Builtin Functions

A builtin functions is a predefined function, which has a special meaning for KBasic and which meaning cannot be changed. Many of them are provided for VB6 and QBasic backward compatibility. The following list contains all KBasic builtin functions.

Table of Contents

__Class__(Macro) , __File__(Macro) , __IsClass__(Macro) , __IsLinux__(Macro) , __IsMacOS__(Macro) , __IsModule__(Macro) , __IsSub__(Macro) , __IsWindows__(Macro) , __Line__(Macro) , __Module__(Macro) , __Scope__(Macro) , __Sub__(Macro) , Abs , Access , Acs , AddHandler , AppActiviate , Append , Array , Asc , Asn , Atn , Beep , Bin , Bin$, Binary (Builtin) , BLOAD , BSAVE , CBCD , CBool , CByte , CChar , CCur , CDate , CDbl , CDec , CEXT , CFIX , ChDir , ChDrive , Chr , Chr$, CInt , Circle , Clear , CLng , Close , CLS , CObj , Color , Command , Command$, Cos , CQUD , CreateObject , CShort , CSng , CsrLin , CType , CurDir , CurDir$, CVD , CVDMBF , CVERR , CVI , CVL , CVS , CVSMBF , Date , Date$, DateAdd , DateDiff , DatePart , DateSerial , DateValue , Day , DDB , Deg , DeleteSetting , Dir , Dir$, DoEvents , DOF , Draw , Environ , Environ$, EOF , ErDev , ErDev$, Erl , Err , Error , Error$, Exp , Fact , Field , FileAttr , FileCopy , FileDateTime , FileLen , Files , Filter , Fix , FN , Format , Format$, FormatCurrency , FormatDateTime , FormatNumber , FormatPercent , Frac , FRE , FreeFile , FV , Get , GetAllSettings , GetAttr , GetAutoServerSettings , GetObject , GetSetting , GetType , Hex , Hex$, Hour , Hypot , IMEStatus , Inkey , Inkey$, Inp , Input , Input$, InputBox , InStr , InStRev , Int , IOCtl , IOCtl$, IPMT , IRR , IsArray , IsBoolean , IsByte , IsCharacter , IsCollection , IsCString , IsCurrency , IsDate , IsDouble , IsEmpty , IsError , IsInteger , IsMissing , IsNull , IsNumeric , IsObject , IsShort , IsSingle , IsLong , IsString , IsVariant , Join , Kill , LCase , LCase$, Left , Left$, Len , Line , Ln , Load , LoadPicture , LoadResData , LoadResPicture , LoadResString , Loc , Locate , Lock , LOF , Log , Logb , LPos , LPrint , LTrim , LTrim$, Max , Mid (Builtin) , Mid$ (Builtin) , Min , Minute , MIRR , MKD$, MkDir , MKDMBF$, MKI$, #MKL$, MKS , MKS$, MKSMBF$, Month , MonthName , MsgBox , MTIMER , Name , Now , NPER , NPV , Nz , Oct , Oct$, Open , Out , Output , Paint , Palette , Partition , PCopy , Peek , PMAP , PMT , Point , Poke , Pos , PPMT , Preset , Print , PrintScreen , PSet , Put , PV , QBCOLOR , Rad , Raise , RaiseEvent , RaiseSignal , Random , Randomize , Rate , RemoveHandler , Replace , Reset , RGB , Right , Right$, RmDir , RND , Round , RTrim , RTrim$, SavePicture , SaveSetting , Screen , Sec , Second , Seek , Seg , SendKeys , SetAttr , Sgn , Shell , Sin , Sleep , Sln , Sound , Space , Space$, Spc , Split , Sqr , Stick , Str , Str$, StrComp , StrConv , String , String$, StrReverse , SYD , Tab , Tan , Time , Time$, TimeSerial , TimeValue , Trim , Trim$, TypeName , UCase , UCase$, UnLoad , UnLock , Using , Val , VarType , View , Weekday , WeekdayName , Width , Window , Write , Year

The following list is recommended for new application development. Instead of using the old builtin functions, you ought to use the new KBasic Framework.

__Class__(Macro) , __File__(Macro) , __IsClass__(Macro) , __IsLinux__(Macro) , __IsMacOS__(Macro) , __IsModule__(Macro) , __IsSub__(Macro) , __IsWindows__(Macro) , __Line__(Macro) , __Module__(Macro) , __Scope__(Macro) , __Sub__(Macro) , Abs , Asc , Bin , CBool , CByte , CDbl , Chr , CInt , Cos , CShort , CSng , DoEvents , Exp , Fact , FileCopy , FileLen , Fix , Format , FormatDateTime , Hex , InputBox , InStr , InStRev , Int , LCase , Left , Len , Log , LTrim , Max , Mid (Builtin) , Min , MsgBox , Nz , Print , Random , Randomize , Replace , Right , Rnd , RTrim , Sgn , Sin , Space , Sqr , Str , StrComp , String , StrReverse , Tan , Trim , TypeName , UCase , Using , Val

The following list contains the reserved builtin functions without functionality yet.

Acs , AddHandler , AppActiviate , Asn , Atn , BLOAD , BSAVE , CBCD , CChar , CDec , CEXT , CFIX , CObj , CQUD , CreateObject , CType , CVD , CVDMBF , CVI , CVL , CVS , CVSMBF , DDB , Deg , DeleteSetting , DOF , Draw , Environ , Environ$, ErDev , ErDev$, Fact , Field , Filter , FN , FormatCurrency , FormatNumber , FormatPercent , Frac , FV , GetAllSettings , GetAutoServerSettings , GetObject , GetSetting , GetType , Hypot , IMEStatus , Inp , IOCtl , IOCtl$, IPMT , IRR , IsCharacter , IsCollection , IsCString , Join , Ln , Load , LoadPicture , LoadResData , LoadResPicture , LoadResString , Logb , LPos , LPrint , MIRR , MKD$, MKDMBF$, MKI$, MKL$, MKS , MKS$, MKSMBF$, MTIMER , NPER , NPV , Out , Paint , Palette , Partition , PCopy , Peek , PMAP , PMT , Point , Poke , PPMT , Preset , PV , QBCOLOR , RaiseEvent , RaiseSignal , Rate , RemoveHandler , Round , SavePicture , SaveSetting , Screen , Sec, Seg , SendKeys , SetAttr , Sln , Sound , Split , Stick , StrConv , SYD , UnLoad , UnLock , View , Width , Window

---

Descriptions

__Class__ (Macro)

This macro is replaced by the current class name at runtime. Use this for more information on your error messages.

```
Class lordoftherings

  Sub gandalf()

    Dim s As String

    If __IsClass__ Then
      s = __Class__
    Else
      s = ""
    End If

    Print "Gandalf is inside the class " + s

  End Sub

End Class


' main part

Dim c As lordoftherings

c = New lordoftherings
c.gandalf()

If __IsClass__ Then
  Print "inside a class"
Else
  Print "is not inside a class"
EndIf
```

__File__ (Macro)

This macro is replaced by the current file name at runtime. Use this for more information on your error messages.

__IsClass__ (Macro)

This macro is replaced by 'True' if current scope is a class at runtime. Use this for more information on your error messages.

__IsLinux__ (Macro)

This macro is replaced by 'True' if your program is running on Linux. Use this for more information on your error messages.

__IsMacOS__ (Macro)

This macro is replaced by 'True' if your program is running on Mac. Use this for more information on your error messages.

__IsModule__ (Macro)

This macro is replaced by 'True' if current scope is inside a module at runtime. Use this for more information on your error messages.

__IsSub__ (Macro)

This macro is replaced by 'True' if current scope is inside a sub or function. Use this for more information on your error messages.

__IsWindows__ (Macro)

This macro is replaced by 'True' if your program is running on Windows. Use this for more information on your error messages.

__Line__ (Macro)

This macro is replaced by the current line at runtime. Use this for more information on your error messages.

__Module__ (Macro)

This macro is replaced by the current module name at runtime. Use this for more information on your error messages.

__Scope__ (Macro)

This macro is replaced by the current scope name at runtime (might be class/module/and/or sub/function. Use this for more information on your error messages.

__Sub__ (Macro)

This macro is replaced by the current sub or function name at runtime. Use this for more information on your error messages.

--------A--------

Abs

Function Abs(EXPRESSION) As Double

Returns the absolute value of an numerical expression.

The absolute value of a number is its positive value. For instance, the absolute value of -3 equals 3 and that of +3 equals 3, too.

The required number argument can be any valid numeric expression.

Example

```
'use Abs to find the difference
'between 2 values
value1 = 11
value2 = 17
Print "The difference is "; Abs(value1 - value2)

Output:
The difference is 6
```

See also Sgn

Access

Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] Access [Read|Write|Read Write] As INTEGEREXPRESSION VB6! QB!

Access is not supported, but you may use still this syntax.

Example

```
Dim TextLine As String, ff As Integer

ff = FreeFile ' next availaible filehandle

Open "c:\kbasic15\examples\test\test.txt" For Input Access Read As #ff ' open
test file

Do While Not EOF(ff) ' while end of file has not been reached
   Line Input #ff, TextLine ' store next line in string
   print TextLine
Loop

Close #ff ' close file
```

Append

Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As INTEGEREXPRESSION VB6! QB!

Opens a file for appending. Supported for backward compatibility.

Example

```
OPTION OLDBASIC

DIM Rec1$, Rec2$

CLS
 OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR APPEND AS #1
 DO
     INPUT "   NAME:        ", Name$
     INPUT "   AGE:         ", Age$
     WRITE #1, Name$, Age$
     INPUT "More entries?"; R$
 LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1

 'print file on screen
 OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR INPUT AS #1
 CLS
 PRINT "Entries of file:": PRINT
DO WHILE NOT EOF(1)
    INPUT #1, Rec1$, Rec2$
    PRINT Rec1$, Rec2$
LOOP
```

```
CLOSE #1
KILL "LIST"
```

Array

Function Array(ARGUMENTS) As Variant VB6! QB!

Creates an array of variant values.

If no arguments are specified, an array of zero length is created.

Example

```
Dim A As Variant
A = Array(10,20,30)
B = A(2)
```

Asc

Function Asc(String) As Integer

Returns the ASCII code for the first character of a STRING. A STRING of length zero returns 0.

Example

```
PRINT ASC("Bernd") ' will show 66
```

See also Chr

--------B--------

Beep

Sub Beep() VB6! QB!

Produces a sound. Depends on the system if it works.

Example

```
Beep
```

Bin

Function Bin(EXPRESSION) As String

Bin or BIN$ returns a string giving the binary (base 2) representation of ‘number’. The return string has as many characters as necessary to represent the integer in binary.

Example

```
PRINT BIN$(128)
```

See also Hex

Binary (Builtin)

Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As INTEGEREXPRESSION VB6! QB!

Opens a file in binary mode. Supported for backward compatibility.

Example

```
OPTION OLDBASIC

DIM Name$, Age$, R$, Rec1$, Rec2$



CLS
OPEN "c:\kbasic15\examples\test\LISTEN2.txt" FOR BINARY AS #1
DO
     INPUT "   NAME:        ", Name$
     INPUT "   AGE:         ", Age$
     WRITE #1, Name$, Age$
     INPUT "More entries?"; R$
LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1

'print file on screen
OPEN "c:\kbasic15\examples\test\LISTEN2.txt" FOR INPUT AS #1
CLS
PRINT "Entries of file:": PRINT
DO WHILE NOT EOF(1)
    INPUT #1, Rec1$, Rec2$
    PRINT Rec1$, Rec2$
LOOP
CLOSE #1
```

--------C--------

CBool

Function CBool(EXPRESSION) As Boolean

It converts any number to a boolean.

Example

```
PRINT CBOOL(300.5012)

' Output:
' true
```

CByte

Function CByte(EXPRESSION) As Byte

It converts any number to a byte.

Example

```
PRINT CByte(30.5012)

' Output:
' 30
```

CCur

Function CCur(EXPRESSION) As Currency VB6! QB!

It converts any number to a currency.

Example

```
PRINT CCUR(8.8)
```

CDate

Function CDate(EXPRESSION) As Date VB6! QB!

It converts any number to a date.

Example

```
'PRINT CDATE(899999998) ' integer not allowed
PRINT CDATE("2006-12-12") ' must be like this format yyyy-mm-dd
```

CDbl

Function CDbl(EXPRESSION) As Double

It converts any number to a double. CDbl takes any number and tries to convert it to a double.

Example

```
PRINT CDBL(300.5012)

' Output:
' 300.5012
```

See also CSng, CInt, CLng

ChDir

Sub ChDir(String) VB6! QB!

Changes the current directory to new one.

Example

```
MKDIR "C:\TEMP\TEST"
CHDIR "C:\TEMP"
FILES
RMDIR "TEST"
```

See also CurDir, MkDir, ChDrive, Files

ChDrive

Sub ChDrive(String) VB6! QB!

Changes to another current drive.

Example

```
CHDRIVE "D" ' change to D:
```

Chr

Function Chr(EXPRESSION) As String

Returns the ASCII character corresponding to the value of Val. EXPRESSION must be a numerical expression.

Example

```
PRINT CHR(34)
```

See also Asc

CInt

Function CInt(EXPRESSION) As Integer

It converts any number to an integer and takes any number and convert it to an integer. This will remove any precision from a single or a double.

Example

```
PRINT CINT(300.5012)
'
' Output:
' 300
```

See also CLng

Circle

Sorry. Not supported yet. VB6! QB!

Clear

Sub Clear() VB6! QB!

It is a method of the error object, which is provided for VB6 backward compatibility. Use exception handling instead, e.g. Try.

Example

```
Dim Msg

On Error Resume Next

Err.Clear
Err.Raise(6)

If Err.Number <> 0 Then

   Msg = "Error # " & Str(Err.Number) & " " _
       & Err.Source & Chr(10) & Err.Description

    Print Msg

End If
```

See also Try

CLng

Function CLng(EXPRESSION) As Long

It takes any number and tries to convert it to a long. This will remove any precision from a single or a double.

Example

```
PRINT CINT(300.5012)
'
' Output:
' 300
```

See also CInt

Close

Sub Close [[#]FileNo As Integer] {[, [#]FileNo As Integer]} VB6! QB!

Closes the specified file(s).

Example

```
Dim I, filename
For I = 1 To 3 ' repeat loop 3 times
  filename = "TEST" & I ' create filename
  Open filename For Output As #I ' open file
  Print #I, "Ein Test." ' write string into file
Next I
Close ' close all 3 opened files
```

See also Open, Reset

CLS

Sub CLS() VB6! QB!

In the terminal screen, CLS will clear the screen and returns the cursor to the upper left corner (line 1, column 1).

Example

```
' CLS clearing the terminal screen
' with a new background color

PRINT "This is to show the CLS command"
INPUT "To clear the screen, press [Return]", keypressed$

' changes the background color:
COLOR (2, 4)
CLS
```

```
PRINT "This is green text on a blue screen!"
```

Color

Sub Color(Fore As Integer [, Back As Integer]) VB6! QB!

Calling COLOR will set the color of either the foreground and optionally the background. Passing only one integer will change the fore color. Passing 2 will change both the fore and background color.

Example

```
COLOR(5)
PRINT "Hi"
COLOR(15,1)
PRINT "Nadja"
```

Command

Function Command() As String VB6! QB!

Returns the arguments which have been given to your application by the OS while starting.

Example

```
PRINT COMMAND()
```

Cos

Function Cos(EXPRESSION) As Double

It returns the cosine of the argument 'number' in radians. EXPRESSION must be a numerical expression.

Example

```
CONST PI=3.141592654
PRINT ATN(TAN(PI/4.0)), PI/4.0 'result: .7853981635 .7853981635
PRINT (COS(180 * (PI / 180))) 'result: -1
PRINT (SIN(90 * (PI / 180))) 'result: 1
PRINT (TAN(45 * (PI / 180))) 'result: 1.000000000205103
```

See also Sin, Tan

CShort

Function CShort(EXPRESSION) As Short

Sorry. Not implemented yet. Use CInt instead of it.

See also CInt

CSng

Function CSng(EXPRESSION) As Single

It converts any number to a single. EXPRESSION must be a numerical expression.

Example

```
PRINT CSNG(300.5012)

' Output:
' 300.5012
```

See also CDbl

CSRLin

Function CSRLin() As Integer VB6! QB!

It returns the current line of the cursor.

Example

```
PRINT CSRLIN()

PRINT "row = " + POS(0)

INPUT s$

PRINT "line = " + CSRLIN

PRINT s$
```

See also Pos, Locate

CurDir

Function CurDir([Drive As String]) As String VB6! QB!

It returns the current path.

Example

```
' Windows:
' current path of C: ist "C:\WINDOWS\SYSTEM32".
' current path of D: ist "D:\kbasic".
' C: is the active drive.
Dim path
path = CurDir' returns "C:\WINDOWS\SYSTEM32".
path = CurDir("C") ' returns "C:\WINDOWS\SYSTEM32".
path = CurDir("D") ' returns "D:\kbasic".
```

CVErr

Function CVErr(EXPRESSION) As Variant VB6! QB!

It returns a user defined error.

Example

```
Option OldBasic

Sub test()
 Print doubleit("395.45bernd")
End Sub

Function doubleit(no)
 If IsNumeric(no) Then
  doubleit = no* 2 ' return result
 Else
  doubleit = CVErr(2001) ' return user defined error
 End If
End Function

test()
```

--------D--------

Date

Function Date() As String VB6! QB!

Date or DATE$ returns the current system date as a string. Setting the date is not possible with this builtin function.

Example

```
PRINT DATE
```

See also Time

DateAdd

Function DateAdd(Interval As String, Number As Integer, DateToChange As Date) As Date

VB6! QB!

Adds something to a date.

interval values:

- yyyy year
- q quarter
- m month
- y day of year
- d day
- w weekday
- ww week
- h hour
- n minute
- s second

Example

```
Dim Date1 As Date
Dim Interval As String
Dim Number As Integer
Dim Msg
Interval = "m"
Date1 = InputBox("Input the date") ' #yyyy-mm-dd#
Number = Val(InputBox("Input the number of months to add"))
Msg = "New date: " & DateAdd(Interval, Number, Date1)
MsgBox Msg
```

See also DateDiff, DatePart, DateSerial, DateValue

DateDiff

Function DateDiff(Interval As String, Date1 As Date, Date2 As Date[, FirstDayOfWeek As String[, FirstWeekOfYear As String]]) As Date VB6! QB!

Returns the number of interval laying between date1 and date2.

interval values:

- yyyy year
- q quarter
- m month
- y day of year
- d day
- w weekday
- ww week
- h hour

- n minute
- s second

Example

```
Dim Date1 As Date
Dim Msg
Date1 = InputBox("Input the date")
Msg = "Days till today: " & DateDiff("d", Now, Date1)
MsgBox Msg
```

See also DateAdd, DatePart, DateSerial, DateValue

DatePart

Function DatePart(Interval As String, DateToAsk As Date [, FirstDayOfWeek As String[, FirstWeekOfYear As String]]) As Integer VB6! QB!

Returns the desired part of a date.

interval values:

- yyyy year
- q quarter
- m month
- y day of year
- d day
- w weekday
- ww week
- h hour
- n minute
- s second

Example

```
Dim Date1 As Date
Dim Msg
Date1 = InputBox("Input a date:")
Msg = "quarter: " & DatePart("q", Date1)
MsgBox Msg
```

See also DateAdd, DateDiff, DateSerial, DateValue

DateSerial

Function DatePart(Year As Integer, Month As Integer, Day As Integer) As Date VB6! QB!

Converts a date given by year, month and day into a big number.

Example

```
Dim Date1
Date1 = DateSerial(1969, 2, 12)' return Date1
```

See also DateAdd, DateDiff, DatePart, DateValue

DateValue

Function DateValue(STRINGEXPRESSION) As Date VB6! QB!

Converts a date given in a string into a date type.

Example

```
Dim Date1
Date1 = DateValue("1979-02-03")
```

See also DateAdd, DateDiff, DatePart, DateSerial

Day

Function Day(DATEEXPRESSION) As Integer VB6! QB!

Returns the day part of an date expression.

Example

```
Dim Date1, Day1
Date1 = #2006-12-12#
Day1 = Day(Date1) ' --> 12
```

Dir

Function Dir([Path As String [, Attribute]]) As String VB6! QB!

Attributes:

- kbNormal 0 normal
- kbHidden 2 hidden
- kbSystem 4 system file
- kbVolume 8 volume name
- kbDirectory 16 directory

Example

```
file1 = Dir("C:\WINDOWS\*.INI")
```

```
Path1 = "c:\"
Name1 = Dir(Path1, kbDirectory) ' first entry
Do While Name1 "" ' loop
If Name1 "." And Name1 ".." Then
If (GetAttr(Path1 & Name1) And kbDirectory) = kbDirectory Then
Print Name1
End If
End If
Name1 = Dir ' next entry
Loop
```

DoEvents

Sub DoEvents()

Enables the application to process events.

Example

```
DoEvents()
```

--------E--------

EOF

Function EOF**([#]FileNo) As Boolean** VB6! QB!

It returns true if end of file has been reached. In other words, it checks if end of file has been reached and return true if it has happened.

Example

```
Dim data
Open "file1" For Input As #1 ' open file for reading
Do While Not EOF(1) ' test for end of file
  Line Input #1, data ' get the data from file
  Print data
Loop
Close #1 ' close file
```

See also Open, Write, LOF, Close, LOC

Erl

Function Erl() As Integer VB6! QB!

It returns the line in which the last error occured.

Example

```
PRINT ERL
```

See also Error, Resume, Err, On Error

Err

Function Err() As Integer VB6! QB!

As for VeryOldBasic, it returns the runtime error code, as for OldBasic it is an object for error handling.

Properties (OldBasic):

- Property Number As Integer (ReadOnly)
- Property Source As String (ReadOnly)
- Property Description As String (ReadOnly)

Methods (OldBasic):

- Sub Clear() ' reset
- Sub Raise(Number As Integer, Source As String, Description As String) ' raise error

Example

```
Dim Msg
On Error Resume Next
Err.Clear
Err.Raise 6
If Err.Number 0 Then
  Msg = "Error # " & Str(Err.Number) & " " _
  & Err.Source & Chr(13) & Err.Description
  MsgBox Msg, , "Error"
End If
```

See also Error, Resume, Err, On Error

Error

Function Error(EXPRESSION) As String VB6! QB!

Simulates an error. EXPRESSION must be a numerical expression.

Example

```
ERROR 4 ' throws an error

Dim errorno
For errorno = 61 To 64
Print Error(errorno)
```

```
Next
```

See also Erl, Resume, Err, On Error

Exp

Function Exp(EXPRESSION) As Double

It returns the exponential value of an expression. EXPRESSION must be a numerical expression.

Example

```
PRINT EXP(0), EXP(1) 'result: 1 2.718282
PRINT LOG(1), LOG(EXP(1)) 'result: 0 1
```

See also Log

--------F--------

FileAttr

Function FileAttr([#]FileNo As Integer, ReturnType As Integer) As Double VB6! QB!

It returns the access mode for an opened file.

if ReturnType = 1 then the following values can be returned:

- Input 1
- Output 2
- Random 4
- Append 8
- Binary 32

Example

```
Dim filehandle, Mode
filehandle = 1
Open "file1" For Append As filehandle
Mode = FileAttr(filehandle, 1) ' returns 8 (Append).
Close filehandle ' close file
```

See also Open

FileCopy

Sub FileCopy(Source As String, Destination As String)

It copies a file from source to destination.

Example

```
FILECOPY "c:\kbasic\examples\test\test.dat", "c:\kbasic\examples\test\test2.dat"

FILECOPY "test.dat", "test2.dat"
FILECOPY "test2.dat", "test.dat"
```

FileDateTime

Function FileDateTime(FileName As String) As Date VB6! QB!

It returns the date of the file.

Example

```
Print FileDateTime("c:\kbasic14\examples\test\liste.txt")
```

FileLen

Function FileLen(FileName As String) As Long

It returns the length of a file in bytes.

Example

```
Print FileLen("c:\kbasic\parser.cpp")
```

Files

Sub Files() VB6! QB!

Provided for QBasic compatibility.

Example

```
PRINT "listening of directory"
FILES
```

Fix

Function Fix(EXPRESSION) As Long

It cuts off the trail of a number. EXPRESSION must be a numeric expression.

Example

```
PRINT FIX(12.49), FIX(12.54) 'result: 12 12
```

See also Int, CInt, CLng

Format

Function Format(STRINGEXPRESSION[{, EXPRESSION,...]}) As String

Sorry. Not implemented yet.

Example

FormatDateTime

Function FormatDateTime(STRINGEXPRESSION[{, EXPRESSION,...]}) As String

Sorry. Not implemented yet.

Example

Fre

Function Fre(EXPRESSION) As Long VB6! QB!

It returns the available memory.

Example

```
PRINT FRE
FRE(-1)
PRINT FRE("")
```

FreeFile

Function FreeFile([Range]) As Integer VB6! QB!

It returns the next free available file handle.

Example

```
Dim Index1, filehandle
For Index1 = 1 To 5

filehandle = FreeFile ' next free available file handle
```

```
Open "TEST" & Index1 For Output As #filehandle
Write #filehandle, "example text."
Close #filehandle
Next
```

See also Open

--------G--------

Get

Function Get([#]FileNo As Integer[, RecordNo As Integer], Variable As AnyType)

It reads a record from file.

Example

```
TYPE TestRecord
  Student AS STRING * 20
  Result AS SINGLE
END TYPE

DIM MyClass AS TestRecord

OPEN "ENDRESULTS.DAT" FOR RANDOM AS #1 LEN = LEN(MyClass)

MyClass.Student = "Bernd Noetscher"
MyClass.Result = 99
PUT #1, 1, MyClass
CLOSE #1

OPEN "ENDRESULTS.DAT" FOR RANDOM AS #1 LEN = LEN(MyClass)
GET #1, 1, MyClass
PRINT "STUDENT:", MyClass.Student
PRINT "SCORE:", MyClass.Result
CLOSE #1

KILL "ENDRESULTS.DAT"
```

See also Type, Put

GetAttr

Function GetAttr(Path As String)

It returns attributes of files or directories.

Possible return values are:

- kbNormal 0 normal
- kbReadOnly 1 read only file
- kbHidden 2 hidden file
- kbSystem 4 system file (not supported yet)
- kbDirectory 16 directory (not supported yet)
- kbArchive 32 (not supported yet)

Example

```
Dim Attr1
' "hidden" has been set for TSTFILE
Attr1 = GetAttr("TSTFILE")      ' returns 2.
```

--------H--------

Hex

Function Hex(EXPRESSION) As String

It returns a string giving the hexadecimal (base 16) value. EXPRESSION must be a numerical expression. It will be rounded to the nearest whole number before being evaluated. Integers (or results of expressions within that range) are returned as a string of up to 4 hexadecimal characters, long integers are returned as a string of up to 8 hexadecimal characters.

Example

```
/*
Characters of Hex (0 - 9, A - F)

Hexadecimal -> Decimal
0 -> 0
1 -> 1
2 -> 2
3 -> 3
4 -> 4
5 -> 5
6 -> 6
7 -> 7
8 -> 8
9 -> 9
A -> 10
B -> 11
C -> 12
D -> 13
E -> 14
F -> 15
10 -> 16
*/
INPUT "Please type in a number: ", number
PRINT "The hexadecimal representation is "; HEX$(number)

' Output:
'
' Please type in a number: 123456
' The hexadecimal representation is 1E240
```

See also Oct

Hour

Function Hour(DATEEXPRESSION) As Integer VB6! QB!

Returns the day part of an date expression.

Example

```
Dim dd As Date = "#2006-12-12 4:35:17"

Dim Time1, Hour1
Time1 = #4:35:17 PM#
Hour1 = Hour(Time1)
```

--------I--------

Inkey

Function Inkey() As String VB6! QB!

Provided for QBasic compatibility. Returns the key code, which was pressed.

Example

```
CLS

PRINT "Press Esc, to stop ..."
DO
LOOP UNTIL INKEY$ = CHR$(27)    '27 is the ASCII-Code for Esc.
```

Input

- **INPUT [;][STRINGEXPRESSION{;,}] VARIABLENAME[,VARIABLENAME…]**

VB6! QB!

Combined screen output and keyboard input.

1. semicolon prevent EOL after pressing return. If the 2. semicolon is replaced by a comma, no question mark will be displayed

The INPUT command prompts for a value to be stored in a variable. Values are entered with the keyboard. The simplest way to use it is just followed by a variable name which will hold the value entered: INPUT n

prompts a quotation mark at the beginning of a line and waits until the user has typed a value and pressed the RETURN key.

KBasic does not make any control whether the variable has been previously used with another value or not and therefore, destroys the previous value of the variable. Because variables need not to be declared (only veryoldbasic or oldbasic with option explicit off) INPUT serves also as initialization for the variable, which assumes value of 0 (zero) when only RETURN is pressed. It is often convenient to explain a bit more what we want from the user so it is preferable to prompt an explanation message on what kind of input we're expecting. This is done by just placing a double quoted text after the keyword, like:

INPUT “How many seconds”; s

As shown the prompt is now clearer and we know what kind of input is expected. The message can be as long as wanted and must be inside double quotes, single quotes are not allowed. Instead of the semicolon a comma can be used to separate message prompt and the variable name, like:

INPUT “Enter time (in seconds): ", s

in this case the usual question mark is suppressed and the cursor is placed immediately afterwards the last character of the message string. The use of one or the other is a choice of the developer but the message and the variable name must be separated by one of the two characters. A semicolon can be used immediately after the INPUT command, like:

INPUT; “Enter age”; a INPUT " and height”; b

this will cause the second INPUT command prompt to be placed immediately after the last character typed by the user:

Enter age? 156 and height? INPUT permits also to enter as many different values as we want. The different values are stored in different variables: INPUT “age and height: ", age, height

this prompts the message and waits till the user has entered the coordinates separate with a comma. It is not possible to use any other separation character here. The two (or three, four, etc.) may be variables of different type, like:

INPUT “Enter full longitude coords (degree, cardinal): ", long, card$

Whenever the number of variables expected is different from the entered ones or a variable type mismatch occurs a “redo from start” warning message is displayed and the command is prompted again to the user.

Example

```
INPUT "How many seconds"; s
INPUT; "Enter longitude"; a
INPUT " and latitude"; b
INPUT "longitude and latitude: ", longitude, latitude
INPUT "Enter full longitude coords (degree, cardinal): ", long, card$
```

- **Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As INTEGEREXPRESSION**

VB6! QB!

Opens a file for writing. Supported for backward compatibility.

Example

```
OPTION OLDBASIC


DIM REC$

CLS
OPEN "c:\kbasic\examples\test\LISTE.TXT" FOR OUTPUT AS #1
DO
    INPUT "   NAME:        ", Name$  'input from keyboard
    INPUT "   Age:         ", Age$
    WRITE #1, Name$, Age$
    INPUT "Type a new entry"; R$
LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1

'print content of file
OPEN "c:\kbasic\examples\test\LISTE.TXT" FOR INPUT AS #1
CLS
PRINT "entries of file:": PRINT
DO WHILE NOT EOF(1)
    LINE INPUT #1, REC$
    PRINT REC$
LOOP
CLOSE #1
```

InputBox

Function InputBox(Prompt As String [, Title As String] [, Default As String]) As String

Get a string from the user using a input box on screen.

Example

```
Dim Msg, Titel, default2, val1

Msg = "Input value between 1 and 3"
Titel = "InputBox-Demo"
default2 = "1"

val1 = InputBox(Msg /*, Titel , default2*/ )

MsgBox("You have inputted: " + val1)
```

InStr

Function InStr([Start As Integer,] Source As String, Find As String) As Integer

InStr finds one string inside another. Returns 0, if Find could not be found. First position in Source is referenced as 1.

Example

```
DIM s As String

s = "Bernd Noetscher's KBasic"
PRINT "string position = "& INSTR(1, s, "KBasic")
```

See also Mid ,Right ,Len ,Mid ,InStRev

InStRev

Function InStRev([Start As Integer,] Source As String, Find As String) As Integer

InStRevsearches the Source string for and occurance of the Find string and returns the index of the Find string in the Source string. 0 is returned if the Find string is not found. Start tells the positon from which the searching starts from.

Example

```
Dim x As String, y As String

x = "This is a string"
y = "s"

Print InStRev(x, y)
```

See also Mid ,Right ,Len ,Mid ,InStr

Int

Function Int(EXPRESSION) As Long

Returns the next integer number < = given number.

Example

```
DIM n AS INTEGER

n = INT(12.54)
PRINT n

n = INT(-99.4)
PRINT n
```

See also Fix ,CInt ,CLng

IsArray

Function IsArray(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents an array type.

Example

```
Dim i[8] As Integer
Dim x As String

Print IsArray(i)
Print IsArray(x)
```

IsBoolean

Function IsBoolean(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents a boolean type.

Example

```
Dim x As Boolean

Print IsBoolean(x)
```

IsByte

Function IsByte(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents a byte type.

Example

```
Dim i As Byte
Dim x As String

Print IsByte(i)
Print IsByte(x)
```

IsCurrency

Function IsCurrency(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents a currency type.

Example

```
Dim c As Currency


c = 23

Print IsCurrency(c)
```

IsDate

Function IsDate(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents a date type.

Example

```
PRINT ISDATE(34)
PRINT ISDATE(#2006-12-12#)
```

IsDouble

Function IsDouble(EXPRESSION) As Boolean VB6! QB!

Returns true if a variable represents a double type.

Example

```
Dim i As Double
Dim x As String

Print IsDouble(i)
Print IsDouble(x)
```

IsEmpty

Function IsEmpty(EXPRESSION) As Boolean VB6! QB!

Returns true if expression represents a empty value.

Example

```
Dim v As Variant
Dim n As Integer

v = Empty

Print IsEmpty(v)
Print IsEmpty(n)

v = 99

Print IsEmpty(v)
```

IsError

Function IsError(EXPRESSION) As Boolean VB6! QB!

Returns true if the expression represents an error type.

Example

```
Dim v As Variant
'Dim v As integer

v = Error

Print IsError(v)
```

IsInteger

Function IsInteger(EXPRESSION) As Boolean VB6! QB!

Returns true if the expression represents an integer type.

Example

```
Dim i As Integer
Dim k As String

Print IsInteger(i)
Print IsInteger(k)
```

IsMissing

Function IsMissing(EXPRESSION) As Boolean VB6! QB!

Returns if an optional argument of a sub/function has not been given = is missing.

Example

```
Sub test(Optional k As String)

  If IsMissing(k) Then
    Print "k is missing"
  Else
    Print "k: " + k
  End If
End Sub


test()
test("hello here is k")
```

IsNull

Function IsNull(EXPRESSION) As Boolean VB6! QB!

Returns true if the expression represents null.

Example

```
Dim o As Object

o = Null

Print IsNull(o)
```

IsNumeric

Function IsNumeric(EXPRESSION) As Boolean VB6! QB!

Returns true if expression represents a numeric value.

Example

```
Dim v As Variant
v = 12
v = "!"

Print IsNumeric(v)
Print IsNumeric(3343.678)
Print IsNumeric("hey")
```

IsObject

Function IsObject(EXPRESSION) As Boolean VB6! QB!

Returns true if expression represents an object value.

Example

```
Class t

End Class

Dim k As New t

Dim o As New Object
Dim z As Integer

Print IsObject(k)
Print IsObject(o)
Print IsObject(z)
```

IsShort

Function IsShort(EXPRESSION) As Boolean VB6! QB!

Returns true if expression represents a short value.

Example

```
Dim i As Short
Dim x As String

Print IsShort(i)
Print IsShort(x)
```

IsSingle

Function IsSingle(EXPRESSION) As Boolean VB6! QB!

Returns true if expression represents a single value.

Example

```
Dim i As Single
Dim x As String
```

```
Print IsSingle(i)
Print IsSingle(x)
```

IsLong

Function IsLong(EXPRESSION) As Boolean

Returns true if expression represents a long value.

Example

```
Dim i As Long
Dim k As String

Print IsLong(i)
Print IsLong(k)
```

IsString

Function IsString(EXPRESSION) As Boolean

Returns true if expression represents a string value.

Example

```
Dim i As Long
Dim k As String

Print IsString(i)
Print IsString(k)
```

IsVariant

Function IsVariant(EXPRESSION) As Boolean

Returns true if expression represents a variant value.

Example

```
Dim i As Variant
Dim x As String

Print IsVariant(i)
Print IsVariant(x)
```

--------K--------

Kill

Sub Kill FileName As String VB6! QB!

Deletes a file specified by a filename. Like for DOS-based Basics, KILL deletes only files.

Example

```
' This deletes the file "test.xml":
KILL "c:\kbasic\examples\test\test.xml"
```

--------L--------

LCase

Function LCase(STRINGEXPRESSION) As String

It returns a new string. It contains the source string converted to all lower case.

LCASE take a string and converts all its characters to lower case. It then returns a copy of the string.

Example

```
DIM src as string
src = "Mr. Big was HERE"
PRINT LCASE( src )

' Output:
' mr. big was here
```

See also UCase

Left

Function Left(STRINGEXPRESSION, Len As Integer) As String

LEFT returns a string containing the first characters of a string.

Example

```
DIM src AS STRING
src = "What a nice day"
PRINT LEFT(src, 4)
```

See also Right, Mid

Len

- **Function Len(STRINGEXPRESSION) As Integer**
- **Function Len(VARIABLENAME) As Integer**

LEN returns the length of a string or the size of a variable in bytes.

Example

```
Dim s As String

s = "Bernd Noetscher's KBasic"

Print Len(s)
''Print s.Len()
''? "hi".Len()

dim x as string
x = "a string"
PRINT LEN(x)

' Output:
' 8
```

See also SizeOf

Line

- **Sub Line[(x1!, y1!)] - (x2!, y2!) [, Color As Integer]]**

VB6! QB!

Draws a line on the screen. Color might be a value between 0…255. Provided for QBasic compatibility.

```
CLS

For a As Integer = 1 To 15
  Line(10, a * 80) - (1000, a * 80), 15
Next

For a = 1 To 15
  Line(a * 80, 10) - (a * 80, 1000), 15
Next


For y As Integer = 1 To 100

  For i As Integer = 1 To 600
    Locate 1, 1 : Print "y=" + y + " : i=" + i
    Line(11 + i + y, 11 + i + y) - (2 * i + y, 11 + i + y), i / 10

  Next

Next
```

- **Sub Line Input [#]FilenNo As Integer, VARIABLENAME**

VB6! QB!

Reads line of text from file into variable. Provided for QBasic compatibility.

Example

```
Dim text2 As String

Open "c:\kbasic14\examples\test\test.txt" For Input As #1       ' open file
Do While Not EOF(1)     ' loop until end of file
        Line Input #1, text2    ' read line into variable
        Print text2
Loop
Close #1
```

See also Print , Open , Write , Input , Inkey

Loc

Function Loc([#]FileNo As Integer) As Long VB6! QB!

Returns the current position for reading or writing in a file.

Example

```
Dim Position1, Line1
Open "file1" For Binary As #1
Do While Not EOF(1)
Line1 = Line1 & Input(1, #1)
Position1 = Loc(1)

Print Line1; Tab; Position1
Loop
Close #1
```

See also EOF , Seek

Locate

Sub Locate [Y As Integer] [,X As Integer] [,Cursor As Integer] VB6! QB!

Sets the cursor position on screen. Provided for QBasic backward compatibility.

Example

```
OPTION OLDBASIC

CLS
LOCATE 5, 5
row% = CSRLIN
column% = POS(0)
PRINT "position 1 (press any key)"
DO
LOOP WHILE INKEY$ = ""
LOCATE (row% + 2), (column% + 2)
PRINT "position 2"
```

See also CSRLin , Pos , Print

LOF

Function LOF([#]FileNo As Integer) As Long VB6! QB!

Returns the length of a file in bytes.

Example

```
OPTION OLDBASIC

INPUT "input filename: "; f$
'f$ = "c:\capture.avi"

OPEN f$ FOR BINARY AS #1
PRINT "file len is = "; LOF(1)
CLOSE
```

See also EOF , Open , Write

Log

Function Log(n As Double) As Long

LOG returns a the natural logaritm of a number. The LOG function calculates the base “e” (or natural) logaritm of a number. Input number must be a positive (i.e. > 0).

Example

```
DIM x AS INTEGER
x = 12
PRINT LOG(x)

' Output:
```

```
' 2.48490665
```

LTrim

Function LTrim(STRINGEXPRESSION) As String

LTRIM function removes the source string's leading spaces, from the beginning of the source string.

Example

```
DIM x as string
x = " My house is on fire."
PRINT LTRIM( x )

' Output:
' My house is on fire.
```

See also RTrim , Trim

--------M--------

Max

Function Max(EXPRESSION, EXPRESSION) As Double

Returns the major value of two values. Both expressions must be numeric.

Example

```
PRINT MAX(44, 3)
```

See also Min

Mid (Builtin)

Function Mid(Variable As String, Start As Integer[, Len As Integer]) As String

Gets a part of a string.

Example

```
OPTION OLDBASIC

text$ = "The dog bites the cat"

text$ = MID$(text$, 10, 1)

PRINT text$
```

See also Trim , InStr

Min

Function Min(EXPRESSION, EXPRESSION) As Double

Returns the minor value of two values. Both expressions must be numeric.

Example

```
PRINT MIN(44, 3)
```

See also Max

Minute

Function Minute(DATEEXPRESSION) As Integer VB6! QB!

Returns the minute part of a time expression.

Example

```
Dim Time1, Minute1
Time1 = #4:35:17 PM#
Minute1 = Minute(Time1) ' Minute1 contains 35.
```

MkDir

Sub MkDir(String) VB6! QB!

Creates a new directory.

Example

```
MKDIR "C:\TEMP\TEST"
CHDIR "C:\TEMP"
FILES
RMDIR "TEST"
```

See also CurDir, ChDir, ChDrive, RmDir

Month

Function Month(DATEEXPRESSION) As Integer VB6! QB!

Returns the month part of a date expression.

Example

```
Dim Date1, Month1
Date1 = #1979-02-02#
Month1 = Month(Date1)    ' Month1 contains 2.
Print Month1
```

MonthName

Function MonthName(Month As Integer, ShortName As Boolean) As Integer VB6! QB!

Returns the month part of a date expression.

Example

```
Dim strMonatsname

strMonatsname = MonthName(1)   ' January
strMonatsname = MonthName(1, True)   ' Jan
```

MsgBox

Sub MsgBox(Prompt As String [, Buttons As Integer] [, Title As String])

Prints a message in a GUI dialog box.

buttons:

- kbOKOnly 0 show only [OK].
- kbOKCancel 1 show [OK] and [Cancel]
- kbAbortRetryIgnore 2
- kbYesNoCancel 3
- kbYesNo 4
- kbRetryCancel 5
- kbCritical 16 Stop symbol
- kbQuestion 32 question mark symbol
- kbExclamation 48 exclamation mark symbol
- kbInformation 64 information mark symbol
- kbDefaultButton1 0
- kbDefaultButton2 256
- kbDefaultButton3 512

return values:

- kbOK 1 OK
- kbCancel 2 Cancel
- kbAbort 3 Abort

- kbRetry 4 Retry
- kbIgnore 5 Ignore
- kbYes 6 Yes
- kbNo 7 No

Example

```
' text in richtext is possible as well
'n = MsgBox("<b>message</b> or <i>not</i>", kbOKOnly, "title text")

'n = MsgBox("message", kbOKOnly, "title text")
'n = MsgBox("message", kbOKCancel, "title text")
'n = MsgBox("message", kbAbortRetryIgnore, "title text")
'n = MsgBox("message", kbYesNoCancel, "title text")
'n = MsgBox("message", kbYesNo, "title text")
'n = MsgBox("message", kbRetryCancel, "title text")
'
'n = MsgBox("message", kbOKOnly Or kbCritical, "title text")
'n = MsgBox("message", kbOKOnly Or kbQuestion, "title text")
'n = MsgBox("message", kbOKCancel Or kbExclamation, "title text")
'n = MsgBox("message", kbOKOnly Or kbInformation, "title text")
'
'n = MsgBox("message", kbYesNoCancel Or kbDefaultButton1, "title text")
'n = MsgBox("message", kbYesNoCancel Or kbDefaultButton2, "title text")
'n = MsgBox("message", kbAbortRetryIgnore Or kbDefaultButton3, "title text")
'

n = MsgBox(" to save succeeding generations from the scourge of war, which twice
in our lifetime has brought untold sorrow to mankind, and", kbOKOnly, "WE THE
PEOPLES OF THE UNITED NATIONS DETERMINED")
```

See also InputBox

--------N--------

Name

Sub Name(OldName As String, NewName As String) VB6! QB!

Renames a file or a directory.

Example

```
NAME "c:\old.txt" AS "c:\new.txt"
```

Now

Function Now() As Date

Returns the current system date.

Example

```
PRINT NOW()
```

Nz

Function Nz(EXPRESSION) As String

Changes to expression from null to nullstring "", if needed.

Example

```
Function test()
  Return Null
End Function

Print "'_" + Nz(test) + "_'" ' --> ""
```

--------O--------

Oct

Function Oct(EXPRESSION) As String VB6! QB!

Returns a string giving the octal (base 8) representation of 'number'.

The return string has as many characters as necessary to represent the integer in octal, or the number specified by the second argument, whichever is larger. Octal numbers are just for fun.

Example

```
PRINT OCT(8)

/*
Oct (0 - 7)

Octal -> Decimal
0 -> 0
1 -> 1
2 -> 2
3 -> 3
4 -> 4
5 -> 5
6 -> 6
7 -> 7
10 -> 8
11 -> 9
12 -> 10
```

```
*/
```

See also Hex

Open

- **Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As [#]FileNo [LEN=RecordLen As Integer]**

VB6! QB!

Opens a file related on the given mode. Supported for backward compatibility.

STRINGEXPRESSION is the name of the file. It can contain path information.

mode

- One of the following modes: APPEND, BINARY, INPUT, OUTPUT or RANDOM.

access (not supported)

- READ, WRITE or READ WRITE.

READ Opens a file only for reading from file WRITE Opens a file only for writing to file READ WRITE Opens a file for writing and reading. READ WRITE is only possible for direct access and sequentiel files, and for files, which are opened for APPEND (sequentiel access)

lock(not supported)

Access permission inside network filesystem: SHARED, LOCK READ, LOCK WRITE or LOCK READ WRITE.

FileNo is an integer between 1 and 255, which names a file, which is opened. RecordLen is for direct access files: It is the length of the record (default is 128 byte). For sequentiel files: The amount of buffer characters (default is 512 Byte).

Example

```
INPUT "Input filename: "; n$
OPEN n$ FOR OUTPUT AS #1
PRINT #1, "This is stored in a file."
CLOSE
OPEN n$ FOR INPUT AS #1
INPUT #1, a$
PRINT "Readed from file: "; a$
CLOSE
```

See also Close , FreeFile , Type

- **Sub Open MODE, [#]FileNo, STRINGEXPRESSION, RecordLen As Integer**

VB6! QB!

Needed to open files (alternate syntax).

MODE "O" or "o" for output, "I" or "i" for input, "A" or "a" for append.

FileNo is an integer between 1 and 255, which names a file, which is opened.

STRINGEXPRESSION is the name of the file. It can contain path information.

RecordLen For direct access files: It is the length of the record (default is 128 byte).

For sequentiel files: The amount of buffer characters (default is 512 Byte).

Example

```
INPUT "What input file to use ??(e.g. C:\file.inp)? ", file$
OPEN "I", #1, file$  ' Open the input file
INPUT "What output file to use ??(e.g. C:\file.out)? ", file$
OPEN "O", #2, file$  ' Open the output file
```

See also Open

Output

Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As INTEGEREXPRESSION

Opens a file for writing. Supported for backward compatibility.

Example

```
OPTION OLDBASIC


CLS
OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR OUTPUT AS #1
DO
    INPUT "   NAME:        ", Name$  'input from keyboard
    INPUT "   Age:         ", Age$
    WRITE #1, Name$, Age$
    INPUT "Type a new entry"; R$
LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1

'print content of file
OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR INPUT AS #1
CLS
PRINT "entries of file:": PRINT
DO WHILE NOT EOF(1)
    LINE INPUT #1, REC$
    PRINT REC$
LOOP
CLOSE #1
```

--------P--------

Pos

Function Pos(EXPRESSION) As Integer VB6! QB!

POS returns the current cursor position in the line. EXPRESSION is obsolete, just write 0 for it.

Provided for QBasic backward compatibility.

Example

```
OPTION OLDBASIC

PRINT POS(0)

INPUT s$

PRINT CSRLIN

PRINT s$
```

See also CSRLin , Locate

Print

- **Sub Print {[[#]FileNo ,] (EXPRESSION Spc(EXPRESSION) Tab(EXPRESSION) [; | ,]) }**

VB6! QB!

Provided for QBasic backward compatibility.

Example

```
PRINT "Hello baby!"; ":-)", "----"

DIM s AS STRING = "1"
DIM s2 AS STRING = "2"
DIM s3 AS STRING = "3"

PRINT s, s2, s3
```

2nd Example

```
OPTION OLDBASIC

DIM Name$, Age$

CLS

OPEN "c:\kbasic14\examples\test\LIST4.txt" FOR OUTPUT AS #1
DO
```

```
  INPUT "   NAME:        ", Name$
  INPUT "   AGE:         ", Age$
  PRINT #1, Name$, Age$
  INPUT "More entries?"; R$
LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1
```

See also Input , Write , Using

Print

- **Sub PSet [Step] (X As Integer ,Y As Integer) [, Color As Integer]**

VB6! QB!

Draws a point on screen. Step is not supported. Provided for QBasic backward compatibility.

Example

```
OPTION OLDBASIC
OPTION EXPLICIT OFF


FOR y% = 0 TO 200
  FOR x% = 0 TO 320
    PSET(x%, y%)
  NEXT
NEXT
```

See also Line

PrintScreen

- **Sub PrintScreen (PrintDialog As Boolean)**

VB6! QB!

Prints the screen. If PrintDialog = True, the print dialog appears before printing.

Put

Function Put([#]FileNo As Integer[, RecordNo As Integer], Variable As AnyType) VB6! QB!

It wrtites a record to a file.

Example

```
TYPE TestRecord
    Student AS STRING * 20
    Result AS SINGLE
END TYPE

DIM meineKlasse AS TestRecord

OPEN "ENDRESULTS.DAT" FOR RANDOM AS #1 LEN = LEN(meineKlasse)
meineKlasse.Student = "Bernd Noetscher"
meineKlasse.Result = 99
PUT #1, 1, meineKlasse
CLOSE #1

OPEN "ENDRESULTS.DAT" FOR RANDOM AS #1 LEN = LEN(meineKlasse)
GET #1, 1, meineKlasse
PRINT "STUDENT:", meineKlasse.Student
PRINT "SCORE:", meineKlasse.Result
CLOSE #1

KILL "ENDRESULTS.DAT"
```

See also Type, Get

--------R--------

Rad

Function Rad(EXPRESSION) As Double

Example

```
PRINT RAD(8)
```

See also Sin, Cos

Raise

Sub Raise(Number As Integer[, Source As String[, Description As String]]) VB6! QB!

It is a method of the error object, which is provided for VB6 backward compatibility. Use exception handling instead, e.g. Try.

Example

```
Dim Msg

On Error Resume Next

Err.Clear
Err.Raise(6)
```

```
If Err.Number <> 0 Then

   Msg = "Error # " & Str(Err.Number) & " " _
       & Err.Source & Chr(10) & Err.Description

    Print Msg

End If
```

See also Try

Random

Sub Open STRINGEXPRESSION For [Input|Output|Append|Binary|Random] As INTEGEREXPRESSION VB6! QB!

Opens a file for random access. Supported for backward compatibility.

Example

```
OPTION OLDBASIC

DIM Rec1$, Rec2$

CLS
 OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR APPEND AS #1
 DO
     INPUT "   NAME:        ", Name$
     INPUT "   AGE:         ", Age$
     WRITE #1, Name$, Age$
     INPUT "More entries?"; R$
 LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1

 'print file on screen
 OPEN "c:\kbasic\examples\test\LISTEN.TXT" FOR INPUT AS #1
 CLS
 PRINT "Entries of file:": PRINT
DO WHILE NOT EOF(1)
    INPUT #1, Rec1$, Rec2$
    PRINT Rec1$, Rec2$
LOOP
CLOSE #1
KILL "LIST"
```

Randomize

Sub Randomize [StartValue As Integer]

Start the random generator.

Example

```
RANDOMIZE TIMER
x% = INT(RND * 6) + 1
y% = INT(RND * 6) + 1
PRINT "2 throws with one dice: 1st throw ="; x%; "and 2nd throw ="; y%
```

Replace

Sub Replace (Str As String, SearchFor As String, ReplaceWith As String)

Replaces string occurances with another string.

Example

```
DIM s = "Das ist alles was wir brauchen. Fang nochmal von vorne an."
DIM search = vorne"
DIM replace = "hinten"
PRINT REPLACE(s, search, replace)
```

Reset

Sub Reset VB6! QB!

Closes all opened files.

Example

```
Reset
```

RGB

Function RGB(Eed As Integer, Green As Integer, Blue As Integer) As Long VB6! QB!

Returns a long value generated by a triple.

Example

```
Dim red As Integer

red = RGB(255, 0, 0)

Print Hex(red)
```

Right

Function Right(STRINGEXPRESSION, Len As Integer) As String

Right returns a string containing the last characters of a string.

Example

```
PRINT RIGHT$("I'm living in Germany", 7)
'PRINT RIGHT$("I'm living in Germany", LEN("Germany"))
```

See also Left, Mid

RmDir

Sub RmDir(String) VB6! QB!

Deletes a complete directory.

Example

```
MKDIR "C:\TEMP\TEST"
CHDIR "C:\TEMP"
FILES
RMDIR "TEST"
```

See also CurDir, ChDir, ChDrive, MkDir

RND

Function RND(EXPRESSION) As Double

Returns an integer pseudo-random number between 0 and int(EXPR)-1 inclusive. If EXPRESSIONis 1, then returns a rational number between 0 (inclusive) and 1. If EXPRESSION is negative then EXPRESSION seeds the random number generator.

Example

```
RANDOMIZE TIMER
x% = INT(RND * 6) + 1
y% = INT(RND * 6) + 1
PRINT "2 turns with one dice: turn 1 ="; x%; "and turn 2 ="; y%
```

See also Randomize

RTrim

Function RTrim(STRINGEXPRESSION) As String

RTRIM function removes the source string's trailing spaces, from the end of the source string.

Example

```
PRINT RTRIM$("  bedazzeled  ")
```

See also LTrim , Trim

--------S--------

Second

Function Second(DATEEXPRESSION) As Integer VB6! QB!

Returns the second part of a date expression.

Example

```
Dim Time1, Second1
Time1 = #4:35:47 PM#
Second1 = Second(Time1) ' Second1 contains 47
```

Seek

Sub Seek #FileNo, RecordPosition As Long VB6! QB!

Returns the current position in file or set the new position in file.

Example

```
Option OldBasic

Type myRecordset ' define type
 id As Integer
 Name2 As String * 20
End Type

Dim DSet1 As myRecordset, MaxSize, DSetNo

' file with random access
Open "c:\kbasic\examples\test\file1.txt" For Random As #1 Len = Len(DSet1)
MaxSize = 10 ' define count of records in file

For DSetNo = MaxSize To 1 Step - 1
  Seek #1, DSetNo ' set position

  DSet1.id = DSetNo
  DSet1.Name2 = "Bernd" + DSetNo * 1000

  Put #1, , DSet1  ' write recordset
Next
```

```
Close #1 ' close file


' file with random access
Open "c:\kbasic\examples\test\file1.txt" For Random As #1 Len = Len(DSet1)
MaxSize = LOF(1) \ Len(DSet1) ' define count of records in file
Print "MaxSize = " + MaxSize

For DSetNo = MaxSize To 1 Step - 1
  Seek #1, DSetNo  ' set position
  Get #1, , DSet1 ' read recordset

  Print DSet1.id
  'Print DSet1.Name2

Next
Close #1 ' close file
```

See also Open ,Get ,Put ,Write ,Print

Sgn

Function Sgn(EXPRESSION) As Integer

SGN returns the sign of the argument 'number', +1 for positive numbers, 0 for 0, and - 1 for negative numbers.

Example

```
PRINT ABS(45.5 - 100!) 'result: 54.5
PRINT SGN(1), SGN(-1), SGN(0) 'result: 1 -1 0
```

See also Abs

Shell

Function Shell(EXPRESSION) As Long

Send a command to the environment. Return value is -255, if an error occured.

Example

```
SHELL ("DIR") ' on Windows
'SHELL ("LS")
```

Sin

Function Sin(EXPRESSION) As Long

SIN returns the sine of the argument 'number' in radians.

Example

```
CONST PI=3.141592654
PRINT ATN(TAN(PI/4.0)), PI/4.0 'result: .7853981635 .7853981635
PRINT (COS(180 * (PI / 180))) 'result: -1
PRINT (SIN(90 * (PI / 180))) 'result: 1
PRINT (TAN(45 * (PI / 180))) 'result: 1.000000000205103
```

See also Cos, Tan

Sleep

Sub Sleep [Seconds As Integer] VB6! QB!

Waits for until a was key pressed, or the after a time period.

Example

```
PRINT "Pausing 10 seconds..."
SLEEP 10
PRINT "Continue..."
```

Space

Function Space(INTEGEREXPRESSION) As String

SPACE function creates a string consisting of spaces. SPACE creates a string of spaces based on x length. This function is similar to the STRING function.

Example

```
PRINT "*" + SPACE(5) + "*"

' Output:
' * *

PRINT SPACE$(4.3 + 2)

PRINT "*" + SPACE(5) + "*"
```

Spc

Function Spc(INTEGEREXPRESSION) As String

Returns a string with a number of spaces. Used together with Print.

Example

```
PRINT "Text1"; SPC(10); "Text2"
```

Sqr

Function Sqr(EXPRESSION) As Long

SQR returns the square root of the argument 'number'.

Example

```
PRINT SQR(25), SQR(2) 'result: 5 1.414214
```

Str

Function Str(EXPRESSION) As String

Converts a number to a string.

Example

```
PRINT STR$(239.546)
```

See also Asc , Val

StrComp

Function StrComp(STRINGEXPRESSION, STRINGEXPRESSION [, ComparisionMode As Integer]) As Integer

Compares two strings.

ComparisionMode : defines how to compare the two strings. Do not need to be given (optional)

Possible values are 0 and 1. The value 0 (default) means a binary compare. The value 1 means a text-based compare. If compare is not given, Option Compare defines how to compare.

Example

```
Dim Text1, Text2, Vergl

Text1 = "ABCD": Text2 = "abcd" '
Verg1 = StrComp(Text1, Text2, 1) ' result:0.
Verg1 = StrComp(Text1, Text2, 0) ' result:-1.
Verg1 = StrComp(Text2, Text1) ' result:1.
```

See also Asc , Val

String

- **Function String(STRINGEXPRESSION, Len As Integer) As String**

STRING function creates a string of characters.

Example

```
PRINT STRING(20, "x")

' Output:
' xxxxxxxxxxxxxxxxxxxx
```

- **Function String(Len As Integer, Ascii As Integer) As String**

STRING function creates a string of characters.

Example

```
PRINT STRING(20, 65)
' would output AAAAAAAAAAAAAAAAAAAA.
```

StrReverse

Function StrReverse(STRINGEXPRESSION) As String

Returns a given string reversed.

Example

```
DIM s = "Mondscheinsonate by Beethoven"
PRINT STRREVERSE(s) ' --> nevohteeB yb etanosniehcsdnoM
```

--------T--------

Tab

Function Tab(EXPRESSION) As String

Print Tabs. Used together with Print.

Example

```
CLS

Print "1", Tab(25) "Hio"

'Print "Hi", "2"
```

Tan

Function Tan(EXPRESSION) As Long

TAN returns the tangent of the argument 'number' in radians.

Example

```
CONST PI=3.141592654
PRINT ATN(TAN(PI/4.0)), PI/4.0 'result: .7853981635 .7853981635
PRINT (COS(180 * (PI / 180))) 'result: -1
PRINT (SIN(90 * (PI / 180))) 'result: 1
PRINT (TAN(45 * (PI / 180))) 'result: 1.000000000205103
```

See also Cos , Sin

Time

Function Time() As String VB6! QB!

Date or TIME$ returns the current system time as a string. Setting the time is not possible with this builtin function.

Example

```
PRINT TIME
```

See also Date

TimeSerial

Function TimeSerial(Hour As Integer, Minute As Integer, Second As Integer) As Long VB6! QB!

Returns a time as a integer.

Example

```
Dim Time1
Time1 = TimeSerial(16, 35, 17) ' in integer format --> 16:35:17
```

TimeValue

Function TimeValue(STRINGEXPRESSION) As Date VB6! QB!

Returns a time given in a string expression as a date.

Example

```
Dim Time1
Time1 = TimeValue("4:35:17 PM") ' return time as date
```

Trim

Function Trim(STRINGEXPRESSION) As String

TRIM function removes the source string's leading and trailing spaces.

TRIM function removes the source string's leading and trailing spaces, from the beginning and end of the source string.The "trimmed" string is return to the function caller.

Example

```
DIM x as string
x = " My house is on fire. "
PRINT TRIM( x )

' Output:
' My house is on fire.
```

See also LTrim , RTrim

TypeName

Function TypeName(VARIABLENAME) As String

Returns the type name of a variable as string

possible returns:

- Byte
- Integer
- Long
- Single
- Double
- Currency
- Date
- String
- Boolean
- Error
- Empty
- Null
- Object
- Nothing

Example

```
Class k

End Class

Enum e
  o
End Enum

Type t
  o As Integer
End Type
```

```
Dim kk As k
Dim ee As e
Dim tt As t
Dim ll As Label

Dim NullVar, Type1, StrVar As String, IntVar As Integer, CurVar As Currency
Dim ArrayVar(1 To 5) As Integer

NullVar = Null ' Null zuweisen.
'NullVar = CVERR(2)
'NullVar = Empty

Type1 = TypeName(StrVar)' returns "String".
Type1 = TypeName(IntVar) ' returns "Integer".
Type1 = TypeName(CurVar)' returns "Currency".

Type1 = TypeName(NullVar) ' returns "Null".

Type1 = TypeName(ArrayVar) ' returns "Integer()"

Type1 = TypeName(kk)

Type1 = TypeName(ee) ' returns the internal id only
Type1 = TypeName(tt) ' returns the internal id only

Type1 = TypeName(ll)
```

UCase

Function UCase(STRINGEXPRESSION) As String

It returns a new string. It contains the source string converted to all upper case.

It takes a string and converts all its characters to upper case. It then returns a copy of the string.

Example

```
DIM src as string
src = "Mr. Big was HERE"
PRINT UCASE( src )

' Output:
' MR. BIG WAS HERE
```

See also UCase

Using

Sub Print Using STRINGEXPRESSION; STRINGEXPRESSION [;|,]

Prints strings or numbers using a specified format

```
Formatted strings:

formatstring may contain \\ or ! or &

you can use "\\" to print a certain number of characters (n-2), so

"\\" will print 2
"\ \" will print 4 characters

A "!" will print only the first character of the string

A "&" will print the whole string

----------
Formatted numbers:
you can choose the width of the printing, the number of decimals and the place
of the decimal point. Also the place of $, kommas and + or minus

- every # is a decimal digit (max 18)
- additional spaces right of the decimal point will be filled with 0
- additional spaces left of the decimal point will be filled with spaces
- exception: 1 > n > -1 when a 0 is before the decimal point
- all numbers rounded to number of digits
- negative numbers with a leading -
- this leading - takes one # in the formatstring

0.468 ##.## 0.47 one leading space
0.468 #.#### 0.4680 no leading space
12.5 ##.## 12.50 no leading space
12.5 ####.# 12.5 two leading spaces

Well, this is a bit confusing:

- a plus at the beginning produces a leading + or - before the number
- a minus at the beginning produces always a - (for positive and negative)
- a plus at the end produces a trailing + or -
- a minus at the end produces a space for positive and a - for negative numbers

Now the $ and stuff:
- a $ at the beginning: $ before the number
- for a negative value, the - is between $ and first digit
- several $ reserve additional spaces, but only one $ is printed
- two * fill the spaces with ***
- you can combine ** and $
- a comma left of decimal point marks thousands (1,000,000 - English style)

Exponentials:
- scientific notation by including 3 - 6 "^" signs
- one for "E", one for +/- and two to four for the exponents

Literals:
literals must be preceeded by a "_":

print using "_##"; 1 'prints #1
print using "#_#"; 1 'prints 1#

- if the number doesn't fit, the formatstring is ignored and the whole number
with a leading "%" is printed
```

- variables may be used as formatstrings

- formatstrings may be emedded into normal text:

```
a = 12.56
x$= "sum"
PRINT USING "The & is $##.##";x$,a
```

The formatstring is a string literal (or variable) containing literal characters to print (such as labels) and special formatting characters. These formatting characters determine the field and the format of the printed string or numbers. Spaces, commas, and semicolons in the expressionlist have the same meaning as they do in a PRINT statement.

The expressionlist contains the string expressions or numeric expressions to be printed, separated by semicolons.

When PRINT USING is used to print strings, you may use one of three formatting characters to format the string field, as described in the following list:

!

Only the first character in the given string is to be printed

\   \

Prints 2+n characters from a string where n is the number of spaces between the two backslashes. If the backslashes are typed with no spaces, two characters are printed, and so on. If the field is longer than the string, the string is left-justified in the field and padded with spaces to the right.

&

Indicates a variable length string field. When the field is specified with the ampersand (&), the string is output without modification.

When PRINT USING is used to print numbers, the following special characters can be used to format the numeric field:

Character

Description

```
#

Represents each digit position.  Digit positions
are always filled.  If the number to be printed has fewer digits than positions
specified, the number is right justified (preceded by spaces) in the field.



.

Prints a decimal point.  A decimal point may be
inserted at any position in the field. If the format string specifies that
a digit is to precede the decimal point, the digit is always printed (as
0, if necessary).  If necessary, numbers are rounded.



+

Causes the sign of the number (plus or minus) to
be printed before the number (if it appears at the beginningat
string) or after (if it appears at the endat string).



-

Causes a negative number to be printed with a trailing
minus sign if it appears at the endat string.



**

Causes leading spaces in the numeric field to be
filled with asterisks.  The double asterisk also specifies positions for
two more digits.



$$

Causes a dollar sign to be printed to the immediate
leftatted number.  The $$ specifies two more digit positions,
one of which is the dollar sign.



**$

Combines the effects of the double-asterisk and
double-dollar sign signals.  Leading spaces are asterisk filled, and a dollar
sign is printed before the number.  The **$ symbols specify three more digit
positions, one of which is the dollar sign. When negative numbers are printed,
the minus sign appears to the immediate left of the dollar sign.
```

,

If the comma appears to the left of the decimal
point, in a format string, it causes a comma to be printed to the left of every third digit left of the decimal point. If it appears at the end of the format string, it is printed as part of the string. A comma specifies another digit position. The comma has no effect if used with exponential (^^^^ or ^^^^^) format.

^^^^

Specifies exponential format. You can also use
five carets (^^^^^) to allow E+xxx to be printed for larger numbers. Any decimal point position may be specified. The significant digits are left justified and the exponent is adjusted. Unless a leading +, trailing +, or - is specified, one digit position is used to the left of the decimal point to print a space or a minus sign.

_

An underscore in the format string prints the next
character as a literal character. A literal underscore is printed as the result of two underscores in the format string.

If he number to be printed is larger than the specified numeric field, a percent sign (%) is printed in front of the number. If rounding causes a number to exceed the field, a percent sign is printed in front of the rounded number. If the number of digits specified exceeds 24, an error message results that reads <tt>Illegal function call.</tt>

Example

```
CLS

' numeric

PRINT USING "###"; 1


'PRINT USING "#####"; 12.12545
'PRINT USING "###.##"; 12.12545 ' rounds automatically

'PRINT USING "+###"; +12.12345
'PRINT USING "+####"; -12.12345

'PRINT USING "x###x"; 12.12345
'PRINT USING "###.###"; 12.12345

'PRINT USING "$$####"; -12.12345
'PRINT USING "$$####"; -1234.12345
'PRINT USING "**####"; -12.12345
'PRINT USING "**$###"; -1.12345
```

```
'PRINT USING "$####"; -1.12345
'PRINT USING "*####"; - 1.12345

'PRINT USING "$$####"; -1.12345
'PRINT USING "####"; -12.12345

'PRINT USING "**$####-x"; -12.12345
'PRINT USING "####-x"; -12.12345
'PRINT USING "####-x"; 12.12345

'PRINT USING "+^^^^"; 12.12345 ' not allowed


'PRINT USING "**^^^^"; 290.12345
'PRINT USING "**^^^^^^"; -999912.12345


'PRINT USING "##,.##"; 1.12345
'PRINT USING "##,.##"; 12.12345
'PRINT USING "##,.##"; 1234.12345
'PRINT USING "##,.##"; 123456.12345
'PRINT USING "##,.##"; 1234567.12345




' string

PRINT USING "x&x x&x"; "Hello World!", "Bernd"
'PRINT USING "x&x x&x"; "Hello World!"
'PRINT USING "x&x x&x"; "Hello World!",
'PRINT USING "x&x x&x"; "Hello World!";
'PRINT USING "x&x"; "Hello World!"
'PRINT USING "&"; "Hello World!"

'PRINT USING "_!_"; "Hello World!"

'PRINT USING "_\   \_"; "Hello World!"

'PRINT "Hello World!"

' escape code

'PRINT USING "x_&x&x"; "Hello World!"
```

See also Print

Val

Function Val(STRINGEXPRESSION) As Double

VAL returns the numerical value of a string.

Example

```
Print Val("344") ' --> 344
Print Val("21st day") ' --> 21
Print Val("BASIC") ' --> 0
```

See also Str

VarType

Function VarType(VARIABLENAME) As Integer VB6! QB!

Returns the type of a variable.

values are:

- kbEmpty 0
- kbNull 1
- kbInteger 2
- kbLong 3
- kbSingle 4
- kbDouble 5
- kbCurrency 6
- kbDate 7
- kbString 8
- kbObject 9
- kbError 10
- kbBoolean 11
- kbVariant 12
- kbByte 17
- kbArray 8192

Example

```
Dim s As String

Print VarType(s)
```

See also Str

--------W--------

Weekday

Function Weekday(DATEEXPRESSION [,FirstDayOfWeek As Integer]) As Integer

VB6! QB!

Returns the weekday of a date.

Example

```
Dim Date1, Weekday1
Date1 = #2006-05-10#
Weekday1 = Weekday(Date1)       ' Weekday1 contains 4
```

WeekdayName

Function WeekdayName(Weekday As Integer [,ShortName As Boolean, [,FirstDay As Integer]]) As Integer

Returns the name of the given weekday.

Example

```
Dim sWDay As String

Dim n As Integer = Weekday(#2006-05-10#)

sWDay = WeekdayName(n)

MsgBox sWDay
```

Write

Sub Write [[#]FileNo As Integer, EXPRESSION, EXPRESSION…

Writes data to the screen or a file.

FileNo is the number of an opened sequentiel file. If no fileno is given, the output is done to the screen. EXPRESSIONs, comma separated, which should be written to file or screen.

While writing between the elements commas and quotations are inserted.

Example

```
CLS
OPEN "LIST" FOR OUTPUT AS #1
DO
INPUT " NAME: ", Name$
INPUT " AGE: ", Age$
WRITE #1, Name$, Age$
INPUT "More entries?"; R$
LOOP WHILE UCASE$(R$) = "Y"
CLOSE #1
'print file on screen
OPEN "LIST" FOR INPUT AS #1
CLS
PRINT "Entries of file:": PRINT
DO WHILE NOT EOF(1)
INPUT #1, Rec1$, Rec2$
PRINT Rec1$, Rec2$
```

```
LOOP
CLOSE #1
KILL "LIST"
```

See also Input , Line , Open , Print

Year

Function Year(DATEEXPRESSION) As Integer VB6! QB!

Returns the year of a date.

Example

```
Dim Date1, Year1
Date1 = #2006-12-12#
Year1 = Year(Date1) ' Year1 contains 1969.
```