Dialog dlg ' show dialog (wait for ok)
   For i = LBound(dlg.list) To UBound(dlg.list)
      Debug.Print dlg.list(i);
   Next i
   Debug.Print
End Sub
```

1.3.5.150 Name

Syntax:
Name OldName$ As NewName$

Group: File

Description:
Rename file OldName$ as NewName$.

Parameter	Description
OldName$	This string value is the path and name of the file. A path relative to the current directory can be used.
NewName$	This is the new file name (and path). A path relative to the current directory can be used.

Example:

```
Sub Main
   Name "AUTOEXEC.BAK" As "AUTOEXEC.SAV"
End Sub
```

1.3.5.151 Now

Syntax:
Now

Group: Time/Date

Description:
Return the current date and time as a date value.

See Also: Date, Time, Timer.

Example:

```
Sub Main
   Debug.Print Now ' example: 1/1/1995 10:05:32 AM
End Sub
```

1.3.5.152 Oct

Syntax:
Oct[$](Num)

Group: String

Description:
Return a octal string.

Parameter	Description
Num	Return an octal encoded string for this numeric value.

See Also: Hex$( ), Str$( ), Val( ).

Example:

```
Sub Main
   Debug.Print Oct$(15) '17
End Sub
```

1.3.5.153 OKButton

Syntax:
OKButton X, Y, DX, DY[, .Field]

Group: User Dialog

Description:
Define an OK button item. Pressing the OK button updates the dlgvar field values and closes the dialog. (Dialog( ) function call returns -1.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this is omitted then the field name is "OK".

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,30,"Please push the OK button"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.154 On Error

Syntax:
On Error GoTo 0
-or-
On Error GoTo label
-or-
On Error Resume Next

Group: Error Handling

Description:
Form 1: Disable the error handler (default).

Form 2: Send error conditions to an error handler.

Form 3: Error conditions continue execution at the next statement.

On Error sets or disables the error handler. Each user defined procedure has its own error handler. The default is to terminate the macro on any error. The Err object's properties are set whenever an error occurs. Once an error has occurred and the error handler is executing any further errors will terminate the macro, unless the Err object has been cleared.

Note: This instruction clears the Err and sets Error$ to null.

Example:

```
Sub Main
    On Error Resume Next
    Err.Raise 1
    Debug.Print "RESUMING, Err=";Err
    On Error GoTo X
    Err.Raise 1
    Exit Sub

X:  Debug.Print "Err=";Err
    Err.Clear
    Debug.Print "Err=";Err
    Resume Next
End Sub
```

1.3.5.155 Open

Syntax:
Open Name$ For mode [Access access] [lock] As _
[#]StreamNum [Len = RecordLen]

Group: File

Description:
Open file Name$ for mode as StreamNum.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.
mode	May be Input, Output, Append, Binary or Random.

access	May be Read, Write or Read Write.
lock	May be Shared, Lock Read, Lock Write or Lock Read Write.
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordLen	This numeric value is the record length for Random mode files. Other file modes ignore this value.

See Also: Close, FileAttr, FreeFile, Reset.

Example:

```
Sub Main
   Open "XXX" For Output As #1
   Print #1,"1,2,""Hello"""
   Close #1
End Sub
```

1.3.5.156 Option

Syntax:

Option Base [0|1]
-or-
Option Compare [Binary | Text]
-or-
Option Explicit
-or-
Option Private Module

Group: Declaration

Description:

Form 1: Set the default base index for array declarations. Affects Dim, Static, Private, Public and ReDim . Does not affect Array, ParamArray or arrays declare in a Type. Option Base 0 is the default.

Form 2: Set the default comparison mode for string.

- Option Compare Binary - compare string text using binary data (default)
- Option Compare Text - compare string text using the collation rules

String comparision using <, <=, =, >, >=, <>, Like and StrComp are affected by this mode's setting.

Form 3: Require all variables to be declared prior to use. Variables are declared using Dim, Private, Public, Static or as a parameter of Sub, Function or Property blocks.

Form 4: Public symbols defined by the module are only accessible from the same project.

Example:

```
Option Base 1
Option Explicit

Sub Main
   Dim A
      Dim C(2) ' same as Dim C(1 To 2)
      Dim D(0 To 2)
   A = 1
   B = 2 ' B has not been declared
End Sub
```

1.3.5.157 OptionGroup

Syntax:
OptionGroup .Field
OptionButton X, Y, DX, DY, Title$[, .Field]
OptionButton X, Y, DX, DY, Title$[, .Field]
...

Group: User Dialog

Description:
Define a optiongroup and option button items.

Parameter	Description
Field	The value of the option group is accessed via this field. This first option button is 0, the second is 1, etc.
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Title$	The value of this string is the title of the option button.

See Also: Begin Dialog, Dim As UserDialog, OptionButton.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      OptionGroup .options
         OptionButton 10,30,180,15,"Option &0"
         OptionButton 10,45,180,15,"Option &1"
         OptionButton 10,60,180,15,"Option &2"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.options = 2
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.options
End Sub
```

1.3.5.158 Picture

Syntax:
Picture X, Y, DX, DY, FileName$, Type[, .Field]

Group: User Dialog

Description:
Define a picture item. The bitmap is automatically sized to fit the item's entire area.

Parameter	Description

X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
FileName$	The value of this string is the .BMP file shown in the picture control.
Type	This numeric value indicates the type of bitmap used. See below.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this identifer is omitted then the first two words of the title are used.

Type	Effect
0	FileName is the name of the bitmap file. If the file does not exist then "(missing picture)" is displayed.
3	The clipboard's bitmap is displayed. Not supported.
+16	Instead of displaying "(missing picture)" a run-time error occurs.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Picture 10,10,180,75,"SAMPLE.BMP",0
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.159 Print

Syntax:
Print #StreamNum, [expr[; ...][;]]

Group: File

Description:
Print the expr(s) to StreamNum. Use ; to separate expressions. A num is it automatically converted to a string before printing (just like Str$( )). If the instruction does not end with a ; then a newline is printed at the end.

See Also: Input, Line Input, Write.

Example:

```
Sub Main
   A = 1
   B = 2
   C$ = "Hello"
   Open "XXX" For Output As #1
   Print #1,A;",";B;",""";C$;"""
   Close #1
End Sub
```

1.3.5.160 Private

Syntax:
Private [WithEvents] name[type][([dim[, ...]])] [As [New] type][, ...]

Group: Declaration

Description:
Create arrays (or simple variables) which are available to the entire macro/module, but not other macros/modules. Dimension var array(s) using the dims to establish the minimum and maximum index value for each dimension. If the dims are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any dims. It must be ReDimensioned before it can be used. The Private statement must be placed outside of Sub, Function or Property blocks.

See Also: Dim, Option Base, Public, ReDim, Static, WithEvents.

Example:

```
Private A0,A1(1),A2(1,1)

Sub Init
   A0 = 1
   A1(0) = 2
   A2(0,0) = 3
End Sub

Sub Main
   Init
   Debug.Print A0;A1(0);A2(0,0) ' 1 2 3
End Sub
```

1.3.5.161 Property

Syntax:
```
[ | Private | Public | Friend ] _
[ Default ] _
Property Get name[type][([param[, ...]])] [As type[()]]
   statements
End Property
-or-
[ | Private | Public | Friend ] _
Property [Let|Set] name[([param[, ...]])]
   statements
End Property
```

Group: Declaration

Description:
User defined property. The property defines a set of statements to be executed when its value is used or changed. A property acts like a variable, except that getting its value calls Property Get and changing its value calls Property Let (or Property Set). Property Get and Property Let with the same name define a property that holds a value. Property Get and Property Set with the same name define a property that holds an object reference. The values of the calling arglist are assigned to the params. (For Property Let and Property Set the last parameter is the value on the right hand side of the assignment operator.)

Property defaults to Public if Private, Public or Friend are not is specified.

See Also: Function, Sub.

Example:

```
Dim X_Value

Property Get X()
   X = X_Value
End Property

Property Let X(NewValue)
   If Not IsNull(NewValue) Then X_Value = NewValue
End Property

Sub Main
   X = "Hello"
   Debug.Print X
   X = Null
   Debug.Print X
End Sub
```

1.3.5.162 Public

Syntax:
Public [WithEvents] name[type][([dim[, ...]])] [As [New] type][, ...]

Group: Declaration

Description:
Create arrays (or simple variables) which are available to the entire macro/module and other macros/ modules. Dimension var array(s) using the dims to establish the minimum and maximum index value for each dimension. If the dims are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any dims. It must be ReDimensioned before it can be used. The Public statement must be placed outside of Sub, Function or Property blocks.

See Also: Dim, Option Base, Private, ReDim, Static, WithEvents.

Example:

```
Public A0,A1(1),A2(1,1)

Sub Init
   A0 = 1
   A1(0) = 2
   A2(0,0) = 3
End Sub

Sub Main
   Init
   Debug.Print A0;A1(0);A2(0,0) ' 1 2 3
End Sub
```

1.3.5.163 PushButton

Syntax:
PushButton X, Y, DX, DY, Title$[, .Field]

Group: User Dialog

Description:
Define a push button item. Pressing the push button updates the dlgvar field values and closes the dialog. (Dialog( ) function call returns the push button's ordinal number in the dialog. The first push button returns 1.)

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Title$	The value of this string is the title of the push button control.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this identifer is omitted then the first two words of the title are used.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,30,"Please push the DoIt button"
      OKButton 40,90,40,20
      PushButton 110,90,60,20,"&Do It"
   End Dialog
   Dim dlg As UserDialog
   Debug.Print Dialog(dlg)
End Sub
```

1.3.5.164 Put

Syntax:
Put StreamNum, [RecordNum], var

Group: File

Description:
Write a variable's value to StreamNum.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1. If this is omitted then the current position (or record number) is used.
var	This variable value is written to the file. For a fixed length variable (like Long) the number of bytes required to store the variable are written. For a Variant variable two bytes which describe its type are written and then the variable value is written accordingly. For a usertype variable each field is written in sequence. For an array variable each element is written in sequence. For a dynamic array variable the number of dimensions and range of each dimension is written prior to writing the array values.

All binary data values are written to the file in little-endian format.

Note: When a writing string (or a dynamic array) to a Binary mode file the string length (or array dimension) information is not written. Only the string data or array elements are written.

See Also: Get, Open.

Example:

```
Sub Main
   Dim V As Variant
   Open "SAVE_V.DAT" For Binary Access Write As #1
   Put #1, , V
   Close #1
End Sub
```

1.3.5.165 QBColor

Syntax:
QBColor(num)

Group: Miscellaneous

Description:
Return the appropriate color defined by Quick Basic.

num	color
0	black
1	blue
2	green
3	cyan
4	red
5	magenta
6	yellow
7	white
8	gray
9	light blue
10	light green
11	light cyan
12	light red
13	light magenta
14	light yellow
15	bright white

See Also: RGB( ).

Example:

```
Sub Main
    Debug.Print Hex(QBColor(1))  "'800000"
    Debug.Print Hex(QBColor(7))  "'C0C0C0"
    Debug.Print Hex(QBColor(8))  "'808080"
    Debug.Print Hex(QBColor(9))  "'FF0000"
    Debug.Print Hex(QBColor(10)) "'FF00"
    Debug.Print Hex(QBColor(12)) "'FF"
    Debug.Print Hex(QBColor(15)) "'FFFFFF"
End Sub
```

1.3.5.166 Randomize

Syntax:
Randomize [Seed]

Group: Math

Description:
Randomize the random number generator.

Parameter	Description
Seed	This numeric value sets the initial seed for the random number generator. If this value is omitted then the current time is used as the seed.

See Also: Rnd( ).

Example:

```
Sub Main
    Randomize
    Debug.Print Rnd ' 0.??????????????
End Sub
```

1.3.5.167 ReDim

Syntax:
ReDim [Preserve] name[type][([dim[, ...]])] [As type][, ...]
-or-
ReDim [Preserve] usertypevar.elem[type][([dim[, ...]])] [As type][, ...]

Group: Declaration

Description:
Redimension a dynamic arrayvar or user defined type array element. Use Preserve to keep the array values. Otherwise, the array values will all be reset. When using preserve only the last index of the array may change, but the number of indexes may not. (A one-dimensional array can't be redimensioned as a two-dimensional array.)

See Also: Dim, Option Base, Private, Public, Static.

Example:

```
Sub Main
    Dim X()
    ReDim X(3)
    Debug.Print UBound(X) ' 3
    ReDim X(200)
    Debug.Print UBound(X) ' 200
End Sub
```

1.3.5.168 Reference

Syntax:
'#Reference {uuid}#vermajor.verminor#lcid#[path[#name]]

Description:
The Reference comment indicates that the current macro/module references the type library identified. Reference comment lines must be the first lines in the macro/module (following the global Attributes). Reference comments are in reverse priority (from lowest to highest). The IDE does not display the reference comments.

Parameter	Description
uuid	Type library's universally unique identifier.
vermajor	Type library's major version number.
verminor	Type library's minor version number.
lcid	Type library's locale identifier.
path	Type library's path.
name	Type library's name.

Example:

'#Reference {00025E01-0000-0000-C000-000000000046}#4.0#0#C:\PROGRAM FILES\COMMON FILES\MICROSOFT SHARED\DAO\DAO350.DLL#Microsoft DAO 3.5 Object Library

1.3.5.169 Rem

Syntax:
Rem ...
-or-
'...

Group: Miscellaneous

Description:
Both forms are comments. The Rem form is an instruction. The ' form can be used at the end of any line. All text from either ' or Rem to the end of the line is part of the comment. That text is not executed.

Example:

```
Sub Main
    Debug.Print "Hello" ' prints to the output window
    Rem the macro terminates at Main's End Sub
End Sub
```

1.3.5.170 Replace

Syntax:
Replace[$](S$, Pat$, Rep$, [Index], [Count])

Group: String

Description:
Replace Pat$ with Rep$ in S$.

Parameter	Description
S$	This string value is searched. Replacements are made in the string returned by Replace.

Pat$	This string value is the pattern to look for.
Rep$	This string value is the replacement.
Index	This numeric value is the starting index in S$. Replace(S,Pat,Rep,N) is equivalent to Replace(Mid(S,N),Pat,Rep). If this is omitted use 1.
Count	This numeric value is the maximum number of replacements that will be done. If this is omitted use -1 (which means replace all occurrences).

See Also: InStr( ), InStrRev( ), Left$( ), Len( ), Mid$( ), Right$( ).

Example:

```
Sub Main
   Debug.Print Replace$("abcabc","b","B")    "'aBcaBc"
   Debug.Print Replace$("abcabc","b","B",,1) "'aBcabc"
   Debug.Print Replace$("abcabc","b","B",3)  "'caBc"
   Debug.Print Replace$("abcabc","b","B",9)  ""
End Sub
```

1.3.5.171 Reset

Syntax:
Reset

Group: File

Description:
Close all open streams for the current macro/module.

See Also: Close, Open.

Example:

```
Sub Main
   ' read the first line of XXX and print it
   Open "XXX" For Input As #1
   Line Input #1,L$
   Debug.Print L$
   Reset
End Sub
```

1.3.5.172 Resume

Syntax:
Resume label
-or-
Resume Next

Group: Error Handling

Description:
Form 1: Resume execution at label.

Form 2: Resume execution at the next statement.

Once an error has occurred, the error handler can use Resume to continue execution. The error handler must use Resume or Exit at the end.

Note: This instruction clears the Err and sets Error$ to null.

Example:

```
Sub Main
   On Error GoTo X
   Err.Raise 1
   Debug.Print "RESUMING"
   Exit Sub

X:  Debug.Print "Err=";Err
   Resume Next
End Sub
```

1.3.5.173 RGB

Syntax:
RGB(red, green, blue)

Group: Miscellaneous

Description:
Return a color. Some useful color constants are predefined:

- vbBlack - same as RGB(0,0,0)
- vbRed - same as RGB(255,0,0)
- vbGreen - same as RGB(0,255,0)
- vbYellow - same as RGB(255,255,0)
- vbBlue - same as RGB(0,0,255)
- vbMagenta - same as RGB(255,0,255)
- vbCyan - same as RGB(0,255,255)
- vbWhite - same as RGB(255,255,255)

See Also: QBColor( ).

Example:

```
Sub Main
   Debug.Print Hex(RGB(255,0,0)) "'FF0000"
End Sub
```

1.3.5.174 Right

Syntax:
Right[$](S$, Len)

Group: String

Description:
Return the last Len chars of S$.

Note: A similar function, RightB, returns the last Len bytes.

Parameter	Description
S$	Return the right portion of this string value. If this value is Null then Null is returned.
Len	Return this many chars. If S$ is shorter than that then just return S$.

See Also: InStr( ), InStrRev( ), Left$( ), Len( ), Mid$( ), Replace$( ).

Example:

```
Sub Main
    Debug.Print Right$("Hello",3) "llo"
End Sub
```

1.3.5.175 RmDir

Syntax:
RmDir Name$

Group: File

Description:
Remove directory Name$.

Parameter	Description
Name$	This string value is the path and name of the directory. A path relative to the current directory can be used.

See Also: MkDir.

Example:

```
Sub Main
    RmDir "C:\WWTEMP"
End Sub
```

1.3.5.176 Rnd

Syntax:
Rnd([Num])

Group: Math

Description:
Return a random number greater than or equal to zero and less than one.

Parameter	Description
Num	See table below.

Num	Description
<0	Return the same number every time, using Num as the seed.
>0	Return the next random number in the sequence.
0	Return the most recently generated number.
omitted	Return the next random number in the sequence.

See Also: Randomize.

Example:

```
Sub Main
    Debug.Print Rnd() ' 0.??????????????
End Sub
```

1.3.5.177 Round

Syntax:
Round([Num][, Places])

Group: Math

Description:
Return the number rounded to the specified number of decimal places.

Parameter	Description
Num	Round this numeric value. If this value is Null then Null is returned.
Places	Round to this number of decimal places. If this is omitted then round to the nearest integer value.

Example:

```
Sub Main
    Debug.Print Round(.5)       ' 0
    Debug.Print Round(.500001)  ' 1
    Debug.Print Round(1.499999) ' 1
    Debug.Print Round(1.5)      ' 2
    Debug.Print Round(11.11)    ' 11
    Debug.Print Round(11.11,1)  ' 11.1
End Sub
```

1.3.5.178 RSet

Syntax:
RSet strvar = str

Group: Assignment

Description:
Assign the value of str to strvar. Shorten str by removing trailing chars (or extend with leading blanks). The previous length strvar is maintained.

See Also: LSet.

Example:

```
Sub Main
    S$ = "123"
    RSet S$ = "A"
    Debug.Print ".";S$;"." "'.  A."
End Sub
```

1.3.5.179 RTrim

Syntax:
RTrim[$](S$)

Group: String

Description:
Return the string with S$'s trailing spaces removed.

Parameter	Description
S$	Copy this string without the trailing spaces. If this value is Null then Null is returned.

See Also: LTrim$( ), Trim$( ).

Example:

```
Sub Main
    Debug.Print ".";RTrim$("  x  ");"." "'.  x."
End Sub
```

1.3.5.180 SaveSetting

Syntax:
SaveSetting AppName$, Section$, Key$, Setting

Group: Settings

Description:
Save the Setting for Key in Section in project AppName. Win16 and Win32s store settings in a .ini file named AppName. Win32 stores settings in the registration database.

Parameter	Description
AppName$	This string value is the name of the project which has this Section and Key.
Section$	This string value is the name of the section of the project settings.
Key$	This string value is the name of the key in the section of the project settings.
Setting	Set the key to this value. (The value is stored as a string.)

Example:

```
Sub Main
    SaveSetting "MyApp","Font","Size",10
End Sub
```

1.3.5.181 Second

Syntax:
Second(dateexpr)

Group: Time/Date

Description:
Return the second of the minute (0 to 59).

Parameter	Description
dateexpr	Return the second of the minute for this date value. If this value is Null then Null is returned.

See Also: Hour( ), Minute( ), Time( ).

Example:

```
Sub Main
    Debug.Print Second(#12:00:01 AM#) ' 1
End Sub
```

1.3.5.182 Seek

Syntax:
Seek [#]StreamNum, Count

Group: File

Description:
Position StreamNum for input Count.

Parameter	Description

StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
Count	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.

See Also: Seek( ).

Example:

```
Sub Main
   Open "XXX" For Input As #1
   Line Input #1,L$
   Seek #1,1 ' rewind to start of file
   Input #1,A
   Close #1
   Debug.Print A
End Sub
```

1.3.5.183 Select Case

Syntax:

```
Select Case expr
[Case caseexpr[, ...]
   statements]...
[Case Else
   statements]
End Select
```

Group: Flow Control

Description:
Select the appropriate case by comparing the expr with each of the caseexprs. Select the Case Else part if no caseexpr matches. (If the Case Else is omitted then skip the entire Select...End Select block.)

caseexpr	Description
expr	Execute if equal.
Is < expr	Execute if less than.
Is <= expr	Execute if less than or equal to.
Is > expr	Execute if greater than.
Is >= expr	Execute if greater than or equal to.
Is <> expr	Execute if not equal to.
expr1 To expr2	Execute if greater than or equal to expr1 and less than or equal to expr2.

See Also: If, Choose( ), Iif( ).

Example:

```
Sub Main
   S = InputBox("Enter hello, goodbye, dinner or sleep:")
   Select Case UCase(S)
   Case "HELLO"
      Debug.Print "come in"
   Case "GOODBYE"
      Debug.Print "see you later"
   Case "DINNER"
      Debug.Print "Please come in."
      Debug.Print "Dinner will be ready soon."
   Case "SLEEP"
      Debug.Print "Sorry."
      Debug.Print "We are full for the night"
   Case Else
      Debug.Print "What?"
   End Select
End Sub
```

1.3.5.184 SendKeys

Syntax:
SendKeys Keys$[, Wait]

Group: Miscellaneous

Description:
Send Keys$ to Windows.

Parameter	Description
Keys$	Send the keys in this string value to Windows. (Refer to table below.)
Wait	If this is not zero then the keys are sent before executing the next instruction. If this is omitted or zero then the keys are sent during the following instructions.

Key	Description
+	Shift modifier key: the following key is a shifted key
^	Ctrl modifier key: the following key is a control key
%	Alt modifier key: the following key is an alt key
(keys)	Modifiers apply to all keys
~	Send Enter key
k	Send k Key (k is any single char)
K	Send Shift k Key (K is any capital letter)
{special n}	special key (n is an optional repeat count)
{mouse x,y}	mouse key (x,y is an optional screen position)
{k}	Send k Key (any single char)
{K}	Send Shift k Key (any single char)
{Cancel}	Send Break Key
{Esc}	Send Escape Key
{Escape}	Send Escape Key
{Enter}	Send Enter Key
{Menu}	Send Menu Key (Alt)

{Help}	Send Help Key (?)
{Prtsc}	Send Print Screen Key
{Print}	Send
{Execute}	Send ?
{Tab}	Send
{Pause}	Send Pause Key
{Tab}	Send Tab Key
{BS}	Send Back Space Key
{BkSp}	Send Back Space Key
{BackSpace}	Send Back Space Key
{Del}	Send Delete Key
{Delete}	Send Delete Key
{Ins}	Send Insert Key
{Insert}	Send Insert Key
{Left}	Send Left Arrow Key
{Right}	Send Right Arrow Key
{Up}	Send Up Arrow Key
{Down}	Send Down Arrow Key
{PgUp}	Send Page Up Key
{PgDn}	Send Page Down Key
{Home}	Send Home Key
{End}	Send End Key
{Select}	Send ?
{Clear}	Send Num Pad 5 Key
{Pad0..9}	Send Num Pad 0-9 Keys
{Pad*}	Send Num Pad * Key
{Pad+}	Send Pad + Key
{PadEnter}	Send Num Pad Enter
{Pad.}	Send Num Pad . Key
{Pad-}	Send Num Pad - Key
{Pad/}	Send Num Pad / Key
{F1..24}	Send F1 to F24 Keys

Mouse:
Mouse movement and button clicks:

- {Move x,y} - move the mouse to (x,y)
- {ClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is the same as {DownLeft x,y}{UpLeft}.)
- {DoubleClickLeft x,y} - move the mouse to (x,y) and click the left button. (This is NOT the same as {ClickLeft x,y}{ClickLeft}.)
- {DownLeft x,y} - move the mouse to (x,y) and push the left button down.
- {UpLeft x,y} - move the mouse to (x,y) and release the left button.
- {...Middle x,y} - similarly named keys for the middle mouse button.
- {...Right x,y} - similarly named keys for the right mouse button.

The x,y values are screen pixel locations, where (0,0) is in the upper-left corner. In all cases the x,y is optional. If omitted, the previous mouse position is used.

See Also: AppActivate, KeyName, Shell( ).

Example:

```
Sub Main
   SendKeys "%S"     ' send Alt-S (Search)
   SendKeys "GoTo~~" ' send G o T o {Enter} {Enter}
End Sub
```

1.3.5.185 Set

Syntax:
Set objvar = objexpr
-or-
Set objvar = New objtype

Group: Assignment

Description:
Form 1: Set objvar's object reference to the object reference of objexpr.

Form 2: Set objvar's object reference to the a new instance of objtype.

The Set instruction is how object references are assigned.

Example:

```
Sub Main
   Dim App As Object
   Set App = CreateObject("WinWrap.CppDemoApplication")
   App.Move 20,30 ' move icon to 20,30
   Set App = Nothing
   App.Quit       ' run-time error (no object)
End Sub
```

1.3.5.186 SetAttr

Syntax:
SetAttr Name$, Attrib

Group: File

Description:
Set the attributes for file Name$. If the file does not exist then a run-time error occurs.

Parameter	Description
Name$	This string value is the path and name of the file. A path relative to the current directory can be used.
Attrib	Set the file's attributes to this numeric value.

Example:

```
Sub Main
   Attrib = GetAttr("XXX")
   SetAttr "XXX",1 ' readonly
   Debug.Print GetAttr("XXX") ' 1
   SetAttr "XXX",Attrib
End Sub
```

1.3.5.187 Sgn

Syntax:
Sgn(Num)

Group: Math

Description:
Return the sign.

Parameter	Description
Num	Return the sign of this numeric value. Return -1 for negative. Return 0 for zero. Return 1 for positive.

See Also: Abs.

Example:

```
Sub Main
   Debug.Print Sgn(9)  ' 1
   Debug.Print Sgn(0)  ' 0
   Debug.Print Sgn(-9) '-1
End Sub
```

1.3.5.188 Shell

Syntax:
Shell(Name$[, WindowType])

Group: Miscellaneous

Description:
Execute program Name$. This is the same as using File|Run from the Program Manager. This instruction can run .COM, .EXE, .BAT and .PIF files. If successful, return the task ID.

Parameter	Description
Name$	This string value is the path and name of the program to run. Command line arguments follow the program name. (A long file name containing a space must be surrounded by literal double quotes.)
WindowType	This controls how the application's main window is shown. See the table below.

WindowType	Value	Effect
vbHide	0	Hide Window
vbNormalFocus	1, 5, 9	Normal Window
vbMinimizedFocus	2	Minimized Window (default)
vbMaximizedFocus	3	Maximized Window

vbNormalNoFocus		
	4, 8	Normal Deactivated Window
vbMinimizedNoFocus		
	6, 7	Minimized Deactivated Window

See Also: AppActivate, SendKeys.

Example:

```
Sub Main
   X = Shell("Calc") ' run the calc program
   AppActivate X
   SendKeys "% R" ' restore calc's main window
   SendKeys "30*2{+}10=",1 '70
End Sub
```

1.3.5.189 ShowPopupMenu

Syntax:
ShowPopupMenu(StrArray$( )[, PopupStyle][, XPos, YPos])

Group: User Input

Description:
Show a popup menu and return the number of the item selected. The item number is the index of the StrArray selected minus LBound(StrArray). The value -1 is returned in no menu item is selected.

Parameter	Description
StrArray$( )	This one-dimensional array of strings establishes the list of choices. All the non-null elements of the array are used.
PopupMenuStyle	This controls how the popup menu is aligned. Any combination of styles may used together. See the table below.
XPos	When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.
YPos	When the menu is put up the alignment will be at this window position. If this is omitted then the current mouse position is used.

PopupStyle	Value	Effect
vbPopupLeftTopAlign	0	Align menu left edge at XPos and top at YPos. (default)
vbPopupUseLeftButton	1	User can select menu choices with the left mouse button only.
vbPopupUseRightButton	2	User can select menu choices with the left or right mouse button.
vbPopupRightAlign	4	Align menu with right edge at the XPos.
vbPopupCenterAlign	8	Align menu center at the XPos.
vbPopupVCenterAlign	16	Align menu center at the YPos.
vbPopupBottomAlign	32	Align menu bottom at the YPos.

Example:

```
Sub Main
    Dim Items(0 To 2) As String
    Items(0) = "Item &1"
    Items(1) = "Item &2"
    Items(2) = "Item &3"
    X = ShowPopupMenu(Items) ' show popup menu
    Debug.Print X ' item selected
End Sub
```

1.3.5.190 Sin

Syntax:
Sin(Num)

Group: Math

Description:
Return the sine.

Parameter	Description
Num	Return the sine of this numeric value. This is the number of radians. There are 2*Pi radians in a full circle.

See Also: Atn, Cos, Tan.

Example:

```
Sub Main
    Debug.Print Sin(1) ' 0.8414709848079
End Sub
```

1.3.5.191 Space

Syntax:
Space[$](Len)

Group: String

Description:
Return the string Len spaces long.

Parameter	Description
Len	Create a string this many spaces long.

See Also: String$( ).

Example:

```
Sub Main
    Debug.Print ".";Space$(3);"." "'.    ."
End Sub
```

1.3.5.192 Split

Syntax:
Split(Str, [Sep], [Max])

Group: Miscellaneous

Description:

Return a string array containing substrings from the original string.

Parameter	Description
Str	Extract substrings from this string value.
Sep	Look for this string value to separate the substrings. (Default: " ")
Max	Create at most this many substrings. (Default -1, which means create as many as are found.)

See Also: Join( ).

Example:

```
Sub Main
    Debug.Print Split("1 2 3")(1) "'2"
End Sub
```

1.3.5.193 Sqr

Syntax:
Sqr(Num)

Group: Math

Description:
Return the square root.

Parameter	Description
Num	Return the square root of this numeric value.

Example:

```
Sub Main
    Debug.Print Sqr(9) ' 3
End Sub
```

1.3.5.194 Static

Syntax:
Static name[type][([dim[, ...]])][As [New] type][, ...]

Group: Declaration

Description:
A static variable retains it value between procedure calls. Dimension var array(s) using the dims to establish the minimum and maximum index value for each dimension. If the dims are omitted then a scalar (single value) variable is defined. A dynamic array is declared using ( ) without any dims. It must be ReDimensioned before it can be used.

See Also: Dim, Option Base, Private, Public, ReDim.

Example:

```
Sub A
   Static X
   Debug.Print X
   X = "Hello"
End Sub

Sub Main
   A
   A ' prints "Hello"
End Sub
```

1.3.5.195 Stop

Syntax:
Stop

Group: Flow Control

Description:
Pause execution. If execution is resumed then it starts at the next instruction. Use End to terminate the macro completely.

Example:

```
Sub Main
   For I = 1 To 10
      Debug.Print I
      If I = 3 Then Stop
   Next I
End Sub
```

1.3.5.196 Str

Syntax:
Str[$](Num)

Group: String

Description:
Return the string representation of Num.

Parameter	Description
Len	Return the string representation of this numeric value. Positive values begin with a blank. Negative values begin with a dash '-'.

See Also: CStr( ), Hex$( ), Oct$( ), Val( ).

Example:

```
Sub Main
   Debug.Print Str$(9*9) ' 81
End Sub
```

1.3.5.197 StrComp

Syntax:
StrComp(Str1,Str2,Comp)

Group: String

Description:
Compare two strings.

Parameter	Description
Str1	Compare this string with Str2. If this value is Null then Null is returned.
Str2	Compare this string with Str1. If this value is Null then Null is returned.
Comp	This numeric value indicates the type of comparison. See Comp table below.

Result	Description
-1	Str1 is less than Str2.
0	Str1 is equal to Str2.
1	Str1 is greater than Str2.
Null	Str1 or Str2 is Null.

Comp	Value	Effect
vbUseCompareOption	-1	Performs the comparison using the Option Compare statement value.
vbBinaryCompare	0	Compares the string's binary data.
vbTextCompare	1	Compares the string's text using the collation rules.
vbDatabaseCompare	2	Microsoft Access only. (Not supported.)

See Also: LCase$( ), Option Compare, StrConv$( ), UCase$( ).

Example:

```
Sub Main
    Debug.Print StrComp("F","e") ' -1
    Debug.Print StrComp("F","e",1) ' 1
    Debug.Print StrComp("F","f",1) ' 0
End Sub
```

1.3.5.198 StrConv

Syntax:
StrConv[$](Str,Conv)

Group: String

Description:
Convert the string.

Parameter	Description
Str	Convert this string value. If this value is Null then Null is returned.
Conv	This numeric value indicates the type of conversion. See conversion table below.

Conv	Value	Effect
vbUpperCase	1	Convert to upper case.
vbLowerCase	2	Convert to lower case.
vbProperCase	3	Convert to proper case. (Not supported.)
vbWide	4	Convert to wide. (Only supported for Win32 in eastern locales.)
vbNarrow	8	Convert to narrow. (Only supported for Win32 in eastern locales.)

vbKatakana	16	Convert to Katakana. (Only supported for Win32 in Japanese locales.)
vbHiragana	32	Convert to Hiragana. (Only supported for Win32 in Japanese locales.)
vbUnicode	64	Convert to Unicode. (Only supported for Win32.)
vbFromUnicode	128	Convert from Unicode. (Only supported for Win32.)

See Also: LCase$( ), StrComp( ), UCase$( ).

Example:

```
Sub Main
   Dim B(1 To 3) As Byte
   B(1) = 65
   B(2) = 66
   B(3) = 67
   Debug.Print StrConv$(B,vbUnicode) "'ABC"
End Sub
```

1.3.5.199 String

Syntax:
String[$](Len, Char|$)

Group: String

Description:
Return the string Len long filled with Char or the first char of Char$.

Parameter	Description
Len	Create a string this many chars long.
Char|$	Fill the string with this char value. If this is a numeric value then use the ASCII char equivalent. If this is a string value use the first char of that string. If this value is Null then Null is returned.

See Also: Space$( ).

Example:

```
Sub Main
   Debug.Print String$(4,65)    "'AAAA"
   Debug.Print String$(4,"ABC") "'AAAA"
End Sub
```

1.3.5.200 StrReverse

Syntax:
StrReverse[$](S)

Group: String

Description:
Return the string with the characters in reverse order.

Parameter	Description
S	Return this string with the characters in reverse order.

Example:

```
Sub Main
   Debug.Print StrReverse$("ABC") 'CBA
End Sub
```

1.3.5.201 Sub

Syntax:

```
[ | Private | Public | Friend ] _
Sub name[([param[, ...]])]
   statements
End Sub
```

Group: Declaration

Description:
User defined subroutine. The subroutine defines a set of statements to be executed when it is called. The values of the calling arglist are assigned to the params. A subroutine does not return a result.

Sub defaults to Public if Private, Public or Friend are not is specified.

See Also: Declare, Function, Property.

Example:

```
Sub IdentityArray(A()) ' A() is an array of numbers
   For I = LBound(A) To UBound(A)
      A(I) = I
   Next I
End Sub

Sub CalcArray(A(),B,C) ' A() is an array of numbers
   For I = LBound(A) To UBound(A)
      A(I) = A(I)*B+C
   Next I
End Sub

Sub ShowArray(A()) ' A() is an array of numbers
   For I = LBound(A) To UBound(A)
      Debug.Print "(";I;")=";A(I)
   Next I
End Sub

Sub Main
   Dim X(1 To 4)
   IdentityArray X() ' X(1)=1, X(2)=2, X(3)=3, X(4)=4
   CalcArray X(),2,3 ' X(1)=5, X(2)=7, X(3)=9, X(4)=11
   ShowArray X()     ' print X(1), X(2), X(3), X(4)
End Sub
```

1.3.5.202 Tan

Syntax:

```
Tan(Num)
```

Group: Math

Description:

Return the tangent.

Parameter	Description
Num	Return the tangent of this numeric value.

See Also: Atn, Cos, Sin.

Example:

```
Sub Main
   Debug.Print Tan(1) ' 1.5574077246549
End Sub
```

1.3.5.203 Text

Syntax:
Text X, Y, DX, DY, Title$[, .Field][, Options]

Group: User Dialog

Description:
Define a text item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Title$	The value of this string is the title of the text control.
Field	This identifier is the name of the field. The dialogfunc receives this name as string. If this identifer is omitted then the first two words of the title are used.
Options	This numeric value controls the alignment of the text. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text is left aligned.
1	Text is right aligned.
2	Text is centered.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   Dialog dlg ' show dialog (wait for ok)
End Sub
```

1.3.5.204 TextBox

Syntax:
TextBox X, Y, DX, DY, .Field$[, Options]

Group: User Dialog

Description:
Define a textbox item.

Parameter	Description
X	This number value is the distance from the left edge of the dialog box. It is measured in 1/8 ths of the average character width for the dialog's font.
Y	This number value is the distance from the top edge of the dialog box. It is measured in 1/12 ths of the character height for the dialog's font.
DX	This number value is the width. It is measured in 1/8 ths of the average character width for the dialog's font.
DY	This number value is the height. It is measured in 1/12 ths of the character height for the dialog's font.
Field	The value of the text box is accessed via this field.
Options	This numeric value controls the type of text box. Choose one value from following table. (If this numeric value omitted then zero is used.)

Option	Description
0	Text box allows a single line of text to be entered.
1	Text box allows multiple lines of text can be entered.
-1	Text box allows a hidden password can be entered.

See Also: Begin Dialog, Dim As UserDialog.

Example:

```
Sub Main
   Begin Dialog UserDialog 200,120
      Text 10,10,180,15,"Please push the OK button"
      TextBox 10,25,180,20,.Text$
      OKButton 80,90,40,20
   End Dialog
   Dim dlg As UserDialog
   dlg.Text$ = "none"
   Dialog dlg ' show dialog (wait for ok)
   Debug.Print dlg.Text$
End Sub
```

1.3.5.205 Time

Syntax:
Time[$]

Group: Time/Date

Description:
Return the current time as a date value.

See Also: Date, Now, Timer.

Example:

```
Sub Main
    Debug.Print Time ' example: 09:45:00 am
End Sub
```

1.3.5.206 Timer

Syntax:
Timer

Group: Time/Date

Description:
Return the number of seconds past midnight. (This is a real number, accurate to about 1/18th of a second.)

See Also: Date, Now, Time.

Example:

```
Sub Main
    Debug.Print Timer ' example: 45188.13
End Sub
```

1.3.5.207 TimeSerial

Syntax:
TimeSerial(Hour, Minute, Second)

Group: Time/Date

Description:
Return a date value.

Parameter	Description
Hour	This numeric value is the hour (0 to 23).
Minute	This numeric value is the minute (0 to 59).
Second	This numeric value is the second (0 to 59).

See Also: DateSerial, DateValue, TimeValue.

Example:

```
Sub Main
    Debug.Print TimeSerial(13,30,0) '1:30:00 PM
End Sub
```

1.3.5.208 TimeValue

Syntax:
TimeValue(Date$)

Group: Time/Date

Description:
Return the time part of date encoded as a string value.

Parameter	Description
Date$	Convert this string value to the time part of date it represents.

See Also: DateSerial, DateValue, TimeSerial.

Example:

```
Sub Main
    Debug.Print TimeValue("1/1/2000 12:00:01 AM")
        '12:00:01 AM
End Sub
```

1.3.5.209 Trim

Syntax:
Trim[$](S$)

Group: String

Description:
Return the string with S$'s leading and trailing spaces removed.

Parameter	Description
S$	Copy this string without the leading or trailing spaces. If this value is Null then Null is returned.

See Also: LTrim$( ), RTrim$( ).

Example:

```
Sub Main
    Debug.Print "";Trim$("  x  ");"." "".x."
End Sub
```

1.3.5.210 Type

Syntax:

```
[ | Private | Public ] _
Type name
    elem [([dim[, ...]])] As [New] type
    [...]
End Type
```

Group: Declaration

Description:
Define a new usertype. Each elem defines an element of the type for storing data. As [New] type defines the type of data that can be stored. A user defined type variable has a value for each elem. Use .elem to access individual element values.

Type defaults to Public if neither Private or Public is specified.

Example:

```
Type Employee
   FirstName As String
   LastName As String
   Title As String
   Salary As Double
End Type

Sub Main
   Dim e As Employee
   e.FirstName = "John"
   e.LastName = "Doe"
   e.Title = "President"
   e.Salary = 100000
   Debug.Print e.FirstName "'John"
   Debug.Print e.LastName  "'Doe"
   Debug.Print e.Title     "'President"
   Debug.Print e.Salary    ' 100000
End Sub
```

1.3.5.211 TypeName

Syntax:
TypeName[$](var)

Group: Variable Info

Description:
Return a string indicating the type of value stored in var.

Parameter	Description
var	Return a string indicating the type of value stored in this variable.

Result	Description
Empty	Variant variable is empty. It has never been assigned a value.
Null	Variant variable is null.
Integer	Variable contains an integer value.
Long	Variable contains a long value.
Single	Variable contains a single value.
Double	Variable contains a double value.
Currency	Variable contains a currency value.
Date	Variable contains a date value.
String	Variable contains a string value.
Object	Variable contains an object reference that is not Nothing. (An object may return a type name specific to that type of object.)
Nothing	Variable contains an object reference that is Nothing.
Error	Variable contains a error code value.
Boolean	Variable contains a boolean value.
Variant	Variable contains a variant value. (Only used for arrays of variants.)
Unknown	Variable contains a non-ActiveX Automation object reference.
Byte	Variable contains a byte value.

( )	Variable contains an array value. The TypeName of the element followed by ( ).

See Also: VarType.

Example:

```
Sub Main
    Dim X As Variant
    Debug.Print TypeName(X) "'Empty"
    X = 1
    Debug.Print TypeName(X) "'Integer"
    X = 100000
    Debug.Print TypeName(X) "'Long"
    X = 1.1
    Debug.Print TypeName(X) "'Double"
    X = "A"
    Debug.Print TypeName(X) "'String"
    Set X = CreateObject("Word.Basic")
    Debug.Print TypeName(X) "'Object"
    X = Array(0,1,2)
    Debug.Print TypeName(X) "'Variant()"
End Sub
```

1.3.5.212 UBound

Syntax:
UBound(arrayvar[, dimension])

Group: Variable Info

Description:
Return the highest index.

Parameter	Description
arrayvar	Return the highest index for this array variable.
dimension	Return the highest index for this dimension of arrayvar. If this is omitted then return the highest index for the first dimension.

See Also: LBound( ).

Example:

```
Sub Main
    Dim A(3,6)
    Debug.Print UBound(A)   ' 3
    Debug.Print UBound(A,1) ' 3
    Debug.Print UBound(A,2) ' 6
End Sub
```

1.3.5.213 UCase

Syntax:
UCase[$](S$)

Group: String

Description:
Return a string from S$ where all the lowercase letters have been uppercased.

Parameter	Description
S$	Return the string value of this after all chars have been converted to lowercase. If this value is Null then Null is returned.

See Also: LCase$( ), StrComp( ), StrConv$( ).

Example:

```
Sub Main
   Debug.Print UCase$("Hello") "'HELLO"
End Sub
```

1.3.5.214 Unlock

Syntax:
Unlock StreamNum
-or-
Unlock StreamNum, RecordNum
-or-
Unlock StreamNum, [start] To end

Group: File

Description:
Form 1: Unlock all of StreamNum.

Form 2: Unlock a record (or byte) of StreamNum.

Form 3: Unlock a range of records (or bytes) of StreamNum. If start is omitted then unlock starting at the first record (or byte).

Note: For sequential files (Input, Output and Append) unlock always affects the entire file.

Parameter	Description
StreamNum	Streams 1 through 255 are private to each macro. Streams 256 through 511 are shared by all macros.
RecordNum	For Random mode files this is the record number. The first record is 1. Otherwise, it is the byte position. The first byte is 1.
start	First record (or byte) in the range.
end	Last record (or byte) in the range.

See Also: Lock, Open.

Example:

```
Sub Main
   Dim V As Variant
   Open "SAVE_V.DAT" For Binary As #1
   Lock #1
   Get #1, 1, V
   V = "Hello"
   Put #1, 1, V
   Unlock #1
   Close #1
End Sub
```

1.3.5.215 Uses

Syntax:
'#Uses "module" [Only:[Win16|Win32]] ...
-or-
'$Include: "module"

Description:
The Uses comment indicates that the current macro/module uses public and friend symbols from the module. The Only option indicates that the module is only loaded for that Windows platform.

Parameter	Description
module	Public and Friend symbols from this module are accessible. If the module name is a relative path then the path is relative to the macro/module containing the Uses comment. For example, if module "A:\B\C\D.BAS" has this uses comment: '#Uses "E.BAS" then it uses "A:\B\C\E.BAS".

See Also: Class Module, Code Module, Object Module.

Example:

```
'Macro A.BAS
'#Uses "B.BAS"
Sub Main
    Debug.Print BFunc$("Hello") "'HELLO"
End Sub

'Module B.BAS
Public Function BFunc$(S$)
    BFunc$ = UCase(S$)
End Function
```

1.3.5.216 Val

Syntax:
Val(S$)

Group: String

Description:
Return the value of the S$.

Parameter	Description
S$	Return the numeric value for this string value. A string value begins with &O is an octal number. A string value begins with &H is a hex number. Otherwise it is decimal number.

Example:

```
Sub Main
    Debug.Print Val("-1000") '-1000
End Sub
```

1.3.5.217 VarType

Syntax:
VarType(var)

Group: Variable Info

Description:
Return a number indicating the type of value stored in var.

Parameter	Description
var	Return a number indicating the type of value stored in this variable.

Result	Value	Description
vbEmpty	0	Variant variable is empty. It has never been assigned a value.
vbNull	1	Variant variable is null.
vbInteger	2	Variable contains an integer value.
vbLong	3	Variable contains a long value.
vbSingle	4	Variable contains a single value.
vbDouble	5	Variable contains a double value.
vbCurrency	6	Variable contains a currency value.
vbDate	7	Variable contains a date value.
vbString	8	Variable contains a string value.
vbObject	9	Variable contains an object reference.
vbError	10	Variable contains a error code value.
vbBoolean	11	Variable contains a boolean value.
vbVariant	12	Variable contains a variant value. (Only used for arrays of variants.)
vbDataObject	13	Variable contains a non-ActiveX Automation object reference.
vbDecimal	14	Variable contains a 96 bit scaled real.
vbByte	17	Variable contains a byte value.
vbUserDefinedType	36	Variable contains a User Defined Type value.
+vbArray	8192	Variable contains an array value. Use VarType( ) And 255 to get the type of element stored in the array.

See Also: TypeName.

Example:

```
Sub Main
    Dim X As Variant
    Debug.Print VarType(X) ' 0
    X = 1
    Debug.Print VarType(X) ' 2
    X = 100000
    Debug.Print VarType(X) ' 3
    X = 1.1
    Debug.Print VarType(X) ' 5
    X = "A"
    Debug.Print VarType(X) ' 8
    Set X = CreateObject("Word.Basic")
    Debug.Print VarType(X) ' 9
    X = Array(0,1,2)
    Debug.Print VarType(X) ' 8204 (8192+12)
End Sub
```

1.3.5.218 Wait

Syntax:
Wait Delay

Group: Miscellaneous

Description:
Wait for Delay seconds.

Example:

```
Sub Main
    Wait 5 ' wait for 5 seconds
End Sub
```

1.3.5.219 Weekday

Syntax:
Weekday(dateexpr)

Group: Time/Date

Description:
Return the weekday.

- vbSunday (1) - Sunday
- vbMonday (2) - Monday
- vbTuesday (3) - Tuesday
- vbWednesday (4) - Wednesday
- vbThursday (5) - Thursday
- vbFriday (6) - Friday
- vbSaturday (7) - Saturday

Parameter	Description
dateexpr	Return the weekday for this date value. If this value is Null then Null is returned.

See Also: Date( ), Day( ), Month( ), WeekdayName( ), Year( ).

Example:

```
Sub Main
   Debug.Print Weekday(#1/1/1900#) ' 2
   Debug.Print Weekday(#1/1/2000#) ' 7
End Sub
```

1.3.5.220 WeekdayName

Syntax:
WeekdayName(NumZ{day}[, CondZ{abbrev}])

Group: Time/Date

Description:
Return the localized name of the weekday.

Parameter	Description
day	Return the localized name of this weekday. (1-7)
abbrev	If this conditional value is True then return the abbreviated form of the weekday name.

See Also: Weekday( ).

Example:

```
Sub Main
   Debug.Print WeekdayName(1) 'Sunday
   Debug.Print WeekdayName(Weekday(Now))
End Sub
```

1.3.5.221 While

Syntax:
```
While condexpr
   statements
Wend
```

Group: Flow Control

Description:
Execute statements while condexpr is True.

See Also: Do, For, For Each, Exit While.

Example:

```
Sub Main
   I = 2
   While I < 10
      I = I*2
   Wend
   Debug.Print I ' 16
End Sub
```

1.3.5.222 With

Syntax:
```
With objexpr
   statements
```

End With

Group: Object

Description:
Method and property references may be abbreviated inside a With block. Use .method or .property to access the object specified by the With objexpr.

Example:

```
Sub Main
   Dim App As Object
   Set App = CreateObject("WinWrap.CppDemoApplication")
   With App
      .Move 20,30 ' move icon to 20,30
   End With
End Sub
```

1.3.5.223 Write

Syntax:
Write #StreamNum, expr[, ...]

Group: File

Description:
Write's expr(s) to StreamNum. String values are quoted. Null values are written as #NULL#. Boolean values are written as #FALSE# or #TRUE#. Date values are written as #date#. Error codes are written as #ERROR number#.

See Also: Input, Line Input, Print.

Example:

```
Sub Main
   A = 1
   B = 2
   C$ = "Hello"
   Open "XXX" For Output As #1
   Write #1,A,B,C$
   Close #1
End Sub
```

1.3.5.224 Year

Syntax:
Year(dateexpr)

Group: Time/Date

Description:
Return the year.

Parameter	Description
dateexpr	Return the year for this date value. If this value is Null then Null is returned.

See Also: Date( ), Day( ), Month( ), Weekday( ).

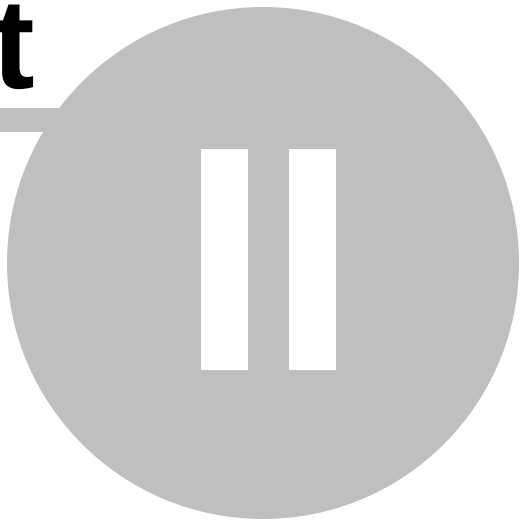
Example:

```
Sub Main
   Debug.Print Year(#1/1/1900#) ' 1900
   Debug.Print Year(#1/1/2000#) ' 2000
End Sub
```

Top Level Intro

This page is printed before a new top-level chapter starts

Part



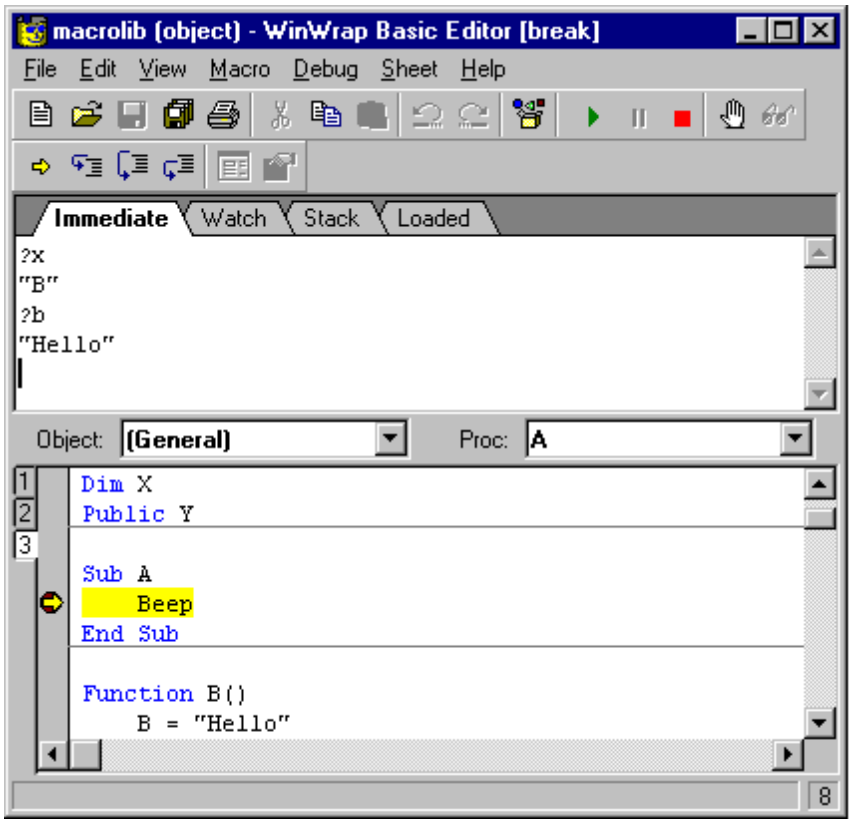
2 Sax Basic Editor

2.1 Sax Basic Editor

The Sax Basic Editor is an interactive design environment for developing, testing and executing Sax Basic scripts.

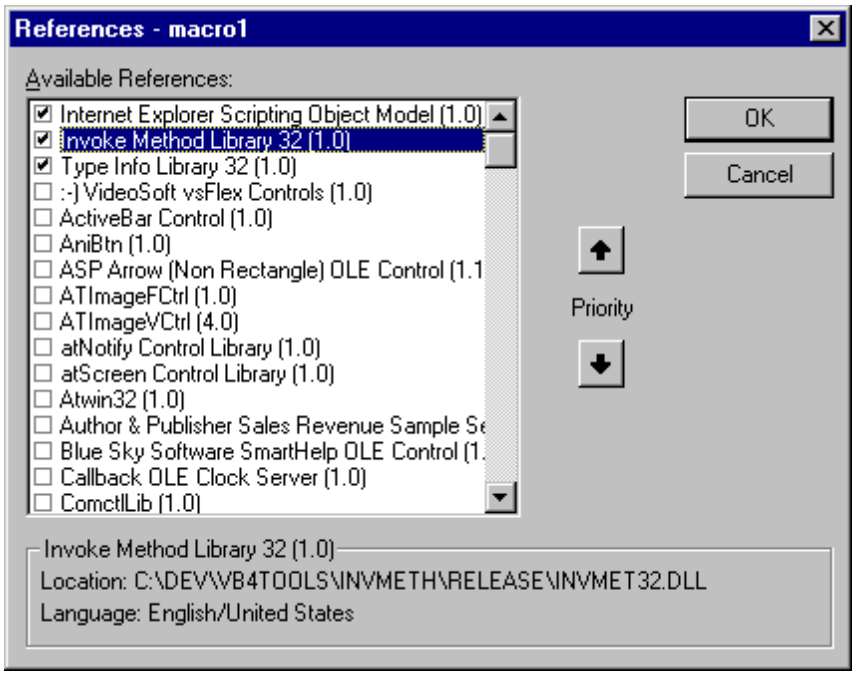
- IDE
- References
- Object Browser
- UserDialog Editor
- Sax Basic Language

2.2 IDE

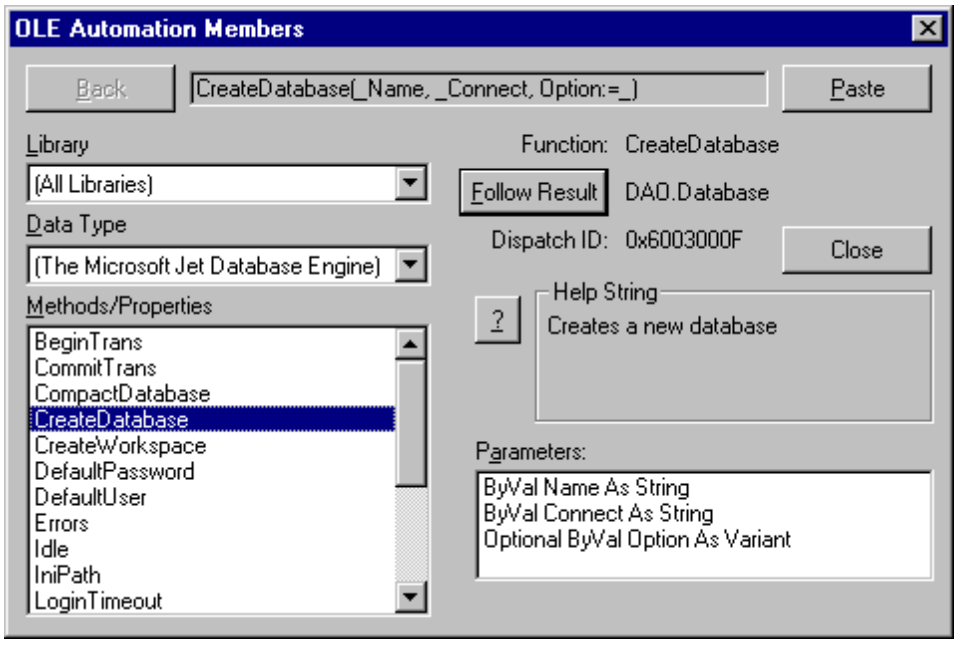


- File, Edit, View, Macro, Debug, Sheet and Help Menus
- Immediate, Watch, Stack and Loaded Windows
- Object and Proc Lists
- Sheet Tabs
- Break Bar

References Overview



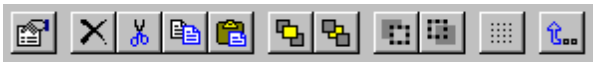
2.3 Object Browser



The Object Browser shows information about all the special data types that are available.

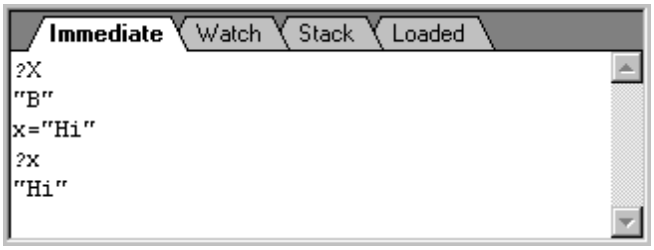
IDE:
Auto-completion in the IDE uses the object browser information to show the current object's methods and properties. To see language extensions, built-in instructions/functions and user-defined procedures/ variables press Ctrl-Space on a blank line in the IDE.

2.4 UserDialog Editor



A UserDialog is described by a Begin Dialog...End Dialog block. To graphically edit a UserDialog place the current selection in a UserDialog block and select Edit|UserDialog.

Immediate Window



Index

- D -

- E -

- F -

- G -

- H -

- I -

- J -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- Y -