

[{bmc_j_a.bmp}](#) [{bmc_j_b-c.bmp}](#) [{bmc_j_d.bmp}](#) [{bmc_j_e.bmp}](#) [{bmc_j_f.bmp}](#) [{bmc_j_g.bmp}](#) [{bmc_j_h-k.bmp}](#) [{bmc_j_l.bmp}](#) [{bmc_j_m.bmp}](#) [{bmc_j_n.bmp}](#) [{bmc_j_o.bmp}](#) [{bmc_j_p.bmp}](#) [{bmc_j_q.bmp}](#) [{bmc_j_r.bmp}](#) [{bmc_j_s.bmp}](#) [{bmc_j_t.bmp}](#) [{bmc_j_u-v.bmp}](#) [{bmc_j_w-z.bmp}](#) [{bmc_j_sym.bmp}](#)

Command Categories

[Arrays](#)

[Clipboard Manipulation](#)

[Conversions](#)

[Date and Time Functions](#)

[Desktop Modifications](#)

[Dialog Creation](#)

[Dialog Display](#)

[Dialog Manipulation](#)

[Dynamic Data Exchange](#)

[DCL Environment Information](#)

[Environment Statements and Functions](#)

[Error Trapping](#)

[File Input and Output](#)

[Flow Control](#)

[Icons](#)

[Keyboard Manipulation](#)

[Math Statements and Functions](#)

[Menus](#)

[Miscellaneous Statements and Functions](#)

[Mouse Events](#)

[Network Functions](#)

[Operators](#)

[Printer Manipulation](#)

[Procedure Statements](#)

[Strings](#)

[Variables and Constants](#)

[Viewport Window Manipulation](#)

[Window Manipulation](#)

Alphabetical Listing

#3 Symbols

'
=
*
=
+
=
=
/
=

¹ BasicReference

² Desktop Control Language Reference

³ cmd_toc_sym

#4 A

[Abs](#)
[ActivateControl](#)
[AddIni](#)
[And](#)
[AnswerBox](#)
[AppActivate](#)
[AppClose](#)
[AppFileName\\$](#)
[AppFind](#)
[AppGetActive\\$](#)
[AppGetPosition](#)
[AppGetState](#)
[AppHide](#)
[AppList](#)
[AppMaximize](#)
[AppMinimize](#)
[AppMove](#)
[AppRestore](#)
[AppSetState](#)
[AppShow](#)
[AppSize](#)
[AppType](#)
[ArrayDims](#)
[ArraySort](#)
[Asc](#)
[AskBox\\$](#)
[AskPassword\\$](#)
[Atn](#)
[ATTR_ARCHIVE](#)
[ATTR_DIRECTORY](#)
[ATTR_HIDDEN](#)
[ATTR_NONE](#)
[ATTR_NORMAL](#)
[ATTR_READONLY](#)
[ATTR_SYSTEM](#)
[ATTR_VOLUME](#)

#5 **B -- C**

Beep
Begin Dialog

⁴ cmd_toc_a

⁵ cmd_toc_bc

ButtonEnabled
ButtonExists
Call
CancelButton
CDBl
ChDir
ChDrive
CheckBox
CheckboxEnabled
CheckboxExists
Chr\$
CInt
Clipboard\$
ClipboardClear
CLng
Close
Combobox
ComboboxEnabled
ComboboxExists
Command\$
Const
Cos
CSng
CStr
CurDir\$

#6 D

Date\$
DateSerial
DateValue
Day
DDEExecute
DDEInitiate
DDEPoke
DDERequest
DDETerminateAll
DDETerminate
DDETimeOut
Declare
DEFtype
DesktopCascade
DesktopSetColors
DesktopSetWallpaper
DesktopTile
Dialog
Dim
Dir\$
DirExists
DiskDrives
DiskFree
Do...Loop
DoEvents
DoKeys

DCLHomeDir\$
DCLOS\$
DCLVersion\$

#7 E

EditEnabled
EditExists
EnableStopScript
End
ENV_BOTH
ENV_DOS
ENV_WINDOWS
Environ\$
EOF
Erl
Err
Error\$
Error
Exclusive
Exit Do
Exit For
Exit Function
Exit Sub
Exp

#8 F

FALSE
FileAttr
FileCopy
FileDateTime
FileDirs
FileExists
FileLen
FileList
FileParse
FileType
FindFile\$
Fix
For...Next
FreeFile
Function...End Function

#9 G

GetAttr
GetCheckbox
GetComboboxItem\$
GetComboboxItemCount
GetEditText\$

⁷ cmd_toc_e

⁸ cmd_toc_f

⁹ cmd_toc_g

GetEnv
GetListboxItem\$
GetListboxItemCount
GetOption
GoSub
Goto
GroupBox

#10 **H -- K**

Hex\$
HLine
Hour
HPage
HScroll
If...Then...Else
Input #
Input\$
InputBox\$
InStr
Int
Item\$
ItemCount
Kill

#11 **L**

LBound
LCase\$
Left\$
Len
Let
Line\$
LineCount
LineInput #
ListBox
ListboxEnabled
ListboxExists
LOF
Log
LTrim\$

#12 **M**

Main
MCI
Menu
MenuItemChecked
MenuItemEnabled
MenuItemExists
Mid\$

¹⁰ cmd_toc_hk

¹¹ cmd_toc_l

¹² cmd_toc_m

Minute
MkDir
Mod
Month
MsgBox
MsgClose
MsgOpen
MsgSetText
MsgSetThermometer

#13 **N**

Name
NetAttach
NetConnectDrive
NetDetach
NetDirectoryRights
NetDisconnectDrive
NetGetDirectoryRights
NetMemberOf
NetStationID
NetUserName
NetworkStatus
Not
Now
NS_ACTIVE
NS_LOGGEDON
Null

#14 **O**

Oct\$
OKButton
On Error
Open
OpenFileName\$
Option Base
OptionButton
OptionEnabled
OptionExists
OptionGroup
Or

#15 **P**

PI
PO_LANDSCAPE
PO_PORTRAIT
PopupMenu
Print #
Print

¹³ cmd_toc_n

¹⁴ cmd_toc_o

¹⁵ cmd_toc_p

PrinterGetOrientation
PrinterSetOrientation
PrintFile
PushButton

#16 **Q**

QueEmpty
QueFlush
QueKeyDn
QueKeys
QueKeyUp
QueMouseClicked
QueMouseDbIClk
QueMouseDbIDn
QueMouseDn
QueMouseMove
QueMouseUp
QueSetRelativeWindow

#17 **R**

Random
Randomize
ReadINI\$
ReadINISection
ReDim
RefreshIni
REM
Reset
RestoreEnv
Resume
Return
Right\$
Rmdir
Rnd
RTrim\$

#18 **S**

SaveEnv
SaveFileName\$
Second
Seek
Select...Case
SelectBox
SelectButton
SelectComboboxItem
SelectListboxItem
SendKeys
SetAttr

¹⁶ cmd_toc_q

¹⁷ cmd_toc_r

¹⁸ cmd_toc_s

SetCheckbox
SetEditText
SetEnv
SetIcon
SetIconTitle
SetOption
Sgn
Shell
ShowIcon
Sin
Sleep
SleepUntil
Snapshot
Space\$
Sqr
Stop
Str\$
StrComp
String\$
StringSort
Sub...End Sub
SystemFreeMemory
SystemFreeResources
SystemMouseTrails
SystemRestart
SystemTotalMemory
SystemWindowsDirectory\$
SystemWindowsVersion\$

#19 **T**

Tan
TextBox
Text
Time
Timer
TimeSerial
TimeValue
Trim\$
TRUE
TYPE_DOS
TYPE_WINDOWS

#20 **U -- V**

UBound
UCase\$
Val
ViewportClear
ViewportClose
ViewportOpen
VK_LBUTTON

¹⁹ cmd_toc_t

²⁰ cmd_toc_uv

VK_RBUTTON
VLine
VPage
VScroll

#21 **W -- Z**

WaitForTaskCompletion
Weekday
While...Wend
WinActivate
WinClose
WinFind
WinList
WinMaximize
WinMinimize
WinMove
WinRestore
WinSize
Word\$
WordCount
Write #
WriteINI
WS_MAXIMIZED
WS_MINIMIZED
WS_RESTORED
Xor
Year

#22 S23 K24 +25 **Arrays**

Description

Function/Statement

Change default lower limit

{bml j_bullet.bmp}Option Base

Declare and initialize

{bml j_bullet.bmp}Dim

{bml j_bullet.bmp}FileDirs

{bml j_bullet.bmp}FileList

{bml j_bullet.bmp}ReDim

{bml j_bullet.bmp}ReadINISection

Find the limits

{bml j_bullet.bmp}LBound

{bml j_bullet.bmp}UBound

Manipulate an array

{bml j_bullet.bmp}ArrayDims

{bml j_bullet.bmp}ArraySort

#26 S27 K28 +29 **Clipboard Manipulation**

Description

Function/Statement

Capture a screen or window to the clipboard

{bml j_bullet.bmp}Snapshot

Clear the clipboard object

{bml j_bullet.bmp}ClipboardClear

Get contents of clipboard

{bml j_bullet.bmp}Clipboard\$

²² CFAM_Arrays

²³ Arrays, command category

²⁴ Arrays

²⁵ CmdFam:001

²⁶ CFAM_Clipboard_Manipulation

²⁷ Clipboard Manipulation, command category

²⁸ Clipboard manipulation

²⁹ CmdFam:003

Description	Function/Statement
ANSI value to string	<u>{bml j_bullet.bmp}Chr\$</u>
Date to serial number	<u>{bml j_bullet.bmp}DateSerial</u> <u>{bml j_bullet.bmp}DateValue</u>
Number to string	<u>{bml j_bullet.bmp}CStr</u> <u>{bml j_bullet.bmp}Hex\$</u> <u>{bml j_bullet.bmp}Oct\$</u> <u>{bml j_bullet.bmp}Str\$</u>
One numeric type to another	<u>{bml j_bullet.bmp}CDBl</u> <u>{bml j_bullet.bmp}CInt</u> <u>{bml j_bullet.bmp}CLng</u> <u>{bml j_bullet.bmp}CSng</u>
String to ASCII value	<u>{bml j_bullet.bmp}Asc</u>
String to number	<u>{bml j_bullet.bmp}Val</u>

Description	Function/Statement
Date to serial number	<u>{bml j_bullet.bmp}DateSerial</u> <u>{bml j_bullet.bmp}DateValue</u>
Get the current date or time	<u>{bml j_bullet.bmp}Date\$</u> <u>{bml j_bullet.bmp}Now</u> <u>{bml j_bullet.bmp}Time\$</u>

³⁰ CFAM_Conversions³¹ Conversions, command category³² Conversions³³ CmdFam:005³⁴ CFAM_Date_and_Time_Functions³⁵ Date and Time Functions³⁶ Date functions;Time functions³⁷ CmdFam:007

Serial number to date	<u>{bml j_bullet.bmp}Day</u>
	<u>{bml j_bullet.bmp}Month</u>
	<u>{bml j_bullet.bmp}Weekday</u>
	<u>{bml j_bullet.bmp}Year</u>
Serial number to time	<u>{bml j_bullet.bmp}Hour</u>
	<u>{bml j_bullet.bmp}Minute</u>
	<u>{bml j_bullet.bmp}Second</u>
Set the date or time	<u>{bml j_bullet.bmp}Date\$</u>
	<u>{bml j_bullet.bmp}Time\$</u>
Time a process	<u>{bml j_bullet.bmp}Timer</u>
Time to serial number	<u>{bml j_bullet.bmp}TimeSerial</u>
	<u>{bml j_bullet.bmp}TimeValue</u>

#38 S39 K40 +41 **Desktop Modifications**

{bml j_bullet.bmp}DesktopCascade
{bml j_bullet.bmp}DesktopSetColors
{bml j_bullet.bmp}DesktopSetWallpaper
{bml j_bullet.bmp}DesktopTile

³⁸ CFAM_Desktop_Modifications

³⁹ Desktop Modifications, command category

⁴⁰ Desktop modifications

⁴¹ CmdFam:009

{bml j_bullet.bmp}Begin Dialog...EndDialog

{bml j_bullet.bmp}CancelButton

{bml j_bullet.bmp}Checkbox

{bml j_bullet.bmp}Combobox

{bml j_bullet.bmp}Dialog

{bml j_bullet.bmp}Dim

{bml j_bullet.bmp}GroupBox

{bml j_bullet.bmp}ListBox

{bml j_bullet.bmp}OKButton

{bml j_bullet.bmp}OptionButton

{bml j_bullet.bmp}OptionGroup

{bml j_bullet.bmp}PushButton

{bml j_bullet.bmp}Text

{bml j_bullet.bmp}TextBox

⁴² CFAM_Dialog_Creation

⁴³ Dialog Creation, command category

⁴⁴ Dialog creation

⁴⁵ CmdFam:010

#46 S47 K48 +49 **Dialog Display**

{bml j_bullet.bmp}AnswerBox
{bml j_bullet.bmp}AskBox\$
{bml j_bullet.bmp}AskPassword\$
{bml j_bullet.bmp}InputBox\$
{bml j_bullet.bmp}MsgBox
{bml j_bullet.bmp}MsgClose
{bml j_bullet.bmp}MsgOpen
{bml j_bullet.bmp}MsgSetText
{bml j_bullet.bmp}MsgSetThermometer
{bml j_bullet.bmp}OpenFileName\$
{bml j_bullet.bmp}PopupMenu
{bml j_bullet.bmp}SaveFileName\$
{bml j_bullet.bmp}SelectBox

⁴⁶ CFAM_Dialog_Display

⁴⁷ Dialog Display, command category

⁴⁸ Dialog display

⁴⁹ CmdFam:011

{bml j_bullet.bmp}ActivateControl
{bml j_bullet.bmp}ButtonEnabled
{bml j_bullet.bmp}ButtonExists
{bml j_bullet.bmp}CheckboxEnabled
{bml j_bullet.bmp}CheckboxExists
{bml j_bullet.bmp}ComboboxEnabled
{bml j_bullet.bmp}ComboboxExists
{bml j_bullet.bmp>EditEnabled
{bml j_bullet.bmp>EditExists
{bml j_bullet.bmp}GetCheckbox
{bml j_bullet.bmp}GetComboboxItem\$
{bml j_bullet.bmp}GetComboboxItemCount
{bml j_bullet.bmp}GetEditText\$
{bml j_bullet.bmp}GetListboxItem\$
{bml j_bullet.bmp}GetListboxItemCount
{bml j_bullet.bmp}GetOption
{bml j_bullet.bmp}ListboxEnabled
{bml j_bullet.bmp}ListboxExists
{bml j_bullet.bmp}OptionEnabled
{bml j_bullet.bmp}OptionExists
{bml j_bullet.bmp>SelectButton
{bml j_bullet.bmp>SelectComboboxItem
{bml j_bullet.bmp>SelectListboxItem
{bml j_bullet.bmp}SetCheckbox
{bml j_bullet.bmp}SetEditText
{bml j_bullet.bmp}SetOption

⁵⁰ CFAM_Dialog_Manipulation

⁵¹ Dialog Manipulation, command category

⁵² Dialog manipulation

⁵³ CmdFam:012

Dynamic Data Exchange (DDE)

{bml j_bullet.bmp}DDEExecute

{bml j_bullet.bmp}DDEInitiate

{bml j_bullet.bmp}DDEPoke

{bml j_bullet.bmp}DDERequest

{bml j_bullet.bmp}DDETerminate

{bml j_bullet.bmp}DDETerminateAll

{bml j_bullet.bmp}DDETimeOut

⁵⁴ CFAM_Dynamic_Data_Exchange

⁵⁵ Dynamic Data Exchange, command category

⁵⁶ Dynamic Data Exchange

⁵⁷ CmdFam:013

#58 S59 K60 +61 **DCL Environment Information**

{bml j_bullet.bmp}DCLHomeDir\$

{bml j_bullet.bmp}DCLOS\$

{bml j_bullet.bmp}DCLVersion\$

⁵⁸ CFAM_DCL_Environment_Information

⁵⁹ DCL Environment Information, command category

⁶⁰ DCL environment information

⁶¹ CmdFam:015

Environment Statements and Functions

Description	Function/Statement
Control an MCI device	<u>{bml j_bullet.bmp}MCI</u>
End the working session	<u>{bml j_bullet.bmp}SystemRestart</u>
Get information about the system	<u>{bml j_bullet.bmp}ReadINI\$</u> <u>{bml j_bullet.bmp}ReadINISection</u> <u>{bml j_bullet.bmp}SystemFreeMemory</u> <u>{bml j_bullet.bmp}SystemFreeResources</u> <u>{bml j_bullet.bmp}SystemTotalMemory</u> <u>{bml j_bullet.bmp}SystemWindowsDirectory\$</u> <u>{bml j_bullet.bmp}SystemWindowsVersion\$</u>
Modify the Windows environment	<u>{bml j_bullet.bmp}AddIni</u> <u>{bml j_bullet.bmp}RefreshIni</u> <u>{bml j_bullet.bmp}SystemMouseTrails</u> <u>{bml j_bullet.bmp}WriteINI</u>
Save and restore the environment	<u>{bml j_bullet.bmp}RestoreEnv</u> <u>{bml j_bullet.bmp}SaveEnv</u>
Set and get environment variables	<u>{bml j_bullet.bmp}Environ\$</u> <u>{bml j_bullet.bmp}GetEnv</u> <u>{bml j_bullet.bmp}SetEnv</u>

Error Trapping

Description	Function/Statement
Get error messages	<u>{bml j_bullet.bmp}Error\$</u>

⁶² CFAM_Environment_Statements_and_Functions

⁶³ Environment Statements and Functions

⁶⁴ Environment commands

⁶⁵ CmdFam:017

⁶⁶ CFAM_Error_Trapping

⁶⁷ Error Trapping, command category

⁶⁸ Error trapping

⁶⁹ CmdFam:019

Get error-status data	<u>{bml j _bullet.bmp}Err</u>
Get or set error-status data	<u>{bml j _bullet.bmp}Err</u>
Simulate run-time errors	<u>{bml j _bullet.bmp}Error</u>
Trap errors while a program is running	<u>{bml j _bullet.bmp}On Error</u>
	<u>{bml j _bullet.bmp}Resume</u>

#70 S71 K72 +73 **File Input and Output**

Description	Function/Statement
Access or create a file	<u>{bml j _bullet.bmp}Open</u>
Close files	<u>{bml j _bullet.bmp}Close</u>
	<u>{bml j _bullet.bmp}Reset</u>
Copy a file	<u>{bml j _bullet.bmp}FileCopy</u>
Get information about a file	<u>{bml j _bullet.bmp}EOF</u>
	<u>{bml j _bullet.bmp}FileAttr</u>
	<u>{bml j _bullet.bmp}FileDateTime</u>
	<u>{bml j _bullet.bmp}FileExists</u>
	<u>{bml j _bullet.bmp}FileLen</u>
	<u>{bml j _bullet.bmp}FileList</u>
	<u>{bml j _bullet.bmp}FileType</u>
	<u>{bml j _bullet.bmp}FindFile\$</u>
	<u>{bml j _bullet.bmp}FreeFile</u>
	<u>{bml j _bullet.bmp}Loc</u>
	<u>{bml j _bullet.bmp}LOF</u>
	<u>{bml j _bullet.bmp}Seek</u>

⁷⁰ CFAM_File_Input_and_Output

⁷¹ File Input and Output, command category

⁷² File input and output;Input, file;Output, file

⁷³ CmdFam:021

Manage disk drives or directories	<u>{bml j_bullet.bmp}ChDir</u>
	<u>{bml j_bullet.bmp}ChDrive</u>
	<u>{bml j_bullet.bmp}CurDir\$</u>
	<u>{bml j_bullet.bmp}DiskDrives</u>
	<u>{bml j_bullet.bmp}DiskFree</u>
	<u>{bml j_bullet.bmp}MkDir</u>
	<u>{bml j_bullet.bmp}Rmdir</u>
	<u>{bml j_bullet.bmp}DirExists</u>
Manage files	<u>{bml j_bullet.bmp}Dir\$</u>
	<u>{bml j_bullet.bmp}FileDirs</u>
	<u>{bml j_bullet.bmp}FileParse</u>
	<u>{bml j_bullet.bmp}Kill</u>
	<u>{bml j_bullet.bmp}Name</u>
Read from a file	<u>{bml j_bullet.bmp}Input #</u>
	<u>{bml j_bullet.bmp}Input\$</u>
	<u>{bml j_bullet.bmp}Line Input #</u>
Set or get file attributes	<u>{bml j_bullet.bmp}GetAttr</u>
	<u>{bml j_bullet.bmp}SetAttr</u>
Set read-write position in a file	<u>{bml j_bullet.bmp}Seek</u>
Write to a file	<u>{bml j_bullet.bmp}Print #</u>
	<u>{bml j_bullet.bmp}Write #</u>

#74 S75 K76 +77 **Flow Control**

Description

Function/Statement

Branch

[{bml j_bullet.bmp}GoSub...Return](#)

[{bml j_bullet.bmp}Goto](#)

[{bml j_bullet.bmp}On Error](#)

Exit or pause the program

[{bml j_bullet.bmp}EnableStopScript](#)

[{bml j_bullet.bmp}End](#)

⁷⁴ CFAM_Flow_Control

⁷⁵ Flow Control, command category

⁷⁶ Flow control

⁷⁷ CmdFam:023

{bml j_bullet.bmp}SleepUntil

{bml j_bullet.bmp}Stop

{bml j_bullet.bmp}WaitForTaskCompletion

Loop

{bml j_bullet.bmp}Do...Loop

{bml j_bullet.bmp}Exit Do

{bml j_bullet.bmp}Exit For

{bml j_bullet.bmp}For...Next

{bml j_bullet.bmp}While...Wend

Make decisions

{bml j_bullet.bmp}If...Then...Else

{bml j_bullet.bmp}Select Case

Pause script

{bml j_bullet.bmp}Sleep

Prevent other applications

{bml j_bullet.bmp}Exclusive

Run another program

{bml j_bullet.bmp}Shell

Yield control to other applications

{bml j_bullet.bmp}DoEvents

#78 S79 K80 +81 **Icons**

{bml j_bullet.bmp}SetIcon

{bml j_bullet.bmp}SetIconTitle

{bml j_bullet.bmp}ShowIcon

⁷⁸ CFAM_Icons

⁷⁹ Icons, command category

⁸⁰ Icons

⁸¹ CmdFam:024

#82 S83 K84 +85 **Keyboard Manipulation**

Description	Function/Statement
	Play back a string of keystrokes <u>{bml j_bullet.bmp}DoKeys</u>
	Record and play keystrokes, using the event queue <u>{bml j_bullet.bmp}QueEmpty</u> <u>{bml j_bullet.bmp}QueFlush</u> <u>{bml j_bullet.bmp}QueKeyDn</u> <u>{bml j_bullet.bmp}QueKeys</u> <u>{bml j_bullet.bmp}QueKeyUp</u> <u>{bml j_bullet.bmp}SendKeys</u>

#86 S87 K88 +89 **Math Statements and Functions**

Description	Function/Statement
General calculations	<u>{bml j_bullet.bmp}Exp</u> <u>{bml j_bullet.bmp}Log</u> <u>{bml j_bullet.bmp}Sqr</u>

- 82 CFAM_Keyboard_Manipulation
- 83 Keyboard Manipulation, command category
- 84 Keyboard manipulation
- 85 CmdFam:025
- 86 CFAM_Math_Statements_and_Functions
- 87 Math Statements and Functions
- 88 Math commands
- 89 CmdFam:027

Generate random numbers	<u>{bml j_bullet.bmp}Random</u>
	<u>{bml j_bullet.bmp}Randomize</u>
	<u>{bml j_bullet.bmp}Rnd</u>
Get absolute value	<u>{bml j_bullet.bmp}Abs</u>
Get the sign of an expression	<u>{bml j_bullet.bmp}Sgn</u>
Numeric conversion	<u>{bml j_bullet.bmp}Fix</u>
	<u>{bml j_bullet.bmp}Int</u>
Trigonometry	<u>{bml j_bullet.bmp}Atn</u>
	<u>{bml j_bullet.bmp}Cos</u>
	<u>{bml j_bullet.bmp}Sin</u>
	<u>{bml j_bullet.bmp}Tan</u>

#90 S91 K92 +93 **Menus**

Description

Function/Statement

Issue menu command

[{bml j_bullet.bmp}Menu](#)

Manipulate menu items

[{bml j_bullet.bmp}MenuItemChecked](#)

[{bml j_bullet.bmp}MenuItemEnabled](#)

[{bml j_bullet.bmp}MenuItemExists](#)

#94 S95 K96 +97 **Miscellaneous Statements and Functions**

⁹⁰ CFAM_Menus

⁹¹ Menus, command category

⁹² Menus

⁹³ CmdFam:029

⁹⁴ CFAM_Miscellaneous_Statements_and_Functions

⁹⁵ Miscellaneous Statements and Functions

⁹⁶ Miscellaneous DCL commands

⁹⁷ CmdFam:031

Description	Function/Statement
Comment	<u>{bml j_bullet.bmp}Rem</u> <u>{bml j_bullet.bmp}'</u>
Get command-line arguments	<u>{bml j_bullet.bmp}Command\$</u>
Sound a beep	<u>{bml j_bullet.bmp}Beep</u>

#98 S99 K100 +101 **Mouse Events**

{bml j_bullet.bmp}QueEmpty
{bml j_bullet.bmp}QueFlush
{bml j_bullet.bmp}QueMouseClicked
{bml j_bullet.bmp}QueMouseDown
{bml j_bullet.bmp}QueMouseDn
{bml j_bullet.bmp}QueMouseMove
{bml j_bullet.bmp}QueMouseUp
{bml j_bullet.bmp}QueSetRelativeWindow

⁹⁸ CFAM_Mouse_Events
⁹⁹ Mouse Events, command category
¹⁰⁰ Mouse events
¹⁰¹ CmdFam:033

Network Functions

{bml j_bullet.bmp}NetAttach

{bml j_bullet.bmp}NetConnectDrive

{bml j_bullet.bmp}NetDetach

{bml j_bullet.bmp}NetDirectoryRights

{bml j_bullet.bmp}NetDisconnectDrive

{bml j_bullet.bmp}NetGetDirectoryRights

{bml j_bullet.bmp}NetMemberOf

{bml j_bullet.bmp}NetStationID

{bml j_bullet.bmp}NetUserName

{bml j_bullet.bmp}NetworkStatus

¹⁰² CFAM_Network_Functions

¹⁰³ Network Functions, command category

¹⁰⁴ Network functions

¹⁰⁵ CmdFam:034

Operators

Description	Function/Statement
Arithmetic	<u>{bml j_bullet.bmp}*</u>
	<u>{bml j_bullet.bmp}+</u>
	<u>{bml j_bullet.bmp}-</u>
	<u>{bml j_bullet.bmp}/</u>
	<u>{bml j_bullet.bmp}\</u>
	<u>{bml j_bullet.bmp}^</u>
	<u>{bml j_bullet.bmp}Mod</u>
Comparison	<u>{bml j_bullet.bmp}<</u>
	<u>{bml j_bullet.bmp}<=</u>
	<u>{bml j_bullet.bmp}<></u>
	<u>{bml j_bullet.bmp}=</u>
	<u>{bml j_bullet.bmp}></u>
	<u>{bml j_bullet.bmp}>=</u>
Logical	<u>{bml j_bullet.bmp}And</u>
	<u>{bml j_bullet.bmp}Not</u>
	<u>{bml j_bullet.bmp}Or</u>
	<u>{bml j_bullet.bmp}Xor</u>

Printer Manipulation

{bml j_bullet.bmp}PrinterGetOrientation
{bml j_bullet.bmp}PrinterSetOrientation
{bml j_bullet.bmp}PrintFile

#114 S115 K116 +117 Procedure Statements

Description	Function/Statement
Call a subroutine	<u>{bml j_bullet.bmp}Call</u>
Declare a reference to an external routine	<u>{bml j_bullet.bmp}Declare</u>
Define a procedure	<u>{bml j_bullet.bmp}Function...End Function</u> <u>{bml j_bullet.bmp}Main</u> <u>{bml j_bullet.bmp}Sub...End Sub</u>
Exit a procedure	<u>{bml j_bullet.bmp}Exit Function</u> <u>{bml j_bullet.bmp}Exit Sub</u>

#118 S119 K120 +121 Strings

Description	Function/Statement
Convert to lowercase or uppercase letters	<u>{bml j_bullet.bmp}LCase\$</u> <u>{bml j_bullet.bmp}UCase\$</u>
Convert string to number	<u>{bml j_bullet.bmp}Val</u>
Create strings of repeating characters	<u>{bml j_bullet.bmp}Space\$</u> <u>{bml j_bullet.bmp}String\$</u>
Find the length of a string	<u>{bml j_bullet.bmp}Len</u>

114 CFAM_Procedure_Statements
115 Procedure Statements
116 Procedures
117 CmdFam:039
118 CFAM_Strings
119 Strings, command category
120 Strings
121 CmdFam:041

Logical operators used in string comparisons

{bml j_bullet.bmp}<=

{bml j_bullet.bmp}<>

{bml j_bullet.bmp}=

{bml j_bullet.bmp}>

{bml j_bullet.bmp}>=

Manipulate strings

{bml j_bullet.bmp}InStr

{bml j_bullet.bmp}Left\$

{bml j_bullet.bmp}LTrim\$

{bml j_bullet.bmp}Mid\$

{bml j_bullet.bmp}Null

{bml j_bullet.bmp}Right\$

{bml j_bullet.bmp}RTrim\$

{bml j_bullet.bmp}StrComp

{bml j_bullet.bmp}StringSort

{bml j_bullet.bmp}Str\$

{bml j_bullet.bmp}Trim\$

Parsing

{bml j_bullet.bmp}Item\$

{bml j_bullet.bmp}ItemCount

{bml j_bullet.bmp}Line\$

{bml j_bullet.bmp}LineCount

{bml j_bullet.bmp}Word\$

{bml j_bullet.bmp}WordCount

Work with ASCII and ANSI values

{bml j_bullet.bmp}Asc

{bml j_bullet.bmp}Chr\$

Logical operators used in string comparisons

{bml j_bullet.bmp}<

{bml j_bullet.bmp}<=

{bml j_bullet.bmp}<>

{bml j_bullet.bmp}=

{bml j_bullet.bmp}>

{bml j_bullet.bmp}>=

#122 S123 K124 +125

Variables and Constants

Description	Function/Statement
Assign value	<u>{bml j_bullet.bmp}Let</u>
Declare variables or constants	<u>{bml j_bullet.bmp}Const</u> <u>{bml j_bullet.bmp}Dim</u> <u>{bml j_bullet.bmp}ReDim</u>
Set default data type	<u>{bml j_bullet.bmp}Deftype</u>

#126 S127 K128 +129

Viewport Window Manipulation

{bml j_bullet.bmp}Print
{bml j_bullet.bmp}ViewportClear
{bml j_bullet.bmp}ViewportClose
{bml j_bullet.bmp}ViewportOpen

122 CFAM_Variables_and_Constants
123 Variables and Constants, command category
124 Variables;Constants
125 CmdFam:043
126 CFAM_Viewport_Window_Manipulation
127 Viewport Window Manipulation, command category
128 Viewport window manipulation
129 CmdFam:045

Window Manipulation

{bml j_bullet.bmp}AppActivate
{bml j_bullet.bmp}AppClose
{bml j_bullet.bmp}AppFileName\$
{bml j_bullet.bmp}AppFind
{bml j_bullet.bmp}AppGetActive\$
{bml j_bullet.bmp}AppGetPosition
{bml j_bullet.bmp}AppGetState
{bml j_bullet.bmp}AppHide
{bml j_bullet.bmp}AppList
{bml j_bullet.bmp}AppMaximize
{bml j_bullet.bmp}AppMinimize
{bml j_bullet.bmp}AppMove
{bml j_bullet.bmp}AppRestore
{bml j_bullet.bmp}AppSetState
{bml j_bullet.bmp}AppShow
{bml j_bullet.bmp}AppSize
{bml j_bullet.bmp}AppType
{bml j_bullet.bmp}HLine
{bml j_bullet.bmp}HPage
{bml j_bullet.bmp}HScroll
{bml j_bullet.bmp}VLine
{bml j_bullet.bmp}VPage
{bml j_bullet.bmp}VScroll
{bml j_bullet.bmp}WinActivate
{bml j_bullet.bmp}WinClose
{bml j_bullet.bmp}WinFind
{bml j_bullet.bmp}WinList
{bml j_bullet.bmp}WinMaximize
{bml j_bullet.bmp}WinMinimize
{bml j_bullet.bmp}WinMove
{bml j_bullet.bmp}WinRestore

¹³⁰ CFAM_Window_Manipulation

¹³¹ Window Manipulation, command category

¹³² Window manipulation

¹³³ CmdFam:047

{bml j_bullet.bmp}WinSize

#134 S135 K136 +137 **'**

Description

Begins a comment line or comments the remainder of the current line.

Syntax

' text

Comments

Causes the compiler to skip all characters between this character and the end of the current line.

See Also

[{bml j_bullet.bmp}Comments](#)

[{bml j_bullet.bmp}REM Statement](#)

#138 S139 K140 +141 *****

Description

Multiplication operator.

Syntax

expression1 * expression2

Returns

Returns the product of expression1 and expression2.

See Also

[{bml j_bullet.bmp}Operators](#)

134 CMD_COMMENT

135 **'**

136 **'**;Comments

137 Commands:0005

138 CMD_MUL

139 *****

140 *****;Multiplication operator

141 Commands:0010

#142 S143 K144 +145 **+**

Description

Addition operator.

Syntax

`expression1 + expression2`

Returns

Returns the sum of `expression1` and `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

#146 S147 K148 +149 **-**

Description

Subtraction operator.

Syntax

`expression1 - expression2`

Returns

Returns the difference between `expression1` and `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

142 CMD_ADD

143 **+**

144 +;Addition operator;Operators

145 Commands:0015

146 CMD_SUBTRACT

147 **-**

148 -;Subtraction operator;Operators

149 Commands:0020

#150 S151 K152 +153 /

Description

Long Division operator.

Syntax

expression1 / expression2

Returns

Returns the quotient of expression1 and expression2.

See Also

[{bml j_bullet.bmp}Operators](#)

#154 S155 K156 +157 <

Description

Less Than operator.

Syntax

expression1 < expression2

Returns

TRUE if expression1 is less than expression2, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if expression1 is less than expression2.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

150 CMD_DIV
151 /
152 /;Division, long, operator;Operators
153 Commands:0025
154 CMD_LT
155 <
156 <;Less Than operator;Operators
157 Commands:0030

#158 S159 K160 +161 <=

Description

Less Than or Equal To operator.

Syntax

expression1 <= expression2

Returns

TRUE if expression1 is less than or equal to expression2, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if expression1 is less than or equal to expression2.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

#162 S163 K164 +165 <>

Description

Not Equal operator.

Syntax

expression1 <> expression2

Returns

TRUE if expression1 is not equal to expression2, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if expression1 is not equal to expression2.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

158 CMD_LT_or_EQ

159 <=

160 <=;Less Than or Equal To operator;Operators

161 Commands:0035

162 CMD_NE

163 <>

164 <>;Not Equal operator;Operators

165 Commands:0040

#166 S167 K168 +169 =

Description

Equal To operator.

Syntax

```
expression1 = expression2
```

Returns

TRUE if `expression1` is equal to `expression2`, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if `expression1` is equal `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

#170 S171 K172 +173 >

Description

Greater Than operator.

Syntax

```
expression1 > expression2
```

Returns

TRUE if `expression1` is greater than `expression2`, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if `expression1` is greater than `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

166 CMD_EQ

167 =

168 =;Equal To operator;Operators

169 Commands:0045

170 CMD_GT

171 >

172 >;Greater Than operator;Operators

173 Commands:0050

#174 S175 K176 +177 **>=**

Description

Greater Than or Equal To operator.

Syntax

`expression1 >= expression2`

Returns

TRUE if `expression1` is greater than or equal to `expression2`, otherwise FALSE. If the two expressions are strings, then the operator performs a case-sensitive comparison between the two expressions, returning TRUE if `expression1` is greater than or equal to `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

[{bml j_bullet.bmp}Strings](#)

#178 S179 K180 +181 ****

Description

Integer Division operator.

Syntax

`expression1 \ expression2`

Returns

Returns the integer portion of the quotient of `expression1` divided by `expression2`. Any fractional part of the answer is dropped. For example `3 \ 1` equals 1 rather than 1.5.

See Also

[{bml j_bullet.bmp}Operators](#)

174 CMD_GT_or_EQ

175 >=

176 >=;Greater Than or Equal To operator;Operators

177 Commands:0055

178 CMD_INT_DIV

179 \

180 \;Division, integer, operator;Operators

181 Commands:0060

#182 S183 K184 +185 **^**

Description

Exponentiation operator.

Syntax

`expression1 ^ expression2`

Returns

Returns `expression1` raised to the power specified by `expression2`.

See Also

[{bml j_bullet.bmp}Operators](#)

#186 S187 K188 +189 **Abs Function**

Description

Returns the absolute value of a given number.

Syntax

`abs (number)`

Comment

`number` is an integer, long integer, single-precision, or double-precision value.

Example

[{bml j_bullet.bmp}Abs Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

182 CMD_EXP_SYM

183 ^

184 ^;Exponentiation operator;Operators

185 Commands:0065

186 CMD_Abs

187 Abs

188 Abs;Absolute value;Math commands

189 Commands:0070

ActivateControl Statement**{bmc no_dos.bmp}****Description**

Sets the focus to the control with the specified name or ID.

Syntax 1

```
ActivateControl ControlName$
```

Syntax 2

```
ActivateControl ControlID%
```

Comments

The control can be referenced using either the `ControlName$` or the `ControlID%`.

For push buttons, radio buttons, or check boxes, `ControlName$` is the name of the actual button. For listboxes, comboboxes, and edit boxes, `ControlName$` is the name that appears within the static text control that immediately precedes it in the window manager list.

A runtime error is generated if a control with `ControlName$` or `ControlID%` cannot be found.

This statement is used primarily to set the focus to a custom control within a dialog box. This is accomplished by setting the focus first to a known control that immediately precedes the custom control, then simulating a TAB key press:

```
ActivateControl "Portrait"
DoKeys "{TAB}"
```

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

AddIni Function**{bmc no_dos.bmp}**

```
190 CMD_ActivateControl
191 ActivateControl
192 ActivateControl;Control, activating;Dialog manipulation
193 Commands:0075
194 CMD_AddIni
195 AddIni
196 AddIni;INI file, adding to;Environment commands
197 Commands: 0077
```

Description

Reads the .INI settings of the source .INI file and adds them to the destination .INI file.

Syntax

```
AddIni(srcfile[, destfile$])
```

Comments

If the `destfile$` is not provided, WIN.INI is used. If the specified destination file does not exist, it is created. If an .INI setting is not present in the destination file, it is added. The function returns TRUE if it completes successfully or FALSE if it doesn't.

It is assumed that all entries in the `srcfile$` .INI file are in standard .INI format. Any entries that do not follow this Windows standard are ignored. If the specified source file does not exist, the function ends.

Example

[{bml j_bullet.bmp}AddIni Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#198 \$199 K200 +201 **And**

Description

And operator.

Syntax

```
expression1 And expression2
```

Returns

TRUE if `expression1` is TRUE and `expression2` is TRUE, otherwise FALSE.

If the two operands are numeric, then the result is the bitwise AND of the two arguments.

Notes

If either of the two operands is a floating point number, then the two operands are first converted to longs, then a bitwise AND is performed.

Example

[{bml j_bullet.bmp}And Example](#)

See Also

[{bml j_bullet.bmp}Operators](#)

198 CMD_And

199 And

200 And;And operator;Operators

201 Commands:0080

Description

Displays a box that prompts the user for a response and indicates which button the user pressed.

Syntax

```
AnswerBox(prompt$ [,button1$ [,button2$ [,button3$]]])
```

Returns

Returns an integer indicating which button was pushed (1 for the first button, 2 for the second, and so on). 0 is returned if the user cancels the dialog box (by pressing Escape).

Comments

The dialog box is sized to hold the entire contents of `prompt$`. The `prompt$` string can contain CR/LF--`Chr$(13)+Chr$(10)`--to separate lines.

The maximum size of the dialog box is 5/8 the width of the screen and 3/4 the height of the screen. If a given line is too long, it will be word wrapped.

The `button$` parameters specify the labels for one or more buttons to appear below the displayed `prompt$`. If no buttons are specified, then "OK" and "Cancel" are used. Up to three buttons can be specified. The width of each button is determined by the width of the widest button.

You can use the '&' symbol to specify an accelerator key in the label of the button.

The dialog box uses the 8 point Helvetica font.

Example

[{bml j_bullet.bmp}AnswerBox Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

```
202 CMD_AnswerBox
203 AnswerBox
204 AnswerBox;Box, answer;Input, button;Dialog display
205 Commands:0085
206 CMD_AppActivate
207 AppActivate
208 AppActivate;Application, activating;Window manipulation
209 Commands:0090
```

Description

Activates the specified top-level window.

Syntax

```
AppActivate WindowName$
```

Comments

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If the application is minimized, it will be restored by this command. If the window being activated is a full-screen DOS application, that application will be restored to full screen.

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box) or if the window is not found.

Example

[{bml j_bullet.bmp}AppActivate Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#210 #211 #212 +213 **AppClose Statement** **{bmc no_dos.bmp}**

Description

Closes the specified top-level application.

Syntax

```
AppClose [WindowName$]
```

Comments

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

[{bml j_bullet.bmp}AppClose Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

210 CMD_AppClose

211 AppClose

212 AppClose;Application, closing;Window manipulation

213 Commands:0095

#214 S215 K216 +217 **AppFileName\$ Function** {bmc no_dos.bmp}

Description

Returns the filename of the program that owns the top-level window of the given title.

Syntax

AppFileName\$ (WindowName\$)

Comments

The WindowName\$ parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

For DOS applications, the filename returned is that of the actual executable program, not WINOLDAP.EXE.

Example

{bml j_bullet.bmp}AppFileName\$ Example

See Also

{bml j_bullet.bmp}Window Manipulation

#218 S219 K220 +221 **AppFind Function** {bmc no_dos.bmp}

Description

Returns the full name of the top-level window, given a partial window name.

Syntax

AppFind\$ (partial_name\$)

Comments

The name is specified using the same format as the WinActivate statement.

Example

{bml j_bullet.bmp}AppFind Example

214 CMD_AppFileName_DOLLAR

215 AppFileName\$

216 AppFileName\$;Application, file name;Window manipulation;File name

217 Commands:0100

218 CMD_AppFind

219 AppFind

220 AppFind;Application, finding;Window manipulation;Title, window;Caption, window;Window title

221 Commands:0105

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#222 S223 K224 +225 **AppGetActive\$ Function** {bmc no_dos.bmp}

Description

Returns the title of the active window.

Syntax

```
AppGetActive$()
```

Comments

This function is used to retrieve the title of the active top-level window. The returned value can be used in subsequent calls to routines that require a title to a window.

If "" is returned, no window is active. This is a rare occurrence, it indicates that Windows may be in an unstable state.

Example 1

[{bml j_bullet.bmp}AppGetActive\\$ Example](#)

Example 2

```
n$ = AppGetActive$()
AppMinimize n$
```

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#226 S227 K228 +229 **AppGetPosition Statement** {bmc no_dos.bmp}

Description

Gets the position of a specified top-level window.

```
222 CMD_AppGetActive_DOLLAR
223 AppGetActive$
224 AppGetActive$;Application, getting active;Window manipulation;Window, getting active
225 Commands:0110
226 CMD_AppGetPosition
227 AppGetPosition
228 AppGetPosition;Application window, getting position;Window manipulation;Position, window
229 Commands:0115
```

Syntax

```
AppGetPosition x%,y%,width%,height%[,WindowName$]
```

Comments

The numeric arguments are filled with the pixel position of the window on the display. If an argument is not a variable reference, then the argument is ignored, as in the following example which only retrieves the position and ignores the width and height:

```
dim x as integer,y as integer
AppGetPosition x,y,0,0,"Program Manager"
```

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, then the window with the focus is used (i.e., the active window).

Example

[{bml j_bullet.bmp}AppGetPosition Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#230 S231 K232 +233 **AppGetState Function** {bmc no_dos.bmp}

Description

Returns an integer representing the state of the top-level window.

Syntax

```
AppGetState (WindowName$)
```

Returns

WS_MAXIMIZED	window is maximized
WS_MINIMIZED	window is minimized
WS_RESTORED	window is restored

Comments

The `WindowNme$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, then the window with the focus is used (i.e., the active window).

Example

[{bml j_bullet.bmp}AppGetState Example](#)

See Also

230 CMD_AppGetState

231 AppGetState

232 AppGetState;Application, getting state;Minimized;Maximized;Restored;Window manipulation

233 Commands:0120

{bml j_bullet.bmp}Window Manipulation

#234 S235 K236 +237 **AppHide Statement** {bmc no_dos.bmp}

Description

Hides the specified top-level window.

Syntax

AppHide [WindowName\$]

Comments

Nothing happens if the window is already hidden.

The WindowName\$ parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If WindowName\$ parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

{bml j_bullet.bmp}AppHide Example

See Also

{bml j_bullet.bmp}Window Manipulation

#238 S239 K240 +241 **AppList Statement** {bmc no_dos.bmp}

Description

Fills the specified string array with the names of all the active applications.

Syntax

AppList ArrayOfAppNames\$

234 CMD_AppHide
235 AppHide
236 AppHide;Application, hiding;Window manipulation
237 Commands:0125
238 CMD_AppList
239 AppList
240 AppList;Application, list;Window manipulation
241 Commands:0130

Comments

Before calling this function, you must declare the array to contain the names of the applications. Use the Dim command.

After calling this function, use the lbound() and ubound() functions to determine the new size of the array.

Example

{bml j_bullet.bmp}AppList Example

See Also

{bml j_bullet.bmp}Window Manipulation

#242 S243 K244 +245 **AppMaximize Statement** {bmc no_dos.bmp}

Description

Maximizes the specified top-level window.

Syntax

AppMaximize [WindowName\$]

Comments

Nothing happens if the window is already maximized or is hidden.

The WindowName\$ parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If WindowName\$ parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

{bml j_bullet.bmp}AppMaximize Example

See Also

{bml j_bullet.bmp}Window Manipulation

242 CMD_AppMaximize
243 AppMaximize
244 AppMaximize;Application, maximizing;Window manipulation
245 Commands:0135

#246 S247 K248 +249 **AppMinimize Statement** {bmc no_dos.bmp}

Description

Minimizes the specified top-level window.

Syntax

```
AppMinimize [WindowName$]
```

Comments

Nothing happens if the window is already minimized or is hidden.

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

{bml j_bullet.bmp}AppMinimize Example

See Also

{bml j_bullet.bmp}Window Manipulation

#250 S251 K252 +253 **AppMove Statement** {bmc no_dos.bmp}

Description

Sets the position of the specified top-level window to a given `x`, `y` pixel location.

Syntax

```
AppMove x%,y%[,WindowName$]
```

Comments

This statement has no effect if the top-level window is maximized.

246 CMD_AppMinimize

247 AppMinimize

248 AppMinimize;Application, minimizing;Window manipulation

249 Commands:0140

250 CMD_AppMove

251 AppMove

252 AppMove;Application window, moving;Window manipulation

253 Commands:0145

It is valid to specify an *x*, *y* location that is not visible.

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

`{bml j_bullet.bmp}AppMove Example`

See Also

`{bml j_bullet.bmp}Window Manipulation`

#254 \$255 K256 +257 **AppRestore Statement** **`{bmc no_dos.bmp}`**

Description

Restores the specified top-level window.

Syntax

`AppRestore [WindowName$]`

Comments

This statement has an effect only if the specified window is either maximized or minimized.

`AppRestore` will do nothing if the specified window is hidden.

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

`{bml j_bullet.bmp}AppRestore Example`

See Also

`{bml j_bullet.bmp}Window Manipulation`

254 `CMD_AppRestore`

255 `AppRestore`

256 `AppRestore;Application, restoring;Window manipulation`

257 `Commands:0150`

#258 S259 K260 +261 **AppSetState Statement** {bmc no_dos.bmp}

Description

Sets the state of the specified top-level window.

Syntax

```
AppSetState [WindowName$]
```

Comments

This statement sets the state of the specified top-level window to one of the following values:

```
WS_MAXIMIZED      maximize the window
WS_MINIMIZED       minimize the window
WS_RESTORED        restore the window
```

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

Example

{bml j_bullet.bmp}AppSetState Example

See Also

{bml j_bullet.bmp}Window Manipulation

#262 S263 K264 +265 **AppShow Statement** {bmc no_dos.bmp}

Description

Shows the specified top-level window.

Syntax

```
AppShow [WindowName$]
```

Comments

```
258 CMD_AppSetState
259 AppSetState
260 AppSetState;Application, setting state;Maximized;Minimized;Restored;Window manipulation
261 Commands:0155
262 CMD_AppShow
263 AppShow
264 AppShow;Application, showing;Window manipulation
265 Commands:0160
```

Nothing happens if the window is already displayed.

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

[{bml j_bullet.bmp}AppShow Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#266 \$267 K268 +269 **AppSize Statement** **{bmc no_dos.bmp}**

Description

Sets the width and height of the specified top-level window. It lets you size sizable and non-sizable windows.

Syntax

```
AppSize width%,height%[,WindowName$]
```

Comments

This statement works only if the specified application is restored (i.e., not minimized or maximized).

The `WindowName$` parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If `WindowName$` parameter is missing, the window with the focus is used (i.e., the active window).

A runtime error results if the window being activated is not enabled (which is the case if that application currently is displaying a modal dialog box).

Example

[{bml j_bullet.bmp}AppSize Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#270 S271 K272 +273 **AppType Function** {bmc no_dos.bmp}

Description

Returns an integer representing the type of application owning the specified window:

Syntax

AppType[(WindowName\$)]

Returns

TYPE_DOS DOS executable
TYPE_WINDOWS Windows executable

Comments

The WindowName\$ parameter is the title of a top-level window. The complete text of the window title is required. The title is not case-sensitive.

If WindowName\$ parameter is missing, the window with the focus is used (i.e., the active window).

Example

{bml j_bullet.bmp}AppType Example

See Also

{bml j_bullet.bmp}Window Manipulation

#274 S275 K276 +277 **ArrayDims Function**

Description

Returns an integer representing the number of dimensions in the specified array.

Syntax

ArrayDims (arrayvariable)

270 CMD_AppType
271 AppType
272 AppType;Application, type of;Window manipulation
273 Commands:0170
274 CMD_ArrayDims
275 ArrayDims
276 ArrayDims;Array, number of dimensions;Arrays
277 Commands:0175

Comments

If the function indicates zero dimensions, the array was declared as an empty array (e.g. `dim x$()`).

Example 1

[{bml j_bullet.bmp}ArrayDims Example](#)

Example 2

```
sub main()
dim f$() 'allocate empty array
files f$, "C:\*.BAT" 'fill the array
if dims(f$) = 0 then exit sub 'exit if no
                        'elements
end sub
```

See Also

[{bml j_bullet.bmp}Arrays](#)

#278 #279 #280 #281 **ArraySort Statement**

Description

Sorts a single-dimensioned array.

Syntax 1

```
ArraySort s$()
```

Syntax 2

```
ArraySort a%()
```

Syntax 3

```
ArraySort a&()
```

Syntax 4

```
ArraySort a!()
```

Syntax 5

```
ArraySort a#()
```

Comments

²⁷⁸ CMD_ArraySort

²⁷⁹ ArraySort

²⁸⁰ ArraySort;Array, sorting;Arrays

²⁸¹ Commands:0180

If a string array is specified, then the routine sorts alphabetically in ascending order (using case-sensitive string comparisons). If a numeric array is specified, the routine sorts lower numbers to the lowest array index locations.

A runtime error results if an array with more than one dimension is specified.

Example 1

[{bml j_bullet.bmp}ArraySort Example](#)

Example 2

```
sub main()  
    dim a$(100)  
    :  
    :  
    ArraySort a$  
    result = SelectBox("Title", "Prompt:", a$)  
end sub
```

See Also

[{bml j_bullet.bmp}Arrays](#)

#282 #283 #284 +285 **Asc Function**

Description

Returns the numeric ASCII code for the first character of the specified string.

Syntax

```
asc(text$)
```

Comments

The return value is an integer between 0 and 255.

Example

[{bml j_bullet.bmp}Asc Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Strings](#)

282 CMD_Asc

283 Asc

284 Asc;Ascii code, converting to;Conversions;Strings

285 Commands:0185

#286 S287 K288 +289 **AskBox\$ Function**

Description

Displays a dialog box that asks the user to enter a string in an edit box, and returns the string the user enters.

Syntax

```
AskBox$(prompt$ [,default$])
```

Returns

Returns a string that the user typed, or returns an empty string indicating that the user canceled the dialog box.

Comments

The dialog box is sized to the appropriate width depending on the width of `prompt`.

When the dialog box is displayed, the edit box has the focus.

If `default` is specified, then this is used as the initial contents of the edit field.

The dialog box has OK and Cancel buttons.

The dialog box uses the 8 point Helvetica font.

The maximum number of characters that can be typed into the edit box is 255.

Example

{bml j_bullet.bmp}AskBox\$ Example

See Also

{bml j_bullet.bmp}Dialog Display

#290 S291 K292 +293 **AskPassword\$ Function**

286 CMD_AskBox_DOLLAR

287 AskBox\$

288 AskBox\$;Box, ask;Input, string;Dialog display

289 Commands:0190

290 CMD_AskPassword_DOLLAR

291 AskPassword\$

292 AskPassword\$;Password, asking;Input, password;Dialog display

293 Commands:0195

Description

Displays a dialog box that asks the user to enter his or her password in an edit box, and returns the string the user enters.

Syntax

AskPassword\$(prompt\$)

Returns

Returns the string that the user typed, or returns an empty string indicating that the user canceled the dialog box.

Comments

Unlike the AskBox command, the user sees asterisks in place of the characters that are actually typed. This allows the input of passwords.

When the dialog box is displayed, the edit box has the focus.

The dialog box is sized to the appropriate width depending on the width of prompt\$.

The dialog box has OK and Cancel buttons.

The dialog box uses the 8 point Helvetica font.

The maximum number of characters that can be typed into the edit box is 255.

Example

{bml j_bullet.bmp}AskPassword\$ Example

See Also

{bml j_bullet.bmp}Dialog Display

#294 #295 #296 #297 Atn Function

Description

Returns a double-precision number representing the arctangent of a given number.

Syntax

atn(number#)

Comments

To calculate the value of PI, use:

$4 * \text{ATN}(1)$

Example

{bml j_bullet.bmp}Atn Example

See Also

²⁹⁴ CMD_Atn

²⁹⁵ Atn

²⁹⁶ Atn;Arctangent;Math commands

²⁹⁷ Commands:0200

#298 \$299 K300 +301 **ATTR_ARCHIVE**

Description

Constant.

Value

32

Comments

Bit position of a file attribute indicating that a file hasn't been backed up. This value is used by GetAttr, SetAttr, and FileList.

#302 \$303 K304 +305 **ATTR_DIRECTORY**

Description

Constant.

Value

16

Comments

Bit position of a file attribute indicating that a file is a directory entry. This value is used by GetAttr, SetAttr, and FileList.

#306 \$307 K308 +309 **ATTR_HIDDEN**

Description

298 CMD_ATTR_ARCHIVE
299 ATTR_ARCHIVE
300 ATTR_ARCHIVE;Attributes, file;File attributes
301 Commands:0205
302 CMD_ATTR_DIRECTORY
303 ATTR_DIRECTORY
304 ATTR_DIRECTORY;Attributes, file;File attributes
305 Commands:0210
306 CMD_ATTR_HIDDEN
307 ATTR_HIDDEN
308 ATTR_HIDDEN;Attributes, file;File attributes
309 Commands:0215

Constant.

Value

2

Comments

Bit position of a file attribute indicating that a file is hidden. This value is used by GetAttr, SetAttr, and FileList.

#310 S311 K312 +313 **ATTR_NONE**

Description

Constant.

Value

64

Comments

Bit position of a file attribute indicating that a file has no other attributes set. This value is used by GetAttr, SetAttr, and FileList.

#314 S315 K316 +317 **ATTR_NORMAL**

Description

Constant.

Value

0

Comments

Bit position of a file attribute indicating that a file has no other attributes set. This value is used by GetAttr, SetAttr, and FileList.

³¹⁰ CMD_ATTR_NONE

³¹¹ ATTR_NONE

³¹² ATTR_NONE;Attributes, file;File attributes

³¹³ Commands:0217

³¹⁴ CMD_ATTR_NORMAL

³¹⁵ ATTR_NORMAL

³¹⁶ ATTR_NORMAL;Attributes, file;File attributes

³¹⁷ Commands:0220

#318 S319 K320 +321 **ATTR_READONLY**

Description

Constant.

Value

1

Comments

Bit position of a file attribute indicating that a file is read-only. This value is used by GetAttr, SetAttr, and FileList.

#322 S323 K324 +325 **ATTR_SYSTEM**

Description

Constant.

Value

4

Comments

Bit position of a file attribute indicating that a file is a system file. This value is used by GetAttr, SetAttr, and FileList.

#326 S327 K328 +329 **ATTR_VOLUME**

Description

Constant.

- 318 CMD_ATTR_READONLY
- 319 ATTR_READONLY
- 320 ATTR_READONLY;Attributes, file;File attributes
- 321 Commands:0225
- 322 CMD_ATTR_SYSTEM
- 323 ATTR_SYSTEM
- 324 ATTR_SYSTEM;Attributes, file;File attributes
- 325 Commands:0230
- 326 CMD_ATTR_VOLUME
- 327 ATTR_VOLUME
- 328 ATTR_VOLUME;Attributes, file;File attributes
- 329 Commands:0235

Value

8

Comments

Bit position of a file attribute indicating that a file is the volume label. This value is used by GetAttr, SetAttr, and FileList.

#330 \$331 K332 +333 **Beep Statement****Description**

Makes a single system beep.

Syntax

```
beep
```

Example

```
{bml j_bullet.bmp}Beep Example
```

#334 \$335 K336 +337 **Begin Dialog...End Dialog Statement****Description**

The BEGIN DIALOG...END DIALOG block defines a dialog box template.

Syntax

```
begin dialog DialogName$,x%,y%,width%,height%  
  
    ...  
  
    ...  
  
end dialog
```

Comments

The *x*, *y* parameters are the dialog coordinates of the upper left hand corner of the dialog box relative to the parent window.

The *width*, *height* parameters specify the width and height dimensions of the dialog box in dialog units.

```
330 CMD_Beep  
331 Beep  
332 Beep  
333 Commands:0240  
334 CMD_Begin_Dialog  
335 Begin Dialog  
336 Begin Dialog;Begin;Dialog;End;Dialog box template;Dialog creation  
337 Commands:0245
```

The `DialogName` parameter specifies the name used to dimension a variable of this type (i.e., a variable that refers to this dialog box template). Once a dialog template has been defined, a variable can be dimensioned using this name:

```
dim MyDlg as DialogName
```

An error is generated if the dialog box template is empty.

A dialog template must have at least one [PushButton](#), [OKButton](#), or [CancelButton](#). Otherwise, there will be no way to close the dialog box.

Dialog units are defined as 1/4 the width of the font in the horizontal direction and 1/8 the height of the font in the vertical direction. All dialogs created by DCL use an 8 point Helvetica font.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#338 #339 #340 +341 **ButtonEnabled Function** **{bmc no_dos.bmp}**

Description

Determines whether the specified button within the current window is enabled.

Syntax 1

```
ButtonEnabled(ButtonName$)
```

Syntax 2

```
ButtonEnabled(ButtonID%)
```

Returns

Returns the integer TRUE if the specified button within the current window is enabled, otherwise this function returns FALSE.

Comments

The button can be specified either by its name (`ButtonName$`) or using its ID (`ButtonID%`).

`ButtonName$` is the text on the button's label.

When a button is enabled, it can be pressed using the [SelectButton](#) statement.

Example

[{bml j_bullet.bmp}ButtonEnabled Example](#)

See Also

338 CMD_ButtonEnabled

339 ButtonEnabled

340 ButtonEnabled;Button, determine if enabled;Dialog manipulation;Push button, determine if enabled

341 Commands:0250

{bml j_bullet.bmp}Dialog Manipulation

#342 S343 K344 +345

ButtonExists Function

{bmc no_dos.bmp}

Description

Determines whether the specified button exists within the current window.

Syntax 1

ButtonExists (ButtonName\$)

Syntax 2

ButtonExists (ButtonID%)

Returns

Returns the integer TRUE if the specified button exists within the current window, otherwise this function returns FALSE.

Comments

The button can be specified either by its name (ButtonName\$) or using its id (ButtonID%). The ButtonName\$ is the text of button's label.

Example

{bml j_bullet.bmp}ButtonExists Example

See Also

{bml j_bullet.bmp}Dialog Manipulation

#346 S347 K348 +349

Call Statement

Description

Transfers control to the specified subroutine, optionally passing arguments.

342 CMD_ButtonExists

343 ButtonExists

344 ButtonExists;Button, determine if exists;Dialog manipulation;Push button, determine if exists

345 Commands:0255

346 CMD_Call

347 Call

348 Call;Procedure statements;Subroutine, calling

349 Commands:0260

Syntax

```
call subroutine_name [(arguments)]
```

Comments

Using this statement is equivalent to:

```
subroutine_name [arguments]
```

Use of the `call` statement is never required.

Example

[{bml j_bullet.bmp}Call Example](#)

See Also

[{bml j_bullet.bmp}Procedure Statements](#)

#350 #351 K352 +353 **CancelButton Statement**

Description

Defines a Cancel button that appears within a dialog box template.

Syntax

```
CancelButton x%,y%,width%,height%
```

Comments

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#354 #355 K356 +357 **CDbl Function**

350 CMD_CancelButton

351 CancelButton

352 CancelButton;Button, cancel;Dialog creation

353 Commands:0265

354 CMD_CDbl

355 CDbl

356 CDbl;Double precision, converting to;Conversions

357 Commands:0270

Description

Returns the double-precision equivalent of the passed numeric expression.

Syntax

`cdbl (number#)`

Comments

This function has the same effect as assigning the numeric expression to a double precision variable.

Example

[{bml j_bullet.bmp}Cdbl Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

#358 #359 #K360 +361 **ChDir Statement**

Description

Changes the current directory on the specified drive.

Syntax

`chdir newdir$`

Comments

This statement will not change to the specified drive.

If you do not include a drive in the `newdir$` parameter, the current drive is assumed.

This statement behaves the same as the DOS "cd" command.

Example

[{bml j_bullet.bmp}ChDir Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

358 CMD_ChDir
359 ChDir
360 ChDir;Directory, changing current;File input and output
361 Commands:0275

#362 S363 K364 +365 ChDrive Statement

Description

Changes the default drive.

Syntax

```
chdrive DriveLetter$
```

Comments

Only the first character of `DriveLetter$` is used. You can use the same string for the `ChDrive` and `ChDir` commands.

`DriveLetter$` is case insensitive.

If `DriveLetter$` is empty, then the current drive is not changed.

Example

{bml j_bullet.bmp}ChDrive Example

See Also

{bml j_bullet.bmp}File Input and Output

#366 S367 K368 +369 CheckBox Statement

Description

Defines a checkbox within a dialog box template.

Syntax

```
CheckBox x%,y%,width%,height%,title$, .Field
```

Comments

Checkboxes can be either on or off, depending on the value of `.Field`.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

```
362 CMD_ChDrive
363 ChDrive
364 ChDrive;Drive, changing default;File input and output
365 Commands:0280
366 CMD_CheckBox
367 CheckBox
368 CheckBox;Check box, defining;Dialog creation
369 Commands:0285
```

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box. The `width` and `height` parameters specify the dimensions of the check box and the label.

On entry to the `Dialog` statement, the `.Field` variable is used to set the initial state of the checkbox. On exit from the `Dialog` statement, the `.Field` variable is used to determine the final state of the checkbox. If the value is 0, the checkbox is unchecked; 1 indicates that the checkbox is checked.

The `title$` parameter may contain an ampersand character to denote an underlined accelerator, such as "&Font" for `Font`.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#370 #371 #372 +373 **CheckboxEnabled Function** **{bmc no_dos.bmp}**

Description

Determines whether the specified checkbox in the current window is enabled.

Syntax 1

`CheckboxEnabled (CheckboxName$)`

Syntax 2

`CheckboxEnabled (CheckboxID%)`

Returns

Returns the integer TRUE if the specified checkbox within the current window is enabled, otherwise this function returns FALSE.

Comments

The checkbox can be specified either by its name (`CheckboxName$`) or using its id (`CheckboxID%`). `CheckboxName$` is the text of the check box label.

When a checkbox is enabled, its state can be set using the [SetCheckbox](#) statement.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

³⁷⁰ CMD_CheckboxEnabled

³⁷¹ CheckboxEnabled

³⁷² CheckboxEnabled;Check box, determine if enabled;Dialog manipulation

³⁷³ Commands:0290

#374 S375 K376 +377

CheckboxExists Function {bmc no_dos.bmp}

Description

Determines whether the specified checkbox exists within the current window.

Syntax 1

```
CheckboxExists (CheckboxName$)
```

Syntax 2

```
CheckboxExists (CheckboxID%)
```

Returns

Returns the integer TRUE if the specified checkbox exists within the current window, otherwise this function returns FALSE.

Comments

The checkbox can be specified either by its name (`CheckboxName$`) or using its ID (`CheckboxID%`). `CheckboxName$` is the text of the check box label.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#378 S379 K380 +381 Chr\$ Function

Description

Returns the character for the specified ASCII code.

Syntax

374 CMD_CheckboxExists

375 CheckboxExists

376 CheckboxExists;Check box, determine if exists;Dialog manipulation

377 Commands:0295

378 CMD_Chr_DOLLAR

379 Chr\$

380 Chr\$;Character, converting to;Conversions;Strings;ASCII code, changing to character

381 Commands:0300

```
chr$(AsciiCode%)
```

Comments

AsciiCode must be an integer between 0 and 255.

The Chr\$() function can be used within constant declarations, as in the following example:

```
const crlf$ = chr$(13) + chr$(10)
```

Example

[{bml j_bullet.bmp}Chr\\$ Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Strings](#)

#382 #383 K384 +385 CInt Function

Description

Returns the integer portion of the given expression. In cases of fractions, this function rounds to the nearest integer.

Syntax

```
cint(number#)
```

Comments

The passed numeric expression must be within the following range:

```
-32768 <= number <= 32767
```

A runtime error results if the passed expression is not within the above range.

This function has the same effect as assigning a numeric expression to a variable of type integer.

Example

[{bml j_bullet.bmp}CInt Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Fix](#)

[{bml j_bullet.bmp}Int](#)

382 CMD_CInt

383 CInt

384 CInt;Integer, converting to;Conversions

385 Commands:0305

#386 S387 K388 +389 **Clipboard\$ Statement and Function**

Description

The `Clipboard$` statement puts a string in the clipboard.

The `Clipboard$` function returns the text contained in the clipboard.

Statement Syntax

```
clipboard$ NewContent$
```

Comments

This statement puts `NewContent$` into the clipboard.

Function Syntax

```
clipboard$()
```

Returns

Text contained in the clipboard.

Comments

If the clipboard doesn't contain text, or the clipboard is empty, then an empty string is returned.

Example

[{bml j_bullet.bmp}Clipboard\\$ Example](#)

See Also

[{bml j_bullet.bmp}Clipboard Manipulation](#)

#390 S391 K392 +393 **ClipboardClear Statement**

Description

Clears (removes the contents of) the clipboard.

```
386 CMD_Clipboard_DOLLAR
387 Clipboard$
388 Clipboard$;Clipboard, filling;Clipboard, contents
389 Commands:0310
390 CMD_ClipboardClear
391 ClipboardClear
392 ClipboardClear;Clipboard, clearing
393 Commands:0315
```

Syntax

ClipboardClear

Example

[{bml j_bullet.bmp}ClipboardClear Example](#)

See Also

[{bml j_bullet.bmp}Clipboard Manipulation](#)

#394 #395 K396 +397 CLng Function

Description

Returns a long integer representing the result of the given numeric expression.

Syntax

clng(number#)

Comments

The passed numeric expression must be within the following range:

-2147483648 <= number <= 2147483647

A runtime error results if the passed expression is not within the above range.

This function has the same effect as assigning a numeric expression to a long variable.

Example

[{bml j_bullet.bmp}CLng Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

#398 #399 K400 +401 Close Statement

Description

```
394 CMD_CLng
395 CLng
396 CLng;Integer, long, converting to;Conversions
397 Commands:0320
398 CMD_Close
399 Close
400 Close;File, closing;Closing files;File input and output
401 Commands:0325
```

Closes open file(s).

Syntax

```
close [[#] filename% [, [#] filename%]]
```

Comments

The file number is assigned by the Open command.

If no arguments are specified, this statement closes all files. Otherwise, this statement closes each specified file.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#402 #403 #404 #405 **Combobox Statement**

Description

Defines a combobox that appears within a dialog box template.

Syntax

```
ComboBox x%,y%,width%,height%,items$(),.Field
```

Comments

The `items$` array must be a single-dimension array of strings. The elements of this array are placed into the combobox when the dialog box is created. The `.Field` parameter defines the name used to extract which string occupies the combobox when the dialog box ends. On exit from the Dialog statement, the `.Field` contains an index to the item that is highlighted in the combobox.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x,y,width,height` parameters are specified in dialog coordinates. The `x,y` position is relative to the upper left corner of the dialog box. The `width` and `height` parameters define the dimensions of the drop-down list box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

402 CMD_Combobox

403 Combobox

404 Combobox;Combo box, defining;Dialog creation

405 Commands:0330

#406 S407 K408 +409 **ComboboxEnabled Function** {bmc no_dos.bmp}

Description

Determines whether the specified combobox is enabled in the current window or dialog box.

Syntax 1

`ComboboxEnabled(name$)`

Syntax 2

`ComboboxEnabled(id%)`

Comment

The combobox can be specified either by `name$` or `id%`. The `name$` specifies the text that appears in the static control that immediately precedes the combobox control in the window list (or dialog template).

This function returns the integer TRUE if the given combobox is enabled within the current window or dialog box, FALSE otherwise.

A runtime error is generated if the specified combobox does not exist.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#410 S411 K412 +413 **ComboboxExists Function** {bmc no_dos.bmp}

Description

Determines whether the specified combobox exists in the current window or dialog box.

Syntax 1

```
406 CMD_ComboboxEnabled
407 ComboboxEnabled
408 ComboboxEnabled;Combo box, determining if enabled;Dialog manipulation
409 Commands:0335
410 CMD_ComboboxExists
411 ComboboxExists
412 ComboboxExists;Combo box, determining if exists;Dialog manipulation
413 Commands:0340
```


ComboboxExists (name\$)

Syntax 2

ComboboxExists (id%)

Comment

The combobox can be specified either by `name$` or `id%`. The `name$` specifies the text that appears in the static control that immediately precedes the combobox control in the window list (or dialog template).

This function returns the integer TRUE if the given combobox exists within the current window or dialog box, FALSE otherwise.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#414 \$415 K416 +417 Command\$ Statement

Description

Returns a string representing the arguments from the command line used to start the application.

Syntax

Command\$

Comment

When running scripts from the DCL editor, you enter command-line arguments through the [Arguments](#) command.

#418 \$419 K420 +421 Const Statement

Description

Declares a constant for use within the current script.

Syntax

const name = expression [,name = expression]...

414 CMD_Command_DOLLAR

415 Command\$

416 Command\$;Command line arguments;Arguments, command line;Procedure statements

417 Commands:0345

418 CMD_Const

419 Const

420 Const;Constant, declaring

421 Commands:0350

Comments

The `name` is only valid within the current DCL script.

The `expression` must be assembled from literals, or other constants. Calls to functions are not allowed.

See Also

[{bmlj_bullet.bmp}Variables and Constants](#)

#422 S423 K424 +425 **Cos Function**

Description

Returns a double-precision number representing the cosine of a given angle.

Syntax

```
cos (angle#)
```

Comments

The `angle` parameter is given in radians.

See Also

[{bmlj_bullet.bmp}Math Statements and Functions](#)

#426 S427 K428 +429 **CSng Function**

Description

Returns a single-precision number representing the result of the given numeric expression.

Syntax

```
CSng (number#)
```

Comments

This function has the same effect as assigning a numeric expression to a single-precision variable.

Example

```
422 CMD_Cos
423 Cos
424 Cos;Cosine;Math commands
425 Commands:0355
426 CMD_CSng
427 CSng
428 CSng;Single precision, converting to;Conversions
429 Commands:0360
```

[{bml j_bullet.bmp}CSng Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

#430 \$431 K432 +433 **CStr Function**

Description

Returns a string representing the result of the given expression.

Syntax

CStr (number#)

Example

[{bml j_bullet.bmp}CStr Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

#434 \$435 K436 +437 **CurDir\$ Function**

Description

Gets the current directory.

Syntax

curdir\$[(drive\$)]

Returns

Returns the current directory on the specified drive. If no drive is specified, the current directory on the current drive is returned.

Comments

A runtime error results if `drive$` is invalid.

```
430 CMD_CStr
431 CStr
432 CStr;String, converting to;Conversions;Strings
433 Commands:0365
434 CMD_CurDir_DOLLAR
435 CurDir$
436 CurDir$;Directory, getting current;File input and output
437 Commands:0370
```

Example

[{bml j_bullet.bmp}CurDir\\$ Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#438 S439 K440 +441 **Date\$ Statement and Function**

Description

The `date$` assignment statement sets the system date. The `date$` function returns the system date.

Assignment Syntax

```
date$ = newdate$
```

Comments

Sets the system date to the specified date. The format for `newdate$` is any of the following:

MM-DD-YYYY

MM-DD-YY

MM/DD/YYYY

MM/DD/YY

Example 1

[{bml j_bullet.bmp}Date\\$ Statement Example](#)

Example 2

```
date$ = "7-28-1992"
```

Function Syntax

```
date$[ () ]
```

Returns

The `date$` function returns the current system date as a 10 character string.

Comments

The format for the returned date is MM-DD-YYYY.

Example

[{bml j_bullet.bmp}Date\\$ Function Example](#)

438 CMD_Date_DOLLAR

439 Date\$

440 Date\$;Date, setting;Date, getting;System date;Date functions

441 Commands:0375

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#442 S443 K444 +445 **DateSerial Function**

Description

Returns a double-precision number representing the specified date. The number is returned in days where Dec 30, 1899 is 0.

Syntax

```
DateSerial (year%,month%,day%)
```

Example

[{bml j_bullet.bmp}DateSerial Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Date and Time Functions](#)

#446 S447 K448 +449 **DateValue Function**

Description

Returns a double-precision number representing the date contained in the specified string argument.

Syntax

```
DateValue (date_string$)
```

Comments

This function interprets the passed `date_string$` parameter looking for a valid date specification. Date specifications vary depending on the international settings contained in the INTL section of the WIN.INI file.

```
442 CMD_DateSerial
443 DateSerial
444 DateSerial;Date, getting;Date functions
445 Commands:0380
446 CMD_DateValue
447 DateValue
448 DateValue;Date, string;Date functions
449 Commands:0385
```

The `date_string$` parameter can contain valid date items separated by date separators such as slash (/), minus (-), or comma (.). The date items must follow the ordering determined by the current date format settings in use by Windows.

Date strings can contain an optional time specification, but this is not used in the formation of the returned value.

Months can appear in their abbreviated formats, such as Jan, Feb, or Mar.

Example

[{bml j_bullet.bmp}DateValue Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Date and Time Functions](#)

#450 \$451 K452 +453 **Day Function**

Description

Returns an integer representing the day of the month for the date encoded in the specified `serial` parameter. The value returned is between 1 and 31 inclusive.

Syntax

```
day(serial#)
```

Comment

You can obtain the value for the `serial#` parameter by using the [DateSerial](#) or [DateValue](#) command.

Example

[{bml j_bullet.bmp}Day Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#454 \$455 K456 +457 **DCLHomeDir\$ Function**

450 CMD_Day

451 Day

452 Day;Date functions

453 Commands:0390

454 CMD_DCLHomeDir_DOLLAR

455 DCLHomeDir\$

456 DCLHomeDir\$;Directory, containing DCL;DCL environment information

457 Commands:0392

Description

Returns the directory containing DCL.

Syntax

DCLhomedir\$()

Comments

This function is used to locate files that are part of the DCL system itself.

Example

{bml j_bullet.bmp}DCLHomeDir\$ Example

See Also

{bml j_bullet.bmp}DCL Environment Information

#458 \$459 K460 +461 **DCLOS\$ Function**

Description

Returns a number indicating the host operating environment.

Syntax

DCLOS\$()

Returns

0 Windows

1 DOS

Example

{bml j_bullet.bmp}DCLOS\$ Example

See Also

{bml j_bullet.bmp}DCL Environment Information

#462 \$463 K464 +465 **DCLVersion\$ Function**

458 CMD_DCLOS_DOLLAR

459 DCLOS\$

460 DCLOS\$;Operating environment, getting;DCL environment information

461 Commands:0394

462 CMD_DCLVersion_DOLLAR

463 DCLVersion\$

464 DCLVersion\$;Version of DCL;DCL environment information

465 Commands:0396

Description

Returns the version of DCL.

Syntax

```
DCLVersion$()
```

Comments

This function returns the major and minor version numbers in the format `major.minor`, as in "1.1".

Example

{bml j_bullet.bmp}DCLVersion\$ Example

See Also

{bml j_bullet.bmp}DCL Environment Information

#466 S467 K468 +469 **DDEExecute Statement** {bmc no_dos.bmp}

Description

Sends an execute message to another application.

Syntax

```
DDEExecute channel%,command$
```

Comments

The `channel` must first be initiated using `DDEInitiate`. An error will result if `channel` is invalid.

If the receiving application does not execute the instructions, a runtime error will be generated.

The format of `command$` depends on the receiving application.

Example

{bml j_bullet.bmp}DDE Example

See Also

{bml j_bullet.bmp}Dynamic Data Exchange

466 CMD_DDEExecute

467 DDEExecute

468 DDEExecute;Dynamic Data Exchange

469 Commands:0398

#470 S471 K472 +473

DDEInitiate Function {bmc no_dos.bmp}

Description

Initializes a DDE link to another application.

Syntax

```
DDEInitiate(app$,topic$)
```

Returns

Returns a unique integer used to subsequently refer to the open DDE channel.

Comments

The function returns 0 if the link cannot be established. This will occur under the following circumstances:

- The specified application is not running
- The topic was invalid for that application
-
-
- Insufficient memory or system resources to establish DDE link

Example

{bml j_bullet.bmp}DDE Example

See Also

{bml j_bullet.bmp}Dynamic Data Exchange

#474 S475 K476 +477

DDEPoke Statement {bmc no_dos.bmp}

Description

Sets the value of a data item in the receiving application associated with an open DDE link.

470 CMD_DDEInitiate

471 DDEInitiate

472 DDEInitiate;Dynamic Data Exchange

473 Commands:0400

474 CMD_DDEPoke

475 DDEPoke

476 DDEPoke;Dynamic Data Exchange

477 Commands:0405

Syntax

DDEPoke channel%,dataItem\$,value\$

Comments

The channel must first be initiated using DDEInitiate. An error will result if channel is invalid.

The format for dataItem\$ and value\$ depends on the receiving application.

Example

{bml j_bullet.bmp}DDE Example

See Also

{bml j_bullet.bmp}Dynamic Data Exchange

#478 S479 K480 +481 **DDERequest Function** {bmc no_dos.bmp}

Description

Gets a data item from a receiving application.

Syntax

DDERequest\$(channel%,dataItem\$)

Returns

Returns a string representing the value of the given data item in the receiving application associated with the open DDE channel.

Comments

The channel must first be initiated using DDEInitiate. An error will result if channel is invalid.

The formats for dataItem\$ and the returned value depend on the receiving application.

Example

{bml j_bullet.bmp}DDE Example

See Also

{bml j_bullet.bmp}Dynamic Data Exchange

478 CMD_DDERequest

479 DDERequest

480 DDERequest;Dynamic Data Exchange

481 Commands:0410

#482 S483 K484 +485 **DDETerminate Statement** {bmc no_dos.bmp}

Description

Closes the specified DDE channel.

Syntax

```
DDETerminate channel%
```

Comments

The `channel` must first be initiated using [DDEInitiate](#). An error will result if `channel` is invalid.

All open DDE channels are automatically terminated when the script ends.

Example

[{bml j_bullet.bmp}DDE Example](#)

See Also

[{bml j_bullet.bmp}Dynamic Data Exchange](#)

#486 S487 K488 +489 **DDETerminateAll Statement** {bmc no_dos.bmp}

Description

Closes all currently open DDE channels.

Syntax

```
DDETerminateAll
```

Comments

If you do not issue this statement, all open DDE channels are automatically terminated when the script ends.

Example

[{bml j_bullet.bmp}DDE Example](#)

```
482 CMD_DDETerminate
483 DDETerminate
484 DDETerminate;Dynamic Data Exchange
485 Commands:0415
486 CMD_DDETerminateAll
487 DDETerminateAll
488 DDETerminateAll;Dynamic Data Exchange
489 Commands:0420
```

See Also

[{bml j_bullet.bmp}Dynamic Data Exchange](#)

#490 S491 K492 +493 **DDETimeout Statement** {bmc no_dos.bmp}

Description

Sets the number of milliseconds that must elapse before a DDE command times out.

Syntax

```
DDETimeout milliseconds&
```

Comments

The default is 10000 (10 seconds).

The timeout should be set before issuing other DDE commands.

Example

[{bml j_bullet.bmp}DDE Example](#)

See Also

[{bml j_bullet.bmp}Dynamic Data Exchange](#)

#494 S495 K496 +497 **Declare Statement** {bmc no_dos.bmp}

Description

Declares a subroutine or function within another DLL.

Syntax 1

```
Declare sub name [lib libname$ [alias aliasname$]] [([argumentlist])]
```

Syntax 2

```
Declare function name [lib libname [alias aliasname$]] [([argumentlist])] [as type]
```

490 CMD_DDETimeout

491 DDETimeout

492 DDETimeout;Dynamic Data Exchange

493 Commands:0425

494 CMD_Declare

495 Declare

496 Declare;Subroutine, in other DLL;Function, in other DLL;Procedure statements

497 Commands:0430

Comments

This statement must precede any call to an external DLL routine.

The `name` parameter is any DCL valid global name. When declaring functions, a type-declaration character can be included to indicate the return type.

The `libname$` needs to be specified along with an optional `aliasname$`. The `libname$` parameter specifies the DLL that contains the external routine. All of the Windows API routines are contained in DLLs, such as "user", "kernel", "gdi", and so on. The file extension ".EXE" is implied if another extension is not given. If the `libname$` parameter is not the real name of the external routine as it appears within the external DLL, then an alias name must be given providing this name. For example, the following two statements declare the same routine.

```
declare function GetCurrentTime lib "user"()
    as integer

declare function GetTime lib "user" alias
    "GetCurrentTime" as integer
```

Use an alias when the name of an external routine conflicts with the name of an internal routine or if the external routine name contains invalid characters.

The optional `argumentlist` specifies the arguments received by the routine. The argument list must match exactly with that of the referenced routine. Otherwise unpredictable results may occur. By default, DCL passes arguments by reference. Many DLL routines require a value rather than a reference. The `byval` keyword does this. For example, the following C routine:

```
int MessageBeep(int);
```

would be declared as follows:

```
declare sub MessageBeep lib "user"
    (byval n as integer)
```

The following shows how a C routine that requires a pointer to an integer would be declared (notice the missing `byval` keyword on the third parameter):

```
int SystemParametersInfo(int,int,
    int far *,int);

declare function SystemParametersInfo
    lib "user"
    (byval action as integer,
    byval uParam as integer,
    pi as integer,
    byval updateINI as integer)
```

Strings are always passed to DLL routines by reference - the `byval` keyword in this case is unnecessary. If a DLL routine modifies a passed string variable, there must be sufficient space within the string to hold the returned characters. This can be accomplished using the `space$()` function:

```
declare sub GetWindowsDirectory lib "user"
    (dir$,len%)
    :
    :
    dim s as string
    s = space$(128)
    GetWindowsDirectory s,128
```

For function declarations, the return type can be specified using a type declaration character (i.e., \$, %, or &), or by specifying the `[as type]` clause. The valid types are `integer`, `long`, and `string`.

The libraries containing the routines are loaded when the routine is called for the first time (i.e., not when the script is loaded). This allows a script to references external DLLs that potentially do not exist.

The `declare` statement must appear before the function is used.

The `declare` statements are only valid during the life of that script.

Example

[{bml j_bullet.bmp}Declare Example](#)

See Also

[{bml j_bullet.bmp}Procedure Statements](#)

#498 \$499 K500 +501 **DEftype Statement**

Description

This statement controls automatic type declaration of variables.

Syntax

`DEFInt letterrange`

`DEFLng letterrange`

`DEFStr letterrange`

`DEFsng letterrange`

`DEFdbl letterrange`

Comments

Normally, if a variable is encountered that hasn't yet been declared with the `dim` statement, or does not appear with an explicit type declaration character, the variable is declared implicitly as an integer (`DEFInt A-Z`). This can be changed using the `DEftype` statement to specify starting letter ranges for *type* other than integer. The *letterrange* parameter is used to specify starting letters. Thus, any variable that begins with a specified character will be declared using the specified *type*.

The syntax for *letterrange* is:

`letter [-letter] [,letter [-letter]]...`

`DEftype` variable types are superseded by an explicit type declaration -- using a type declaration character or using the `dim` statement.

This statement only affects compiling of scripts.

Example

[{bml j_bullet.bmp}DEftype Example](#)

See Also

498 CMD_DEftype

499 DEftype

500 DEftype;Variable types, automatic

501 Commands:0435

[{bml j_bullet.bmp}Variables and Constants](#)

#502 \$503 K504 +505 **DesktopCascade Statement** {bmc no_dos.bmp}

Description

Cascades all non-minimized top-level windows.

Syntax

DesktopCascade

Example

[{bml j_bullet.bmp}DesktopCascade Example](#)

See Also

[{bml j_bullet.bmp}Desktop Modifications](#)

#506 \$507 K508 +509 **DesktopSetColors Statement** {bmc no_dos.bmp}

Description

Changes the system colors to one of the predefined color sets in the CONTROL.INI file.

Syntax

DesktopSetColors ControlPanelItemName\$

Example

[{bml j_bullet.bmp}DesktopSetColors Example](#)

See Also

[{bml j_bullet.bmp}Desktop Modifications](#)

502 CMD_DesktopCascade
503 DesktopCascade
504 DesktopCascade;Cascading windows;Desktop modifications;Windows, cascading
505 Commands:0440
506 CMD_DesktopSetColors
507 DesktopSetColors
508 DesktopSetColors;Colors, setting;Desktop modifications
509 Commands:0445

#510 #511 K512 +513 **DesktopSetWallpaper Statement** {bmc no_dos.bmp}

Description

Changes the Windows wallpaper.

Syntax

```
DesktopSetWallPaper filename$,tile%
```

Comments

This statement changes the Windows wallpaper to the bitmap specified by `filename$`. The wallpaper will be tiled is `tile` is TRUE, otherwise the bitmap will be centered on the desktop.

To remove the wallpaper, set the `filename$` parameter to "":

```
DesktopSetWallPaper "",true
```

This statement makes permanent changes to the wallpaper by writing the new wallpaper information to the WIN.INI file.

Example

{bml j_bullet.bmp}DesktopSetWallpaper Example

See Also

{bml j_bullet.bmp}Desktop Modifications

#514 #515 K516 +517 **DesktopTile Statement** {bmc no_dos.bmp}

Description

Tiles all non-minimized top-level windows.

Syntax

```
DesktopTile
```

Example

{bml j_bullet.bmp}DesktopTile Example

510 CMD_DesktopSetWallpaper

511 DesktopSetWallpaper

512 DesktopSetWallpaper;Wallpaper, setting;Desktop modifications

513 Commands:0450

514 CMD_DesktopTile

515 DesktopTile

516 DesktopTile;Tiling windows;Desktop modifications;Windows, tiling

517 Commands:0455

See Also

[{bml j_bullet.bmp}Desktop Modifications](#)

#518 \$519 K520 +521 **Dialog Statement and Function**

Description

The `Dialog` statement displays a dialog box. The `Dialog` function displays a dialog box and returns an integer representing the button that was pushed.

Statement Syntax

```
Dialog UserDlg as dialog
```

Comments

Displays the dialog box specified by the dialog template `UserDlg`.

If the user selects Cancel, a runtime error is generated. This error can be trapped using `ON ERROR`.

The `Dialog` statement returns only after the user presses OK, Cancel, or another push button.

Function Syntax

```
Dialog(UserDlg as dialog)
```

Returns

-1 OK button was pushed.

0 Cancel button was pushed.

>0 A push button was selected. The returned number represents which button was pushed based on its order in the dialog template. 1 represents the first button, 2 represents the second, etc.

Comments

This function displays the dialog box associated with the template `UserDlg`, and returns which button was pushed. Unlike the statement form, this function will not generate a runtime error when Cancel is selected.

Examples

[{bml j_bullet.bmp}Dialog Statement and Function Example](#)

[{bml j_bullet.bmp}A Comprehensive Dialog Example](#)

See Also

518 `CMD_Dialog`

519 `Dialog`

520 `Dialog;Dialog box, displaying;Dialog creation`

521 `Commands:0460`

Description

Declares a list of variables and their corresponding types and sizes.

A special form of the DIM statement is used to store a dialog definition in memory.

Syntax

```
dim name [(<subscripts>)] [as type] [,name [(<subscripts>)] [as type]]...
```

Comments

If a type declaration character is used (such as %, &, or \$), the optional [as type] expression is not allowed.

Up to 60 array dimensions are allowed.

The total size of an array (not counting space for strings) is limited to 42K.

Dynamic arrays are declared by not specifying any bounds:

```
dim a()
```

Any DIM statements declared within a subroutine or function are local to that subroutine or function.

If a variable is seen that has not been explicitly declared with DIM, it is implicitly declared using the type specifier character (% , \$, or &). If this character is missing, then integer is assumed.

Example

[{bml j_bullet.bmp}Dim Example](#)

Use with Dialogs

A special form of the DIM statement stores the preceding dialog definition ([BeginDialog...EndDialog](#)) in memory.

The syntax of this statement is:

```
dim dialog_name as UserDialog
```

Where `dialog_name` is a variable you define and `UserDialog` is a DCL reserved word.

For an example of this command, see [Dialog Examples](#).

See Also

[{bml j_bullet.bmp}Arrays](#)

[{bml j_bullet.bmp}Dialog Creation](#)

522 CMD_Dim

523 Dim

524 Dim;Array, declaring;Dialog definition, storing;Arrays;Variables, declaring

525 Commands:0465

[{bml j_bullet.bmp}ReDim](#)

[{bml j_bullet.bmp}Variables and Constants](#)

#526 #527 K528 +529 **DirExists Function** {bmc no_dos.bmp}

Description

Determines whether the specified `path$` exists and is a directory.

Syntax

```
DirExists (path$)
```

Comments

The function returns the integer TRUE or FALSE.

The path can be a partial path, such as "windows".

Example

[{bml j_bullet.bmp}DirExists Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#530 #531 K532 +533 **Dir\$ Function**

Description

Searches a disk directory.

Syntax

```
dir$ ([filespec$])
```

Returns

```
526 CMD_DirExists
527 DirExists
528 DirExists;Directory, determining if exists;File input and output
529 Commands: 0467
530 CMD_Dir_DOLLAR
531 Dir$
532 Dir$;Directory, searching;File input and output
533 Commands:0470
```

If `filespec$` is specified, then this function returns the first file matching that `filespec`. If `filespec$` is not specified, then this function returns the next file matching the initial `filespec`.

Comments

The `filespec$` argument can include wildcards, such as `*` and `?`.

An error is generated if `dir$` is called without first calling it with a valid `filespec`.

If there is no matching `filespec`, then an empty string is returned.

If no path is specified on `filespec$`, then the current directory is used.

This function does not find hidden files and directories.

Example

[{bml j_bullet.bmp}Dir\\$ Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#534 \$535 K536 +537 **DiskDrives Statement**

Description

This statement grabs all of the valid drive letters and packs them into the specified array.

Syntax

```
DiskDrives list$()
```

Comments

The array is resized to hold the exact number of valid drives.

The `list$` parameter must be a single dimension array of strings.

Use the functions [lbound\(\)](#) and [ubound\(\)](#) to determine the size of the resultant array.

Example

[{bml j_bullet.bmp}DiskDrives Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

⁵³⁴ CMD_DiskDrives

⁵³⁵ DiskDrives

⁵³⁶ DiskDrives;Drives, list;File input and output

⁵³⁷ Commands:0475

#538 #539 K540 +541 **DiskFree Function**

Description

Returns a long integer representing the free space (in bytes) available on the specified drive.

Syntax

```
DiskFree([drive$])
```

Comments

If `drive$` is empty or not specified, then the current drive is assumed.

Only the first character of the `drive$` string is used.

Example

[{bml j_bullet.bmp}DiskFree Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#542 #543 K544 +545 **Do...Loop Statement**

Description

This statement repeats a block of DCL statements while a condition is TRUE or until a condition is TRUE.

Syntax 1

```
do {while | until} condition
    statements
loop
```

```
538 CMD_DiskFree
539 DiskFree
540 DiskFree;Free space;File input and output
541 Commands:0480
542 CMD_Do_Loop
543 Do...Loop
544 Do...Loop;Do;Loop;Flow control
545 Commands:0485
```

Comment

Using `do while` causes the statements in the loop to be executed while the specified condition is true.

Using `do until` causes the statements in the loop to be executed until the specified condition is true.

The *condition* parameter specifies any Boolean expression.

Syntax 2

```
do
    statements
loop {while | until} condition
```

Comment

Syntax 2 has the same effect as Syntax 1.

Syntax 3

```
do
    statements
loop
```

Comment

If the `{while | until}` conditional clause is not specified, then the loop repeats the statements forever (or until an `exit do` statement is encountered).

Example

[{bml j_bullet.bmp}Do...Loop Example](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

#546 #547 K548 +549 **DoEvents Statement** **{bmc no_dos.bmp}**

Description

The `DoEvents` statement and function yield control to other applications.

Statement Syntax

```
DoEvents
```

Function Syntax

```
DoEvents [ () ]
```

Returns

```
546 CMD_DoEvents
547 DoEvents
548 DoEvents;Yielding to other applications;Flow control
549 Commands:0490
```

The function returns the value 0.

Comments

When running in exclusive mode, the only way other applications can multitask is for the script to call this statement/function.

Example

{bml j_bullet.bmp}DoEvents Example

See Also

{bml j_bullet.bmp}Flow Control

#550 \$551 K552 +553 **DoKeys Statement** {bmc no_dos.bmp}

Description

Uses the Windows journaling mechanism to play keystrokes into the Windows environment.

Syntax

DoKeys KeyString\$

Comments

The format for KeyString\$ is the same as that used for QueKeys.

This statement will not affect the current event queue.

Example

{bml j_bullet.bmp}DoKeys Example

See Also

{bml j_bullet.bmp}Keyboard Manipulation

#554 \$555 K556 +557 **EditEnabled Function** {bmc no_dos.bmp}

550 CMD_DoKeys
551 DoKeys
552 DoKeys;Keystrokes, playing;Keyboard manipulation
553 Commands:0495
554 CMD_EditEnabled
555 EditEnabled
556 EditEnabled;Edit box, determining if enabled;Dialog manipulation
557 Commands:0515

Description

Determines if an edit box is enabled within the current window or dialog box.

Syntax 1

```
EditEnabled(name$)
```

Syntax 2

```
EditEnabled(id%)
```

Returns

Returns the integer TRUE if the given edit box is enabled within the active window or dialog box, FALSE otherwise.

Comments

If enabled, the edit box can be given the focus using the [ActivateControl](#) statement.

The `name$` parameter specifies the text that appears within the static control that immediately precedes the edit control in the window list (or dialog template). Alternatively, the `id%` of the edit box can be specified.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#558 \$559 K560 +561 **EditExists Function** {bmc no_dos.bmp}

Description

Determines if an edit box exists within the current window or dialog box.

Syntax 1

```
EditExists(name$)
```

Syntax 2

```
EditExists(id%)
```

Returns

Returns the integer TRUE if the given edit box exists within the active window or dialog box, FALSE otherwise.

Comments

If there is no active window, FALSE will be returned.

558 CMD_EditExists
559 EditExists
560 EditExists;Edit box, determining if exists;Dialog manipulation
561 Commands:0520

The `name$` parameter specifies the text that appears within the static control that immediately precedes the edit control in the window list (or dialog template). Alternatively, the `id%` of the edit box can be specified.

Example

[`{bml j_bullet.bmp}`Dialog Examples](#)

See Also

[`{bml j_bullet.bmp}`Dialog Manipulation](#)

#562 \$563 K564 +565 **EnableStopScript Statement** `{bmc no_dos.bmp}`

Description

Controls whether a script may be halted by the user.

Syntax

`EnableStopScript condition%`

Comments

The `condition%` parameter can be one of the following constants:

- ESS_ENABLE (enable script breaking). This is the default.
- ESS_DISABLE (disable script breaking).
-
-
- ESS_ENABLE_INTERACTIVE (allows the script to be broken only when executing in interactive [debug] mode).

Example

[`{bml j_bullet.bmp}`EnableStopScript Example](#)

See Also

[`{bml j_bullet.bmp}`Flow Control](#)

#566 \$567 K568 +569 **End Statement**

```
562 CMD_EnableStopScript
563 EnableStopScript
564 EnableStopScript;Script, allowing to be stopped;Flow control;Stopping script execution
565 Commands: 0522
566 CMD_End
567 End
568 End;Script, ending;Flow control
569 Commands:0523
```

Description

Terminates execution of the current script.

Syntax

end

Comments

This statement can appear multiple times in a script, so that you conditionally end the script. It can also appear in functions and subroutines.

All open files are closed. All open DDE channels are closed.

Example

{bml j_bullet.bmp}End Example

See Also

{bml j_bullet.bmp}Flow Control

#570 \$571 K572 +573 **ENV_BOTH**

Description

Constant meaning both DOS and Widows; used by the SetEnv command.

#574 \$575 K576 +577 **ENV_DOS**

Description

Constant meaning DOS environment; used by the GetEnv, SetEnv, RestoreEnv, and SaveEnv commands.

570 CMD_ENV_BOTH
571 ENV_BOTH
572 ENV_BOTH
573 Commands:0524
574 CMD_ENV_DOS
575 ENV_DOS
576 ENV_DOS
577 Commands:0525

#578 \$579 K580 +581 **ENV_WINDOWS**

Description

Constant meaning Windows environment; used by the GetEnv, SetEnv, RestoreEnv, and SaveEnv commands.

578 CMD_ENV_WINDOWS
579 ENV_WINDOWS
580 ENV_WINDOWS
581 Commands:0526

#582 \$583 K584 +585

Environ\$ Function {bmc no_win.bmp}

Description

Returns the value of the specified environment variable.

Note: For greater convenience, we recommend using the [GetEnv](#) function instead.

Syntax 1

```
environ$(variable$)
```

Syntax 2

```
environ$(VariableNumber%)
```

Comments

If `variable$` is specified, then this function looks for that variable in the environment. If the variable name cannot be found, then an empty string is returned.

If `VariableNumber` is specified, then this function looks for the Nth variable within the environment (the first variable being number 1). If there is no such environment variables, an empty string is returned. Otherwise, the entire entry from the environment is returned in the format:

```
variable=value
```

Example

[{bml j_bullet.bmp}Environ\\$ Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#586 \$587 K588 +589

EOF Function

Description

```
582 CMD_Environ_DOLLAR
583 Environ$
584 Environ$;Variables, environment;Environment commands
585 Commands:0530
586 CMD_EOF
587 EOF
588 EOF;End of file;File, end of;File input and output
589 Commands:0535
```

Determines whether end of file has been reached.

Syntax

```
eof(filename%)
```

Returns

Returns the integer TRUE if the end of file has been reached for the given file, otherwise FALSE.

Comments

The `filename` parameter is a number that is used by DCL to refer to the open file -- the number passed to the [open](#) statement.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#590 \$591 K592 +593 **Erl Function**

Description

Not used.

Syntax

```
erl[() ]
```

Returns

Returns the integer 0 (DCL does not support line numbers).

See Also

[{bml j_bullet.bmp}Error Trapping](#)

#594 \$595 K596 +597 **Err Statement and Function**

590 CMD_Erl
591 Erl
592 Erl;Error trapping
593 Commands:0540
594 CMD_Err
595 Err
596 Err;Error trapping
597 Commands:0545

Description

The `err` function returns an integer representing the runtime error that caused the current error trap. The `err` statement sets the value returned by the `err` function to a specific value.

Statement Syntax

```
err = value%
```

Function Syntax

```
err[()]
```

Comments

The `err` function can only be used while within an error trap.

When a function or statement ends, the value returned by `err` is reset to 0.

Example

[{bml j_bullet.bmp}Err Example](#)

See Also

[{bml j_bullet.bmp}Error Trapping](#)

#598 \$599 K600 +601 **Error Statement**

Description

Simulates the occurrence of the specified runtime error.

Syntax

```
error errornumber%
```

Comments

The `errornumber` parameter can be a built-in error number, or a user defined error number. The `err` function can be used within the error trap handler to determine the value of the error.

Example

[{bml j_bullet.bmp}Error Statement and Function Example](#)

See Also

[{bml j_bullet.bmp}Error Trapping](#)

598 CMD_Error
599 Error
600 Error;Error trapping
601 Commands:0550

#602 S603 K604 +605 **Error\$ Function**

Description

Returns the text corresponding to the given error number or the most recent error.

Syntax

```
error$ [(errornumber%)]
```

Comments

If `errornumber` is omitted, then the function returns the text corresponding to the most recent runtime error. If no runtime error has occurred, then an empty string is returned ("").

If the `error` statement was used to generate a user-defined runtime error, this function will return an empty string ("").

Example

[{bml j_bullet.bmp}Error Statement and Function Example](#)

See Also

[{bml j_bullet.bmp}Error Trapping](#)

#606 S607 K608 +609 **Exclusive Statement** **{bmc no_dos.bmp}**

Description

Sets or unsets exclusive mode.

Syntax

```
exclusive NewState%
```

Comments

When set (`NewState` is TRUE), then the script will not yield control to other applications - no other applications will execute concurrently. When a script is not running in exclusive mode (`NewState` is FALSE), then other applications can multitask while the script is running.

```
602 CMD_Error_DOLLAR
603 Error$
604 Error$;Error trapping
605 Commands:0555
606 CMD_Exclusive
607 Exclusive
608 Exclusive;Exclusive mode;Flow control
609 Commands:0560
```

By default, scripts do not run in exclusive mode.

If a script is running in exclusive mode and there is an infinite loop, you will be unable to abort the script (the computer will hang). Thus, caution must be taken to make sure that a script has no errors or infinite loops.

Scripts running exclusively run faster than scripts not in exclusive mode.

If a dialog box is encountered (MsgBox, AskBox\$, AnswerBox, ...) while a script is running in `exclusive` mode, then that dialog box is system modal, meaning that a user can only interact with that dialog and not any other applications that may be running concurrently. All other dialog boxes, applications, and buttons will be "locked out".

Example

{bml j_bullet.bmp}Exclusive Example

See Also

{bml j_bullet.bmp}Flow Control

#610 \$611 K612 +613 **Exit Do Statement**

Description

Exits a `do ...loop`.

Syntax

```
exit do
```

Comments

This statement can only appear within a `do ...loop` statement. It causes execution to continue with the next statement after the `loop` clause.

See Also

{bml j_bullet.bmp}Flow Control

#614 \$615 K616 +617 **Exit For Statement**

⁶¹⁰ `CMD_Exit_Do`

⁶¹¹ `Exit Do`

⁶¹² `Exit Do;Exit;Do;Flow control`

⁶¹³ `Commands:0565`

⁶¹⁴ `CMD_Exit_For`

⁶¹⁵ `Exit For`

⁶¹⁶ `Exit For;Exit;For;Flow control`

⁶¹⁷ `Commands:0570`

Description

Exits a `for...next` loop.

Syntax

```
exit for
```

Comments

This statement ends the `for...next` block in which it appears. Execution will continue on the line immediately after the `NEXT` statement.

This statement can only appear within a `for...next` block.

Example

[{bml j_bullet.bmp}Exit Statement Examples](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

#618 #619 #620 #621 **Exit Function Statement**

Description

Exits the current function.

Syntax

```
exit function
```

Comments

This statement ends execution of the function in which it appears. Execution will continue on the statement or function following the call to this function.

This statement can only appear within a function.

Example

[{bml j_bullet.bmp}Exit Statement Examples](#)

See Also

[{bml j_bullet.bmp}Procedure Statements](#)

618 CMD_Exit_Function

619 Exit Function

620 Exit Function;Exit;Function;Flow control

621 Commands:0575

#622 S623 K624 +625 **Exit Sub Statement**

Description

Exits the current subroutine.

Syntax

```
exit sub
```

Comments

This statement ends the current subroutine. Execution is transferred to the statement following the call to the current subroutine.

This statement can appear anywhere within a subroutine. It cannot appear within a function.

Example

[{bml j_bullet.bmp}Exit Statement Examples](#)

See Also

[{bml j_bullet.bmp}Procedure Statements](#)

#626 S627 K628 +629 **Exp Function**

Description

Returns a double-precision number representing the value of *e* raised to the power of *x*.

Syntax

```
exp (x#)
```

Comments

Range of *x*: 0 <= *x* <= 709.782712893

A runtime error is generated if *x* is out of the above specified range.

Example

```
622 CMD_Exit_Sub
623 Exit Sub
624 Exit Sub;Exit;Sub;Flow control
625 Commands:0580
626 CMD_Exp
627 Exp
628 Exp;Exponentiation;Math commands
629 Commands:0585
```

{bml j_bullet.bmp}Exp Example

See Also

{bml j_bullet.bmp}Math Statements and Functions

#630 \$631 K632 +633 **FALSE**

Description

Constant.

Returns

0

Comments

Used in conditionals and Boolean expressions

#634 \$635 K636 +637 **FileAttr Function**

Description

Returns an integer representing the file mode or file handle.

Syntax

`FileAttr (filenumber%, attribute%)`

Returns

Returns the file mode (if `attribute` is 1), or the operating system file handle (if `attribute` is 2).

Comments

If `attribute` is 1, then one of the following values is returned:

- 1 input
- 2 output
- 8 append

630 CMD_FALSE

631 FALSE

632 FALSE

633 Commands:0590

634 CMD_FileAttr

635 FileAttr

636 FileAttr;Attributes, file;File attributes;File input and output

637 Commands:0595

The `filenumber` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#638 S639 K640 +641 **FileCopy Function** {bmc no_dos.bmp}

Description

Copies the specified file(s) to the given destination.

Syntax

`FileCopy(src$, dest$)`

Comments

The function returns the integer TRUE if successful; otherwise it returns FALSE.

Wildcards are permitted in the `src$` string. They are the same as the DOS wildcards.

If the `src$` parameter specifies wildcards, all files matching the `src$` parameter are copied to `dest$` with the proper modification made to the destination file name so that it matches the source file according to the wildcards.

Example

[{bml j_bullet.bmp}FileCopy Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#642 S643 K644 +645 **FileDateTime Function**

Description

638 CMD_FileCopy
639 FileCopy
640 FileCopy;File, copying;File input and output
641 Commands: 0597
642 CMD_FileDateTime
643 FileDateTime
644 FileDateTime;File, date and time;File input and output
645 Commands:0600

Returns a double-precision number representing the date and time of the given file. The number is returned in days where Dec 20, 1899 is 0.

Syntax

```
FileDateTime(filename$)
```

Comments

This function retrieves the date and time of the file specified by `filename$`. A runtime error results if the file does not exist. The value returned can be used with the date/time functions (i.e., `year()`, `month()`, `day()`, `weekday()`, `minute()`, `second()`, `hour()`) to extract the individual elements.

Example

[{bml j_bullet.bmp}FileDateTime Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#646 #647 #648 #649 **FileDirs Statement**

Description

Fills an array with directory names from disk.

Syntax

```
FileDirs array$() [,dirspec$]
```

Comments

The `array$()` is any previously declared string array. The `FileDirs` function reallocates this array to exactly hold all of the directory names matching a given specification.

The `dirspec$` parameter specifies the file search mask, such as:

```
T*.C:\*.*
```

If the `dirspec$` parameter is not specified, then `*.*` is used, which fills the array with all the sub-directory names within the current directory.

Example

[{bml j_bullet.bmp}FileDirs Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

[{bml j_bullet.bmp}File Input and Output](#)

646 CMD_FileDirs
647 FileDirs
648 FileDirs;Directories, list;File input and output;Arrays
649 Commands:0605

#650 \$651 K652 +653 **FileExists Function**

Description

Determines if a given filename is valid.

Syntax

```
FileExists(filename$)
```

Returns

Returns the integer TRUE if the `filename$` is a valid file, FALSE otherwise.

Example

[{bml j_bullet.bmp}FileExists Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#654 \$655 K656 +657 **FileLen Function**

Description

Returns a long integer representing the length of the specified file in bytes.

Syntax

```
FileLen(filename$)
```

Comments

This function is used to retrieve the length of a file without first opening the file. A runtime error results if the file does not exist.

Example

[{bml j_bullet.bmp}FileLen Example](#)

See Also

```
650 CMD_FileExists
651 FileExists
652 FileExists;File, determines if exists;File input and output
653 Commands:0610
654 CMD_FileLen
655 FileLen
656 FileLen;File, length of;File input and output
657 Commands:0615
```

[{bml j_bullet.bmp}File Input and Output](#)

[{bml j_bullet.bmp}LOF](#)

#658 \$659 K660 +661 **FileList Statement**

Description

Fills an array with filenames from disk.

Syntax

```
FileList array$() [,filespec$ [,fileattr%]]
```

Comments

The `array$()` is any previously declared string array. The `files` function reallocates this array to exactly hold all of the files matching a given `filespec`.

The `filespec$` parameter specifies the file search mask, such as:

```
*.EXE      *.DOC      t*.DO?
```

If the `filespec$` parameter is not specified, `*.*` is used.

The `fileattr` parameter is a number indicating what types of files you want included in the list. It can be any combination of the following:

Constant	Value	Description
ATTR_NORMAL	0	Read-only, archive, subdirectory, and files with no attributes
ATTR_READONLY	1	Read-only files
ATTR_HIDDEN	2	Hidden files
ATTR_SYSTEM	4	System files
ATTR_VOLUME	8	Volume label
ATTR_DIRECTORY	16	MS-DOS directories
ATTR_ARCHIVE	32	Files changed since last backup
ATTR_NONE	64	Files with no attributes

If the `fileattr` parameter is not specified, then the value 97 is used (ATTR_READONLY or ATTR_ARCHIVE or ATTR_NONE or ATTR_DIRECTORY). This value retrieves the same set of files normally returned by the DOS `dir` command.

Example

[{bml j_bullet.bmp}FileList Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

[{bml j_bullet.bmp}File Input and Output](#)

658 CMD_FileList

659 FileList

660 FileList;File names, list;File input and output;Arrays;Directories, list

661 Commands:0620

#662 S663 K664 +665 **FileParse Statement**

Description

Takes a given filename and extracts a given portion of the filename from it.

Syntax

```
FileParse$(filename$[,operation])
```

Comments

The `filename$` parameter can specify an valid DOS filename (does not have to exist). For example:

```
..\TEST.DAT
C:\SHEETS\TEST.DAT
TEST.DAT
```

The optional `operation` parameter specifies which portion of the `filename$` to extract. It can be any of the following values.

0	full name	C:\SHEETS\TEST.DAT
1	drive	C
2	path	C:\SHEETS
3	name	TEST.DAT
4	root	TEST
5	extension	DAT

If `operation` is not specified, then the full name is returned. A runtime error will result if `operation` is not one of the above values.

Example

[{bml j_bullet.bmp}FileParse Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#666 S667 K668 +669 **FileType Function**

```
662 CMD_FileParse
663 FileParse
664 FileParse;File name, parsing;Parsing file name;File input and output
665 Commands:0625
666 CMD_FileType
667 FileType
668 FileType;File, determine type;File input and output
669 Commands:0630
```


Description

Returns an integer representing the file type.

Syntax

```
FileType(filename$)
```

Returns

Returns one of the following file type constants:

TYPE_DOS	a DOS executable file
TYPE_WINDOWS	a Windows executable file

Comments

This function is used to determine whether a file is a Windows executable or DOS executable. If one of the above values is not returned, then the file type is unknown.

Example

[{bml j_bullet.bmp}FileType Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#670 #671 #672 +673 **FindFile\$ Function** **{bmc no_dos.bmp}**

Description

Searches for `file$` and, if found, returns a full path to it. If the file is not found, the return value is a null-string.

Syntax

```
fullpath$=FindFile$(file$)
```

Comments

The search follows normal Windows search order: current directory, Windows directory, system directory, and then the PATH environment variable.

Example

[{bml j_bullet.bmp}FindFile\\$ Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

⁶⁷⁰ CMD_FindFile_DOLLAR

⁶⁷¹ FindFile\$

⁶⁷² FindFile\$;File, searching for;File input and output

⁶⁷³ Commands: 0632

#674 \$675 K676 +677 **Fix Function**

Description

Returns the integer part of `number`.

Syntax

```
fix(number#)
```

Comments

This function returns the integer part of the given value by removing the fractional part. The sign is preserved. No rounding occurs. For example:

```
fix(4.5)  'returns 4
fix(-4.5) 'returns -4
```

Example

[{bml j_bullet.bmp}Fix Example](#)

See Also

[{bml j_bullet.bmp}CInt](#)

[{bml j_bullet.bmp}Int](#)

[{bml j_bullet.bmp}Math Statements and Functions](#)

#678 \$679 K680 +681 **For...Next Statement**

674 CMD_Fix

675 Fix

676 Fix;Integer, extracting;Math commands

677 Commands:0635

678 CMD_For_Next

679 For...Next

680 For...Next;For;Next;Flow control

681 Commands:0640

Description

Repeats a block of statement a specified number of times, incrementing a loop counter by a given increment each time through the loop.

Syntax

```
for counter = start to end [step increment]
    ...
next [counter]
```

Comments

If `increment` is not specified, then 1 is assumed.

The first time through the loop, `counter` is equal to `start`. Each time through the loop, `increment` is added to `counter` by the amount specified in `increment`.

The `for...next` statement continues executing until:

- An `exit for` statement is encountered

OR

-
-
-
- When `counter` is greater than `end`.

If `end > start` then `increment` must be positive. If `end < start`, then `increment` must be negative.

The `for...next` statements can be nested. In such a case, the `next [counter]` statement applies to the innermost `for...next`.

The `next [counter]` can be optimized for next loops by separating each counter with a comma. The ordering or the counters must be consistent with the nesting order (innermost counter appearing before outermost counter):

```
next i,j
```

Example

[{bml j_bullet.bmp}For...Next Example](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

#682 S683 K684 +685 FreeFile Function

682 CMD_FreeFile

683 FreeFile

684 FreeFile;File, next free;File input and output

685 Commands:0645

Description

Returns the next available file number.

Syntax

```
FreeFile[()]
```

Comment

The returned integer is suitable for use in the open statement.

Example

{bml j_bullet.bmp}FreeFile Example

See Also

{bml j_bullet.bmp}File Input and Output

#686 #687 #688 #689 **Function...End Function Statement**

Description

Creates a user-defined function.

Syntax

```
function name[(parameter [as <type>]...)] [as <type>]
    ...
    ...
    name = <expression>
end function
```

Comments

The return value is determined by the statement:

```
name = <expression>
```

The name of the function following DCL naming conventions. It can include type declaration characters: %, &, and \$.

If no assignment is encountered before the function exits, then 0 will be returned for numeric functions, and an empty string will be returned for string functions.

The type of the return value is determined by the `as <type>` clause on the function statement itself. As an alternative, a type declaration character can be added to the function name:

```
function Test() as string
    Test = "Hello World"
```

686 CMD_Function_End_Function

687 Function...End Function

688 Function...End Function;Function;End;Procedure statements

689 Commands:0650

```

end function

function Test$()
    Test = "Hello World"
end function

```

Parameters are passed to a function by reference, meaning that any modifications to a passed parameter changes that variable in the caller. To avoid this, simply enclose variable names in parenthesis, as in the following example function calls:

```
i = UserFunction(10,12,(j))
```

If a function is not to receive a parameter by reference, then the optional `byval` keyword can be used:

```

function Test(byval FileName as string)
    as string
end function

```

A function returns to the caller when either of the following statements is encountered:

```

end function
exit function

```

The function definition must precede the statements that call the function.

The function cannot be inside a subroutine (including `sub (main)`) or inside another function.

Functions can be recursive.

See Also

[{bml;j_bullet.bmp}Procedure Statements](#)

#690 \$691 K692 +693 **GetAttr Function**

Description

Returns an integer representing the attributes of the specified file.

Syntax

```
GetAttr(filename$)
```

Returns

The attribute value returned contains the sum of the following attributes.

Constant	Value	Description
ATTR_NORMAL	0	Read-only, archive, subdirectory, and files with no attributes
ATTR_READONLY	1	Read-only files
ATTR_HIDDEN	2	Hidden files
ATTR_SYSTEM	4	System files
ATTR_VOLUME	8	Volume label

690 CMD_GetAttr

691 GetAttr

692 GetAttr;Attributes, file;File attributes;File input and output

693 Commands:0655

ATTR_DIRECTORY	16	MS-DOS directories
ATTR_ARCHIVE	32	Files changed since last backup
ATTR_NONE	64	Files with no attributes

These attributes are the same as those used by DOS.

Example

[{bml j_bullet.bmp}GetAttr Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#694 S695 K696 +697 **GetCheckbox Function** {bmc no_dos.bmp}

Description

Returns an integer representing the state of the specified check box.

Syntax 1

GetCheckbox (name\$)

Syntax 2

GetCheckbox (id%)

Comments

This function is used to determine the state of a check box, given its `name$` (the text of its label) or `id%`. The return value will be one of the following:

- 0 check box has no check
- 1 check box contains a check
- 2 check box is grayed

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#698 \$699 K700 +701

GetComboboxItem\$ Function {bmc no_dos.bmp}

Description

Returns the text corresponding to an item number in a combobox.

Syntax 1

```
GetComboboxItem$ (name$, ItemNumber%)
```

Syntax 2

```
GetComboboxItem$ (id%, ItemNumber$)
```

Comments

The combobox must exist within the current window or dialog box, otherwise a runtime error is generated.

You can use the `name$` or `id%` parameter to specify the combobox. The `name$` parameter specifies the text that appears in the static control that immediately precedes the combobox control in the window list (or dialog template).

The `ItemNumber%` parameter is the line number of the desired combobox item.

An empty string will be returned if the combobox does not contain textual items.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#702 \$703 K704 +705

GetComboboxItemCount Function {bmc no_dos.bmp}

Description

Returns an integer representing the number of items in the specified combobox.

698 CMD_GetComboboxItem_DOLLAR

699 GetComboboxItem\$

700 GetComboboxItem\$; Combo box, getting text of item; Dialog manipulation

701 Commands:0665

702 CMD_GetComboboxItemCount

703 GetComboboxItemCount

704 GetComboboxItemCount; Combo box, getting item count; Dialog manipulation

705 Commands:0670

Syntax 1

GetComboBoxItemCount (name\$)

Syntax 2

GetComboBoxItemCount (id%)

Comments

You can use the `name$` or `id%` parameter to specify the combobox. `Name$` specifies the text that appears in the static control that immediately precedes the combobox control in the window list (or dialog template).

A runtime error is generated if the specified combobox does not exist within the current window or dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#706 #707 #708 +709

GetEditText\$ Function **{bmc no_dos.bmp}**

Description

Returns the textual content of the specified edit box.

Syntax 1

GetEditText\$ (name\$)

Syntax 2

GetEditText\$ (id%)

Comments

The name of an edit control is determined by scanning the window list (or dialog template) for a static control labeled `name$` that is immediately followed by an edit control. A runtime error is generated if an edit control with that name cannot be found within the active window.

For edit controls that do not have a preceding static control, the `id%` can be used to absolutely reference the control.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

706 CMD_GetEditText_DOLLAR

707 GetEditText\$

708 GetEditText\$;Control, getting text of;Dialog manipulation

709 Commands:0675

GetEnv Function {bmc no_dos.bmp}

Description

Returns the value of the given environment variable for DOS or WINDOWS.

Syntax

```
GetEnv$(var$[, mode$])
```

Comments

The `mode$` parameter can be `ENV_DOS` or `ENV_WINDOWS`.

If `mode$` is `ENV_DOS`, the variable is returned from the DOS environment, otherwise it is returned from the Windows environment.

If mode is unspecified, the default is `ENV_WINDOWS`.

Example

{bml j_bullet.bmp}GetEnv Example

See Also

{bml j_bullet.bmp}Environment Statements and Functions

⁷¹⁰ CMD_GetEnv

⁷¹¹ GetEnv

⁷¹² GetEnv;Environment variables;DOS environment;Windows environment;Variables, environment;Environment commands

⁷¹³ Commands: 0677

#714 S715 K716 +717

GetListboxItem\$ Function {bmc no_dos.bmp}

Description

Returns the string corresponding to a list box item.

Syntax 1

```
GetListboxItem$(name$,item%)
```

Syntax 2

```
GetListboxItem$(id%,item%)
```

Comments

This function retrieves the text of a given item in a listbox. The `item%` parameter is the item's position in the list, where 1 is the first item. The `item%` parameter must be between 1 and the number of items in the listbox.

The listbox can be specified using either its `id%` or the `name$` (label) of the static control that immediately precedes the listbox control in the window list (or dialog template).

A runtime error is generated if the specified listbox cannot be found within the active window.

Example

{bml j_bullet.bmp}Dialog Examples

See Also

{bml j_bullet.bmp}Dialog Manipulation

#718 S719 K720 +721

GetListboxItemCount Function {bmc no_dos.bmp}

Description

Returns an integer representing the number of items in the specified listbox.

714 CMD_GetListboxItem_DOLLAR

715 GetListboxItem\$

716 GetListboxItem\$;List box, getting text of item;Dialog manipulation

717 Commands:0680

718 CMD_GetListboxItemCount

719 GetListboxItemCount

720 GetListboxItemCount;List box, getting item count;Dialog manipulation

721 Commands:0685

Syntax 1

`GetListboxItemCount (name$)`

Syntax 2

`GetListboxItemCount (id%)`

Comments

The listbox can be specified using either its id% or the `name$` (label) of the static control that immediately precedes the listbox control in the window list (or dialog template).

A runtime error is generated if the specified listbox cannot be found within the active window.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#722 S723 K724 +725 **GetOption Function** **{bmc no_dos.bmp}**

Description

Determines whether a given option button is checked.

Syntax 1

`GetOption (name$)`

Syntax 2

`GetOption (id%)`

Returns

Returns the integer TRUE if the option is set, FALSE otherwise.

Comments

The option button must exist within the current window or dialog box.

The option button can be referenced given its `name$` (label) or its id%. A runtime error will be generated if the given option button does not exist.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

722 CMD_GetOption

723 GetOption

724 GetOption;Option button, determine if selected;Dialog manipulation

725 Commands:0690

#726 S727 K728 +729 **GetUserName Function**

Description

Use the NetUserName function instead.

#730 S731 K732 +733 **GoSub Statement**

Description

Executes the specified subroutine.

Syntax

```
gosub label
```

Comments

This statement causes execution to continue at the specified label. Execution can later be returned to the statement following the `gosub` by using the return statement.

The `label` parameter must be a label within the current function or subroutine. The `gosub` parameter outside the context of the current function or subroutine is not allowed.

Note: It is a sounder programming practice to write a named subroutine using a Sub...End Sub block.

Example

{bml j_bullet.bmp}GoSub Example

See Also

{bml j_bullet.bmp}Flow Control

#734 S735 K736 +737 **Goto Statement**

726 CMD_GetUserName

727 GetUserName

728 GetUserName

729 Commands:0692

730 CMD_GoSub

731 GoSub

732 GoSub;Flow control

733 Commands:0695

734 CMD_Goto

735 Goto

736 Goto;Label;Flow control

737 Commands:0700

Description

Transfers execution to the line containing the specified label.

Syntax

```
goto <label>
```

Comments

The compiler will produce an error if `label` does not exist.

The `label` must appear within the same subroutine or function as the `goto`.

Labels must begin with a letter and end with a colon. Keywords cannot be used as labels. Labels are not case sensitive.

Example

[{bml j_bullet.bmp}Goto Example](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

#738 \$739 K740 +741 **GroupBox Statement**

Description

Defines a groupbox within a dialog box template.

Syntax

```
GroupBox x%,y%,width%,height%,title$
```

Comments

A groupbox is simply a visual element used to enclose other controls within a dialog box.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x,y,width,height` parameters are specified in dialog coordinates. The `x,y` position is relative to the upper left corner of the dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

738 CMD_GroupBox

739 GroupBox

740 GroupBox;Group box, defining;Dialog creation

741 Commands:0705

Description

Returns a string containing the hexadecimal equivalent of the specified number.

Syntax

```
hex$(number&)
```

Comments

The returned string contains only the number of hexadecimal digits necessary to represent the number, up to a maximum of 8.

The `number` parameter can be any type, but is rounded to the nearest whole number before converting to hexadecimal. If the passed number is an integer, then a maximum of 4 digits are returned; otherwise, up to 8 digits can be returned.

Example

{bml j_bullet.bmp}Hex\$ Example

See Also

{bml j_bullet.bmp}Conversions

Description

Scrolls the window with the focus left or right by the specified number of lines. This feature is useful when the contents are wider than the window.

Syntax

```
HLine [lines%]
```

Comments

If the `lines%` parameter is omitted, then the window is scrolled right by 1 line.

Example

{bml j_bullet.bmp}HLine Example

See Also

```
742 CMD_Hex_DOLLAR
743 Hex$
744 Hex$;Hexadecimal, converting to;Conversions
745 Commands:0710
746 CMD_HLine
747 HLine
748 HLine;Scrolling horizontally;Window manipulation
749 Commands:0715
```

Description

Returns an integer representing the hour of the day encoded in the specified `serial#` parameter. The value returned is between 0 and 23 inclusive.

Syntax

```
hour(serial#)
```

Comment

You can obtain the value for the `serial` parameter by using the [TimeSerial](#) or [TimeValue](#) command.

Example

[{bml j_bullet.bmp}Hour Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

Description

Scrolls the window with the focus left or right by the specified number of pages. This feature is useful when the contents are wider than the window.

Syntax

```
HPage [pages%]
```

Comments

If the `pages%` parameter is omitted, then the window is scrolled right by 1 page.

Example

[{bml j_bullet.bmp}HPage Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

750 CMD_Hour

751 Hour

752 Hour;Time functions

753 Commands:0720

754 CMD_HPage

755 HPage

756 HPage;Scrolling horizontally;Window manipulation

757 Commands:0725

#758 S759 K760 +761 **HScroll Statement** {bmc no_dos.bmp}

Description

Sets the thumb mark on the horizontal scroll bar attached to the current window.

Syntax

```
HScroll percentage%
```

Comments

The position is given as a percentage of the total range associated with that scroll bar. For example, if the `percentage%` parameter is 50, then the thumb is positioned in the middle of the scroll bar.

Example

{bml j_bullet.bmp}HScroll Example

See Also

{bml j_bullet.bmp}Window Manipulation

#762 S763 K764 +765 **If...Then...Else Statement**

Description

Conditionally executes a statement or group of statements.

Syntax 1

```
if <condition> then <statement>
[else <statement>]
```

Syntax 2

```
if <condition> then
    [<statement>]
```

```
758 CMD_HScroll
759 HScroll
760 HScroll;Scrolling horizontally;Window manipulation
761 Commands:0730
762 CMD_If_Then_Else
763 If...Then...Else
764 If...Then...Else;If;Then;Else;Flow control
765 Commands:0735
```



```

[elseif <condition> then
    [<statement>]]
[else
    [<statement>]]
end if

```

Comments

In the single line version, the <statement> must be a single statement. Optionally, many statements can be separated using the colon (:).

Example

{bml j_bullet.bmp}If...Then...Else Example

See Also

{bml j_bullet.bmp}Flow Control

#766 #767 #768 #769 Input # Statement

Description

Reads comma-delimited data from a file into variables.

Syntax

```
input [#]filename%,variable[,variable]...
```

Comments

This statement reads data from the file referenced by `filename%` into the given variables.

Each `variable` must be type matched to the data in the file. For example, a string variable must be matched to a string in the file.

All data items are separated by commas in the file. Leading spaces are ignored. Strings must be enclosed in quotes:

```
10,"Hello World",192,6
```

The `filename%` parameter is a number that is used by DCL to refer to the open file -- the number passed to the open statement.

The `filename%` must reference a file opened in `input` mode.

See Also

{bml j_bullet.bmp}File Input and Output

⁷⁶⁶ CMD_Input_POUND

⁷⁶⁷ Input #

⁷⁶⁸ Input #;Reading data;File, reading from;File input and output

⁷⁶⁹ Commands:0740

#770 \$771 K772 +773 **Input\$ Function**

Description

Returns a character string containing the first `numbytes` characters read from the given file.

Syntax

```
input$(numbytes%, [#]filename%)
```

Comments

The `input$` function reads all characters, including spaces and carriage returns.

The `filename%` must reference a file opened in `input` mode.

The `filename%` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

See Also

{bml j_bullet.bmp}File Input and Output

#774 \$775 K776 +777 **InputBox\$ Function**

Description

Presents a dialog box displaying a prompt and returns the user's response.

Syntax

```
InputBox$(prompt$ [,title$ [,default$ [,x%,y%]]])
```

Returns

Returns the text contained in the edit box when the user presses OK. If the user Cancels the dialog box, an empty string is returned.

Comments

A default response can be specified in the `default$` parameter.

770 CMD_Input_DOLLAR

771 Input\$

772 Input\$;Reading data;File, reading from;File input and output

773 Commands:0745

774 CMD_InputBox_DOLLAR

775 InputBox\$

776 InputBox\$;Box, input;Input, string;Dialog display

777 Commands:0750

The `prompt$` parameter can contain multiple lines, each separated with a Carriage Return/Line Feed (`chr$(13) + chr$(10)`).

The `title$` parameter specifies the text that appears in the dialog box's caption. If the `title$` parameter is not specified, no title appears in the dialog's caption.

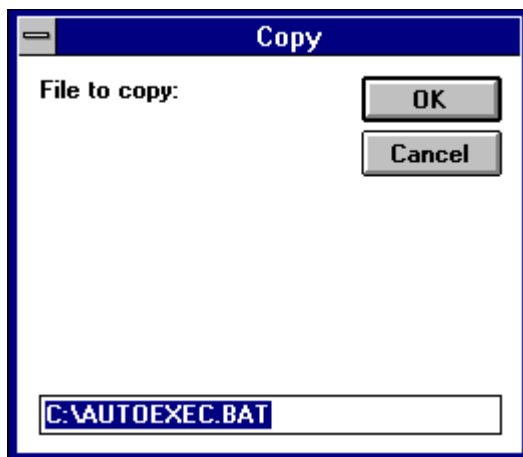
The `x` and `y` parameters are specified in twips ($1/20^{\text{th}}$ of a point or $1/1440$ of an inch). This allows the dialog box to be positioned in a device independent manner. If the position is not specified, then the dialog box is positioned on or near the object containing the executing script.

Example 1

[{bml j_bullet.bmp}InputBox\\$ Example](#)

Example 2

```
s$ = InputBox$("File to copy:",  
"Copy", "C:\AUTOEXEC.BAT")
```



See Also

[{bml j_bullet.bmp}Dialog Display](#)

#778 #779 #780 #781 InStr Function

Description

Searches a string for a substring.

Syntax

```
instr([start%,] search$,find$)
```

Returns

778 CMD_InStr

779 InStr

780 InStr;Substring;String, searching;Strings

781 Commands:0755

Returns an integer representing the character position of `find$` within `search$`.

Comments

If the string is found, its character position within `search$` is returned, with 1 being the character position of the first character.

If `start%` is specified, then the search starts at that character position within `search$`. The `start%` parameter must be between 1 and 65535. If not specified, the search starts at the beginning (`start% = 1`).

If the string is not found, or `start%` is greater than the length of `search$`, or if `search$` is empty, then 0 is returned.

Example

[{bml j_bullet.bmp}InStr Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#782 #783 #784 #785 Int Function

Description

Returns the integer part of a given number.

Syntax

`int (number#)`

Comments

This function returns the first integer less than (rounds down) the given value. The sign is preserved.

Example

[{bml j_bullet.bmp}Int Example](#)

See Also

[{bml j_bullet.bmp}Fix](#)

[{bml j_bullet.bmp}CInt](#)

[{bml j_bullet.bmp}Math Statements and Functions](#)

782 CMD_Int

783 Int

784 Int; Integer, extracting; Conversions

785 Commands: 0760

#786 \$787 K788 +789 Item\$ Function

Description

Gets a set of contiguous, delimited items from a text string..

Syntax

```
item$(text$,first%,last% [,delimiters$])
```

Returns

Returns all of the items between `first` and `last` within the specified text.

Comments

The `first` parameter specifies the first item in the sequence to return. The lowest value for `first` is 1. All items between `first` and `last` are returned.

By default, items are separated by commas and end-of-lines. This can be changed by specifying different delimiters in the `delimiters$` parameter.

If `first` is greater than the number of items in `text$`, then an empty string is returned.

If `last` is greater than the number of items in `text$`, then all items from `first` to the end of `text` are returned.

Example

{bml j_bullet.bmp}Item\$ and ItemCount Example

See Also

{bml j_bullet.bmp}Strings

#790 \$791 K792 +793 ItemCount Function

Description

Returns an integer representing the number of items in the specified text.

786 CMD_Item_DOLLAR

787 Item\$

788 Item\$;String, getting items from;Strings

789 Commands:0765

790 CMD_ItemCount

791 ItemCount

792 ItemCount;String, getting item count;Strings

793 Commands:0770

Syntax

```
ItemCount(text$ [,delimiters$])
```

Comments

By default, items are separated by commas and end-of-lines. This can be changed by specifying different delimiters in the `delimiters$` parameter. For example, to parse items using a backslash:

```
n = itemCount(text$,"\\")
```

The first `text$` item is 1.

Example

[{bml j_bullet.bmp}Item\\$ and ItemCount Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#794 \$795 K796 +797 Kill Statement

Description

Deletes one or more files.

Syntax

```
kill filespec$
```

Comments

This command deletes all files matching `filespec$`.

The `filespec$` parameter can contain wildcards, such as `*` and `?`.

This function behaves the same as the "del" command in DOS.

Example

[{bml j_bullet.bmp}Kill Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#798 \$799 K800 +801 LBound Function

794 CMD_Kill

795 Kill

796 Kill;File, deleting;File input and output

797 Commands:0775

798 CMD_LBound

799 LBound

800 LBound;Array, finding lowest element;Arrays

801 Commands:0780

Description

Determines the smallest subscript for a dimension of an array.

Syntax

```
lbound(ArrayVariable() [,dimension%])
```

Returns

Returns an integer representing the lower bound of the specified dimension of the specified array variable. If the array has no dimension, a runtime error is returned.

Comments

The first dimension is assumed if `dimension` is not specified (i.e., `dimension = 1`).

Example

[{bml j_bullet.bmp}LBound Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

#802 \$803 K804 +805 **LCase\$ Function**

Description

Returns the lower case equivalent of the specified string.

Syntax

```
lcase$(str$)
```

Example

[{bml j_bullet.bmp}LCase\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

802 CMD_LCase_DOLLAR
803 LCase\$
804 LCase\$;Lowercase, converting to;Strings
805 Commands:0785

#806 S807 K808 +809 Left\$ Function

Description

Returns the leftmost NumChars% characters from a given string.

Syntax

```
left$(str$,NumChars%)
```

Comments

If NumChars% is 0, then an empty string is returned.

If NumChars% is greater than or equal to the number of characters in the specified string, then the entire string is returned.

Example

{bml j_bullet.bmp}Left\$ Example

See Also

{bml j_bullet.bmp}Strings

#810 S811 K812 +813 Len Function

Description

Returns an integer representing the number of characters in a given string.

Syntax

```
len(str$)
```

Comments

If str is empty, 0 is returned.

Example

{bml j_bullet.bmp}Len Example

```
806 CMD_Left_DOLLAR
807 Left$
808 Left$;String, extracting from left;Strings
809 Commands:0790
810 CMD_Len
811 Len
812 Len;String, length of;Strings
813 Commands:0795
```


See Also

[{bml j_bullet.bmp}Strings](#)

#814 S815 K816 +817 **Let Statement**

Description

Assigns the result of an expression to a variable.

Syntax

```
[let] variable = expression
```

Comments

`let` is supported for compatibility with other implementations of DCL.

Example

[{bml j_bullet.bmp}Let Example](#)

See Also

[{bml j_bullet.bmp}Variables and Constants](#)

#818 S819 K820 +821 **Line\$ Function**

Description

Gets a set of lines (delimited by CR/LFs) from the specified text.

Syntax

```
line$(text$,first%[,last%])
```

Returns

Returns a single line or group of lines between `first` and `last`.

Comments

```
814 CMD_Let
815 Let
816 Let;Variable, assigning value to
817 Commands:0800
818 CMD_Line_DOLLAR
819 Line$
820 Line$;Text, getting lines of;Strings
821 Commands:0805
```

Lines are delimited by CR/LF pairs.

If `last` is not specified, then only one line is returned.

If `first` is greater than the number of lines in `text$`, an empty string is returned.

If `last` is greater than the number of lines in `text$`, all lines from `first` to the end of `text` are returned.

Example

[{bml j_bullet.bmp}Line\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#822 #823 #824 +825 **LineCount Function**

Description

Returns an integer representing the number of lines in the specified text.

Syntax

```
LineCount(text$)
```

Comments

Lines are delimited by CR/LF pairs.

Example

[{bml j_bullet.bmp}LineCount Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#826 #827 #828 +829 **LineInput # Statement**

Description

Reads a line into a string variable.

```
822 CMD_LineCount
823 LineCount
824 LineCount;Text, line count;Strings
825 Commands:0810
826 CMD_LineInput_POUND
827 LineInput #
828 LineInput #;String, reading line into;File input and output
829 Commands:0815
```

Syntax

```
lineinput [#]filename%,text$
```

Comments

This statement reads an entire line into the given string variable `text$`. The file is read up to the next carriage return. The file pointer is positioned after the terminating CR/LF.

The `filename%` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

The `filename%` must reference a file opened in `input` mode.

The `text$` parameter is any string variable reference.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#830 #831 #832 +833 **ListBox Statement**

Description

Defines a listbox that appears within a dialog box template.

Syntax

```
Listbox x%,y%,width%,height%,items$(),.Field
```

Comments

The `items$` array must be a single-dimension array of strings. The elements of this array are placed into the listbox when the dialog box is created. The `.Field` parameter defines the name used to extract which string occupies the listbox when the dialog box ends. On exit from the [Dialog](#) statement, the `.Field` will contain an index to the item that is highlighted in the listbox.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x,y,width,height` parameters are specified in dialog coordinates. The `x,y` position is relative to the upper left corner of the dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

830 CMD_ListBox

831 ListBox

832 ListBox;List box, defining;Dialog creation

833 Commands:0820

#834 S835 K836 +837

ListboxEnabled Function {bmc no_dos.bmp}

Description

Determines whether a listbox is enabled within the current window or dialog box.

Syntax

```
ListboxEnabled(name$ | id%)
```

Returns

Returns the integer TRUE if the given listbox is enabled within the active window or dialog box, FALSE otherwise.

Comments

If there is no active window, FALSE is returned.

The `name$` parameter specifies the text that appears within the static control that immediately precedes the listbox control in the window list (or dialog template). Alternatively, the `id%` of the listbox can be specified.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#838 S839 K840 +841

ListboxExists Function {bmc no_dos.bmp}

Description

Determines whether a listbox exists within the current window or dialog box.

Syntax

```
ListboxExists(name$ | id%)
```

834 CMD_ListboxEnabled

835 ListboxEnabled

836 ListboxEnabled;List box, determining if enabled;Dialog manipulation

837 Commands:0825

838 CMD_ListboxExists

839 ListboxExists

840 ListboxExists;List box, determining if exists;Dialog manipulation

841 Commands:0830

Returns

Returns the integer TRUE if the given listbox exists within the active window or dialog box, FALSE otherwise.

Comments

If there is no active window, FALSE is returned.

The `name$` parameter specifies the text that appears within the static control that immediately precedes the listbox control in the window list (or dialog template). Alternatively, the `id%` of the listbox can be specified.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#842 S843 K844 +845 Loc Function

Description

Returns an integer representing the position of the file pointer in the given file.

Note: We recommend using the [Seek](#) function instead.

Syntax

```
loc(filenumber%)
```

Comments

The `filenumber%` parameter is a number that is used by DCL to refer to the open file -- the number passed to the [open](#) statement.

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#846 S847 K848 +849 LOF Function

```
842 CMD_Loc
843 Loc
844 Loc;File pointer;File input and output
845 Commands:0835
846 CMD_LOF
847 LOF
848 LOF;File, length of;File input and output
849 Commands:0840
```

Description

Returns an integer representing the number of bytes in the given file.

Syntax

```
lof(filename%)
```

Comments

The `filename` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#850 #851 K852 +853 **Log Function**

Description

Returns a double-precision number representing the natural logarithm of a given number.

Syntax

```
log(number#)
```

Comments

The value of `number` must be greater than 0.

Example

[{bml j_bullet.bmp}Log Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

#854 #855 K856 +857 **LTrim\$ Function**

850 CMD_Log

851 Log

852 Log;Math commands

853 Commands:0845

854 CMD_LTrim_DOLLAR

855 LTrim\$

856 LTrim\$;Leading spaces, removing;Strings

857 Commands:0850

Description

Returns the specified string with the leading spaces removed.

Syntax

```
ltrim$(str$)
```

Example

{bml j_bullet.bmp}LTrim\$ Example

See Also

{bml j_bullet.bmp}Strings

#858 S859 K860 +861 **Main Statement**

Description

Defines the Main subroutine for the script.

Syntax

```
sub main()  
end sub
```

Comments

This defines the subroutine that receives execution control from the host application.

See Also

{bml j_bullet.bmp}Procedure Statements

#862 S863 K864 +865 **MCI Function {bmc no_dos.bmp}**

Description

Executes MCI (multimedia) command.

```
858 CMD_Main  
859 Main  
860 Main;Procedure statements  
861 Commands:0855  
862 CMD_MCI  
863 MCI  
864 MCI;Multimedia;Environment commands  
865 Commands:0860
```

Syntax

```
mci(command$,result$ [,error$])
```

Returns

Returns the integer 0 if the function was successful; otherwise it returns an error number.

Comments

If an error occurs, then the optional `error$` parameter is set to the text corresponding to the error.

If the `command$` returns a value, then that value is contained in `result$`.

The `mci` function accepts any MCI command as defined in the *Multimedia Programmers Reference* in the Windows 3.1 SDK.

Example

[{bml j_bullet.bmp}MCI Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#866 #867 #868 +869 **Menu Statement** **{bmc no_dos.bmp}**

Description

Issues the specified menu command from the active window.

Syntax

```
Menu MenuItem$
```

Comments

The `MenuItem` parameter specifies the complete menu item name, each menu level separated with a period. For example, the "Open" command on the "File" menu is represented by: "File.Open". Cascading menu items may have multiple periods, one for each popup menu, such as "File.Layout.Vertical". Menu items can also be specified using numeric index values. For example, to select the third menu item from the File menu, use "File.#3". To select the 4th item from the third menu, use "#3,#4". Separators count as items.

Items from an application's system menu can be selected by beginning the menu item specification with a period, such as ".Restore" or ".Minimize".

A runtime error will result if the menu item specification does not specify a menu item. For example, "File" specifies a menu popup, rather than a menu item. The menu item "File.Blank Blank" is not a valid menu item.

When comparing menu item names, this statement removes periods (.), spaces, and &. Further, all characters after a backspace or tab are removed. Thus, the menu item "&Open...\aCtrl+F12" translates simply to "Open".

#866 CMD_Menu

#867 Menu

#868 Menu;Menus

#869 Commands:0865

A runtime error is generated if the menu item cannot be found or is not enabled at the time that this statement is encountered.

Example

{bml j_bullet.bmp}Menu Example

See Also

{bml j_bullet.bmp}Menus

#870 S871 K872 +873 **MenuItemChecked Function** {bmc no_dos.bmp}

Description

Determines whether a menu item in the active window is checked.

Syntax

MenuItemChecked (MenuItemName\$)

Returns

Returns the integer TRUE if the given menu item exists and is checked, FALSE otherwise.

Comments

The `MenuItemName$` parameter specifies a complete menu item or menu item popup following the same format as that used by the Menu statement.

See Also

{bml j_bullet.bmp}Menus

#874 S875 K876 +877 **MenuItemEnabled Function** {bmc no_dos.bmp}

Description

Determines whether a menu item in the active window is enabled.

Syntax

MenuItemEnabled (MenuItemName\$)

870 CMD_MenuItemChecked
871 MenuItemChecked
872 MenuItemChecked;Menu item, determining if checked;Menus
873 Commands:0870
874 CMD_MenuItemEnabled
875 MenuItemEnabled
876 MenuItemEnabled;Menu item, determining if enabled;Menus
877 Commands:0875

Returns

Returns the integer TRUE if the given menu item exists and is enabled, FALSE otherwise.

Comments

The `MenuItemName$` parameter specifies a complete menu item or menu item popup following the same format as that used by the Menu statement.

See Also

{bml j_bullet.bmp}Menus

#878 #879 K880 +881 **MenuItemExists Function** {bmc no_dos.bmp}

Description

Determines whether a menu item in the active window exists.

Syntax

```
MenuItemExists(MenuItemName$)
```

Returns

Returns the integer TRUE if the given menu item exists, FALSE otherwise.

Comments

The `MenuItemName` parameter specifies a complete menu item or menu item popup following the same format as that used by the Menu statement.

Examples

```
sub main()  
if MenuItemExists("File.Open") then beep  
if MenuItemExists("File") then  
    MsgBox "There is a File menu."  
end sub
```

See Also

{bml j_bullet.bmp}Menus

878 CMD_MenuItemExists
879 MenuItemExists
880 MenuItemExists;Menu item, determining if exists;Menus
881 Commands:0880

Description

The `mid$` function returns a substring. The `mid$` function can also replace a substring with new text.

Syntax 1

```
mid$(str$,start% [,length%])
```

Returns

Returns a substring from the specified string. The substring starts at character position `start%` for `length%` number of characters.

Comments

If `length%` is not specified, then the entire string starting at `start%` is returned.

If `start%` is greater than the length of `str$`, then an empty string is returned.

Syntax 2

```
mid$(str$,start%[,length%]) = newvalue$
```

Comments

This statement replaces one part of a string with another. The `str$` parameter specifies the string variable containing the substring to replace. The substring within `str$` which is replaced starts at the character position specified by `start%` for `length%` number of characters. If `length%` is not specified, then the rest of the string is assumed.

The `newvalue$` parameter is any string or string expression. The resultant string is never longer than the original length of `str$`. Any extra characters at the end of `newvalue$` are ignored.

Example

[{bml j_bullet.bmp}Mid\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#886 S887 K888 +889 Minute Function

Description

Returns an integer representing the minute of the day encoded in the specified `serial#` parameter. The value returned is between 0 and 59 inclusive.

Syntax

```
minute(serial#)
```

Comment

You can obtain the value for the `serial#` parameter by using the [TimeSerial](#) or [TimeValue](#) command.

Example

[{bml j_bullet.bmp}Minute Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#890 S891 K892 +893 Mkdir Statement

Description

Creates a new directory.

Syntax

```
mkdir dir$
```

Comments

This command behaves just like the DOS "md" command.

Example

[{bml j_bullet.bmp}Mkdir Example](#)

See Also

886 CMD_Minute

887 Minute

888 Minute;Time functions

889 Commands:0890

890 CMD_Mkdir

891 Mkdir

892 Mkdir;Directory, making new;File input and output

893 Commands:0895

{bml j_bullet.bmp}File Input and Output

#894 S895 K896 +897 **Mod**

Description

Modulo operator.

Syntax

`expression1 mod expression2`

Returns

Returns the remainder of `expression1 / expression2`.

Notes

The two operands are converted to whole numbers before performing the modulo operation.

Example

{bml j_bullet.bmp}Mod Example

See Also

{bml j_bullet.bmp}Operators

#898 S899 K900 +901 **Month Function**

Description

Returns an integer representing the month of the date encoded in the specified `serial#` parameter. The value returned is between 1 and 12 inclusive.

Syntax

`month(serial#)`

Comment

894 CMD_Mod
895 Mod
896 Mod;Modulo operator;Operators
897 Commands:0900
898 CMD_Month
899 Month
900 Month;Date functions
901 Commands:0905

You can obtain the value for the `serial#` parameter by using the [DateSerial](#) or [DateValue](#) command.

Example

[{bml j_bullet.bmp}Month Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#902 S903 K904 +905 **MsgBox Statement and Function**

Description

The `MsgBox` statement displays a message box; the `MsgBox` function displays a message box and returns an integer representing the button that was pressed.

Statement Syntax

```
MsgBox msg$ [,type% [,title$]]
```

Function Syntax

```
MsgBox(msg$ [,type% [,title$]])
```

Returns

The `MsgBox` function returns one of the following numbers:

- 1OK was pressed
- 2Cancel was pressed
- 3Abort was pressed
- 4Retry was pressed
- 5Ignore was pressed
- 6Yes was pressed
- 7No was pressed

Comments

The dialog box is sized to hold the entire contents of `msg$`. The `msg$` string can contain CR/LF to separate lines. If a given line is too long, it will be word wrapped.

The `type` parameter is the sub of the any of the following values:

- 0display OK button only
- 1display OK, Cancel buttons

⁹⁰² `CMD_MsgBox`

⁹⁰³ `MsgBox`

⁹⁰⁴ `MsgBox`;Box, message;Message box, displaying;Dialog display

⁹⁰⁵ Commands:0910


2display Abort, Retry, Ignore buttons


3display Yes, No, Cancel buttons


4display Yes, No buttons

5display Retry, Cancel buttons

16 display "stop" icon 

32 display "question mark" icon 

48 display "exclamation point" icon 

64 display "information" icon 

0first button is the default button

256 second button is the default button

512 third button is the default button

0 Application modal - the current application is suspended until dialog box is closed

4096 System modal - all applications are suspended until the dialog box is closed

The default value for `type` is 0 (display only the OK button, making it the default).

The default value for `title$` is "DCL".

Example

{bml j_bullet.bmp}MsgBox Example

See Also

{bml j_bullet.bmp}Dialog Display

#906 #907 #908 #909 **MsgClose Statement** {bmc no_dos.bmp}

906 CMD_MsgClose

907 MsgClose

908 MsgClose;Message window, closing;Dialog display

909 Commands:0915

Description

Closes a message window that was opened with the [MsgOpen](#) statement.

Syntax

MsgClose

Comments

Nothing will happen if there is no open message window.

Example

[{bml j_bullet.bmp}Message Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#910 S911 K912 +913 **MsgOpen Statement** **{bmc no_dos.bmp}**

Description

Displays a window with a message.

Syntax

MsgOpen msg\$,timeout%,isCancel%, isThermometer%[,x%,y%]

Comments

The message can be displayed permanently, or for a specified number of seconds, or until an optional Cancel button is pressed.

The displayed message can be changed by calling the [MsgSetText](#) statement

The `timeout%` parameter causes the window to be removed after that number of seconds. The `timeout%` parameter has no effect if its value is zero.

The `isCancel` parameter controls whether or not a Cancel button appears within the window beneath the displayed message. If TRUE, then a Cancel button appears. If not specified, or FALSE, then no Cancel button is created. If a user presses the Cancel button at runtime, a trappable runtime error is generated. In this manner, a message box can be displayed and processing can continue as normal, aborting only when the process is canceled by pressing the Cancel button.

The `isThermometer` parameter controls whether there is a thermometer. If TRUE, then a thermometer is created between the text and the optional Cancel button. The thermometer initially indicated 0% complete, and can be changed using the [MsgSetThermometer](#) statement.

⁹¹⁰ CMD_MsgOpen

⁹¹¹ MsgOpen

⁹¹² MsgOpen;Message window, opening;Dialog display

⁹¹³ Commands:0920

The optional *x*, *y* parameters specify the location of the upper left corner of the message box, in twips (1/20th of a point or 1/1440 of an inch). If this point is not specified, then the window is centered on top of the parent.

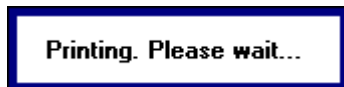
Only one message window can be opened at any one time. The message window is removed automatically when a script terminates.

Example 1

{bml j_bullet.bmp}Message Example

Example 2

```
MsgOpen "Printing. Please wait...",  
0,FALSE,FALSE
```



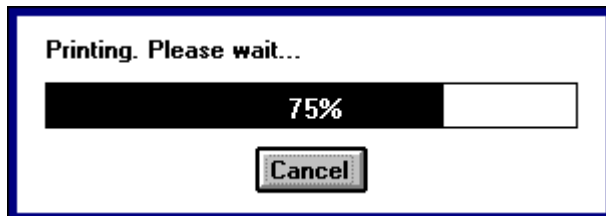
Example 3

```
MsgOpen "Printing. Please wait... ",  
0,TRUE,FALSE
```



Example 4

```
MsgOpen "Printing. Please wait... ",  
0,TRUE,TRUE  
MsgSetThermometer 75
```



See Also

{bml j_bullet.bmp}Dialog Display

#914 #915 #916 #917 **MsgSetText Statement** **{bmc no_dos.bmp}**

⁹¹⁴ CMD_MsgSetText

⁹¹⁵ MsgSetText

⁹¹⁶ MsgSetText;Message window, changing text;Dialog display

⁹¹⁷ Commands:0925

Description

Changes the text within an open message box (one that was previously opened with [MsgOpen](#)).

Syntax

```
MsgSetText newtext$
```

Comments

The message box is resized to accommodate the new text.

A runtime error will result if a message box is not currently open (using [MsgOpen](#)).

Example

[{bml j_bullet.bmp}Message Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#918 S919 K920 +921 **MsgSetThermometer Statement** {bmc no_dos.bmp}

Description

Changes the percentage filled in the thermometer of an open message box (one that was previously opened with [MsgOpen](#)).

Syntax

```
MsgSetThermometer percentage%
```

Comments

A runtime error will result if a message box is not currently open (using [MsgOpen](#)), or if the value of `percentage%` is not between 0 and 100 inclusive.

Example

[{bml j_bullet.bmp}Message Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#922 S923 K924 +925 **Name Statement**

918 CMD_MsgSetThermometer

919 MsgSetThermometer

920 MsgSetThermometer;Message window, thermometer;Dialog display

921 Commands:0930

922 CMD_Name

923 Name

924 Name;File, renaming;File input and output

925 Commands:0932

Description

Renames a file.

Syntax

```
name oldfile$ as newfile$
```

Example

{bml j_bullet.bmp}Name Example

See Also

{bml j_bullet.bmp}File Input and Output

#926 S927 K928 +929 **NetAttach Function** {bmc no_dos.bmp}

Description

Attaches the current user to the specified server using the given user ID and password.

Syntax

```
NetAttach(server$, user$, password$)
```

Comments

This function returns the integer TRUE if it is successful in attaching to the specified server or FALSE if the user is already attached, the password verification fails, or the server is not available.

Example

{bml j_bullet.bmp}NetAttach Example

See Also

{bml j_bullet.bmp}Network Functions

#930 S931 K932 +933 **NetConnectDrive Function** {bmc no_dos.bmp}

926 CMD_NetAttach

927 NetAttach

928 NetAttach;Attaching to server;Server, attaching to;Network functions

929 Commands: 0934

930 CMD_NetConnectDrive

931 NetConnectDrive

932 NetConnectDrive;Drive, connecting to network;Network functions

933 Commands: 0936

Description

Connects a local drive letter with a network path.

Syntax

```
NetConnectDrive(localpath$, networkpath$ [, [root%], passwd$])
```

Comments

Returns the integer TRUE if the drive is successfully mapped, otherwise returns FALSE.

If `root%` is specified, then the drive is created with a *false root*. If `passwd$` is specified it is used in attempting to connect the network drive. Note that `root%` and `passwd$` may not be supported in all networks.

`root%` defaults to FALSE, and `passwd$` defaults to none.

`localpath$` should be specified in the form "E:".

`networkpath$` is a valid network path, meaning *server/volume:...* or a UNC pathname.

Example

[{bml j_bullet.bmp}NetConnectDrive Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#934 #935 K936 +937 **NetDetach Function** {bmc no_dos.bmp}

Description

Detaches the current user from the specified file server.

Syntax

```
NetDetach(server$)
```

Comment

Returns the integer TRUE is returned if the drive is successfully detached, otherwise returns FALSE.

Caution

No safety checks are performed during the detach operation. Detaching from a server is a dangerous thing to do, and this function in no way guarantees that it will be done without causing some sort of system failure.

Example

[{bml j_bullet.bmp}NetDetach Example](#)

⁹³⁴ CMD_NetDetach

⁹³⁵ NetDetach

⁹³⁶ NetDetach;Detaching from server;Server, detaching from;Network functions

⁹³⁷ Commands: 0938

See Also

[{bml j_bullet.bmp}Network Functions](#)

#938 S939 K940 +941 **NetDirectoryRights Function** {bmc no_dos.bmp}

Description

Returns the integer TRUE if the current user has the given rights for the specified network directory path.

Syntax

```
NetDirectoryRights (path$, rights$)
```

Comments

The `path$` parameter may be any legal reference to a directory. This includes full paths with drive letter, server/volume, or UNC; or partial paths that are relative to the current directory.

The `rights$` parameter is a series of characters that have network specific meanings. For example, "ROS" on NetWare means Read, Open & Search permissions. If the user has **all** of these permissions for the specified path, the return value is TRUE.

Example

[{bml j_bullet.bmp}NetDirectoryRights Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#942 S943 K944 +945 **NetDisconnectDrive Function**

Description

Disconnects a local drive letter from a network path.

Syntax

```
NetDisconnectDrive (networkdrive$)
```

Comments

```
938 CMD_NetDirectoryRights
939 NetDirectoryRights
940 NetDirectoryRights;Directory, rights to;Network directory, rights;Network functions
941 Commands: 0940
942 CMD_NetDisconnectDrive
943 NetDisconnectDrive
944 NetDisconnectDrive;Drive, disconnecting from network;Network functions
945 Commands: 0941
```

The function takes a single string parameter which represents the drive letter that is to be disconnected. It returns an integer value 0 or -1. A 0 (or FALSE) indicates function failure, a -1 (or TRUE) indicates success.

Example

[{bml j_bullet.bmp}NetDisconnectDrive Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#946 S947 K948 +949 **NetGetDirectoryRights Function** {bmc no_dos.bmp}

Description

Returns a string representing the effective rights for the current user on the specified path.

Syntax

NetGetDirectoryRights\$(path\$)

Comments

The rights are identified as a series of characters compatible with those used by the network operating system that is in use.

Example

[{bml j_bullet.bmp}NetGetDirectoryRights Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#950 S951 K952 +953 **NetMemberOf Function** {bmc no_dos.bmp}

Description

Determines whether the current user is a member of the specified group.

Syntax

946 CMD_NetGetDirectoryRights

947 NetGetDirectoryRights

948 NetGetDirectoryRights;User, rights;Rights, network;Network functions

949 Commands: 0942

950 CMD_NetMemberOf

951 NetMemberOf

952 NetMemberOf;User, group membership;Network groups;Network functions

953 Commands: 0944

NetMemberOf (group\$, server\$)

Comments

This function returns the integer TRUE or FALSE.

If the `server$` parameter is specified then the groups on that server are searched, otherwise the primary server is used.

If the user `USERA` is a member of the group `SMALLGRP`, and `SMALLGRP` is a member of `BIGGROUP`, `USERA` is a member of `BIGGROUP`.

Example

[{bml j_bullet.bmp}NetMemberOf Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#954 #955 #956 +957 **NetStationID Function** {bmc no_dos.bmp}

Description

Returns a network-dependent station id as a string.

Syntax

NetStationID\$()

Comments

On Novell networks, the station's Ethernet address is returned.

Example

[{bml j_bullet.bmp}NetStationID Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#958 #959 #960 +961 **NetUserName Function** {bmc no_dos.bmp}

954 CMD_NetStationID

955 NetStationID

956 NetStationID;Network station;Network functions

957 Commands: 0946

958 CMD_NetUserName

959 NetUserName

960 NetUserName;Server, user name;User name, server;Network functions

961 Commands: 0948

Description

Returns the user name associated with the given server.

Syntax

```
NetUserName$([server$])
```

Comments

If server isn't specified then the primary server is used.

Example

[{bml j_bullet.bmp}NetUserName Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

#962 S963 K964 +965 **NetworkStatus Function** {bmc no_dos.bmp}

Description

Returns an integer representing the network status as a 16 bit (WORD) set of flags.

Syntax

```
NetworkStatus()
```

Comments

Each bit of the returned status has different significance. Currently, there are two bit flags.

NS_ACTIVE (0x0001) The network redirector is loaded

NS_LOGGEDON (0x0002) A user is actively logged onto a server/the network.

Normally, this function is compared to the value 3 to determine if additional network calls can or should be made.

Example

[{bml j_bullet.bmp}NetworkStatus Example](#)

See Also

[{bml j_bullet.bmp}Network Functions](#)

⁹⁶² CMD_NetworkStatus

⁹⁶³ NetworkStatus

⁹⁶⁴ NetworkStatus;Network functions

⁹⁶⁵ Commands: 0950

Description

Not operator.

Syntax

NOT *expression1*

Returns

TRUE if *expression1* is FALSE, otherwise returns TRUE.

If the operand is numeric, then the result is the bitwise NOT of the argument.

Notes

If the operand is a floating point value (either single or double), then it is first converted to a long, then a bitwise NOT is performed.

Example

{bml j_bullet.bmp}Not Example

See Also

{bml j_bullet.bmp}Operators

Description

Returns a double-precision number representing the current date and time. The number is returned in days since Dec 20, 1899.

Syntax

now ()

Example

{bml j_bullet.bmp}Now Example

966 CMD_Not
967 Not
968 Not;Not operator;Operators
969 Commands:0952
970 CMD_Now
971 Now
972 Now;Date functions
973 Commands:0954

See Also

[{bml.j_bullet.bmp}Date and Time Functions](#)

#974 S975 K976 +977 **NS_ACTIVE**

Description

Constant used by the [NetworkStatus](#) command.

#978 S979 K980 +981 **NS_LOGGEDON**

Description

Constant used by the [NetworkStatus](#) command.

974 CMD_NS_ACTIVE
975 NS_ACTIVE
976 NS_ACTIVE
977 Commands:0955
978 CMD_NS_LOGGEDON
979 NS_LOGGEDON
980 NS_LOGGEDON
981 Commands:0956

#982 S983 K984 +985 **Null Function**

Description

Returns a null string (a string that contains no characters and requires no storage).

Syntax

```
null[ () ]
```

Comments

An empty string ("") can also be used to remove all characters from a string. However, an empty string still requires some memory for storage. Null strings require no memory.

Example

[{bml j_bullet.bmp}Null Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#986 S987 K988 +989 **Oct\$ Function**

Description

Returns a string containing the octal equivalent of the specified number.

Syntax

```
oct$(number%)
```

Comments

The returned string contains only the number of octal digits necessary to represent the number.

The `number` parameter can be any type, but is rounded to the nearest whole number before converting to octal.

Example

```
982 CMD_Null
983 Null
984 Null;String, null;Strings
985 Commands:0958
986 CMD_Oct_DOLLAR
987 Oct$
988 Oct$;Octal, converting to;Conversions
989 Commands:0960
```

[{bml j_bullet.bmp}Oct\\$ Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

#990 \$991 K992 +993 **OKButton Statement**

Description

Defines an OK button that appears within a dialog box template.

Syntax

```
OKButton x%,y%,width%,height%
```

Comments

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#994 \$995 K996 +997 **On Error Statement**

Description

Defines the action taken when a trappable runtime error occurs.

Syntax

```
on error {goto <label> | resume next | goto 0}
```

Comments

```
990 CMD_OKButton
991 OKButton
992 OKButton;Button, OK;Dialog creation
993 Commands:0965
994 CMD_On_Error
995 On Error
996 On Error;Label;Error trapping
997 Commands:0970
```

The form `on error goto <label>` causes execution to transfer to the specified label when a runtime error occurs.

The form `on error resume next` causes execution to continue to the next line after the line that caused the error.

The form `on error goto 0` causes any existing error trap to be removed.

If an error trap is in effect when the script ends, then an error will be generated.

An error trap is only active within the subroutine or function in which it appears.

Once an error trap has gained control, appropriate action should be taken, and then control should be resumed using the `resume` statement.

If an error occurs within the error handler, the current routines error trap is disabled and a runtime error results.

Example

[{bml j_bullet.bmp}On Error Example](#)

See Also

[{bml j_bullet.bmp}Error Trapping](#)

[{bml j_bullet.bmp}Flow Control](#)

#998 \$999 K1000 +1001 **Open Statement**

⁹⁹⁸ CMD_Open

⁹⁹⁹ Open

¹⁰⁰⁰ Open;Opening a file;File, opening;File input and output

¹⁰⁰¹ Commands:0975

Description

Opens a file.

Syntax

```
open filename$ [for {input | output | append}] as  
[#] filenumber%
```

Comments

This statement opens a file for a given mode, assigning the open file to the supplied `filenumber`.

The `filename` parameter is a string expression that contains a valid DOS filename.

The `filenumber` parameter is a number between 1 and 255. The `FreeFile()` function can be used to determine an available file number.

The different modes are defined as follows:

Input	Opens an existing file for input.
Output	Opens an existing file, truncating its length to zero, or creates a new file.

Append	Opens an existing file, positioning the file pointer at the end of the file, or creates a new file.
--------	---

If the `[for mode]` is missing, then `append` is used.

Example

[{bml j_bullet.bmp}Input/Output Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#1002 S1003 K1004 +1005 **OpenFileName\$ Function**

Description

Displays the common file open dialog box (from COMMDLG.DLL), allowing the user to select a file.

Syntax

```
1002 CMD_OpenFileName_DOLLAR  
1003 OpenFileName$  
1004 OpenFileName$;Box, open file;Dialog display;Files, listing  
1005 Commands:0980
```

OpenFileName\$(title\$,extensions\$)

Returns

Returns the full DOS pathname of the file the user selected, or an empty string if the user canceled the dialog box.

Comments

The `title$` parameter specifies the title that appears on the dialog box's caption.

The `extensions$` parameter specifies the available file types. This string should be in the following format:

"*type:ext[,ext][;type:ext[,ext]]...*"

where *ext* is a valid file extension, like *.BAT or *.?F?, and *type* is a string that identifies this type to the user.

Example

{bml j_bullet.bmp}OpenFileName\$ Example

See Also

{bml j_bullet.bmp}Dialog Display

#1006 S1007 K1008 +1009 **Option Base Statement**

Description

Sets the lower bound for array declarations. By default, the lower bound used for all array declarations is 0.

Syntax

option base {0 | 1}

Comments

This statement must appear outside of any functions or subroutines.

Example

{bml j_bullet.bmp}Option Base Example

See Also

{bml j_bullet.bmp}Arrays

1006 CMD_Option_Base

1007 Option Base

1008 Option Base;Array, changing the lower bound;Arrays

1009 Commands:0985

#1010 S1011 K1012 +1013

OptionButton Statement

Description

Defines a push button with the specified text that appears within a dialog box template.

Syntax

```
OptionButton x%,y%,width%,height%,title$
```

Comments

The `title$` parameter may contain an ampersand character to denote an underlined accelerator, such as "&Font" for Font.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box.

Example

```
{bml j_bullet.bmp}Dialog Examples
```

See Also

```
{bml j_bullet.bmp}Dialog Creation
```

#1014 S1015 K1016 +1017

OptionEnabled Function

{bmc no_dos.bmp}

Description

Determines whether an option button is enabled within the current window or dialog box.

Syntax 1

```
OptionEnabled(name$)
```

Syntax 2

1010 CMD_OptionButton

1011 OptionButton

1012 OptionButton; Option button, defining;Dialog creation

1013 Commands:0990

1014 CMD_OptionEnabled

1015 OptionEnabled

1016 OptionEnabled;Option button, determining if enabled;Dialog manipulation

1017 Commands:0995

`OptionEnabled(id%)`

Returns

Returns the integer TRUE if the specified option button is enabled within the current window or dialog box, otherwise this function returns FALSE.

Comments

If an option button is enabled, its value can be set using the [SetOption](#) statement.

The option button can be referenced given either its `name$` (caption) or its `id%`.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#1018 S1019 K1020 +1021 **OptionExists Function** {bmc no_dos.bmp}

Description

Determines whether an option button exists within the current window or dialog box.

Syntax 1

`OptionExists(name$)`

Syntax 2

`OptionExists(id%)`

Returns

Returns the integer TRUE if the specified option button exists within the current window or dialog box, otherwise this function returns FALSE.

Comments

The option button can be referenced given either its `name$` (caption) or its `id%`.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

1018 CMD_OptionExists

1019 OptionExists

1020 OptionExists;Option button, determining if exists;Dialog manipulation

1021 Commands:1000

OptionGroup Statement

Description

Starts a group of option buttons within a dialog box template and defines the name used to determine which option button (from the group of option buttons) is selected when the dialog box ends.

Syntax

```
OptionGroup .Field
```

Comments

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

On exit from the Dialog statement, the `.Field` will contain the index of the selected option button, with 0 being the first option button.

Example

{bml j_bullet.bmp}Dialog Examples

See Also

{bml j_bullet.bmp}Dialog Creation

Or

Description

Or operator.

Syntax

```
expression1 OR expression2
```

Returns

TRUE if either `expression1` is TRUE or `expression2` is TRUE, otherwise FALSE.

If the two operands are numeric, the result is the bitwise OR of the two arguments.

1022 CMD_OptionGroup

1023 OptionGroup

1024 OptionGroup;Option button, defining group;Dialog creation

1025 Commands:1005

1026 CMD_Or

1027 Or

1028 Or;Or operator;Operators

1029 Commands:1010

Notes

If either of the two operands is a floating point number, the two operands are first converted to longs, then a bitwise OR is performed.

Example

{bml j_bullet.bmp}Or Example

See Also

{bml j_bullet.bmp}Operators

#1030 S1031 K1032 +1033 **PI**

Description

Pi constant.

Syntax

PI

Returns

3.141592653589793238462643383279

Notes

PI can also be determined using the following formula:

$$4 * \text{atn}(1)$$

Example

{bml j_bullet.bmp}PI Example

#1034 S1035 K1036 +1037 **PO_LANDSCAPE**

Description

Constant used with the `PrinterSetOrientation` statement to align the paper horizontally.

Returns

2

1030 CMD_PI

1031 PI

1032 PI

1033 Commands:1015

1034 CMD_PO_LANDSCAPE

1035 PO_LANDSCAPE

1036 PO_LANDSCAPE;Printer orientation

1037 Commands:1020

See Also

[{bml j_bullet.bmp}PrinterGetOrientation](#)

[{bml j_bullet.bmp}PrinterSetOrientation](#)

#1038 S1039 K1040 +1041 **PO_PORTRAIT**

Description

Constant used with the `PrinterSetOrientation` statement to align the paper vertically.

Returns

1

See Also

[{bml j_bullet.bmp}PrinterGetOrientation](#)

[{bml j_bullet.bmp}PrinterSetOrientation](#)

#1042 S1043 K1044 +1045 **PopupMenu Function** **{bmc no_dos.bmp}**

Description

Creates a popup menu using the string elements in the given array and returns an integer representing the user's response.

Syntax

`PopupMenu (MenuItems$ ())`

Returns

Returns the index of the selected item. If no item is selected (the popup menu is canceled), then a value of 1 less than the lower bound is returned.

Comments

Each array element is used as a menu item. An empty string results in a separator bar in the menu.

The popup menu is created with the upper left corner at the current mouse position.

```
1038 CMD_PO_PORTRAIT
1039 PO_PORTRAIT
1040 PO_PORTRAIT;Printer orientation
1041 Commands:1025
1042 CMD_PopupMenu
1043 PopupMenu
1044 PopupMenu;Menu popup;Dialog display
1045 Commands:1030
```

A runtime error results if `MenuItem$` is not a single-dimension array.

Only one popup menu can be displayed at a time. An error will result if another script executes this function while a popup menu is visible.

Example

[{bml j_bullet.bmp}PopupMenu Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#1046 S1047 K1048 +1049 **Print Statement**

Description

Writes data to a viewport window.

Syntax

```
print expression [{, | ;} expression]...
```

Comments

Strings are written in their literal form, with no enclosing quotes.

Integers and longs are written with an initial space reserved for the sign (space = positive). Additionally, there is a space following each number.

Each `expression` is separated either with a comma (,) or a semicolon (;). A comma means that the next expression is output in the next print zone. A semicolon means that the next expression is output immediately after the current expression. Print zones are defined every 14 spaces.

If the last expression is not followed by a comma or semicolon, then a carriage return is printed to the file. If the last expression in the list ends with a semicolon, no carriage return is printed - the next print statement will output information immediately following the expression. If the last expression in the list ends with a comma, the file pointer is positioned at the start of the next print zone on the current line.

If no viewport window is open, then the statement is ignored. Printing information to a viewport window is a convenient way to output debugging information.

Example

[{bml j_bullet.bmp}Print Example](#)

See Also

[{bml j_bullet.bmp}Viewport Window Manipulation](#)

¹⁰⁴⁶ CMD_Print

¹⁰⁴⁷ Print

¹⁰⁴⁸ Print;Writing to viewport;Viewport, writing to;Viewport window manipulation;Output, displaying

¹⁰⁴⁹ Commands:1035

Print # Statement

Description

Writes data to a disk file.

Syntax

```
print #filename%, expression [{, | ;} expression]...
```

Comments

The `filename` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

Strings are written in their literal form, with no enclosing quotes.

Integers and longs are written with an initial space reserved for the sign (space = positive). Additionally, there is a space following each number.

Each `expression` is separated either with a comma (,) or a semicolon (;). A comma means that the next expression is output in the next print zone. A semi-colon means that the next expression is output immediately after the current expression. Print zones are defined every 14 spaces.

If the last expression is not followed by a comma or semi-colon, then a carriage return is printed to the file. If the last expression in the list ends with a semi-colon, no carriage return is printed - the next print statement will output information immediately following the expression. If the last expression in the list ends with a comma, the file pointer is positioned at the start of the next print zone on the current line.

The `write` statement always outputs information ending with a carriage-return. Thus, if a `print` statement is followed by a `write` statement, the file pointer is positioned on a new line.

The `print` statement can only be used with files that are opened in `output` or `append` modes.

Example

{bml j_bullet.bmp}Print # Example

See Also

{bml j_bullet.bmp}File Input and Output

PrinterGetOrientation Function

{bmc no_dos.bmp}

```
1050 CMD_Print_POUND
1051 Print #
1052 Print #;Writing to file;File, writing to;File input and output
1053 Commands:1040
1054 CMD_PrinterGetOrientation
1055 PrinterGetOrientation
1056 PrinterGetOrientation;Printer orientation
1057 Commands:1045
```

Description

Retrieves the orientation of the default printer--the printer specified in the `device=` line in the `[windows]` section of the WIN.INI file.

Syntax

```
PrinterGetOrientation()
```

Returns

Returns the integer `PO_PORTRAIT` if the printer orientation is set to portrait, otherwise if returns `PO_LANDSCAPE`.

Comments

This function loads the printer driver and therefore may be slow.

Example

{bml j_bullet.bmp}PrinterGetOrientation Example

See Also

{bml j_bullet.bmp}Printer Manipulation

#1058 S1059 K1060 +1061 **PrinterSetOrientation Statement** {bmc no_dos.bmp}

Description

Sets the orientation of the default printer--the printer specified in the `device=` line in the `[windows]` section of the WIN.INI file.

Syntax

```
PrinterSetOrientation NewSetting%
```

Comments

NewSetting% is `PO_PORTRAIT` or `PO_LANDSCAPE`.

This command loads the printer driver for the default printer, and therefore may be slow.

Example

{bml j_bullet.bmp}PrinterSetOrientation Example

See Also

{bml j_bullet.bmp}Printer Manipulation

1058 CMD_PrinterSetOrientation
1059 PrinterSetOrientation
1060 PrinterSetOrientation;Printer orientation
1061 Commands:1050

#1062 S1063 K1064 +1065

PrintFile Function {bmc no_dos.bmp}

Description

Invokes the Windows 3.1 shell functions that cause an application to execute and print a file.

Syntax

```
PrintFile(filename$)
```

Returns

Returns an integer representing the ID of the executing task.

Comments

This function is only available under Windows 3.1.

The application to be executed must be associated with the file extension of the file specified by this command. This association is established in the [Extensions] section of the WIN.INI file. For example, if the Notepad application is associated with the .TXT extension, the Notepad application is started when DCL executes a PrintFile command for a .TXT file.

This command does not support .EXE, .COM, .BAT, or .PIF files.

Example

{bml j_bullet.bmp}PrintFile Example

See Also

{bml j_bullet.bmp}Printer Manipulation

#1066 S1067 K1068 +1069

PushButton Statement

Description

Defines a push button within a dialog box template.

Syntax

1062 CMD_PrintFile

1063 PrintFile

1064 PrintFile;File, printing;Printing a file

1065 Commands:1055

1066 CMD_PushButton

1067 PushButton

1068 PushButton;Push button, defining;Dialog creation;Button, defining

1069 Commands:1060

PushButton x%,y%,width%,height%,title\$

Comments

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

When a push button is selected, the Dialog statement ends.

The x, y, width, height parameters are specified in dialog coordinates. The x, y position is relative to the upper left corner of the dialog box.

The title\$ parameter may contain an ampersand character to denote an underlined accelerator, such as "&Font" for Font.

Example

{bml j_bullet.bmp}Dialog Examples

See Also

{bml j_bullet.bmp}Dialog Creation

#1070 S1071 K1072 +1073 **QueEmpty Statement** {bmc no_dos.bmp}

Description

Empties the current event queue.

Syntax

QueEmpty

Comments

After this statement, QueFlush will do nothing.

Example

{bml j_bullet.bmp}Queue Example

See Also

{bml j_bullet.bmp}Keyboard Manipulation

{bml j_bullet.bmp}Mouse Events

#1074 S1075 K1076 +1077 **QueFlush Statement** {bmc no_dos.bmp}

1070 CMD_QueEmpty

1071 QueEmpty

1072 QueEmpty;Event queue, emptying;Mouse events;Keyboard manipulation

1073 Commands:1065

1074 CMD_QueFlush

1075 QueFlush

1076 QueFlush;Event queue, playing back;Mouse events;Keyboard manipulation

1077 Commands:1070

Description

Plays back events that are stored in the current event queue.

Syntax

```
QueFlush isSaveState%
```

Comments

After `QueFlush` is finished, the queue is empty.

The `QueFlush` statement uses the Windows journaling mechanism to replay the mouse and keyboard events stored in the queue. As a result, the mouse position may be changed. Furthermore, events can be played into any Windows application, including DOS applications running in a window.

If `isSaveState` is `TRUE`, then `QueFlush` saves the state of the CAPSLOCK, NUMLOCK, SCROLL LOCK, and INSERT, and restores the state after the `QueFlush` is complete. If `FALSE`, these states are not restored.

The function does not return until the entire queue has been played.

Example

[{bml j_bullet.bmp}Queue Example](#)

See Also

[{bml j_bullet.bmp}Keyboard Manipulation](#)

[{bml j_bullet.bmp}Mouse Events](#)

#1078 S1079 K1080 +1081 **QueKeyDn Statement** {bmc no_dos.bmp}

Description

Appends key down events for the specified keys to the end of the current event queue.

Syntax

```
QueKeyDn Keys$
```

Comments

The format for `Keys$` is the same as for the [QueKeys](#) `KeyString$` parameter, with the exception that parentheses are illegal.

The [QueFlush](#) command is used to play back the events stored in the current event queue.

Example

```
1078 CMD_QueKeyDn
1079 QueKeyDn
1080 QueKeyDn;Event queue, appending to;Keys;Keyboard manipulation
1081 Commands:1075
```

[{bml j_bullet.bmp}Queue Example](#)

See Also

[{bml j_bullet.bmp}Keyboard Manipulation](#)

[{bml j_bullet.bmp}QueKeys](#) for list of special keys.

#1082 S1083 K1084 +1085 **QueKeys Statement** **{bmc no_dos.bmp}**

Description

Appends keystroke information to the current event queue.

Syntax

QueKeys KeyString\$

Comments

To specify any key on the keyboard, simply use that key, such as "a" for lower case a, or "A" for upper case a.

Sequences of keys are specified by appending them together: "abc" or "dir /w".

The keys +, ^, ~, and % are special and must be specified within brackets. For example, to specify the percent, use "%".

The keys {}[] also have special meaning. To specify one of these keys, enclose it within brackets, such as "{ }".

Keys that are not displayed when you press that key are described within brackets, such as {ENTER} or {UP}. A list of these keys follows:

{BACKSPACE}	{BS}	{BREAK}	{CAPSLOCK}
{CLEAR}	{DELETE}	{DEL}	{DOWN}
{END}	{ENTER}	{ESCAPE}	{ESC}
{HELP}	{HOME}	{INSERT}	{LEFT}
{NUMLOCK}	{NUMPAD0}	{NUMPAD1}	{NUMPAD2}
{NUMPAD3}	{NUMPAD4}	{NUMPAD5}	{NUMPAD6}
{NUMPAD7}	{NUMPAD8}	{NUMPAD9}	{NUMPAD/}
{NUMPAD*}	{NUMPAD-}	{NUMPAD+}	{NUMPAD.}
{PGDN}	{PGUP}	{PRTSC}	{RIGHT}
{TAB}	{UP}	{F1}	{SCROLLLOCK}
{F2}	{F3}	{F4}	{F5}
{F6}	{F7}	{F8}	{F9}
{F10}	{F11}	{F12}	{F13}
{F14}	{F15}	{F16}	

Keys can be combined with SHIFT, CTRL, and ALT using the reserved keys "+", "^", and "%" respectively:

Shift+Enter	" + {ENTER} "
Ctrl+C	" ^ c "
Alt+F2	" % { F2 } "

1082 CMD_QueueKeys

1083 QueKeys

1084 QueKeys;Event queue, appending to;Keys;Keyboard manipulation

1085 Commands:1080

To specify a modifier key combined with a sequence of consecutive keys, group the key sequence within parentheses, as in the following example:

```
Shift+A, Shift+B, Shift+C      "+(abc) "
Ctrl+F1, Ctrl+F2               "^({F1}{F2}) "
```

Use "~" as a shortcut for embedding ENTER within a key sequence:

```
"ab~de"
```

To embed quotes, use two quotes in a row:

```
"This is a ""test"" of the system"
```

Key sequences can be repeated using a repeat count within brackets:

```
"{a 10}"           produces 10 "a" keys
"{ENTER 2}"        produces 2 Enter keys
```

Example

[{bml j_bullet.bmp}Queue Example](#)

See Also

[{bml j_bullet.bmp}Keyboard Manipulation](#)

#1086 S1087 K1088 +1089 **QueKeyUp Statement** {bmc no_dos.bmp}

Description

Appends key up events for the specified keys to the end of the current event queue.

Syntax

```
QueKeyUp Keys$
```

Comments

The format for `Keys$` is the same as for the [QueKeys](#) `KeyString$` parameter, with the exception that parentheses are illegal.

The [QueFlush](#) command is used to play back the events stored in the current event queue.

Example

[{bml j_bullet.bmp}Queue Example](#)

See Also

[{bml j_bullet.bmp}Keyboard Manipulation](#)

[{bml j_bullet.bmp}QueKeys](#) for list of special keys.

1086 CMD_QueKeyUp
1087 QueKeyUp
1088 QueKeyUp;Event queue, appending to;Keys;Keyboard manipulation
1089 Commands:1085

#1090 S1091 K1092 +1093

QueMouseClicked Statement

{bmc no_dos.bmp}

Description

Adds a mouse click to the current event queue.

Syntax

```
QueMouseClicked button%,x%,y%
```

Comments

A mouse click consists of a mouse button down at position x , y , immediately followed by a mouse button up.

The `button` parameter specifies which button to queue: either `VK_LBUTTON` or `VK_RBUTTON`.

The `QueFlush` command is used to play back the events stored in the current event queue.

Example

{bml j_bullet.bmp}Queue Example

See Also

{bml j_bullet.bmp}Mouse Events

#1094 S1095 K1096 +1097

QueMouseDblClk Statement

{bmc no_dos.bmp}

Description

Adds a mouse double click to the current event queue.

Syntax

```
QueMouseDblClk button%,x%,y%
```

Comments

A mouse double click consists of a mouse DN/UP/DN/UP at position x , y . The events are queued in such a way that a double-click is registered during queue playback.

1090 CMD_QueMouseClicked

1091 QueMouseClicked

1092 QueMouseClicked;Event queue, appending to;Mouse;Mouse events

1093 Commands:1090

1094 CMD_QueMouseDblClk

1095 QueMouseDblClk

1096 QueMouseDblClk;Event queue, appending to;Mouse;Mouse events

1097 Commands:1095

The `button` parameter specifies which button to queue: either VK_LBUTTON or VK_RBUTTON.

The QueFlush command is used to play back the events stored in the current event queue.

Example

{bml j_bullet.bmp}Queue Example

See Also

{bml j_bullet.bmp}Mouse Events

#1098 S1099 K1100 +1101 **QueMouseDblDn Statement** {bmc no_dos.bmp}

Description

Adds a mouse double down to the current event queue.

Syntax

`QueMouseDblDn button%, x%, y%`

Comments

A double down consists of a mouse DN/UP/DN at position `x%`, `y%`.

The `button%` parameter specifies which button to queue: either VK_LBUTTON or VK_RBUTTON.

The QueFlush command is used to play back the events stored in the current event queue.

Example

{bml j_bullet.bmp}Queue Example

See Also

{bml j_bullet.bmp}Mouse Events

#1102 S1103 K1104 +1105 **QueMouseDn Statement** {bmc no_dos.bmp}

Description

Adds a mouse down to the current event queue.

1098 CMD_QueMouseDblDn

1099 QueMouseDblDn

1100 QueMouseDblDn;Event queue, appending to;Mouse;Mouse events

1101 Commands:1100

1102 CMD_QueMouseDn

1103 QueMouseDn

1104 QueMouseDn;Event queue, appending to;Mouse;Mouse events

1105 Commands:1105

Syntax

`QueMouseDn button%,x%,y%`

Comments

The `button` parameter specifies which button to queue: either `VK_LBUTTON` or `VK_RBUTTON`.

The `QueFlush` command is used to play back the events stored in the current event queue.

Example

`{bml j_bullet.bmp}Queue Example`

See Also

`{bml j_bullet.bmp}Mouse Events`

#1106 S1107 K1108 +1109 **QueMouseMove Statement** **{bmc no_dos.bmp}**

Description

Adds a mouse move to the current event queue.

Syntax

`QueMouseMove button%,x%,y%`

Comments

The `button` parameter specifies which button to queue: either `VK_LBUTTON` or `VK_RBUTTON`.

The `QueFlush` command is used to play back the events stored in the current event queue.

Example

`{bml j_bullet.bmp}Queue Example`

See Also

`{bml j_bullet.bmp}Mouse Events`

#1110 S1111 K1112 +1113 **QueMouseUp Statement** **{bmc no_dos.bmp}**

1106 CMD_QueMouseMove

1107 QueMouseMove

1108 QueMouseMove;Event queue, appending to;Mouse;Mouse events

1109 Commands:1110

1110 CMD_QueMouseUp

1111 QueMouseUp

1112 QueMouseUp;Event queue, appending to;Mouse;Mouse events

1113 Commands:1115

Description

Adds a mouse up to the current event queue.

Syntax

```
QueueMouseUp button%,x%,y%
```

Comments

The `button` parameter specifies which button to queue: either `VK_LBUTTON` or `VK_RBUTTON`.

The `QueueFlush` command is used to play back the events stored in the current event queue.

Example

[{bml j_bullet.bmp}Queue Example](#)

See Also

[{bml j_bullet.bmp}Mouse Events](#)

#1114 #1115 #1116 #1117 **QueueSetRelativeWindow Statement** [{bmc no_dos.bmp}](#)

Description

Adjusts all mouse positions relative to the specified window.

Syntax

```
QueueSetRelativeWindow hWnd%
```

Comments

This statement affects all subsequent `Que...` commands.

The `hWnd%` parameter is a handle to a window in the Windows environment. If `hWnd%` is 0, then the window with the focus is used (i.e., the active window).

The `QueueFlush` command is used to play back the events stored in the current event queue.

Example 1

[{bml j_bullet.bmp}Queue Example](#)

Example 2

```
sub main()  
'Adjust mouse coordinates relative to Notepad  
hWnd = WinFind("Notepad")  
QueueSetRelativeWindow hWnd  
end sub
```

See Also

[{bml j_bullet.bmp}Mouse Events](#)

#1114 `CMD_QueueSetRelativeWindow`

#1115 `QueueSetRelativeWindow`

#1116 `QueueSetRelativeWindow`;Event queue, adjusting mouse positions;Mouse;Mouse events

#1117 `Commands:1120`

#1118 S1119 K1120 +1121 **Random Function**

Description

Generates a random number.

Syntax

```
random(min&,max&)
```

Returns

Returns a long integer greater than or equal to `min` and less than or equal to `max`.

Example

[{bml j_bullet.bmp}Random Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

#1122 S1123 K1124 +1125 **Randomize Function**

Description

Initializes the random number generator with a new seed.

Syntax

```
randomize [seed&]
```

Comments

If `seed` is not specified, then the current value of the system clock is used.

Example

[{bml j_bullet.bmp}Randomize Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

1118 CMD_Random

1119 Random

1120 Random;Random number, generating;Math commands

1121 Commands:1125

1122 CMD_Randomize

1123 Randomize

1124 Randomize;Random number generator, initializing;Math commands

1125 Commands:1130

#1126 S1127 K1128 +1129 **ReadINI\$ Function**

Description

Returns the value of the specified item from the specified INI file.

Syntax

```
ReadINI$(section$,item$[,filename$])
```

Comments

The `filename$` parameter, if specified, contains the name of the INI file to read. Otherwise, the WIN.INI file is used.

The `section$` parameter specifies the section that contains the desired variable, such as "windows". Section names are specified without the enclosing brackets.

The `item$` parameter specifies which item to retrieve the value of.

Example

{bml j_bullet.bmp}ReadINI\$ Example

See Also

{bml j_bullet.bmp}Environment Statements and Functions

#1130 S1131 K1132 +1133 **ReadINISection Statement**

Description

Reads all of the item names from a given section of the specified INI file.

Syntax

```
ReadINISection section$,items$() [,filename$]
```

Comments

The `filename$` parameter, if specified, contains the name of an INI file. Otherwise, the WIN.INI file is used.

1126 CMD_ReadINI_DOLLAR

1127 ReadINI\$

1128 ReadINI\$;INI file, reading item from;Environment commands

1129 Commands:1135

1130 CMD_ReadINISection

1131 ReadINISection

1132 ReadINISection;INI file, reading section;Environment commands

1133 Commands:1140

The `section$` parameter specifies the section that contains the desired variables, such as "windows". Section names are specified without the enclosing brackets.

The `items$` parameter refers to the item name on the left side of the = sign. This parameter must be a single dimension array of strings (see [Dim](#) statement). On return, this will contain one array element for each variable in the specified INI section. Use the [lbound](#) and [ubound](#) functions to determine its size on return.

Example

[{bml j_bullet.bmp}ReadINISection Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1134 #1135 #1136 #1137 **ReDim Statement**

Description

Redimensions an array, specifying a new upper and lower bound for a given arrays dimensions.

Syntax

```
redim variablename (subscriptRange) [as type],...
```

Comments

The *variablename* parameter specifies the name of an existing array (previously declared using the [dim](#) statement).

Caution: The ReDim statement deletes any data already in the array.

The *subscriptRange* parameter specifies the new upper and lower bounds for each dimension of the array using the following syntax:

```
[lower% to] upper% [, [lower% to] upper%]...
```

If *lower%* is not specified, then 0 is used (or the value set using the `option base` statement).

Arrays can be dynamically dimensioned by first declaring them with no initial size using the `dim` statement:

```
dim a$()
```

Then, using the `redim` statement, the actual size can be specified:

```
redim a$(0 to 8,6 to 10)
```

The number of dimensions of an array cannot be changed once the array has been given dimensions--either by declaring it with initial dimensions using `dim` or by a previous use of `redim`.

1134 CMD_ReDim

1135 ReDim

1136 ReDim;Array, redimensioning;Arrays

1137 Commands:1145

The *type* parameter can be used to specify the array element type. The following can be used: integer, long, string.

Example

[{bml j_bullet.bmp}ReDim Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

[{bml j_bullet.bmp}Variables and Constants](#)

#1138 \$1139 K1140 +1141 **REM Statement**

Description

Causes the compiler to skip all characters on that line.

Syntax

REM text

Example

[{bml j_bullet.bmp}REM Example](#)

See Also

[{bml j_bullet.bmp}Comments](#)

#1142 \$1143 K1144 +1145 **RefreshIni Statement** **{bmc no_dos.bmp}**

Description

Reloads the WIN.INI file from disk, thus refreshing all WIN.INI settings.

Syntax

RefreshIni

Comments

This forces all hand-edited changes to the WIN.INI file to be activated.

```
1138 CMD_REM
1139 REM
1140 REM;Comments
1141 Commands:1150
1142 CMD_RefreshIni
1143 RefreshIni
1144 RefreshIni;WIN.INI, refreshing;Environment commands
1145 Commands: 1152
```

Example

[{bml j_bullet.bmp}RefreshIni Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1146 S1147 K1148 +1149 **Reset Statement**

Description

Closes all open files, writing out all I/O buffers.

Syntax

```
reset
```

Example

[{bml j_bullet.bmp}Reset Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#1150 S1151 K1152 +1153 **RestoreEnv Function** **{bmc no_dos.bmp}**

Description

Restores the set of environment variables from the virtual stack for DOS or for Windows saved by the [SaveEnv](#) function.

Syntax

```
RestoreEnv ( [ $mode ] )
```

Comments

Returns the integer TRUE if the function is successful, otherwise FALSE.

Separate stacks are kept for the Windows and DOS environments.

1146 CMD_Reset

1147 Reset

1148 Reset;File input and output

1149 Commands:1155

1150 CMD_RestoreEnv

1151 RestoreEnv

1152 RestoreEnv;Environment variables;DOS environment;Windows environment;Variables, environment;Environment commands

1153 Commands: 1157

The `mode$` parameter can be `ENV_DOS` or `ENV_WINDOWS`.

If mode is unspecified, the default is `ENV_WINDOWS`.

Example

[{bml j_bullet.bmp}RestoreEnv Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1154 \$1155 K1156 +1157 **Resume Statement**

Description

Ends an error handler and continues execution.

Syntax

```
resume {[0] | next | label}
```

Comments

The form `resume [0]` causes execution to continue with the statement that caused the error.

The form `resume next` causes execution to continue with the statement following the statement that caused the error.

The form `resume label` causes execution to continue at the specified label.

Example

[{bml j_bullet.bmp}Resume Example](#)

See Also

[{bml j_bullet.bmp}Error Trapping](#)

#1158 \$1159 K1160 +1161 **Return Statement**

Description

1154 CMD_Resume

1155 Resume

1156 Resume;Error trapping

1157 Commands:1160

1158 CMD_Return

1159 Return

1160 Return;Flow control

1161 Commands:1165

Transfers execution control to the statement following the most recent gosub.

Syntax

```
return
```

Comments

A runtime error results if a `return` statement is encountered without a corresponding `gosub` statement.

Example

[{bml j_bullet.bmp}Return Example](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

#1162 S1163 K1164 +1165 **Right\$ Function**

Description

Returns the rightmost `NumChars` characters from a specified string.

Syntax

```
right$(str$,NumChars$)
```

Comments

If `NumChars` is greater than or equal to the length of the string, then the entire string is returned.

If `NumChars` is 0, then an empty string is returned.

Example

[{bml j_bullet.bmp}Right\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#1166 S1167 K1168 +1169 **Rmdir Statement**

1162 CMD_Right_DOLLAR
1163 Right\$
1164 Right\$;String, extracting from right;Strings
1165 Commands:1170
1166 CMD_Rmdir
1167 Rmdir
1168 Rmdir;Directory, removing;File input and output
1169 Commands:1175

Description

Removes the specified directory.

Syntax

```
rmdir dir$
```

Comments

This command behaves just like the DOS "rd" command.

Example

[{bml j_bullet.bmp}Rmdir Example](#)

See Also

[{bml j_bullet.bmp}File Input and Output](#)

#1170 S1171 K1172 +1173 **Rnd Function****Description**

Returns a single-precision random number between 0 and 1.

Syntax

```
rnd[ (number#) ]
```

Comments

If `number#` is omitted, the next random number is returned. Otherwise, the `number#` parameter has the following meaning:

```
number# < 0 Always returns the same number  
number# = 0 Returns the last number generated  
number# > 0 Returns the next random number
```

Example

[{bml j_bullet.bmp}Rnd Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

1170 CMD_Rnd

1171 Rnd

1172 Rnd;Random number;Math commands

1173 Commands:1180

#1174 S1175 K1176 +1177

RTrim\$ Function

Description

Returns the string with the trailing spaces removed.

Syntax

```
rtrim$(str$)
```

Example

{bml j_bullet.bmp}RTrim\$ Example

See Also

{bml j_bullet.bmp}Strings

#1178 S1179 K1180 +1181

SaveEnv Function

{bmc no_dos.bmp}

Description

Pushes the current environment variable set onto a virtual stack so that they can later be restored by the RestoreEnv function.

Syntax

```
SaveEnv ([mode$])
```

Comments

Returns the integer TRUE if the function is successful, otherwise FALSE.

Separate stacks are kept for the Windows and DOS environments.

The `mode$` parameter can be ENV_DOS or ENV_WINDOWS.

If mode is unspecified, the default is ENV_WINDOWS.

Example

{bml j_bullet.bmp}SaveEnv Example

1174 CMD_RTrim_DOLLAR

1175 RTrim\$

1176 RTrim\$;Trailing spaces, removing;Strings

1177 Commands:1185

1178 CMD_SaveEnv

1179 SaveEnv

1180 SaveEnv;Environment variables;DOS environment;Windows environment;Variables,
environment;Environment commands

1181 Commands: 1187

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1182 S1183 K1184 +1185 **SaveFileName\$ Function**

Description

Displays the common file save dialog box (from COMMDLG.DLL), allowing the user to select a file. If the file already exists, the user is prompted to overwrite it.

Syntax

```
SaveFileName$(title$ [,extensions$])
```

Returns

Returns a full DOS pathname of the file that the user selected.

Comments

The `title$` parameter specifies the title that appears on the dialog box's caption.

The `extensions$` parameter specifies the available file types. This string should be in the following format:

```
"type:ext[,ext][;type:ext[,ext]]..."
```

where `ext` is a valid file extension, like *.BAT or *.?F?, and `type` is a string that identifies this type to the user.

By default, the first extension in appearing within `extensions` is used.

Example

[{bml j_bullet.bmp}SaveFileName\\$ Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#1186 S1187 K1188 +1189 **Second Function**

```
1182 CMD_SaveFileName_DOLLAR
1183 SaveFileName$
1184 SaveFileName$;File, saving;Dialog display
1185 Commands:1190
1186 CMD_Second
1187 Second
1188 Second;Time functions
1189 Commands:1195
```

Description

Returns an integer representing the second of the day encoded in the specified `serial#` parameter. The value returned is between 0 and 59 inclusive.

Syntax

```
second(serial#)
```

Comment

You can obtain the value for the `serial#` parameter by using the [TimeSerial](#) or [TimeValue](#) command.

Example

[{bml j_bullet.bmp}Second Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#1190 S1191 K1192 +1193 **Seek Statement and Function**

Description

The `seek` function returns the file pointer for a given file. The `seek` statement sets the file pointer.

Function Syntax

```
seek(filenumber%)
```

Statement Syntax

```
seek [#] filenumber%,position&
```

Comments

The `filenumber` parameter is a number that is used by DCL to refer to the open file -- the number passed to the [open](#) statement.

See Also

[{bml j_bullet.bmp}File Input and Output](#)

1190 CMD_Seek

1191 Seek

1192 Seek;File, setting pointer;File input and output

1193 Commands:1200

Select...Case Statement

Description

Executes a block of DCL statements depending on the value of a given expression.

Syntax

```
select case testexpression
[case expressionlist
    [statement_block]]
[case expressionlist
    [statement_block]]
:
[case else
    [statement_block]]
end select
```

Comments

The `select case` statement uses the following arguments:

<i>testexpression</i>	Any numeric or string expression
<i>statement_block</i>	Any group of DCL statements
<i>expressionlist</i>	Any of the following:
	<i>expression</i> [, <i>expression</i>]...
	<i>expression</i> to <i>expression</i>
	<i>is relational_operator expression</i>

If the *testexpression* matches any of the expressions contained in *expressionlist*, the accompanying block of DCL statements is executed.

The resultant type of *expression* must be the same as that of *testexpression*.

Multiple expression ranges can be used within a single `case` clause. For example:

```
case 1 to 10,12,15, is > 40
```

Only the *statement_block* associated with the first matching expression will be executed.

A `select...end select` expression can also be represented with the `if...then` expression. The use of the `select` statement, however, may be more readable.

Example

{bml j_bullet.bmp}Select...Case Example

```
1194 CMD_Select_Case
1195 Select...Case
1196 Select...Case;Select;Case;Flow control
1197 Commands:1205
```

See Also

[{bml j_bullet.bmp}Flow Control](#)

#1198 S1199 K1200 +1201

SelectBox Function

{bmc no_dos.bmp}

Description

Displays a dialog box containing a listbox of strings. If the user selects an item, the index of that item is returned.

Syntax

```
SelectBox(title$,prompt$,items$)
```

Returns

Returns an integer representing the index of the item that the user selected. The first item is 0. The value -1 is returned if the user selects Cancel.

Comments

The `items$` parameter must be a single-dimension array of strings, otherwise a runtime error is generated.

The `title$` parameter specifies the name that appears in the dialog box's caption. The `prompt$` parameter specifies the name that appears immediately above the listbox containing the array items.

The dialog box uses the 8 point Helvetica font.

Example

[{bml j_bullet.bmp}SelectBox Example](#)

See Also

[{bml j_bullet.bmp}Dialog Display](#)

#1202 S1203 K1204 +1205

SelectButton Statement

{bmc no_dos.bmp}

1198 CMD_SelectBox

1199 SelectBox

1200 SelectBox;Select box, defining;Dialog display

1201 Commands:1210

1202 CMD_SelectButton

1203 SelectButton

1204 SelectButton;Button, selecting;Dialog manipulation

1205 Commands:1215

Description

Simulates a mouse click on a button, given the button's name or ID.

Syntax 1

```
SelectButton ButtonName$
```

Syntax 2

```
SelectButton ButtonID%
```

Comments

You can reference the button by `name$` (caption) or `id%`.

A runtime error is generated if a button with the given `name$` or `id%` cannot be found in the active window.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#1206 S1207 K1208 +1209 **SelectComboboxItem Statement** {bmc no_dos.bmp}

Description

Selects an item from a combobox, given the name or ID of the combobox and the name or line number of the item.

Syntax

```
SelectComboboxItem name$ | id%,ItemName$ | ItemNumber%[,isDoubleClick%]
```

Comments

The combobox can be referenced either by `name$` or by `id%`. `Name$` specifies the text that appears in the static control that immediately precedes the combobox control in the window list (or dialog template).

A runtime error is generated if a combobox with the given `name$` or `id%` cannot be found within the active window, or if an item with the given name or line number cannot be found within that combobox.

If the second parameter is either 0 or an empty string (""), all selections will be removed from the combobox.

The optional `isDoubleClick%` parameter specifies if this item is to be selected via a double click or single click. If not specified, then the item is selected using a single click.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

```
1206 CMD_SelectComboboxItem
1207 SelectComboboxItem
1208 SelectComboboxItem;Combo box, selecting item;Dialog manipulation
1209 Commands:1220
```

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#1210 S1211 K1212 +1213

SelectListBoxItem Statement

{bmc no_dos.bmp}

Description

This statement selects an item from a list box, given the name or ID of the listbox and the name or line number of the item.

Syntax

```
SelectListBoxItem name$ | id%,ItemName$ | ItemNumber%[,isDoubleClick%]
```

Comments

The listbox must exist within the current window or dialog, otherwise a runtime error will be generated.

The listbox can be referenced either by `name$` or by `id%`. `Name$` specifies the text that appears in the static control that immediately precedes the listbox control in the window list (or dialog template). A runtime error is generated if a listbox with that name cannot be found within the active window.

If the second parameter is 0 or an empty string (""), then all selections are removed from the listbox. For multi-select listboxes, `SelectListBoxItem` will select additional items (i.e., it will not remove the selection from the currently selected items).

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#1214 S1215 K1216 +1217

SendKeys Statement

{bmc no_dos.bmp}

Description

Sends the specified keys to the active application, optionally waiting for the keys to be processed before continuing.

1210 CMD_SelectListBoxItem
1211 SelectListBoxItem
1212 SelectListBoxItem;List box, selecting item;Dialog manipulation
1213 Commands:1225
1214 CMD_SendKeys
1215 SendKeys
1216 SendKeys;Keys, sending;Keyboard manipulation
1217 Commands:1230

Syntax

```
SendKeys KeyString$ [,wait%]
```

Comments

The format for `KeyString$` is the same as that used for [DoKeys](#).

If `wait` is TRUE (or not specified), the statement waits for the keys to be completely processed before continuing. Otherwise, execution continues immediately.

Example

[{bml j_bullet.bmp}SendKeys Example](#)

See Also

[{bml j_bullet.bmp}Keyboard Manipulation](#)

#1218 S1219 K1220 +1221 SetAttr Statement

Description

Changes the attributes of the specified file to the given value.

Syntax

```
SetAttr filename$,attribute%
```

Comments

A runtime error results if the file cannot be found.

The `attribute%` parameter contains the sum of the following attributes.

Constant	Value	Description
ATTR_NORMAL	0	Read-only, archive, subdirectory, and files with no attributes
ATTR_READONLY	1	Read-only files
ATTR_HIDDEN	2	Hidden files
ATTR_SYSTEM	4	System files
ATTR_VOLUME	8	Volume label
ATTR_DIRECTORY	16	MS-DOS directories
ATTR_ARCHIVE	32	Files changed since last backup
ATTR_NONE	64	Files with no attributes

These attributes are the same as those used by DOS.

Example

[{bml j_bullet.bmp}SetAttr Example](#)

See Also

1218 CMD_SetAttr

1219 SetAttr

1220 SetAttr;Attributes, file;File attributes;File input and output

1221 Commands:1235

{bml j_bullet.bmp}File Input and Output

#1222 S1223 K1224 +1225 **SetCheckbox Statement**

Description

Sets the state of the checkbox with the given name or ID.

Syntax 1

```
SetCheckbox name$,state%
```

Syntax 2

```
SetCheckbox id%,state%
```

Comments

The checkbox can be specified either by its `name$` or using its `id%`. `Name$` is the text of the checkbox label.

If `state` is 1, the box is checked. If `state` is 0, the check is removed. If `state` is 2, then the box is grayed (only applicable for 3-state check boxes).

A runtime error is generated if a check box with the specified name cannot be found in the active window.

This statement has the side-effect of setting the focus to the given checkbox.

Example

{bml j_bullet.bmp}Dialog Examples

See Also

{bml j_bullet.bmp}Dialog Manipulation

#1226 S1227 K1228 +1229 **SetEditText Statement** {bmc no_dos.bmp}

Description

```
1222 CMD_SetCheckbox
1223 SetCheckbox
1224 SetCheckbox;Check box, setting state;Dialog manipulation
1225 Commands:1240
1226 CMD_SetEditText
1227 SetEditText
1228 SetEditText;Text box, setting contents;Dialog manipulation
1229 Commands:1245
```

Sets the content of an edit control, given its name or ID.

Syntax 1

```
SetEditText name$,content$
```

Syntax 2

```
SetEditText id%,content$
```

Comments

The `name$` parameter specifies the text that appears within the static control that immediately precedes the edit control in the window list (or dialog template). Alternatively, the `id%` of the edit box can be specified.

This statement has the side-effect of setting the focus to the given edit control.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Manipulation](#)

#1230 \$1231 K1232 +1233 **SetEnv Function** {bmc no_dos.bmp}

Description

Sets the specified environment variable to the given value for DOS, Windows, or Both.

Syntax

```
SetEnv(var$, value$[, mode$])
```

Comments

When you are running Windows, there are two environments. One environment is used for Windows applications and one for DOS applications.

To control which environment you set variables for, use the `mode$` parameter. The `mode$` can be:

- ENV_DOS, which sets environment variables for DOS applications.
- ENV_WINDOWS, which sets environment variables for Windows.

1230 CMD_SetEnv

1231 SetEnv

1232 SetEnv;Environment variables;DOS environment;Windows environment;Variables, environment;Environment commands

1233 Commands: 1246

-- ENV_BOTH, which sets environment variables for both DOS and Windows.

If it is an invalid value, or not set, the default ENV_WINDOWS is used.

Changes made to Windows environment variables are available to all Windows applications—including those that have already been launched.

The function returns TRUE if successful, or FALSE otherwise. If the functions fails, the environment may be full.

Example

[{bml j_bullet.bmp}SetEnv Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1234 S1235 K1236 +1237 **SetIcon Statement** {bmc no_dos.bmp}

Description

Specifies which icon to show when [ShowIcon](#) is called.

Syntax

```
SetIcon iconfile$ [,icon%]
```

Comments

The `iconfile$` specifies an .EXE, .DLL, or .ICO file. `Icon%` is the zero-based index into the list of icons the file contains. If unspecified, `icon%` is assumed to be 0.

If the icon cannot be found or is not a valid format, or if the value of `icon%` exceeds the number of icons in the file, a runtime error is generated.

This function has no affect on scripts run from the editor or debugger.

See Also

[{bml j_bullet.bmp}Icons](#)

#1238 S1239 K1240 +1241 **SetIconTitle Statement** {bmc no_dos.bmp}

1234 CMD_SetIcon
1235 SetIcon
1236 SetIcon;Icons
1237 Commands: 1247
1238 CMD_SetIconTitle
1239 SetIconTitle
1240 SetIconTitle;Icons
1241 Commands: 1248

Description

Sets the title to be displayed under the icon (if shown) for a compiled script.

Syntax

```
SetIconTitle title$
```

Comments

This function has no affect on scripts run from the editor or debugger.

See Also

[{bml.j_bullet.bmp}Icons](#)

#1242 S1243 K1244 +1245

SetOption Statement

{bmc no_dos.bmp}

Description

Clicks on the specified option button.

Syntax 1

SetOption name\$

Syntax 2

SetOption id%

Comments

The option button can be referenced either by its name\$ (caption) or its id%.

A runtime error is generated if the option button cannot be found within the active window.

Example

{bml j_bullet.bmp}Dialog Examples

See Also

{bml j_bullet.bmp}Dialog Manipulation

#1246 S1247 K1248 +1249

Sgn Function

Description

Returns an integer indicating if a number is less than, greater than, or equal to zero.

Syntax

sgn (number)

Comments

Returns 1 if number is greater than 0.

1242 CMD_SetOption

1243 SetOption

1244 SetOption;Option button, setting;Dialog manipulation

1245 Commands:1250

1246 CMD_Sgn

1247 Sgn

1248 Sgn;Sign;Math commands

1249 Commands:1255

Returns 0 if `number` is equal to 0.

Returns -1 if `number` is less than 0.

The `number` parameter is a numeric expression of any type.

Example

[{bml j_bullet.bmp}Sgn Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

#1250 S1251 K1252 +1253 **Shell Function**

Description

Runs the program (with any specified parameters) contained in `command$`.

Syntax

```
shell(command$ [,WindowStyle%])
```

Returns

If successful, the function returns an integer representing the task ID. For other return codes, see "Errors" below.

Comments

The optional `WindowStyle%` parameter specifies the state of the application window after execution. It can be any of the following values:

- 1 Normal window with focus
- 2 Minimized with focus
- 3 Maximized with focus
- 4 Normal window without focus
- 7 Minimized without focus

An error is generated if unsuccessful. A return value less than or equal to 32 specifies an error.

Errors

This function returns the value 31 if there is no association for the specified file type or if there is no association for the specified action within the file type. The other possible error values are as follows:

1250 CMD_Shell

1251 Shell

1252 Shell;Program, starting;Application, starting;Flow control;Starting applications

1253 Commands:1260

- 0 System was out of memory, executable file was corrupt, or relocations were invalid.
- 2 File was not found.
- 3 Path was not found.
- 5 Attempt was made to dynamically link to a task, or there was a sharing or network-protection error.
- 6 Library required separate data segments for each task.
- 8 There was insufficient memory to start the application.
- 10 Windows version was incorrect.
- 11 Executable file was invalid. Either it was not a Windows application or there was an error in the .EXE image.
- 12 Application was designed for a different operating system.
- 13 Application was designed for MS-DOS 4.0.
- 14 Type of executable file was unknown.
- 15 Attempt was made to load a real-mode application (developed for an earlier version of Windows).
- 16 Attempt was made to load a second instance of an executable file containing multiple data segments that were not marked read-only.
- 19 Attempt was made to load a compressed executable file. The file must be decompressed before it can be loaded.
- 20 Dynamic-link library (DLL) file was invalid. One of the DLLs required to run this application was corrupt.
- 21 Application requires Microsoft Windows 32-bit extensions.

Example

{bml j_bullet.bmp}Shell Example

See Also

{bml j_bullet.bmp}Flow Control

#1254 S1255 K1256 +1257 **ShowIcon Statement** **{bmc no_dos.bmp}**

1254 CMD_ShowIcon
 1255 ShowIcon
 1256 ShowIcon;Icons
 1257 Commands: 1262

Description

Displays an icon on the desktop while the script is running.

Syntax

```
ShowIcon [show%]
```

Comments

Normally, an icon does not show up on the desktop while a compiled script is running.

If `show%` is set to TRUE (the default value), an icon will appear on the desktop when the script is run. The purpose of the icon is to allow the user to stop the script if the user has been permitted this authority (through the [EnableStopScript](#) command).

This function has no affect on scripts run from the editor or debugger.

See Also

[{bml j_bullet.bmp}Icons](#)

#1258 S1259 K1260 +1261

Sin Function

Description

Returns a double-precision number representing the sine of a given angle.

Syntax

```
sin(angle#)
```

Comments

The `angle#` parameter is given in radians.

Example

[{bml j_bullet.bmp}Sin Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

#1262 S1263 K1264 +1265

Sleep Statement

Description

Causes the script to pause for a specified number of milliseconds.

Syntax

```
sleep milliseconds&
```

Comments

While paused, other applications can execute.

Example

[{bml j_bullet.bmp}Sleep Example](#)

See Also

[{bml j_bullet.bmp}Flow Control](#)

1258 CMD_Sin

1259 Sin

1260 Sin;Sine;Math commands

1261 Commands:1265

1262 CMD_Sleep

1263 Sleep

1264 Sleep;Script, pausing;Flow control

1265 Commands:1270

SleepUntil Function {bmc no_dos.bmp}

Description

Pauses the script until the specified time is reached.

Syntax

```
SleepUntil(time$[,showdialog%[,text$[,caption$[,cancel%[,minimizebox]]]]],  
usenetttime%)
```

Comments

The integer TRUE is returned if the pause is completed normally. FALSE is returned if an error is encountered or if the user cancels the operation.

The `time$` value must be in one of the following formats:

12 Hour - "HH:MM xm"

24 Hour - "HH:MM"

HH = hours (may be 1 or 2 digits for 12 hour format, but must be 2 digits for 24 hour)

MM = minutes

xm = am/pm indicator (the 'x' will be replaced by 'a' or 'p')

Colons (":") are expected between the hour and minute portions of the time. In the 12-hour format, a space is expected before the "xm" portion.

Any deviation from this time format will generate a runtime error.

You can set `showdialog%` to TRUE or FALSE. If TRUE, a dialog is displayed with the specified features. `text$` specifies the text to display in the dialog box. The dialog resizes to display the text. If text not specified, no text is used. The `caption$` parameter specifies the caption to be used. If not given, then "Sleeping Until"+`time$` is used.

If `cancel%` is specified, a Cancel button appears in the dialog box. Clicking on the Cancel button stops the sleep and returns FALSE.

If `minimizebox%` is specified, then the dialog can be minimized. It will show a clock icon.

Set `usenetttime%` to TRUE to use network time instead of DOS time.

Example

{bml j_bullet.bmp}SleepUntil Example

See Also

{bml j_bullet.bmp}Flow Control

1266 CMD_SleepUntil

1267 SleepUntil

1268 SleepUntil;Script, pausing;Flow control

1269 Commands: 1272

#1270 S1271 K1272 +1273

Snapshot Statement

{bmc no_dos.bmp}

Description

Takes a snapshot of a particular section of the screen and saves it to the clipboard.

Syntax

snapshot [spec%]

Comments

The `spec` parameter specifies the screen area as follows:

0Entire screen

1Client area of the active application

2Entire window of the active application

3Client area of the active window

4Entire window of the active window

Before the snapshot is taken, each application is updated. This ensures that any application in the middle of drawing will have a chance to finish before the snapshot is taken.

There is a slight delay if the specified window is large.

Example

{bml j_bullet.bmp}Snapshot Example

See Also

{bml j_bullet.bmp}Clipboard Manipulation

#1274 S1275 K1276 +1277

Space\$ Function

Description

Returns a string containing the specified number of spaces.

1270 CMD_Snapshot

1271 Snapshot

1272 Snapshot;Screen, capturing

1273 Commands:1275

1274 CMD_Space_DOLLAR

1275 Space\$

1276 Space\$;String, spaces;Strings

1277 Commands:1280

Syntax

space\$ (NumSpaces%)

Comments

NumSpaces must be between 0 and 32767.

Example

[{bml j_bullet.bmp}Space\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#1278 S1279 K1280 +1281

Sqr Function

Description

Returns a double-precision number representing the square root of a given value.

Syntax

sqr (number#)

Comments

The `number` parameter must be greater than or equal to 0.

Example

[{bml j_bullet.bmp}Sqr Example](#)

See Also

[{bml j_bullet.bmp}Math Statements and Functions](#)

#1282 S1283 K1284 +1285

Stop Statement

Description

Stops execution of a script.

1278 CMD_Sqr

1279 Sqr

1280 Sqr;Square root;Math commands

1281 Commands:1285

1282 CMD_Stop

1283 Stop

1284 Stop;Script, stopping;Flow control

1285 Commands:1290

Syntax

stop

Comments

This command terminates execution of the current script and displays the message: "Stopped at line X", where X is the line number containing the stop statement. All open files are closed. All open DDE channels are closed.

Example

{bml j_bullet.bmp}Stop Example

See Also

{bml j_bullet.bmp}Flow Control

#1286 S1287 K1288 +1289 **StrComp Function**

Description

Compares two strings.

Syntax

```
StrComp(string1$,string2$ [,compare%])
```

Returns

Returns an integer value indicating the result of comparing the two string arguments:

```
0    string1$ = string2
1    string1$ > string2$
```

```
-1   string1$ < string2$
```

Comments

The `StrComp` function compares two strings and returns an integer indicating the result of the comparison. The comparison can be either case-sensitive or case-insensitive, depending on the value of the optional `compare%` parameter:

```
0    Case-sensitive comparison
1    Case-insensitive comparison
```

If `compare%` is not specified, then the comparison is case-sensitive (0).

Example

{bml j_bullet.bmp}StrComp Example

```
1286 CMD_StrComp
1287 StrComp
1288 StrComp,String,comparing,Strings
1289 Commands:1295
```

See Also

[{bml j_bullet.bmp}Strings](#)

#1290 S1291 K1292 +1293 **StringSort Function** {bmc no_dos.bmp}

Description

Sorts a one-dimensional array of strings in ascending order.

Syntax

```
StringSort list$()
```

Comments

If an array of more than one dimension is specified, a runtime error is generated.

See Also

[{bml j_bullet.bmp}Strings](#)

#1294 S1295 K1296 +1297 **Str\$ Function**

Description

Returns a string representation of the given number.

Syntax 1

```
str$(number%)
```

Syntax 2

```
str$(number&)
```

Syntax 3

```
str$(number!)
```

Syntax 4

```
1290 CMD_StringSort
1291 StringSort
1292 StringSort;String, sorting;Strings
1293 Commands: 1297
1294 CMD_Str_DOLLAR
1295 Str$
1296 Str$;String, converting to;Strings
1297 Commands:1300
```

```
str$(number#)
```

Comments

If `number` is negative, then the returned string will contain a leading minus sign.

If `number` is positive, then the returned string will contain a leading space.

Singles are printed using only 7 significant digits. Doubles are printed using 15-16 significant digits.

Example

[{bml j_bullet.bmp}Str\\$ Example](#)

See Also

[{bml j_bullet.bmp}Conversions](#)

[{bml j_bullet.bmp}Strings](#)

#1298 \$1299 K1300 +1301 String\$ Function

Description

Returns a string of `number%` length consisting of a repetition of the specified filler character.

Syntax 1

```
string$(number%,CharCode%)
```

Syntax 2

```
string$(number%,str$)
```

Comments

If `CharCode` is specified, the character with this ASCII value is used as the filler character.

If `str` is specified, then the first character of this string is used as the filler character.

Example

[{bml j_bullet.bmp}String\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

1298 CMD_String_DOLLAR

1299 String\$

1300 String\$;Filler;String, filler characters;Strings

1301 Commands:1305

Sub...End Sub Statement

Description

Declares a subroutine.

Syntax

```
sub name[(parameter [as type]...)]
end sub
```

Comments

Parameters are passed to a subroutine by reference, meaning that any modification to a passed parameter changes that variable in the caller. To avoid this, simply enclose variable names in parentheses, as in the following example function calls:

```
UserSub 10,12,(j)
```

If a subroutine is not to receive a parameter by reference, the optional `byval` keyword can be used:

```
sub Test byval FileName as string
end sub
```

A subroutine terminates when one of the following statements is encountered:

```
end sub
exit sub
```

The name of the subroutine must follow DCL naming conventions. It cannot include type declaration characters.

Subroutines can be recursive.

See Also

[{bml j_bullet.bmp}Procedure Statements](#)

SystemFreeMemory Function

Description

Returns a long integer representing the number of bytes of free memory.

```
1302 CMD_Sub_End_Sub
1303 Sub...End Sub
1304 Sub...End Sub;Sub;EndSub;End;Subroutine;Procedure statements
1305 Commands:1310
1306 CMD_SystemFreeMemory
1307 SystemFreeMemory
1308 SystemFreeMemory;Memory, free;Environment commands
1309 Commands:1315
```


Syntax

`SystemFreeMemory()`

Example

[{bml j_bullet.bmp}SystemFreeMemory Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1310 S1311 K1312 +1313

SystemFreeResources Function

{bmc no_dos.bmp}

Description

Returns an integer representing the percentage of free system resources.

Syntax

`SystemFreeResources()`

Comments

The returned value is between 0 and 100.

Example

[{bml j_bullet.bmp}SystemFreeResources Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1314 S1315 K1316 +1317

SystemMouseTrails Statement

{bmc no_dos.bmp}

Description

Turns on or off mouse trails.

Syntax

`SystemMouseTrails state%`

1310 CMD_SystemFreeResources

1311 SystemFreeResources

1312 SystemFreeResources;Resources, free;Environment commands

1313 Commands:1320

1314 CMD_SystemMouseTrails

1315 SystemMouseTrails

1316 SystemMouseTrails;Mouse trails;Environment commands

1317 Commands:1325

Comments

The setting is saved in the INI file permanently.

This option is only available under Windows 3.1.

Example

[{bml j_bullet.bmp}SystemMouseTrails Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1318 S1319 K1320 +1321 **SystemRestart Statement** {bmc no_dos.bmp}

Description

Restarts Windows, much like the Window Setup program.

Syntax

SystemRestart

Example

[{bml j_bullet.bmp}SystemRestart Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1322 S1323 K1324 +1325 **SystemTotalMemory Function**

Description

Returns a long integer representing the total available free memory in Windows in bytes.

Syntax

SystemTotalMemory()

Example

[{bml j_bullet.bmp}SystemTotalMemory Example](#)

```
1318 CMD_SystemRestart
1319 SystemRestart
1320 SystemRestart;Restarting Windows;Environment commands
1321 Commands:1330
1322 CMD_SystemTotalMemory
1323 SystemTotalMemory
1324 SystemTotalMemory;Memory, total;Environment commands
1325 Commands:1335
```

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1326 S1327 K1328 +1329

SystemWindowsDirectory\$ Function

Description

Returns the directory where Windows is stored, such as "C:\WINDOWS".

Syntax

```
SystemWindowsDirectory$()
```

Example

[{bml j_bullet.bmp}SystemWindowsDirectory\\$ Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

#1330 S1331 K1332 +1333

SystemWindowsVersion\$ Function

[{bmc no_dos.bmp}](#)

Description

Returns the Windows version, such as "3.0" or "3.1".

Syntax

```
SystemWindowsVersion$()
```

Example

[{bml j_bullet.bmp}SystemWindowsVersion\\$ Example](#)

See Also

[{bml j_bullet.bmp}Environment Statements and Functions](#)

1326 CMD_SystemWindowsDirectory_DOLLAR

1327 SystemWindowsDirectory\$

1328 SystemWindowsDirectory\$;Windows directory;Directory, Windows;Environment commands

1329 Commands:1340

1330 CMD_SystemWindowsVersion_DOLLAR

1331 SystemWindowsVersion\$

1332 SystemWindowsVersion\$;Windows version;Version, Windows;Environment commands

1333 Commands:1345

#1334 S1335 K1336 +1337

Tan Function

Description

Returns a double-precision number representing the tangent of the specified angle.

Syntax

`tan(angle#)`

Comments

The `angle` parameter is given in radians.

Example

{bml j_bullet.bmp}Tan Example

See Also

{bml j_bullet.bmp}Math Statements and Functions

#1338 S1339 K1340 +1341

Text Statement

Description

Defines a text control in a dialog template.

Syntax

`Text x%,y%,width%,height%,title$`

Comments

The purpose of `Text` controls is simply to display text.

The `title$` parameter will be truncated if the width of the control is insufficient to hold the entire content.

The `title$` parameter may contain an ampersand character to denote an underlined accelerator, such as "&Font" for Font.

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

1334 CMD_Tan

1335 Tan

1336 Tan;Tangent;Math commands

1337 Commands:1350

1338 CMD_Text

1339 Text

1340 Text;Text, defining;Dialog creation

1341 Commands:1355

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#1342 \$1343 K1344 +1345 **TextBox Statement**

Description

Defines a text-entry field that appears within a dialog box template.

Syntax

`TextBox x as integer,y as integer,width as integer,height as integer,.Field`

Comments

This statement can only appear within a dialog box template definition (BEGIN DIALOG...END DIALOG).

The `x`, `y`, `width`, `height` parameters are specified in dialog coordinates. The `x`, `y` position is relative to the upper left corner of the dialog box.

When the dialog box is created, the content of the `.Field` is used to set the initial content of the textbox. When the `Dialog` statement returns, the `.Field` is used to determine the final content of the textbox. The `.Field` always contains a string.

Example

[{bml j_bullet.bmp}Dialog Examples](#)

See Also

[{bml j_bullet.bmp}Dialog Creation](#)

#1346 \$1347 K1348 +1349 **Time\$ Statement and Function**

1342 CMD_TextBox

1343 TextBox

1344 TextBox;Text box, defining;Dialog creation

1345 Commands:1360

1346 CMD_Time

1347 Time

1348 Time;Time, setting;Time, getting;System time;Time functions

1349 Commands:1365

Description

The `time$` assignment statement sets the system time; the `time$` function returns the system time.

Assignment Syntax

```
time$ = newtime$
```

Comments

This statement sets the system time to the time contained in the specified string.

The format for `newtime$` must be one of the following:

HH

HH:MM

HH:MM:SS

A 24 hour clock is used.

Example

[{bml j _bullet.bmp}Time Statement Example](#)

Function Syntax

```
time$()
```

Returns

Returns the system time as an 8 character string.

Comments

The format of the returned string is HH:MM:SS.

Example

[{bml j _bullet.bmp}Time Function](#)

See Also

[{bml j _bullet.bmp}Date and Time Functions](#)

#1350 \$1351 K1352 +1353 **Timer Function**

Description

Returns a long integer representing the number of seconds that have occurred since midnight.

1350 CMD_Timer

1351 Timer

1352 Timer;Time since midnight;Time functions

1353 Commands:1370

Syntax

timer

Example

[{bml j _bullet.bmp}Timer Example](#)

See Also

[{bml j _bullet.bmp}Date and Time Functions](#)

#1354 \$1355 K1356 +1357 **TimeSerial Function**

Description

Returns a double-precision number representing the given time with today's date. The number is returned in days where Dec 30, 1899 is 0.

Syntax

TimeSerial(hour%,minute%,second%)

Example

[{bml j _bullet.bmp}TimeSerial Example](#)

See Also

[{bml j _bullet.bmp}Date and Time Functions](#)

#1358 \$1359 K1360 +1361 **TimeValue Function**

Description

Returns a double-precision number representing the time contained in the specified string argument.

Syntax

TimeValue(time_string\$)

Comments

1354 CMD_TimeSerial
1355 TimeSerial
1356 TimeSerial;Time, getting;Time functions
1357 Commands:1375
1358 CMD_TimeValue
1359 TimeValue
1360 TimeValue;Time, string;Time functions
1361 Commands:1380

This function interprets the passed `time_string$` parameter looking for a valid time specification. Time specifications vary depending on the international settings contained in the INTL section of the WIN.INI file.

The `time_string$` parameter can contain valid time items separated by time separators such as colon (:) or period (.). The time items must follow the ordering determined by the current time format settings in use by Windows.

Time strings can contain an optional date specification, but this is not used in the formation of the returned value.

If a particular time item is missing, then the missing time items are set to zero. For example, the string "10 pm" would be interpreted as "22:00:00".

Example

[{bml j_bullet.bmp}TimeValue Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#1362 \$1363 K1364 +1365 **Trim\$ Function**

Description

Removes leading and trailing spaces.

Syntax

`trim$(str$)`

Returns

Returns a copy of the passed string expression (`str$`) with leading and trailing spaces removed.

Example

[{bml j_bullet.bmp}Trim\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#1366 \$1367 K1368 +1369 **TRUE**

1362 CMD_Trim_DOLLAR

1363 Trim\$

1364 Trim\$;Leading spaces, removing;Trailing spaces, removing;Strings

1365 Commands:1385

1366 CMD_TRUE

1367 TRUE

1368 TRUE

1369 Commands:1390

Description

Constant.

Returns

-1

Comments

Used in conditionals and boolean expressions.

#1370 S1371 K1372 +1373 **TYPE_DOS**

Description

Constant.

Returns

1

Comment

Used with the AppType function to indicate a DOS application.

#1374 S1375 K1376 +1377 **TYPE_WINDOWS**

Description

Constant.

Syntax

2

Comment

Used with the AppType function to indicate a Windows application.

1370 CMD_TYPE_DOS

1371 TYPE_DOS

1372 TYPE_DOS

1373 Commands:1395

1374 CMD_TYPE_WINDOWS

1375 TYPE_WINDOWS

1376 TYPE_WINDOWS

1377 Commands:1400

#1378 S1379 K1380 +1381

UBound Function

Description

Returns an integer representing the upper bound of the specified dimension of the specified array variable.

Syntax

```
ubound(ArrayVariable() [,dimension%])
```

Comments

The first dimension (1) is assumed if `dimension` is not specified.

Example

[{bml j_bullet.bmp}UBound Example](#)

See Also

[{bml j_bullet.bmp}Arrays](#)

#1382 S1383 K1384 +1385

UCase\$ Function

Description

Returns the uppercase equivalent of the specified string.

Syntax

```
ucase$(str$)
```

Example

[{bml j_bullet.bmp}UCase\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

1378 CMD_UBound

1379 UBound

1380 UBound;Array, finding highest element;Arrays

1381 Commands:1405

1382 CMD_UCase_DOLLAR

1383 UCase\$

1384 UCase\$;Uppercase, converting to;Strings

1385 Commands:1410

Val Function

Description

Converts a given string expression to a double-precision number.

Syntax

```
val (number$)
```

Comments

The `number$` parameter can contain any of the following:

- Leading minus sign (for non hex or octal numbers only)
- Hexadecimal number in the format: &H<hex digits>
- Octal number in the format: &O<octal digits>

- Floating point number, which can contain a decimal point and optional exponent

Spaces, tabs, and linefeeds are ignored.

If `number$` does not contain a number, 0 is returned.

The `val()` function continues to read characters from the string up to the first non-numeric character.

The `val()` function always returns a double-precision floating point value. This value is forced to the data type of the assigned variable.

Example 1

{bml j_bullet.bmp}Val Example

Example 2

The following table shows valid strings and their numeric equivalent:

"1 2 3"	123
"12.3"	12.3
"&HFFFF"	-1
"&O77"	64

" 12.345E-02" .12345

See Also

[Conversions](#)

[{bml j_bullet.bmp}Strings](#)

#1390 S1391 K1392 +1393 **ViewportClear Statement** {bmc no_dos.bmp}

Description

Clears the open viewport window.

Syntax

ViewportClear

Comments

The statement has no effect if no viewport is opened.

Example

[{bml j_bullet.bmp}Viewport Example](#)

See Also

[{bml j_bullet.bmp}Viewport Window Manipulation](#)

#1394 S1395 K1396 +1397 **ViewportClose Statement** {bmc no_dos.bmp}

Description

Closes an open viewport window.

Syntax

ViewportClose

Example

[{bml j_bullet.bmp}Viewport Example](#)

```
1390 CMD_V viewportClear
1391 ViewportClear
1392 ViewportClear;Viewport, clearing;Viewport window manipulation
1393 Commands:1420
1394 CMD_V viewportClose
1395 ViewportClose
1396 ViewportClose;Viewport, closing;Viewport window manipulation
1397 Commands:1425
```

See Also

[{bml j_bullet.bmp}Viewport Window Manipulation](#)

#1398 S1399 K1400 +1401

ViewportOpen Statement

{bmc no_dos.bmp}

Description

Opens a viewport window.

Syntax

```
ViewportOpen [title$ [,x%,y% [,width%,height%]]]
```

Comments

The optional `title$` parameter specifies the text to appear in the viewport's caption. The `x` and `y` parameters specify an optional initial position in twips, and the optional `width` and `height` parameters specify an optional initial width and height for the viewport window.

This statement has no effect if a viewport window is already open.

Combined with the [print](#) statement, a viewport window is a convenient place to output debugging information.

The viewport window is closed when the DCL host application is terminated.

The buffer size for the viewport is 32K. Information from the start of the buffer is removed to make room for additional information being appended to the end of the buffer.

The following keys work within a viewport window:

Up	Scroll up by one line
Down	Scroll down by one line
Home	Scroll to the first line in the viewport window
End	Scroll to the last line in the viewport window
PgUp	Scroll the viewport window down by one page
PgUp	Scroll the viewport window up by one page
Ctrl+PgUp	Scroll the viewport window left by one page

Ctrl+PgDn Scroll the viewport window right by one page

1398 CMD_VisportOpen

1399 ViewportOpen

1400 ViewportOpen;Viewport, opening;Viewport window manipulation

1401 Commands:1430

Only 1 viewport window can be open at any one time. Any scripts with `print` statements will output information into the same viewport window.

Example

[{bml j_bullet.bmp}Viewport Example](#)

See Also

[{bml j_bullet.bmp}Viewport Window Manipulation](#)

#1402 S1403 K1404 +1405 **VK_LBUTTON**

Description

Constant used with the `QueMouse...` commands to represent the left button.

Value

1

See Also

[{bml j_bullet.bmp}Mouse Events](#)

#1406 S1407 K1408 +1409 **VK_RBUTTON**

Description

Constant used with the `QueMouse...` commands to represent the right button.

Value

2

See Also

[{bml j_bullet.bmp}Mouse Events](#)

1402 CMD_VK_LBUTTON

1403 VK_LBUTTON

1404 VK_LBUTTON

1405 Commands:1435

1406 CMD_VK_RBUTTON

1407 VK_RBUTTON

1408 VK_RBUTTON

1409 Commands:1440

#1410 S1411 K1412 +1413

VLine Statement {bmc no_dos.bmp}

Description

Scrolls the window with the focus up or down by the specified number of lines.

Syntax

```
VLine [lines%]
```

Comments

If the `lines` parameter is omitted, then the window is scrolled down by 1 line.

Example

[{bml j_bullet.bmp}VLine Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#1414 S1415 K1416 +1417

VPage Statement {bmc no_dos.bmp}

Description

Scrolls the window with the focus up or down by the specified number of pages.

Syntax

```
VPage [pages%]
```

Comments

If the `pages` parameter is omitted, then the window is scrolled down by 1 page.

Example

[{bml j_bullet.bmp}VPage Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

1410 CMD_VLine

1411 VLine

1412 VLine;Scrolling vertically;Window manipulation

1413 Commands:1445

1414 CMD_VPage

1415 VPage

1416 VPage;Scrolling vertically;Window manipulation

1417 Commands:1450

#1418 S1419 K1420 +1421

VScroll Statement {bmc no_dos.bmp}

Description

Sets the thumb mark on the vertical scroll bar attached to the current window.

Syntax

```
VScroll percentage%
```

Comments

The position is given as a percentage of the total range associated with that scroll bar. For example, if the `percentage%` parameter is 50, then the thumb is positioned in the middle of the scroll bar.

Example

{bml j_bullet.bmp}VScroll Example

See Also

{bml j_bullet.bmp}Window Manipulation

#1422 S1423 K1424 +1425

WaitForTaskCompletion Function {bmc no_dos.bmp}

Description

Waits until the task specified by `taskid%` is exited.

Syntax

```
WaitForTaskCompletion taskid%
```

Comments

The `taskid%` is the return value from the Shell statement.

If `taskid%` is not currently running, control returns immediately.

Example

{bml j_bullet.bmp}WaitForTaskCompletion Example

1418 CMD_VScroll

1419 VScroll

1420 VScroll;Scrolling vertically;Window manipulation

1421 Commands:1455

1422 CMD_WaitForTaskCompletion

1423 WaitForTaskCompletion

1424 WaitForTaskCompletion;Program, running;Application, running;Task, waiting for completion;Flow control

1425 Commands: 1457

See Also

[{bml j_bullet.bmp}Flow Control](#)

#1426 S1427 K1428 +1429 **Weekday Function**

Description

Returns an integer representing the day of the week of the date encoded in the specified `serial` parameter. The value returned is between 1 and 7 inclusive where 1 is Sunday.

Syntax

```
weekday(serial#)
```

Comment

You can obtain the value for the `serial#` parameter by using the [DateSerial](#) or [DateValue](#) command.

Example

[{bml j_bullet.bmp}Weekday Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#1430 S1431 K1432 +1433 **While...Wend Statement**

Description

Repeats a statement or group of statements while a condition is TRUE.

Syntax

```
while <condition>
    [<statement>]
wend
```

Example

[{bml j_bullet.bmp}While...Wend Example](#)

```
1426 CMD_Weekday
1427 Weekday
1428 Weekday;Date functions
1429 Commands:1460
1430 CMD_While_Wend
1431 While...Wend
1432 While...Wend;While;Wend;Flow control
1433 Commands:1465
```

See Also

[{bml j_bullet.bmp}Flow Control](#)

#1434 \$1435 K1436 +1437

WinActivate Statement

[{bmc no_dos.bmp}](#)

Description

Activates the window with the given name or window handle.

Syntax 1

```
WinActivate window_name$
```

Syntax 2

```
WinActivate hWnd%
```

Comments

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

If the `hWnd%` parameter is specified rather than the `window_name$` parameter, then focus is set immediately to the window with that handle.

Windows without captions cannot be activated using this command.

Example

[{bml j_bullet.bmp}WinActivate Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

1434 CMD_WinActivate

1435 WinActivate

1436 WinActivate;Window, activating;Window manipulation

1437 Commands:1470

#1438 S1439 K1440 +1441

WinClose Statement {bmc no_dos.bmp}

Description

Closes the given window.

Syntax

```
WinClose [window_name$]
```

Comments

If no `window_name$` parameter is specified, the window with the focus is closed.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the [AppClose](#) command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window.

Example

[{bml j_bullet.bmp}WinClose Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#1442 S1443 K1444 +1445

WinFind Function {bmc no_dos.bmp}

1438 CMD_WinClose

1439 WinClose

1440 WinClose;Window, closing;Window manipulation

1441 Commands:1475

1442 CMD_WinFind

1443 WinFind

1444 WinFind;Window, finding;Window manipulation

1445 Commands:1480

Description

Returns an integer representing a handle to the specified window.

Syntax

WinFind(name\$)

Comments

The name\$ is specified using the same format as that used by the WinActivate statement.

Example

{bml j_bullet.bmp}WinFind Example

See Also

{bml j_bullet.bmp}Window Manipulation

#1446 S1447 K1448 +1449 **WinList Function** {bmc no_dos.bmp}

Description

Fills the passed array with the handles to all the top-level windows.

Syntax

WinList hWnd\$()

Comments

After calling this function, use the lbound() and ubound() functions to determine the new size of the array.

Example

{bml j_bullet.bmp}WinList Example

See Also

{bml j_bullet.bmp}Window Manipulation

#1450 S1451 K1452 +1453 **WinMaximize Statement** {bmc no_dos.bmp}

1446 CMD_WinList
1447 WinList
1448 WinList;Windows, list;Window manipulation
1449 Commands:1485
1450 CMD_WinMaximize
1451 WinMaximize
1452 WinMaximize;Window, maximizing;Window manipulation
1453 Commands:1490

Description

This command maximizes the specified window.

Syntax

```
WinMaximize [window_name$]
```

Comments

If no `window_name$` parameter is specified, the window with the focus is maximized.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the AppMaximize command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window.

Example

{bml j_bullet.bmp}WinMaximize Example

See Also

{bml j_bullet.bmp}Window Manipulation

#1454 \$1455 K1456 +1457 **WinMinimize Statement** {bmc no_dos.bmp}

Description

Minimizes the specified window.

Syntax

```
WinMinimize [window_name$]
```

Comments

If no `window_name$` parameter is specified, the window with the focus is minimized.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

```
1454 CMD_WinMinimize
1455 WinMinimize
1456 WinMinimize;Window, minimizing;Window manipulation
1457 Commands:1495
```

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the AppMinimize command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window.

Example

{bml j_bullet.bmp}WinMinimize Example

See Also

{bml j_bullet.bmp}Window Manipulation

#1458 S1459 K1460 +1461 **WinMove Statement** {bmc no_dos.bmp}

Description

Moves the specified window.

Syntax

```
WinMove x%,y% [window_name$]
```

Comments

This command moves the given window to the given x,y position. If no `window_name$` parameter is specified, then the window with the focus is moved.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the AppMove command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window. When moving child windows, remember that the `x%` and `y%` parameters are relative to the client area of the parent window.

1458 CMD_WinMove

1459 WinMove

1460 WinMove;Window, moving;Window manipulation

1461 Commands:1500

Example

[{bml j_bullet.bmp}WinMove Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

#1462 S1463 K1464 +1465 **WinRestore Statement** {bmc no_dos.bmp}

Description

Restores the specified window.

Syntax

```
WinRestore [window_name$]
```

Comments

If no `window_name$` parameter is specified, the window with the focus is restored.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the `AppRestore` command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window.

Example

[{bml j_bullet.bmp}WinRestore Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

1462 CMD_WinRestore

1463 WinRestore

1464 WinRestore;Window, restoring;Window manipulation

1465 Commands:1505

WinSize Statement {bmc no_dos.bmp}

Description

Resizes the specified window.

Syntax

```
WinSize width%,height% [window_name$]
```

Comments

This command resizes the given window to the specified width and height. If no `window_name$` parameter is specified, the window with the focus is resized.

The `window_name$` parameter specifies the name that appears on the desired application's title bar. The parameter is not case-sensitive. Optionally, a partial name can be used, such as "Word" for "Microsoft Word".

A hierarchy of windows can be specified by separating each window name with a vertical bar (|), as in the following example:

```
WinActivate "Notepad|Find"
```

In the above example, the top level windows are first searched for a name that includes the string "Notepad". When found, the windows owned by the found window are searched for one that contains the string "Find" in its window title.

This command differs from the [AppSize](#) command in that this command operates on the current window rather than the current top-level window. The current window can be an MDI child window, a popup window, or a top-level window.

Example

[{bml j_bullet.bmp}WinSize Example](#)

See Also

[{bml j_bullet.bmp}Window Manipulation](#)

Word\$ Function

```
1466 CMD_WinSize
1467 WinSize
1468 WinSize;Window, resizing;Window manipulation
1469 Commands:1510
1470 CMD_Word_DOLLAR
1471 Word$
1472 Word$;Words, getting from text;Strings
1473 Commands:1515
```


Description

Extracts words from a text source.

Syntax

```
word$(text$,first%[,last%])
```

Returns

Returns a single word or sequence of words between `first` and `last`.

Comments

The `first` parameter specifies the first word in the sequence to return. If `last` is not specified, then only that word is returned. If `last` is specified, then all words between `first` and `last` will be returned, including all spaces, tabs, and end-of-lines that occur between those words.

Word are separated by spaces, tabs, and end-of-lines.

If `first` is greater than the number of words in `text$`, then an empty string is returned.

If `last` is greater than the number of words in `text$`, then all words from `first` to the end of text are returned.

Example

[{bml j_bullet.bmp}Word\\$ Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

#1474 S1475 K1476 +1477 WordCount Function

Description

Returns an integer representing the number of words in the specified text.

Syntax

```
WordCount(text$)
```

Comments

Word are separated by spaces, tabs, and end-of-lines.

Example

[{bml j_bullet.bmp}WordCount Example](#)

See Also

[{bml j_bullet.bmp}Strings](#)

1474 CMD_WordCount

1475 WordCount

1476 WordCount;Words, count;Strings

1477 Commands:1520

#1478 S1479 K1480 +1481

Write # Statement

Description

Writes a list of expressions to the specified file.

Syntax

```
write [#]filename% [,expressionlist]
```

Comments

The file referenced by `filename` must be opened in either `output` or `append` mode.

The `filename` parameter is a number that is used by DCL to refer to the open file -- the number passed to the `open` statement.

See Also

{bml j_bullet.bmp}File Input and Output

#1482 S1483 K1484 +1485

WriteINI Statement

Description

Writes a new value into an INI file.

Syntax

```
WriteINI section$,ItemName$,value$[,filename$]
```

Comments

The `filename$` parameter, if specified, contains the name of an INI file. If `filename$` is not specified, the WIN.INI file is used.

The `section$` parameter specifies the section which contains the desired variables, such as "windows". Section names are specified without the enclosing brackets.

The `ItemName$` parameter specifies which item from within the given section you want to change. If `ItemName$` is an empty string (""), then the entire section specified by `section$` is deleted.

1478 CMD_Write_POUND

1479 Write #

1480 Write #;File, writing to;File input and output

1481 Commands:1525

1482 CMD_WriteINI

1483 WriteINI

1484 WriteINI;INI file, writing to;Environment commands

1485 Commands:1530

The `value$` parameter specifies the new value for the given item. If `value$` is an empty string (""), then the item specified by `ItemName$` is deleted from the INI file.

Example

{bml j_bullet.bmp}WriteINI Example

See Also

{bml j_bullet.bmp}Environment Statements and Functions

#1486 S1487 K1488 +1489 **WS_MAXIMIZED**

Description

Constant used with the AppSetState and AppGetState statements to indicate a maximized window state.

Value

1

#1490 S1491 K1492 +1493 **WS_MINIMIZED**

Description

Constant used with the AppSetState and AppGetState statements to indicate a minimized window state.

Value

2

#1494 S1495 K1496 +1497 **WS_RESTORED**

Description

Constant used with the AppSetState and AppGetState statements to indicate a normal window state.

1486 CMD_WS_MAXIMIZED
1487 WS_MAXIMIZED
1488 WS_MAXIMIZED
1489 Commands:1535
1490 CMD_WS_MINIMIZED
1491 WS_MINIMIZED
1492 WS_MINIMIZED
1493 Commands:1540
1494 CMD_WS_RESTORED
1495 WS_RESTORED
1496 WS_RESTORED
1497 Commands:1545

Value

3

#1498 S1499 K1500 +1501 **Xor**

Description

Xor operator.

Syntax

`expression1 XOR expression2`

Returns

If both operands are relational, then `XOR` returns the logical exclusive OR of `expression1` and `expression2`. In other words, `XOR` returns `TRUE` only if both operands are not equal.

If both operands are numeric, the result is the bitwise XOR of the arguments.

Notes

If either of the two operands is a floating point number, the two operands are first converted to longs, then a bitwise XOR is performed.

Example

[{bml j_bullet.bmp}Xor Example](#)

See Also

[{bml j_bullet.bmp}Operators](#)

#1502 S1503 K1504 +1505 **Year Function**

Description

Returns an integer representing the year of the date encoded in the specified `serial` parameter. The value returned is between 100 and 9999 inclusive.

1498 `CMD_Xor`

1499 `Xor`

1500 `Xor`;Xor operator;Operators

1501 `Commands:1550`

1502 `CMD_Year`

1503 `Year`

1504 `Year`;Date functions

1505 `Commands:1555`

Syntax

`year(serial#)`

Comment

You can obtain the value for the `serial#` parameter by using the [DateSerial](#) or [DateValue](#) command.

Example

[{bml j_bullet.bmp}Year Example](#)

See Also

[{bml j_bullet.bmp}Date and Time Functions](#)

#1506 This is an ID assigned to the control by the program. It can be obtained through Windows diagnostic utilities, such as Spy.