

# USER MANUAL

## Quick3270 Macro Language Reference

© <2026> <DN-Computing>

# Contents

<b>Chapter 1</b>	<b>1 Macro Language Reference Manual</b>
<b>Chapter 2</b>	<b>2 Preamble</b>
<b>Chapter 3</b>	<b>3 Introduction</b>
<b>Chapter 4</b>	<b>4 Variables and Data Types</b>
	4 Data Types
	5 Variable identifiers
	5 Label identifiers
	5 Declaring Variable
	5 Dim Statement
	6 Const Statement.
	6 Build-In Symbolic Constants.
<b>Chapter 5</b>	<b>6 Data Type Functions</b>
	6 Is
	6 IsEmpty
	7 IsNull
	7 IsNumeric
	7 VarType
<b>Chapter 6</b>	<b>8 Control Commands</b>
	8 Call Statement
	8 Do While...Loop
	8 End
	8 Exit
	9 Exit Session
	9 Exit Sub
	9 For...Next Statement
	9 GoSub
	10 GoTo

# Contents

	10	If...Then...Else Statement
	11	Quit
	11	Return
	11	Stop
	11	Sub...End Sub Statement
	12	While...Wend Statement
	12	Whith...End With Statement
<b>Chapter 7</b>	<b>13</b>	<b>String functions</b>
	13	Chr or Chr\$
	13	CStr
	13	InStr
	14	LCase or LCase\$
	14	Left or Left\$
	14	Len
	15	Mid or Mid\$
	15	Replace or Replace\$
	15	Right or Right\$
	16	Space or Space\$
	16	Str or Str\$
	16	StrComp
	17	Trim
	17	UCase or UCase\$
	17	Val
<b>Chapter 8</b>	<b>17</b>	<b>File functions and statements</b>
	17	Open
	18	Close
	18	LineInput or Line Input
	19	Print #
	19	Write #

# Contents

	19	Seek #
	20	Get #
	20	Put #
	20	EOF
	21	FreeFile
	21	Loc
	21	LOF
	21	Kill
<b>Chapter 9</b>	<b>21</b>	<b>Date / Time functions</b>
	22	Build-In Symbolic Constants.
	22	Date\$
	22	Date
	23	Time\$
	23	Time
	23	Day
	23	Month
	24	Year
	24	DateAdd
	25	DateSerial
	25	DateValue
	26	Format
	27	Now
	28	TimeSerial
	28	TimeValue
<b>Chapter 10</b>	<b>28</b>	<b>COM functions</b>
	28	CreateObject
	29	GetObject
<b>Chapter 11</b>	<b>30</b>	<b>Miscellaneous functions and statements</b>
	30	Beep
	30	Decrypt or Decrypt\$

# Contents

	30	Encrypt or Encrypt\$
	30	Environ or Environ\$
	31	GetOpenFileName
	31	GetSpecialFolderLocation
	31	InputDialog
	32	MsgBox
	34	PasswordBox
	34	Rem
	34	Shell
<b>Chapter 12</b>	<b>35</b>	<b>Quick3270 functions and statements</b>
	35	ConvertCol
	36	ConvertRow
	36	EditCopy
	36	EditCopyAdd
	36	EditPaste
	37	EditSelectAll
	37	GetCol
	37	GetCols
	37	GetRow
	38	GetRows
	38	GetRowCount
	38	GetColor
	39	GetIpAddress
	39	GetIpAddressV6
	39	GetMacroKeystroke
	40	GetOIA
	41	GetString
	41	GetText
	41	GetTextRect

# Contents

42	GetTimeoutValue
42	GetVisible
42	GetWindowTitle
42	MoveTo
43	MoveRelative
43	OpenPrinter
43	ClosePrinter
43	PrintScreenFontSize
44	PrintScreen
44	PutString
45	ReceiveFile
45	Search
46	SearchPS
46	SendDarkKeys
46	SendFile
47	SendKeys
49	SetText
50	SetTimeoutValue
50	SetVisible
50	SetWindowTitle
51	Updated
51	WaitForAttrib
51	WaitForCursor
52	WaitForCursorMove
52	WaitForKbdUnlock
53	WaitForString
53	WaitHostQuiet
<b>Chapter 13</b>	<b>53 Support for IBM® Personal Communications™ macros</b>
54	autECLSession

# Contents

<b>54</b>	<code>autECLSession.Name</code>
<b>55</b>	<code>autECLSession.SetConnectionByName</code>
<b>55</b>	<code>autECLSession.StopCommunication</code>
<b>55</b>	<code>autECLSession.StartCommunication</code>
<b>55</b>	<code>autECLPS</code>
<b>55</b>	<code>autECLOIA</code>
<b>55</b>	<code>autECLWinMetrics</code>
<b>56</b>	<code>autECLXfer</code>
<b>56</b>	<code>autECLPS</code>
<b>56</b>	<code>autECLPS.NumRows</code>
<b>56</b>	<code>autECLPS.NumCols</code>
<b>57</b>	<code>autECLPS.CursorPosRow</code>
<b>57</b>	<code>autECLPS.CursorPosCol</code>
<b>57</b>	<code>autECLPS.ConnType</code>
<b>57</b>	<code>autECLPS.SetCursorPos</code>
<b>58</b>	<code>autECLPS.SendKeys</code>
<b>60</b>	<code>autECLPS.SearchText</code>
<b>61</b>	<code>autECLPS.GetText</code>
<b>61</b>	<code>autECLPS.SetText</code>
<b>61</b>	<code>autECLPS.GetTextRect</code>
<b>62</b>	<code>autECLPS.Wait</code>
<b>62</b>	<code>autECLPS.WaitForCursor</code>
<b>63</b>	<code>autECLPS.WaitWhileCursor</code>
<b>63</b>	<code>autECLPS.WaitForString</code>
<b>64</b>	<code>autECLPS.WaitForAttrib</code>
<b>64</b>	<code>autECLFieldList</code>
<b>65</b>	<code>autECLXfer</code>
<b>65</b>	<code>autECLXfer.ReceiveFile</code>
<b>65</b>	<code>autECLXfer.SendFile</code>
<b>66</b>	<code>autECLWinMetrics</code>
<b>66</b>	<code>autECLWinMetrics.WindowTitle</code>
<b>66</b>	<code>autECLWinMetrics.Visible</code>
<b>67</b>	<code>autECLWinMetrics.Minimized</code>
<b>67</b>	<code>autECLWinMetrics.Maximized</code>
<b>67</b>	<code>autECLWinMetrics.Restored</code>
<b>67</b>	<code>autECLWinMetrics.Name</code>
<b>67</b>	<code>autECLOIA</code>
<b>67</b>	<code>autECLOIA.WaitForInputReady</code>

# Contents

	68	autECLOIA.WaitForAppAvailable
	68	autECLOIA.WaitForSystemAvailable
	68	autECLFieldList
	69	autECLFieldList.Refresh
	69	autECLFieldList.Count
	69	autECLFieldList.FindFieldByRowCol
	70	autECLFieldList.FindFieldByText
	70	autECLFieldList(FieldIndex).Display
	70	autECLFieldList(FieldIndex).GetText
	71	autECLFieldList(FieldIndex).SetText
	71	autECLFieldList(FieldIndex).StartCol
	71	autECLFieldList(FieldIndex).StartRow
	71	autECLFieldList(FieldIndex).EndCol
	71	autECLFieldList(FieldIndex).EndRow
	72	autECLFieldList(FieldIndex).Length
	72	autECLFieldList(FieldIndex).Modified
	72	autECLFieldList(FieldIndex).Protected
	72	autECLFieldList(FieldIndex).HighIntensity
	72	autECLFieldList(FieldIndex).PenDetectable
	72	autECLFieldList(FieldIndex).Numeric
	72	autSystem.Inputnd
<b>Chapter 14</b>	<b>73</b>	<b>Support for Attachmate's Extra! X-treme™ macros</b>
<b>Chapter 15</b>	<b>74</b>	<b>Support for Micro Focus Chameleon™/Rumba™ macros</b>
	74	EMReadScreen
	74	EMSendKey
	74	EMWaitCursor
	75	FileWrite
	75	FileAppend
<b>Chapter 16</b>	<b>75</b>	<b>Support for TN3270 Plus™ macros</b>
	76	Chr
	77	DDEExecute
	77	DDEInitiate
	78	DDEPoke
	78	DDERequest
	79	DDETerminate

# Contents

	79	Command
	79	CursorTo
	80	Key
	80	MsgBox
	81	Replace
	82	Run
	82	Type
	83	Wait
	83	WaitFor
<b>Chapter 17</b>	<b>84</b>	<b>Macro Samples</b>
	84	Select an Excel file using GetOpenFileName
	84	Creating a Reference to an Excel Object
	84	File I/O
	84	Connecting to a data source and executing SQL Queries
	85	Find Values from screen and use the WScript object
	85	Simple logon script
	86	Using LDAP to retrieve the user name and password and start logon script
	87	IBM Personal CommunicationsVBScript sample supported by Quick3270
	88	IBM Personal CommunicationsMacro sample supported by Quick3270
	89	Attachmate's Extra! macro sample supported by Quick3270



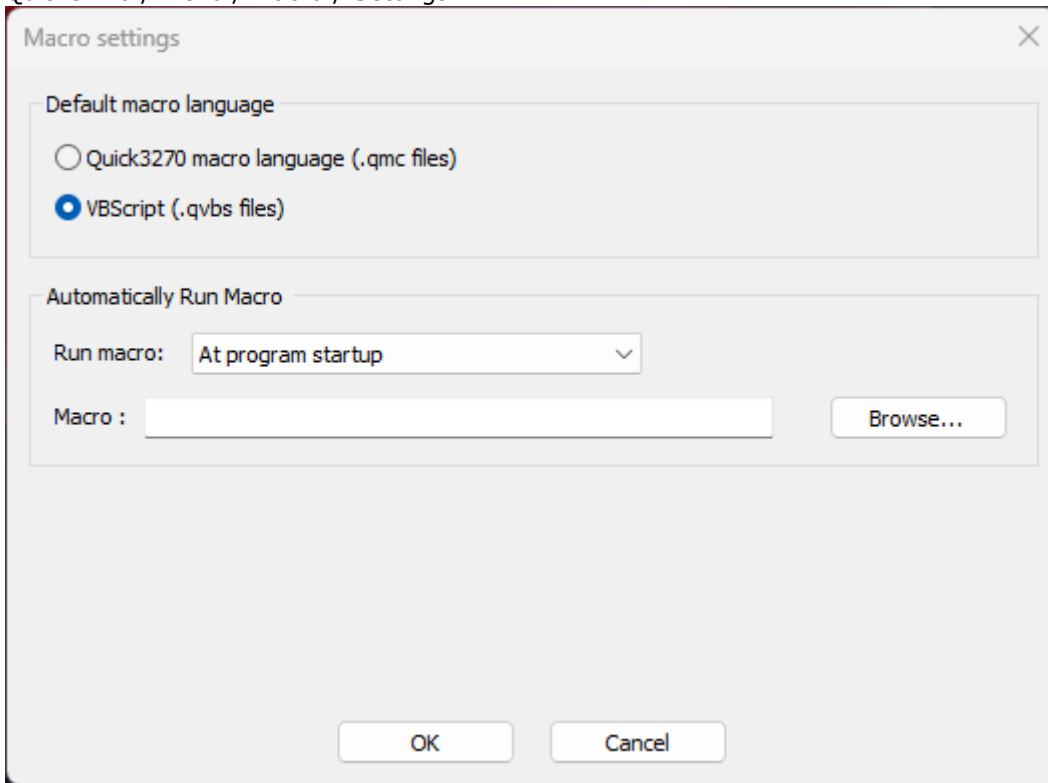
74, rue Verte  
34980 Saint-Gély-du-Fesc – France  
Phone: +33 675 369 257

Email: [contact@dn-computing.com](mailto:contact@dn-computing.com)  
www: <http://www.dn-computing.com/>

The Quick3270 software supports two macro languages: The internal Quick3270 macro interpreter and VBScript.

Since version 5.40, VBScript is the default macro language.

To select the default macro language :  
Quick3270 / Menu / Macro / Settings...



1. The internal Quick3270 macro interpreter, with a syntax similar to the **VBA language**. The syntax, keywords, and functions of this language **are described in this document**. This language is used for Quick3270 macros (.qmc files), as well as macro files from the following software: Extra! X-treme™, Rumba™, HobLink™, QWS3270™, and TN3270 Plus™.
2. Since version 5.30 of Quick3270, the Windows **VBScript** interpreter has been added as an optional macro language (.qvbs files). The VBScript language is more complete and provides more features than the Quick3270 interpreter. **VBScript itself is not described in this document**, but documentation can easily be found online (for example on the VbsEdit website).

However, the emulator specific functions described in the chapter "[Quick3270 Functions and Statements](#)" are also available in VBScript. These functions and statements can be accessed directly or through a virtual object named Emu.

The Emu object was created to make macro code easier to read.

For example, in VBScript you can use:

```
SendKeys "<Enter>"  
or  
Emu.SendKeys "<Enter>"
```

As VBScript is also the interpreter used by IBM Personal Communications™, this option improves

compatibility of Personal Communications™ macros (.mac files) in Quick3270.

The VBScript interpreter is also used for macros from the Reflection™ software (with the addition of the RIBM object).

\* Since version 6.00, Quick3270 includes the **VbsEdit**™ macro editor, which supports syntax highlighting and debugging.

## Note:

Microsoft has announced the end of life of VBScript. Around 2027, VBScript will be disabled by default. At a later date (not yet announced), VBScript will be permanently removed.

Our current project is to extend the Quick3270 macro interpreter so that VBScript syntax is also supported. In this way, when VBScript is no longer available in Windows, the interpreter built into Quick3270 will automatically and transparently execute these macros

A macro is a series of recorded or coded commands you can run to complete tasks more quickly. Macros allow you to condense complex Quick3270 commands into a single command or keystroke.

There are two ways to create a macro with Quick3270. You can record the keystrokes you enter as you're working, and/or you can use an editor in conjunction with the Quick3270 Macro Language to code your macro.

## To automate a series of tasks.

Instead of entering your user ID, password, and pressing the ENTER key each time you log on to your mainframe account, you can automate the process with a macro.

## To automate an often-used sequence of commands and actions.

If you routinely open a mainframe application as part of your log on sequence, you can create a macro to open the application for you. If you've worked with macros or done some Basic programming before, much of the language information will be familiar to you.

## Record a Keystroke Sequence

If you regularly do the same things when you work with a host system, it is convenient to record the keystrokes you make and have Quick3270 play them back if you want to do the same job again. All your keystrokes can be saved to a file; when you play the file back (Run), everything that happened will be reproduced.

## Macro Language

The macro language allows you to create scripts for use with Quick3270. Presented here are some script functions that are recognized by Quick3270. Creating or modifying macros should be done by those who have programming experience and are knowledgeable in VBA or VBScript.

## Debugging a Macro

You can add the *debug* instruction in your macro code to display a dialog box that allows checking the list of variables (value and type).

This debug dialog box allows even to run the macro line per line.

```
haystack = "The quick brown fox"

Result = Replace(haystack, "brown", "orange")
'returns "The quick orange fox"

tempRBA = "1"
```

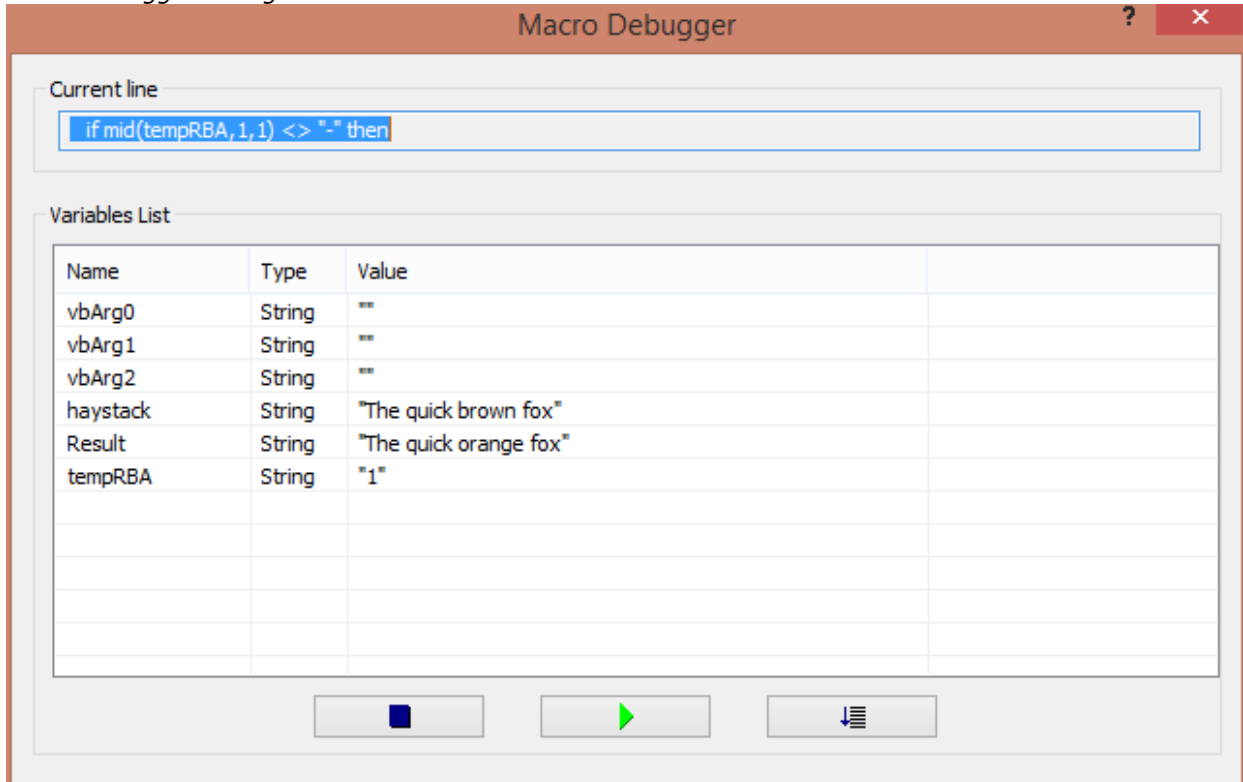
```

debug

if mid(tempRBA,1,1) <> "1" then
    result = MsgBox("Macro Error", vbOK, "Continue")
endif

```

Macro debugger Dialog box



In Quick3270 Basic, you use variables and constants to store values. Variables can contain data represented by any supported data type.

## Data Types

The following table lists the fundamental data types that the Macro language supports.

Data type	Description	Range
Integer	4-byte integer	– 2,147,483,648 to 2,147,483,647.
String	String of characters	Zero to 65535 characters
Boolean	2 bytes	True or False
Date	8 bytes	January 1, 100 to December 31, 9999
Double	8 bytes	Signed IEEE 64-bit (8-byte) double-precision floating-point numbers
Object	4 bytes	Any object reference.

**Note:** Quick3270 macro language supports only single dimension arrays.

## Variable identifiers

The first character must be an alphabetic (A-Z, a-z) or an underscore character "\_". Subsequent characters can be alphabetic, underscore or numeric (0-9). Variable identifiers are not case sensitive. The maximum length is 32.

## Label identifiers

Label identifiers consist of alphabetic, underscore or numeric characters, and are not case sensitive. The maximum length is 32.

```
goto label
```

```
...
```

```
label:
```

## Declaring Variable

A variable is created the first time a value is assigned to it, as shown in the following samples, or with the *Dim* statement.

```
MyBoolean = True
MyInteger = 459
MyString = "Hello world"
MyObject = CreateObject("Excel.Application")
```

## Naming Rules of variables

- A variable name must begin with an alphabet.
- It cannot be more than 255 characters.
- The variable name must not contain any special character like %, &, !, #, @ or \$.
- And finally, it has to be unique within the same scope.

## Type Character

In addition to specifying a data type in a declaration statement, you can force the data type of some programming elements with a *type character*.

The following table shows the available identifier type characters with examples of usage.

Identifier type character	Data type	Example
%	Integer	Dim L%
#	Double	Dim X#
\$	String	Dim V\$ = "String"

## Dim Statement

### Description

Declares variables and allocates storage space

### Syntax

**Dim** *varname* **As** type

```
Dim MyVar As Integer
Dim MyString As String
Dim MyDate As Date
Dim MyObject As Object
```

## Const Statement.

### Description

Declares constants for use in place of literal values.

### Syntax

**Const** *constname* = *expression*

```
Const MyVar = 459  
Const MyString = "Hello world"
```

## Build-In Symbolic Constants.

The macro language provides a number of predefined constants that can be used anywhere in your code in place of the actual values

Constant	Equivalent	Description
vbCrLf	Chr(13) + Chr(10)	Carriage return–linefeed combination.
vbCr	Chr(13)	Carriage return character
vbLf	Chr(10)	Linefeed character.
vbTab	Chr(9)	Tab character.
vbFormFeed	Chr(12)	Form feed character
vbBack	Chr(8)	Form feed character.

Other constants, specific to **MsgBox**, and Date variables are defined. You can get the list in the description of the **MsgBox** function

## Is

### Description

Compares two object expressions and returns a value that indicates whether the expressions refer to the same object.

### Syntax

If (*expression2* **Is** *expression2*)

### Remarks

Also use the **Is** operator to test whether an object variable has been **Set** to **Nothing**.

## IsEmpty

### Description

Returns a Boolean value indicating whether a variable has been initialized.

### Syntax

## **IsEmpty**(*expression*)

The *expression* argument can be any expression.

### Remarks

**IsEmpty** returns **True** if expression is uninitialized, or explicitly set to Empty; otherwise, it returns **False**. **False** is always returned if expression contains more than one variable.

## IsNull

### Description

Returns a Boolean value that indicates whether an expression contains no valid data (Null).

### Syntax

#### **IsNull**(*expression*)

The *expression* argument can be any expression.

### Remarks

**IsNull** returns **True** if expression is Null, that is, it contains no valid data; otherwise, **IsNull** returns **False**. If expression consists of more than one variable, Null in any constituent variable causes True to be returned for the entire expression.

## IsNumeric

### Description

Returns a **Boolean** value indicating whether an expression can be evaluated as a number.

### Syntax

#### **IsNumeric**(*expression*)

The *expression* argument can be any expression.

### Remarks

**IsNumeric** returns **True** if the entire expression is recognized as a number; otherwise, it returns **False**.

## VarType

### Description

Returns a value indicating the subtype of a variable.

### Syntax

#### **VarType**(*VarName*)

The *varname* argument can be any variable.

### Return Values

The **VarType** function returns the following values:

Constant	Value	Description
vbEmpty	0	Empty (uninitialized)

vbInteger	3	Integer
vbString	8	String
vbObject	9	Automation object
vbBoolean	11	Boolean

## Call Statement

### Description

Transfers control to another macro file.  
 When the **Exit** command in the second macro file is encountered, execution returns to the command that follows **Call** statement  
 Up to 10 levels of macro files can be called

### Syntax

**Call** *MacroFilename*  
**Do While...Loop**

### Description

Repeats a block of statements while a **Boolean** condition is **True** or until the condition becomes **True**.

### Syntax

**Do While** *condition*  
*Version [statements]*  
**Loop**

The Do While...Loop statement syntax has these parts:

Part	Description
condition	Numeric or string expression that evaluates to True or False. If condition is Null, condition is treated as False.
statements	One or more statements executed while condition is True.

End

### Description

Terminates an application.

### Syntax

**End**  
**Exit**

### Description

Immediately exits the execution of the current macro file and returns to the calling macro file if any.

### Syntax

## Exit Exit Session

### Description

Immediately stops the execution of the macro and exits the Quick3270 program, with no user prompt.

### Syntax

**Exit Session**  
Exit Sub

### Description

Immediately exits the Sub procedure in which it appears. The Exit Sub statement is equivalent to the Return statement.

### Syntax

**Exit Sub**  
For...Next Statement

### Description

Repeats a group of statements a specified number of times.

### Syntax

**For** *counter = start To end* [**Step** *step*]  
    [*statements*]  
    [**Exit For**]  
    [*statements*]

### Next

The **For...Next** statement syntax has these parts:

Part	Description
counter	Numeric variable used as a loop counter.
start	Initial value of counter.
end	Final value of counter.
step	Amount counter is changed each time through the loop. If not specified, step defaults to one.
statements	One or more statements between For and Next that are executed the specified number of times.
<b>Exit For</b>	Optional. Transfers control out of the <b>For</b> loop.
<b>Next</b>	Required. Terminates the definition of the <b>For</b> loop.

## GoSub

### Description

Subroutine call. When you use the **GoSub** command, execution is transferred to the specified label and continues until a **Return/End Sub** command is encountered, at which point execution returns to the command that follows **GoSub**.

## Syntax

**GoSub** *label*  
or  
**GoSub** *Subroutine name*

where *label* is defined by the **Label** identifier and Subroutine name defined by a **Sub** procedure. In case of a Sub procedure, the **GoSub** statement is optional. A Sub procedure can be called directly by its procedure name.

```
GosubDemo ' or.Gosub GosubDemo
```

```
Sub GosubDemo ()
...
End Sub ' following the GoSub statement.
```

## GoTo

### Description

Branches unconditionally to a specified label.

### Syntax

**GoTo** *label*

where *label* is defined by the Label command.

## If...Then...Else Statement

### Description

Conditionally executes a group of statements, depending on the value of an expression.

### Syntax

**If** *condition* **Then** *statements* [**Else** *elstatements* ]

Or, you can use the block form syntax:

```
If condition Then
  [statements]
[ElseIf condition-n Then
  [elseifstatements]] . . .
[Else
  [elstatements]]
Endif
```

The **If...Then...Else** statement syntax has these parts:

Part	Description
condition	One or more of the following two types of expressions: A numeric or string expression that evaluates to True or False. If condition is Null, condition is treated as False.

statements	One or more statements separated by colons; executed if condition is True.
condition-n	Same as condition.
elseifstatements	One or more statements executed if the associated condition-n is True.
elsestatements	One or more statements executed if no previous condition or condition-n expression is True.

## Remarks

You can use the single-line form (first syntax) for short, simple tests. However, the block form (second syntax) provides more structure and flexibility than the single-line form and is usually easier to read, maintain, and debug.

## Quit

### Description

Terminates a macro (same as Exit).

### Syntax

#### Quit

Return

### Description

Returns to the command that follows the last executed **GoSub** command.

### Syntax

#### Return

## Stop

### Description

Terminates a macro (same as Exit).

**Stop** statements can be placed anywhere in a program to suspend its execution.

### Syntax

#### Stop

Sub...End Sub Statement

### Description

Executes a series of statements within the body of the Sub procedure.

### Syntax

**Sub** *name* [ (*parameter*) ]  
[*statements*]

#### End Sub

The Sub...End Sub statement syntax has these parts:

Part	Description
------	-------------

name	Name of the Sub; follows standard variable naming conventions..
statements	Any group of statements to be executed within the body of the Sub procedure.
parameter	Optional. Variable name representing the parameters of this procedure. <b>Added version 5.21</b>

## Remarks

If condition is **True**, all statements in statements are executed until the **Wend** statement is encountered. Control then returns to the **While** statement and condition is again checked. If condition is still **True**, the process is repeated. If it is not **True**, execution resumes with the statement following the **Wend** statement.

**While...Wend** loops can be nested to any level. Each **Wend** matches the most recent **While**.

## While...Wend Statement

### Description

Executes a series of statements as long as a given condition is True.

### Syntax

**While** *condition*

*Version* [*statements*]

**Wend**

The While...Wend statement syntax has these parts:

Part	Description
condition	Numeric or string expression that evaluates to True or False. If condition is Null, condition is treated as False.
statements	One or more statements executed while condition is True.

## Remarks

If condition is **True**, all statements in statements are executed until the **Wend** statement is encountered. Control then returns to the **While** statement and condition is again checked. If condition is still **True**, the process is repeated. If it is not **True**, execution resumes with the statement following the **Wend** statement.

**While...Wend** loops can be nested to any level. Each **Wend** matches the most recent **While**.

## Whith...End With Statement

### Description

Executes a series of statements making repeated reference to a single object or structure.

### Syntax

**Whith** *object*

[*statements*]

**End With**

The While...Wend statement syntax has these parts:

Part	Description
------	-------------

object	Required. Variable or expression. Can evaluate to any data type, including elementary types.
statements	Optional. One or more statements between With and End With that run on object.

## Remarks

**With...End With** allows you to perform a series of statements on a specified object without requalifying the name of the object.

**Nesting Structures.** Quick3270 don't allow to nest **With...End With** structures.

## Chr or Chr\$

## Description

Returns the character associated with the specified ANSI character code.

## Syntax

**Chr**(*charcode*)

The *charcode* argument is a number that identifies a character

## Remarks

Numbers from 0 to 31 are the same as standard, nonprintable ASCII codes. For example, **Chr**(10) returns a linefeed character.

The following example uses the **Chr** function to return the character associated with the specified character code:

```
Dim MyChar
MyChar = Chr(65)      ' Returns A.
MyChar = Chr(97)     ' Returns a.
MyChar = Chr(62)     ' Returns >.
MyChar = Chr(37)     ' Returns %.
```

## CStr

## Description

The following table describes the return values for **CStr** for different data types of expression

If expression type is	CStr returns
Boolean Data Type	A string containing "True" or "False".
Date Data Type	A string containing a <b>Date</b> value (date and time) in the short date format of your system.
Numeric Data Types	A string representing the number.

## InStr

## Description

Returns the position of the first occurrence of one string within another.

## Syntax

# String functions

**InStr**(*string1*, *string2*)

## Parameters

Part	Description
string1	Required. String expression being searched.
string2	Required. String expression searched for.

## Return Values

The **InStr** function returns the following values:

If	InStr returns
string2 is not found	0
string2 is found within string1	Position at which match is found

## LCase or LCase\$

### Description

Returns a string converted to lowercase.

### Syntax

**LCase**(*string*) or **LCase\$**(*string*)

## Left or Left\$

### Description

Returns a String containing a specified number of characters from the left side of a string.

### Syntax

**Left**(*string*, *length*)

## Parameters

Part	Description
string	Required. String expression from which the leftmost characters are returned.
length	Required. Numeric expression indicating how many characters to return.

## Len

### Description

Returns the number of characters in a string.

### Syntax

**Len**(*string*)

## Mid or Mid\$

### Description

Returns a specified number of characters from a string.

### Syntax

**Mid**(*string*, *start*[, *length*])

### Parameters

Part		Description
string	String	String expression from which characters are returned.
start	Integer	Character position in string at which the part to be taken begins..
length	Integer	Number of characters to return. If omitted or if there are fewer than length characters in the text (including the character at start), all characters from the start position to the end of the string are returned.

## Replace or Replace\$

### Description

Returns a string in which a specified substring has been replaced with another substring.

### Syntax

**Replace**(*expression*, *find*, *replacement*)

### Parameters

Part	Description
expression	Required. String expression containing substring to replace.
find	Required. Substring being searched for.
replacement	Required. Replacement substring

### Example

```
expression = "The quick brown fox"  
Result = Replace(expression, "brown", "orange")
```

'returns "The quick orange fox"

## Right or Right\$

### Description

Returns a String containing a specified number of characters from the right side of a string.

### Syntax

**Right**(*string*, *length*)

## Parameters

Part	Description
string	Required. String expression from which the rightmost characters are returned.
length	Required. Numeric expression indicating how many characters to return.

## Space or Space\$

### Description

Returns a string consisting of the specified number of spaces.

### Syntax

**Space**(*number*) or **Space\$**(*number*)

The number argument is the number of spaces you want in the string.

### Remarks

The following example uses the Space function to return a string consisting of a specified number of spaces:

```
Dim MyString
MyString = Space(10) ' Returns a string with 10 spaces.
MyString = "Hello" & Space(10) & "World" ' Insert 10 spaces between two
strings.
```

## Str or Str\$

### Description

Returns a String representation of a number.

### Syntax

**Str**(*number*)

The required number argument is a Integer containing any valid numeric expression.

### Remarks

When numbers are converted to strings, a leading space is always reserved for the sign of number. If number is positive, the returned string contains a leading space and the plus sign is implied.

## StrComp

### Description

Returns a value indicating the result of a string comparison.

### Syntax

**StrComp**(*string1*, *string2*)

## Parameters

# String functions

Part	Description
string1	Required. Any valid string expression.
string2	Required. Any valid string expression.

## Return Values

The **StrComp** function has the following return values:

If	StrComp returns
string1 is less than string2	-1
string1 is equal to string2	0
string1 is greater than string2	1

## Trim

### Description

Returns a string that contains a copy of a specified string without leading or trailing spaces

### Syntax

**Trim**(*expression*)  
**UCase** or **UCase\$**

### Description

Returns a string converted to uppercase.

### Syntax

**UCase**(*string*) or **UCase\$**(*string*)  
**Val**

### Description

Returns the numbers contained in a string as a numeric value of appropriate type.

### Syntax

**Val**(*string*)

The required string argument is any valid string expression.

### Remarks

The **Val** function stops reading the string at the first character it can't recognize as part of a number. Symbols and characters that are often considered parts of numeric values, such as dollar signs and commas, are not recognized. However, the function recognizes the radix &H (for hexadecimal).

## Open

### Description

Enables input/output (I/O) to a file.

## Syntax

**Open** *pathname* **For mode** **As** [#]*filename* [**Len=***reclength*]

The Open statement syntax has these parts:

Part	Type	Description
Pathname	String	Required. String expression that specifies a file name - may include directory or folder, and drive.
Mode		Required. Keyword specifying the file mode: Append, Binary, Input, Output, or Random. If unspecified, the file is opened for Random access.
Filename	Integer	Required. A valid file number in the range 1 to 10, inclusive. Use the FreeFile function to obtain the next available file number.
Reclength	Integer	Optional. Number less than or equal to 32,767 (bytes). For files opened for random access, this value is the record length. For sequential files, this value is the number of characters buffered.

## Remarks

You must open a file before any I/O operation can be performed on it. Open allocates a buffer for I/O to the file and determines the mode of access to use with the buffer.

## Close

## Description

Concludes input/output (I/O) to a file opened using the **Open** statement.

## Syntax

**Close** *filename*

The *filename* is any valid file number:

## Remarks

When the **Close** statement is executed, the association of a file with its file number ends.

## LineInput or Line Input

## Description

Reads a single line from an open sequential file and assigns it to a String variable.

## Syntax

**LineInput** #*filename*, *varname*

The **LineInput** # statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
varname	String	Required. Valid String variable name.

## Remarks

The **LineInput #** statement reads from a file one character at a time until it encounters a carriage return (Chr(13)) or carriage return–linefeed (Chr(13) + Chr(10)) sequence.

## Print #

### Description

Writes display-formatted data to a sequential file.

### Syntax

**Print #** *filename*, [*outputlist*]

The **Print #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
outputlist*		Optional. Expression or list of expressions to print.

## Remarks

Data written with **Print #** is usually read from a file with **LineInput #**.

outputlist\* Quick3270 supports only one expression. Not possible currently to specify a list of expressions delimited by commas.

## Write #

### Description

Writes data to a sequential file.

### Syntax

**Write #** *filename*, [*outputlist*]

The **Write #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
outputlist		Optional. One or more comma-delimited numeric expressions or string expressions to write to a file.

## Seek #

### Description

Sets the position for the next read/write operation within a file opened using the Open statement.

### Syntax

**Seek #** *filename*, *position*

The **Seek #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
position	Integer	Required. Number in the range 1 – 2,147,483,647, inclusive, that indicates where the next read/write operation should occur.

## Get #

### Description

Reads data from an open disk file into a variable.

### Syntax

**Get #** *filename*, [*recnumber*], *varname*

The **Get #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
recnumber	Integer	Optional. Byte number at which reading begins
varname	String	Required. Valid variable name into which data is read.

## Put #

### Description

Writes data from a variable to a disk file.

### Syntax

**Put #** *filename*, [*recnumber*], *varname*

The **Put #** statement syntax has these parts:

Part	Type	Description
filename	Integer	Required. Any valid file number.
recnumber	Integer	Optional. Byte number at which reading begins
varname	String	Required. Valid variable name into which data is read.

## EOF

### Description

Returns the Boolean value True when the end of a file has been reached.

### Syntax

**EOF**(*filename*)

The required filename argument is an Integer containing any valid file number.

## FreeFile

### Description

Returns an Integer representing the next file number available for use by the Open statement.

### Syntax

**FreeFile()**

### Remarks

Use FreeFile to supply a file number that is not already in use.

## LOC

### Description

Returns an Integer specifying the current read/write position within an open file.

### Syntax

**Loc**(*filenumber*)

The required filenumber argument is an Integer containing any valid file number.

## LOF

### Description

Returns an Integer representing the size, in bytes, of a file opened using the Open statement.

### Syntax

**LOF**(*filenumber*)

The required filenumber argument is an Integer containing any valid file number.

## Kill

### Description

Deletes a file from a disk.

### Syntax

**Kill** *pathname*

The required pathname argument is a string expression that specifies one a file name to be deleted. The pathname may include the directory or folder, and the drive.

Date variables are stored as IEEE 64-bit (8-byte) floating-point numbers that represent dates ranging from 1 January 100 to 31 December 9999 and times from 0:00:00 to 23:59:59. Any recognizable literal date values can be assigned to Date variables. Date literals must be enclosed within number signs (#).

Sample of date literals:

```
#January 1, 1993#  
#1 Jan 93#  
# 8/23/1970 3:45:39AM #  
# 8/23/1970 #  
# 3:45:39AM #  
# 3:45:39 #  
# 13:45:39 #
```

# 1AM #

Date variables display dates according to the short date format recognized by your computer. Times display according to the time format (either 12-hour or 24-hour) recognized by your computer.

**Remark**

Support of date variables and related functions have been added in version 4.37

**Build-In Symbolic Constants.**

The macro language provides a number of predefined constants that can be used anywhere in your code in place of the actual values

Constant	Value	Description
vbSunday	1	Sunday (default).
vbMonday	2	Monday
vbTuesday	3	Tuesday.
vbWednesday	4	Wednesday.
vbThursday	5	Thursday
vbFriday	6	Friday.
vbSaturday	7	Saturday
vbUseSystemDayOfWeek	0	Use the day of the week specified in your system settings for the first day of the week.
vbFirstJan1	1	Start with week in which January 1 occurs (default).
vbFirstFourDays	2	Start with the first week that has at least four days in the new year.
vbFirstFullWeek	3	Start with the first full week of the year.

## Date\$

**Description**

Returns a string containing the current system date.

**Syntax**

**d = Date\$()**

Date

**Description**

Returns a Date Variable containing the current system date.

**Syntax**

**d = Date**

**Remarks**

Added in version 5.21

## Time\$

### Description

Returns a string containing the current system time.

### Syntax

**d = Time\$()**  
Time

### Description

Returns a Date variable containing the current system time.

### Syntax

**d = Time**

### Remarks

Added in version 5.21

## Day

### Description

Returns an Integer specifying a whole number between 1 and 31, inclusive, representing the day of the month.

### Syntax

**MyDay = Day(date)**

### Example

```
Dim MyDate As Date
Dim MyDay As Integer
MyDate = #May 20, 1963# ' Assign a date.
MyDay = Day(MyDate)   ' MyDay contains 20
```

## Month

### Description

Returns an Integer specifying a whole number between 1 and 12, inclusive, representing the month of the year.

### Syntax

**MyMonth = Month(date)**

### Example

```
Dim MyDate As Date
Dim MyMonth As Integer
MyDate = #May 20, 1963# ' Assign a date.
MyMonth = Month(MyDate) ' MyMonth contains 5
```

# Date / Time functions

## Year

### Description

Returns an Integer specifying a whole number representing the year.

### Syntax

**MyYear = Year**(date)

### Example

```
Dim MyDate As Date
Dim MyYear As Integer
MyDate = #May 20, 1963# ' Assign a date.
MyYear = Year(MyDate) ' MyYear contains 1963
```

## DateAdd

### Description

Returns a Date value containing a date to which a specified time interval has been added.

### Syntax

**DateAdd**(*interval*, *number*, *date*)

### Parameters

Part	Type	Description
interval	String	Required. String expression that is the interval of time you want to add..
number	String	Required. Numeric expression that is the number of intervals you want to add. It can be positive (to get dates in the future) or negative (to get dates in the past).
date	String	Required. Date or literal representing date to which the interval is added.

### Settings

Currently supported *interval* argument:

Setting	Description
yyyy	Year
m	Month
d	Day
h	Hour
n	Minut
s	Second

### Remarks

You can use the **DateAdd** function to add or subtract a specified time interval from a date. For example, you can use **DateAdd** to calculate a date 30 days from today or a time 45 minutes from now.

To add days to **date**, you can use Day of Year ("y"), Day ("d"), or Weekday ("w").

The **DateAdd** function won't return an invalid date. The following example adds one month to January 31:

```
DateAdd("m", 1, "31-Jan-95")
```

In this case, **DateAdd** returns 28-Feb-95, not 31-Feb-95. If **date** is 31-Jan-96, it returns 29-Feb-96 because 1996 is a leap year.

If the calculated date would precede the year 100 (that is, you subtract more years than are in **date**), an error occurs.

## DateSerial

### Description

Returns a Date value for a specified year, month, and day.

### Syntax

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

### Parameters

Part	Type	Description
year	Integer	Required. Number between 100 and 9999, inclusive, or a numeric expression
month	Integer	Required. Any numeric expression.
day	Integer	Required. Any numeric expression.

### Example

```
Dim MyDate
' MyDate contains the date for May 20, 1963.
MyDate = DateSerial(1963, 5, 20) ' Return a date.
```

## DateValue

### Description

Returns a date value from the string expression representing a date.

### Syntax

```
DateValue(date)
```

### Parameter

Part	Type	Description
date	String	Required. String expression representing a date from January 1, 100 through December 31, 9999.

		However, date can also be any expression that can represent a date, a time, or both a date and time, in that range.
--	--	---

**Example**

```
Dim MyDate As Date
MyDate = DateValue("May 20, 1963") ' Return a date
```

**Format****Description**

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

**Syntax**

**Format**(*expression*[, *format*])

**Parameters**

Part	Type	Description
expression	Date	Required. Any valid date expression.
format	String	Optional. Any named or user defined format expression.

**Predefined Named Date/Time formats.**

The following table identifies the predefined date and time format names:

Format Name	Description
General Date	Displays a date and/or time. Date display is determined by your application's current culture value.
Long Date	Display a date according to your system's long date format.
Medium Date	Display a date using the medium date format appropriate for the language version of the host application.
Short Date	Display a date using your system's short date format.
Long Time	Display a time using your system's long time format; includes hours, minutes, seconds.
Medium Time	Display time in 12-hour format using hours and minutes and the AM/PM designator.
Short Time	Display a time using the 24-hour format, for example, 17:45.

**User-Defined Date/Time Formats.**

Descriptions preceded by an asterisk: They are differences with the Format function used in VBA (Visual Basic for Applications).

The following table identifies characters you can use to create user-defined date/time formats:

Format Name	Description
d	Display the day as a number without a leading zero (1 – 31).

dd	Display the day as a number with a leading zero (01 – 31).
ddd	Display the day as an abbreviation (Sun – Sat).
dddd	Display the day as a full name (Sunday – Saturday).
M	*Display the month as a number without a leading zero (1 – 12).
MM	*Display the month as a number with a leading zero (01 – 12).
MMM	*Display the month as an abbreviation (Jan – Dec).
MMMM	*Display the month as a full month name (January – December).
y	Year represented only by the last digit.
yy	Year represented only by the last two digits. A leading zero is added for single-digit years.
yyyy	Year represented by a full four or five digits, depending on the calendar used.
h	*Hours without leading zeros for single-digit hours (12-hour clock).
hh	*Hours with leading zeros for single-digit hours (12-hour clock).
H	*Hours without leading zeros for single-digit hours (24-hour clock).
HH	*Hours with leading zeros for single-digit hours (24-hour clock).
m	*Minutes without leading zeros for single-digit minutes.
mm	*Minutes with leading zeros for single-digit minutes.
s	*Seconds without leading zeros for single-digit seconds.
ss	*Seconds with leading zeros for single-digit seconds.

**Example**

```
Dim MyTime As Date
Dim MyDate As Date
Dim MyStr As String
```

```
MyTime = #17:04:23#
MyDate = #January 27, 1993#
```

```
' Returns current system time in the system-defined long time format.
MyStr = Format(Now, "Long Time")
```

```
' Returns current system date in the system-defined long date format.
MyStr = Format(Now, "Long Date")
```

```
MyStr = Format(MyTime, "h:m:s")           ' Returns "17:4:23".
MyStr = Format(MyDate, "dddd, MMM d yyyy") ' Returns "Wednesday,
                                           ' Jan 27 1993".
```

**Now****Description**

Returns a Date value containing the current system date and time.

**Syntax**

# Date / Time functions

**d = Now**  
TimeSerial

## Description

Returns a Date containing the time for a specific hour, minute, and second.

## Syntax

**TimeSerial**(*hour, minute, second*)

## Parameters

Part	Type	Description
hour	Integer	Required. Number between 0 (12:00 A.M.) and 23 (11:00 P.M.), inclusive.
minute	Integer	Required. Any numeric expression.
second	Integer	Required. Any numeric expression.

## Example

```
Dim MyTime As Date
MyTime = TimeSerial(16, 35, 17) ' Serial representation of 4:35:17 PM.
```

## TimeValue

## Description

Returns a date value from the string expression representing a time.

## Syntax

**TimeValue**(*time*)

## Parameter

Part	Type	Description
time	String	Required. String expression string expression representing a time from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.), inclusive.

## Example

```
Dim MyTime As Date
MyTime = TimeValue("4:35:17 PM") ' Return a time.
```

In Quick3270 Macro, you can access an object and use the originating software application to change properties and methods of that object. You can programmatically manipulate the data in these objects. Before you can use an object in a procedure, however, you must access the software application associated with the object by assigning it to an object variable. Next, you attach an object name (with or without properties and methods) to the variable to manipulate the object.

## CreateObject

## Description

Creates and returns a reference to an Automation object.

**Syntax****CreateObject**(*servername.typename*)**Parameters**

Part	Type	Description
servername	String	Required. The name of the application providing the object.
typename	String	Required. The type or class of the object to create.

**Remarks**

Automation servers provide at least one type of object. For example, a word-processing application may provide an application object, a document object, and a toolbar object.

To create an Automation object, assign the object returned by **CreateObject** to an object variable:

```
Set ExcelSheet = CreateObject("Excel.Sheet")
```

**GetObject****Description**

Returns a reference to an Automation object from a file.

**Syntax****GetObject**([*pathname*] [, *class*])**Parameters**

Part	Type	Description
pathname	String	Optional. Full path and name of the file containing the object to retrieve. If pathname is omitted, class is required.
class	String	Optional. Class of the object.

The class argument uses the syntax *appname.objecttype* and has these parts:

Part	Type	Description
appname	String	Required. Name of the application providing the object.
objecttype	String	Required. Type or class of object to create.

**Remarks**

Use the **GetObject** function to access an Automation object from a file and assign the object to an object variable. Use the Set statement to assign the object returned by **GetObject** to the object variable. For example:

```
Set CADObject = GetObject("C:\CAD\SCHEMA.CAD")
```

When this code is executed, the application associated with the specified pathname is started and the object in the specified file is activated. If *pathname* is a zero-length string (""), **GetObject** returns a new object instance of the specified type. If the *pathname* argument is omitted, **GetObject** returns a currently active object of the specified type. If no object of the specified type exists, an error occurs.

## Beep

**Description**

Sounds a tone through the computer's speaker. The frequency and duration of the beep depends on hardware, which may vary among computers

**Syntax****Beep**

Decrypt or Decrypt\$

**Description**

Returns the String decrypted.

**Syntax**

**Decrypt** (str) or **Decrypt \$**(str)

**Parameter**

Part	Type	Description
str	String	Required; String expression that will be decrypted.

**Remark**

Added in version 4.30

Encrypt or Encrypt\$

**Description**

Returns the String encrypted.

**Syntax**

**Encrypt**(str) or **Encrypt \$**(str)

**Parameter**

Part	Type	Description
str	String	Required; String expression that will be encrypted.

**Remark**

Added in version 4.30

Environ or Environ\$

**Description**

Returns the String associated with an operating system environment variable.

**Syntax**

**Environ**(str) or **Environ\$**(str)

**Parameter**

Part	Type	Description
str	String	Required; String expression containing the name of an environment variable.

**Remark**

Added in version 4.30

**GetOpenFileName****Description**

This function creates a system-defined dialog box that enables the user to select a file to open.

**Syntax**

**GetOpenFileName**(*title*, *pathname* , *filter*)

**Parameters**

Part	Type	Description
title	String	Required. String to be placed in the title bar of the dialog box
pathname	String	Required. String that can specify the initial directory.
filter	String	Required. String containing pairs filters separated by the ' ' character.

**GetSpecialFolderLocation****Description**

This function retrieves the full path of a known folder identified by the folder's CSIDL.

**Syntax**

**GetOpenFileName**(*CSIDL*)

Some values for CSIDL:

```

CSIDL_APPDATA = &H1A           ' Application data for the current user
CSIDL_COMMON_APPDATA = &H23    ' Application data for all users.
CSIDL_PROGRAM_FILES = &H26    ' Program Files folder
CSIDL_COMMON_DOCUMENTS = &H23 ' Documents that are common to all users
...

```

**Remark**

Added in version 4.40

**InputBox****Description**

Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns the contents of the text box.

**Syntax**

**InputBox**(*prompt*[, *title*][, *default*])

## Parameters

Part	Type	Description
prompt	String	Required. String expression displayed as the message in the dialog box.
title	String	Optional. String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.
default	String	Optional. String expression displayed in the text box as the default response if no other input is provided. If you omit default, the text box is displayed empty.

## Remarks

If the user clicks OK or presses ENTER, the **InputBox** function returns whatever is in the text box. If the user clicks Cancel, the function returns a zero-length string ("").

## MsgBox

### Description

Displays a message in a dialog box, waits for the user to click a button, and returns a value indicating which button the user clicked.

### Syntax

**Result** = **MsgBox**(*prompt*[, *buttons*][, *title*])

or

**MsgBox** *prompt*, *buttons*][, *title*]

## Parameters

Part	Type	Description
prompt	String	String expression displayed as the message in the dialog box. The maximum length of prompt is approximately 1024 characters, depending on the width of the characters used. If prompt consists of more than one line, you can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line. You can even use the system defined constants like vbCR, vbLf, vbCrLf...
buttons	Integer	Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. See Settings section for values. If omitted, the default value for buttons is 0.
title	String	String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.

## MsgBox Constants

The following constants are used with the MsgBox function to identify what buttons and icons appear on a message box and which button is the default. In addition, the modality of the **MsgBox** can be specified. Since these constants are built into Quick3270 Scripting language, you don't have to define them before using them. Use them anywhere in your code to represent the values shown for each.

Constant	Value	Description
vbOKOnly	0	Display OK button only.
vbOKCancel	1	Display OK and Cancel buttons.
vbAbortRetryIgnore	2	Display Abort, Retry, and Ignore buttons.
vbYesNoCancel	3	Display Yes, No, and Cancel buttons.
vbYesNo	4	Display Yes and No buttons.
vbRetryCancel	5	Display Retry and Cancel buttons.
vbCritical	16	Display Critical Message icon.
vbQuestion	32	Display Warning Query icon.
vbExclamation	48	Display Warning Message icon.
vbInformation	64	Display Information Message icon.
vbDefaultButton1	0	First button is the default.
vbDefaultButton2	256	Second button is the default.
vbDefaultButton3	512	Third button is the default.
vbDefaultButton4	768	Fourth button is the default.
vbApplicationModal	0	Application modal. The user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal. This constant provides an application modal message box that always remains on top of any other programs you may have running.
vbMsgBoxHelpButton	16384	Adds Help button to the message box
vbMsgBoxSetForeground	65536	Specifies the message box window as the foreground window
vbMsgBoxRight	524288	Text is right aligned
vbMsgBoxRtlReading	1048576	Specifies text should appear as right-to-left reading on Hebrew and Arabic systems

### MsgBox Return Values.

The following constants are used with the **MsgBox** function to identify which button a user has selected. These constants are built into Quick3270 Scripting language; you don't have to define them before using them.

Constant	Value	Description
vbOK	1	OK button was clicked.
vbCancel	2	Cancel button was clicked.
vbAbort	3	Abort button was clicked.
vbRetry	4	Retry button was clicked.

vbIgnore	5	Ignore button was clicked.
vbYes	6	Yes button was clicked.
vbNo	7	No button was clicked.

## PasswordBox

### Description

Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns the contents of the text box. The Input text is hidden and replaced by asterisk '\*'.

### Syntax

**PasswordBox**(*prompt*[, *title*][, *default*])

### Parameters

Part	Type	Description
prompt	String	Required; String expression displayed as the message in the dialog box.
title	String	Optional; String expression displayed in the title bar of the dialog box. If you omit title, the application name is placed in the title bar.
default	String	String expression displayed in the text box as the default response if no other input is provided. If you omit default, the text box is displayed empty.

### Remarks

If the user clicks OK or presses ENTER, the PasswordBox function returns whatever is in the text box. If the user clicks Cancel, the function returns a zero-length string ("").

## Rem

### Description

Used to include explanatory remarks in a program. You can use an apostrophe (') instead of the Rem keyword.

### Syntax

**Rem** *Comment*

You can also use the following syntax:  
' comment

## Shell

### Description

Runs an executable program and returns an Integer representing the program's task ID if successful, otherwise it returns zero.

### Syntax

**Shell**(*pathname*[, *windowstyle*])

**Parameters**

Part	Type	Description
pathname	String	Required; Name of the program to execute and any required arguments or command-line switches; may include directory or folder and drive.
windowstyle	Integer	Optional. Integer corresponding to the style of the window in which the program is to be run. If windowstyle is omitted, the program is started minimized with focus. In Attachmate's Extra! compatibility mode, the default windowstyle is normal with focus.

The windowstyle named argument has these values:

Constant	Value	Description
vbHide	0	Window is hidden and focus is passed to the hidden window.
vbNormalFocus	1	Window has focus and is restored to its original size and position.
vbMinimizedFocus	2	Window is displayed as an icon with focus.
vbMaximizedFocus	3	Window is maximized with focus.
vbNormalNoFocus	4	Window is restored to its most recent size and position. The currently active window remains active.
vbMinimizedNoFocus	6	Window is displayed as an icon. The currently active window remains active.

**Remarks**

Each pair of adjacent double quotation marks ( " ") within the string literal is interpreted as one double quotation character in the string.

```
Result = Shell("""C:\Program Files\display.exe"" -a -q", vbNormalFocus)
```

Therefore, the preceding example presents the following string to the Shell function:  
"C:\Program Files\display.exe" -a -q

If the path is not enclosed in quotation marks, Windows looks for a file called Program.exe in the C:\ directory, instead of display.exe in the C:\Program Files directory.

**ConvertCol****Description**

Returns the column corresponding to the given presentation space positional value.

**Syntax**

**ConvertCol**(location)

**Parameters**

Part	Type	Description
------	------	-------------

location	Integer	Required; Host presentation space position. The lower limit for valid input is 1. The upper limit for valid input ranges from 1920 to 3564 depending on how the host presentation space is configured..
----------	---------	---

## ConvertRow

### Description

Returns the row corresponding to the given presentation space positional value.

### Syntax

**ConvertRow**(*location*)

### Parameters

Part	Type	Description
location	Integer	Required; Host presentation space position. The lower limit for valid input is 1. The upper limit for valid input ranges from 1920 to 3564 depending on how the host presentation space is configured..

## EditCopy

### Description

Copies the content of the selected presentation space area to the clipboard. If no area is selected, the entire presentation space is copied to the clipboard.

### Syntax

**EditCopy**  
**EditCopyAdd**

### Description

Append the content of the selected presentation space area to the clipboard. If no area is selected, the entire presentation space is appended to the clipboard.

### Syntax

**EditCopyAdd**  
**EditPaste**

### Description

Copies the content of the clipboard to the presentation space at the current cursor location.

### Syntax

**EditPaste** [*Options*]

### Options

Part	Description
RowOriented	Paste rows of text that end with carriage returns or line feeds. If not specified, paste is field oriented. Carriage returns, line feeds or tab stops will jump

	to the next field.
FieldTruncation	If the end of a field is encountered, the remaining characters will be used to continue the paste operation. If not specified, the remaining characters until carriage returns, line feeds or tab stops will be lost.
WordWrap	Avoid words being split across fields and lines
Protected	Pastes from the clipboard to the selected area of the destination screen, regardless of whether these fields are protected or unprotected.
Continue	Keep the not pasted characters into the clipboard for a next paste operation.
UpdateCursorPos	Moves the cursor to the end of the pasted data.
TabJump	When a tab character is encountered, the following text data will be pasted into the next field. If not selected, the tab character is handled as a space character.
TabStop	The tab stop is replaced by default by 4 space characters.
FillWithSpaces	Fill the end of a field with space characters. If not selected, the end of the field is filled with null characters.

## EditSelectAll

### Description

Selects the entire screen.

### Syntax

**EditSelectAll**

## GetCol

### Description

Returns an Integer representing the column position of the cursor.

### Syntax

**GetCol()**  
**GetCols**

### Description

Returns an Integer representing the number of columns of the presentation space.

### Syntax

**GetCols()**  
**GetRow**

### Description

Returns an Integer representing the row position of the cursor.

**Syntax**

**GetRow()**  
**GetRows**

**Description**

Returns an Integer representing the number of rows of the presentation space.

**Syntax**

**GetRows()**  
**GetRowCount**

**Description**

Returns an Integer representing the number of occurrence of a specified character at a given column.

**Syntax**

**GetRowCount**(*token*, *column* [, *firstrow* [,*lastrow*]])

**Parameters**

Part	Type	Description
token	String	Required; The token to search for. If the string contains more than one character, the function uses the first character.
column	Integer	Required. The column where to search the token.
firstrow	Integer	Optional. The first row where to begin to search. Default is 1.
lastrow	Integer	Optional. The row where to stop the search. Default is the last row of the presentation space.

**Remarks**

The purpose of this function is to quickly count rows that contain data. For example if a transaction displays a result as a list.

**GetColor**

**Description**

Returns an Integer representing the color code at the given screen location.

**Syntax**

**GetColor**(*row*, *col*)

**Parameters**

Part	Type	Description
row	Integer	The row where the color attribute is read.
col	Integer	The column where color attribute is read.

**GetColor Return Values.**

The following values are returned:

- 0 – default Color/Green
- 1 – Blue
- 2 – Red
- 3 – Pink
- 4 – Green
- 5 – Turquoise
- 6 – Yellow
- 7 – White

**GetIpAddress****Description**

This function returns the IP address of the computer running Quick3270.

**Syntax**

String = **GetIpAddress**(index)

**Parameter**

Part	Type	Description
index	Integer	Required. As a Pc can have several network adapters, it can have several IP addresses. So the GetIpAddress function uses an index, the first address is on index = 0. It's possible to get all the IP addresses by incrementing the index until a zero length string is returned. This version returns only IPv4 addresses.

**GetIpAddressV6****Description**

This function returns the IPv6 address of the computer running Quick3270.

**Syntax**

String = **GetIpAddressV6**(index)

**Parameter**

Part	Type	Description
index	Integer	Required. As a Pc can have several network adapters, it can have several IP addresses. So the GetIpAddressV6 function uses an index, the first address is on index = 0. It's possible to get all the IP addresses by incrementing the index until a zero length string is returned. This version returns only IPv6 addresses.

**Remarks**

Added in version 4.37.

**GetMacroKeystroke****Description**

This function returns the Virtual Keycode of the keystroke that started the macro. It returns 0 in case the macro was started from the menu.

## Syntax

KeyCode = **GetMacroKeystroke()**

The list of the Keycodes:

[http://msdn.microsoft.com/de-de/library/Oz084th3\(v=vs.90\).aspx](http://msdn.microsoft.com/de-de/library/Oz084th3(v=vs.90).aspx)

The Upper word contains the modifier (Alt / Ctrl / Shift)

## Sample:

```
WshShell = CreateObject("WScript.Shell")
Result = GetMacroKeystroke()

KeyStr$ = "Key pressed: "

if Result And &H100 Then
    KeyStr$ = KeyStr$ & "Alt + "
Endif

if Result And &H200 Then
    KeyStr$ = KeyStr$ & "Ctrl + "
Endif

if Result And &H400 Then
    KeyStr$ = KeyStr$ & "Shit + "
Endif

' Remove the modifier value from Result and check the VK_KEY value
Result = Result And &HFF

if Result = 0 Then
    KeyStr$ = KeyStr$ & "None"
Endif

if Result = &H70 Then
    KeyStr$ = KeyStr$ & "F1"
Endif

if Result = &H71 Then
    KeyStr$ = KeyStr$ & "F2"
Endif

WshShell.popup(KeyStr$)

End
```

## Remarks

Added in version 4.30.

## GetOIA

### Description

Returns the string representing the content of the OIA line.

### Syntax

String **GetOIA()**

## GetString

### Description

Returns the text from the specified screen location.

### Syntax

**GetString**(*row, col, length*)

### Parameters

Part	Type	Description
row	Integer	The row where the text string begins.
col	Integer	The column where the text string begins.
length	Integer	The length of the text string..

### Remarks

Null and non-printable characters are replaced by spaces.

Added in version 4.12

## GetText

### Description

Returns characters from the presentation space.

### Syntax

**GetText** ([*Row* [, *Col* [, *Len*]])

### Parameters

Part	Type	Description
Row	Integer	Optional. Row at which to begin the retrieval in the presentation space. Default value is 1.
Col	Integer	Optional. Column at which to begin the retrieval in the presentation space. Default value is 1.
Len	Integer	Optional. Number of characters to retrieve from the presentation space. Default value is the number of characters of the presentation space.

### Remarks

Null and non-printable characters are replaced by spaces.

Added in version 4.37

## GetTextRect

### Description

Returns characters from a rectangular area in the presentation space.

No wrapping takes place in the text retrieval; only the rectangular area is retrieved.

**Syntax****GetTextRect**(*StartRow, StartCol, EndRow, EndCol*)**Parameters**

Part	Type	Description
StartRow	Integer	Required. Row at which to begin the retrieval in the presentation space.
StartCol	Integer	Required. Column at which to begin the retrieval in the presentation space.
EndRow	Integer	Required. Row at which to end the retrieval in the presentation space
EndCol	Integer	Required. Column at which to end the retrieval in the presentation space..

**Remarks**

Null and non-printable characters are replaced by spaces.

Added in version 4.31

**GetTimeoutValue****Description**

Returns the number of milliseconds used by Wait operations.

**Syntax****GetTimeoutValue**()**GetVisible****Description**

Returns the visible state of the terminal Window.

**Syntax****GetVisible**()**Returns**

The function returns **True** if application Window is Visible.

**GetWindowTitle****Description**

Returns the title of the terminal Window.

**Syntax****GetWindowTitle**()**MoveTo****Description**

Moves the cursor to the specified screen location.

**Syntax****MoveTo**(*row, col*)**Parameters**

Part	Type	Description
row	Integer	The row location.
col	Integer	The column location.

**MoveRelative****Description**

Moves the cursor relative from the current screen location.

**Syntax****MoveRelative**(*row, col*)**Parameters**

Part	Type	Description
row	Integer	Required. The number of rows to move.
col	Integer	Required. The number of columns to move.

**OpenPrinter****Description**

Allocates the default Windows printer for the PrintScreen function.

**Syntax****OpenPrinter****Remark**

OpenPrinter must be called before the PrintScreen function.

**ClosePrinter****Description**

Frees the resources allocated by the OpenPrinter function.

**Syntax****ClosePrinter****PrintScreenFontSize****Description**

PrintScreenFontSize is a global variable used to specify the font size used by the PrintScreen function. If this variable is not set or set to 0, the default font size is used.

## Syntax

```
PrintScreenFontSize = 8.  
PrintScreen
```

## Description

Prints the presentation space.

## Syntax

**PrintScreen** [*Options*]

The options must be separated by a space character.

Valid options are:

*BW*: Print in Black & White

*HEADER*: Print the page header (overwrites the default settings)

*NOHEADER*: Don't print the page header (overwrites the default settings)

*FORMFEED*: Execute a form feed before the print job

## Sample

```
OpenPrinter  
PrintScreen  
PrintScreen BW  
PrinterFontSize=8  
PrintScreen NOHEADER FORMFEED  
PrinterFontSize=0  
PrintScreen  
ClosePrinter  
End
```

## PutString

## Description

Puts text in the specified location on the screen.

## Syntax

**PutString** *str* [,*row* , *col*]

## Parameters

Part	Type	Description
str	String	Required. Text that you want to put on the screen.
row	Integer	Optional. Integer specifying the row in which to put the text.
col	Integer	Optional. Integer specifying the column in which to put the tex.

## Remark

If row and column are omitted, the current caret location is used.

## ReceiveFile

**Description**

This function allows to receive a file from a host session.

**Syntax**

**ReceiveFile** *PcFile HostFile Options*

or

Boolean **ReceivedFile**(*PcFile,HostFile [,Options]*)

The **ReceiveFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. String expression, pc file name.
HostFile	String	Required. String expression, host file name
Options	String	Optional. File transfer options

**Remarks**

Function added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the EHLAPI Receive File function.

## Search

**Description**

Search for the first occurrence of text in the Presentation Space.

**Syntax**

Boolean **Search**(*Token [,row, col]*)

**Parameters**

Part	Type	Description
Token	String	Required. String to search for.
row	Integer	Optional. Row position to start search in Presentation Space
col	Integer	Optional. Column position to start search in Presentation Space

## SearchPS

### Description

Search in the presentation space for the provided string.

### Syntax

Boolean **SearchPS**(*Token* [,*address*])

### Parameters

Part	Type	Description
Token	String	Required. String to search for.
row	Integer	Optional. Presentation Space location from where to search the token

### Remark

Similar to the HLLAPI SearchPS

### Sample

```
nAddress = SearchPS(".PAM2")
nRow = ConvertRow(nAddress)
nCol = ConvertCol(nAddress)
Message = ".PAM2 found at Row " + Str$(nRow) + ", Col " + Str$(nCol)
rc = msgbox(Message , vbokonly + vbExclamation, "SearchPS Test")
Exit
```

## SendDarkKeys

### Description

Puts text in the specified location on the screen. The text is encrypted.

### Syntax

**SendDarkKey** str

### Parameters

Part	Type	Description
str	String	Required. String expression that you want to put on the screen.

### Remarks

The **SendDarkKey** is used only by the macro recorder. All text typed in a "hidden" field will be stored encrypted. The SendDarkKey allows reading this encrypted text and sending the original text back to the emulator. This function is not intended to be used by the user. However you can Copy / Paste a **SendDarkKey** statement generated by the macro recorder into another macro file

## SendFile

### Description

This function allows to send a file to a host session.

**Syntax****SendFile** *PcFile HostFile Options*

or

Boolean **SendFile**(*PcFile,HostFile,Options*)The **SendFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. String expression, pc file name.
HostFile	String	Required. String expression, host file name
Options	String	Optional. File transfer options

**Remarks**

Function added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the EHLAPI Send File function.

**SendKeys****Description**

Send keystrokes at the current cursor location.

**Syntax****SendKeys** *str*The **SendKeys** syntax has these arguments:

Part	Type	Description
str	String	Required. String expression that you want to put on the screen.

**Remarks**

Host function keys can be specified with mnemonics.

- Mnemonics must be enclosed in angle.
- Mnemonics are case insensitive

Table of mnemonics representing applicable function keys.

Description	Mnemonics	Emulation
Alternate Cursor	<AltCursor>	3270 and 5250
Attention	<Attn>	3270 and 5250
Back Space	<Backspace>	3270 and 5250
Back Tab	<Backtab>	3270 and 5250
Back Word Tab	<BackWord>	3270 and 5250
CapsLock	<CapsLock>	3270 and 5250
Clear	<Clear>	3270 and 5250
Cursor down	<Down>	3270 and 5250
Cursor left	<Left>	3270 and 5250
Cursor right	<Right>	3270 and 5250
Cursor up	<Up>	3270 and 5250
Cursor left fast	<Left2>	3270 and 5250
Cursor right fast	<Right2>	3270 and 5250
Cursor Select	<CursorSelect>	3270 and 5250
Delete char	<Delete>	3270 and 5250
Duplicate	<Dup>	3270 and 5250
Edit-Select All	<Edit-Select All>	3270 and 5250
Edit-Cut	<Edit-Cut>	3270 and 5250
Edit-Copy	<Edit-Copy>	3270 and 5250
Edit-Paste	<Edit-Paste>	3270 and 5250
Edit-Paste Continue	<Edit-Paste Continue>	3270 and 5250
Edit-Copy Append	<Edit-Copy Append>	3270 and 5250
Edit-Undo Paste	<Edit-Undo Paste>	3270 and 5250
Enter	<Enter>	3270 and 5250
Erase EOF	<EraseEOF>	3270 and 5250
Erase Input	<EraseInput>	3270 and 5250
Erase Field	<EraseField>	3270 and 5250
End of Field	<EOF>	3270 and 5250
Field Exit	<Field Exit>	5250
Field Minus	<Field ->	5250
Field Plus	<Field +>	5250
Field Mark	<FieldMark>	3270 and 5250

Description	Mnemonics	Emulation
Forward Word Tab	<ForwardWord>	3270 and 5250
Help	<Help>	5250
Home	<Home>	3270 and 5250
Host Print	<Host Print>	5250
Insert Toggle	<Insert>	3270 and 5250
Jump to next session	<Jump>	3270 and 5250
Last Field	<Last Field>	3270 and 5250
New Line	<NewLine>	3270 and 5250
Num Lock	<NumLock>	3270 and 5250
Page Down or Roll Up	<Page Down>	5250
Page Up or Roll Down	<Page Up>	5250
Print Screen - Default	<PrintScreen>	3270 and 5250
Print Screen - File	<PrintScreen to File>	3270 and 5250
Print Screen - Printer	<PrintScreen to Printer>	3270 and 5250
Reset	<Reset>	3270 and 5250
Ruler Toggler	<Rule>	3270 and 5250
Scroll Lock	<ScrLock>	3270 and 5250
System Request	<SysReq>	3270 and 5250
Tab	<Tab>	3270 and 5250
Pa1	<Pa1>	3270 and 5250
Pa2	<Pa2>	3270 and 5250
Pa3	<Pa3>	3270 and 5250
Pf1 to Pf24	<Pf1> – Pf24>	3270 and 5250

**Sample**

```
SendKeys "Logon<Enter>"
```

**SetText****Description**

Send the given text to the current field.  
Unlike SendKeys, mnemonics are not supported.

**Syntax**

**SetText** *str*

The **SetText** syntax has this argument:

Part	Type	Description
str	String	Required. String expression to set in field.

**Remarks**

Function added in version 4.30

**SetTimeoutValue****Description**

Specifies the number of milliseconds used by Wait operations.

**Syntax**

**SetTimeoutValue** *delay*

The **SetTimeoutValue** syntax has these arguments:

Part	Type	Description
delay	Integer	Required. Number of milliseconds.

**SetVisible****Description**

Sets the visible state of the terminal Window.

**Syntax**

**SetVisible** *state*

The **SetVisible** syntax has these arguments:

Part	Type	Description
state	Boolean	Required. TRUE if visible, FALSE if invisible.

**SetWindowTitle****Description**

Sets the application title bar text.

**Syntax**

**SetWindowTitle** *title*

The **SetWindowTitle** syntax has these arguments:

Part	Type	Description
title	String	Required. String to be displayed in the title bar.

## Updated

**Description**

Returns **True** if the screen was updated since the last call of this function.

**Syntax**

**Updated()**  
**WaitForAttrib**

**Description**

Waits until the specified attribute is displayed at the specified screen location.  
Returns True if the Attribute Value is found,

**Syntax**

Boolean **WaitForAttrib**(*row, col, Waitdata, [optional] MaskData, [optional] plane, [optional] TimeOut, [optional] WaitForIr*)

**Parameters**

Part	Type	Description
row	Integer	Required. Row position of the attribute.
col	Integer	Required. Column position of the attribute.
Waitdata	Integer	Required. Specifies the value of the attribute to wait for.
MaskData	Integer	Optional. Specifies the value to use as a mask with the attribute. The default value is &HFF.
plane	Integer	Optional. Specifies the plane of the attribute to get. The plane can have the following values: 1      Text Plane 2      Color Plane 3      Field Plane (default) 4      Extended Field Plane The default value is 3.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue
WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False

**Returns**

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

**WaitForCursor****Description**

Waits until the cursor is at the specified screen location

**Syntax**

**WaitForCursor**(*row, col, [optional] TimeOut*)

#### Parameters

Part	Type	Description
row	Integer	Required. Row position of the cursor.
col	Integer	Required. Column position of the cursor.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

#### Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

#### WaitForCursorMove

#### Description

Waits until the cursor has moved the specified number of rows and columns from its current position.

#### Syntax

Boolean **WaitForCursorMove**(*row, col, [optional] TimeOut*)

#### Parameters

Part	Type	Description
row	Integer	Required. The number of rows to move..
col	Integer	Required. The number of columns to move..
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

#### Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

#### WaitForKbdUnlock

#### Description

The function waits until the OIA of the connection indicates that the connection is able to accept keyboard input.

#### Syntax

Boolean **WaitForKbdUnlock**(*[optional] TimeOut*)

#### Parameters

Part	Type	Description
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

**Returns**

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

**WaitForString****Description**

Waits until the specified text appears on the specified screen location. Search string is case-sensitive.

**Syntax**

Boolean **WaitForString**(*str, row, col, [optional] TimeOut*)

**Parameters**

Part	Type	Description
str	String	Required. The text string that you want to wait for.
row	Integer	Required. The row where you expect the string to appear.
col	Integer	Required. The column where you expect the string to appear.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.

**Returns**

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

**WaitHostQuiet****Description**

Waits for the host to not send data for a specified number of milliseconds.

**Syntax**

Boolean **WaitHostQuiet**(*settletime*)

**Parameters**

Part	Type	Description
settletime	Integer	Required. The amount of time, in milliseconds, that the host should remain "quiet."

**Returns**

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

The possibility to run IBM<sup>®</sup> Personal Communications™ macros has been added in version 3.95 of Quick3270.

The IBM<sup>®</sup> Personal Communications compatibility mode allows Quick3270 to run VBScript or macros generated by the macro recorder.

Advanced VBScripts/macros, developed by users, are supported only if they use the methods and properties available in the Quick3270 compatibility mode.

# Support for IBM® Personal Communications™ macros 13

The following conditions must be verified to recognize an IBM® Personal Communications VBScript:

- File extension must be .mac
- VBScript Macros must start with the following header:

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=xxx
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
```

## Note:

It is also possible to rename the .mac file to .qvbs, Quick3270 VBScript macro. In this case, just edit and remove the header in the .mac file (the text from [PCOMM SCRIPT HEADER]... to ...[PCOMM SCRIPT SOURCE]).

The following chapters describes each object's methods and properties that are currently available in Quick3270 (.mac and .qvbs macro files).

- [autECLSession](#)
- [autECLPS](#)
  - [autECLFieldList](#)
- [autECLOIA](#)
- [autECLWinMetrics](#)
- [autECLXfer](#)

## autECLSession

The autECLSession object provides general emulator related services and contains pointers to other key objects.

### Properties

- [Name](#)
  
- [autECLPS](#) object
- [autECLOIA](#) object
- [autECLXfer](#) object
- [autECLWinMetrics](#) object

### Methods

- [SetConnectionByName](#)
- [StartCommunication](#)
- [StopCommunication](#)

## autECLSession.Name

### Description

This property returns the short name of the session, a character ID (A-Z).

### Syntax

Name = **autECLSession.Name**

**macros**`autECLSession.SetConnectionByName`**Description**

This function is simply ignored by Quick3270. With Quick3270 it's not possible to connect to another session.

**Syntax**

```
autECLSession.SetConnectionByName(name)
```

**Parameter**

Part	Type	Description
name	String	Required. One-character string short name of the connection (A-Z).

`autECLSession.StopCommunication`**Description**

This function disconnects the session to the host data stream.

**Syntax**

```
autECLSession.StopCommunication()  
autECLSession.StartCommunication
```

**Description**

This function connects the session to the host data stream.

**Syntax**

```
autECLSession.StartCommunication()
```

`autECLPS`

The `autECLPS` object performs operations on a presentation space

- [autECLPS](#)  
`autECLOIA`

The `autECLOIA` object retrieves status from the Host Operator Information Area.

- [autECLOIA](#)  
`autECLWinMetrics`

The `autECLWinMetrics` object performs operations on an emulator window. It allows you to perform window rectangle and position manipulation.

- [autECLWinMetrics](#)

**macros**  
autECLXfer

The autECLXfer object provides file transfer services.

- [autECLXfer](#)

**autECLPS**

The autECLPS object performs operations on a presentation space.

**Properties**

- [NumRows](#)
- [NumCols](#)
- [CursorPosRow](#)
- [CursorPosCol](#)
- [ConnType](#)
  
- [autECLFieldList](#) object

**Methods**

- [SetCursorPos](#)
- [SendKeys](#)
- [SearchText](#)
- [GetText](#)
- [SetText](#)
- [GetTextRect](#)
- [Wait](#)
- [WaitForCursor](#)
- [WaitWhileCursor](#)
- [WaitForAttrib](#)

**autECLPS.NumRows****Description**

Returns the number of rows in the presentation space for the connection associated with the autECLPS object.

**Syntax**

**Dim Rows as Long**

**Rows = autECLSession.autECLPS.NumRows**

**autECLPS.NumCols****Description**

Returns the number of columns in the presentation space for the connection associated with the autECLPS object.

**Syntax**

**Dim Cols as Long**

**Cols = autECLSession.autECLPS.NumCols**

macros  
autECLPS.CursorPosRow

**Description**

Returns current row position of the cursor in the presentation space.

**Syntax**

**CurPosRow = autECLSession.autECLPS.CursorPosRow**

autECLPS.CursorPosCol

**Description**

Returns current column position of the cursor in the presentation space.

**Syntax**

**CurPosCol = autECLSession.autECLPS.CursorPosCol**

autECLPS.ConnType

**Description**

Returns the connection type for which autECLPS was set. ConnType is a string data type and is read-only.

**Syntax**

**Dim ConnType as String**

**ConnType = autECLSession.autECLPS.ConnType**

String Returned	Meaning
DISP3270	3270 display
DISP5250	5250 display.
ASCII	VT emulation

autECLPS.SetCursorPos

**Description**

Sets the position of the cursor in the presentation space. The position set is in row and column units.

**Syntax**

**autECLSession.autECLPS.SetCursorPos** *Row, Col*

**Parameters**

Par t	Type	Description
Row	Integer	The row position of the cursor in the presentation space.

## macros

Col	Integer	The column position of the cursor in the presentation space.
-----	---------	--

**Remarks**

Added in version 5.21

autECLPS.SendKeys

**Description**

Send keystrokes at the current cursor location.

**Syntax**

autECLSession.autECLPS.SendKeys *str*

**Parameter**

Part	Type	Description
str	String	Required. Text that you want to put on the screen.

**Remarks**

Host function keys can be specified with mnemonics.

- Mnemonics must be enclosed in square brackets.
- Mnemonics are case insensitive

Table of mnemonics representing applicable function keys.

Description	Mnemonics IBM Personal Communications™ compatibility mode	Emulation
Alternate Cursor	[altcsr]	3270 and 5250
Attention	[attn]	3270 and 5250
Back Space	[backspace]	3270 and 5250
Back Tab	[backtab]	3270 and 5250
Back Word Tab	[wordleft]	3270 and 5250
CapsLock	[capslock]	3270 and 5250
Clear	[clear]	3270 and 5250
Cursor down	[down]	3270 and 5250
Cursor left	[left]	3270 and 5250
Cursor right	[right]	3270 and 5250
Cursor up	[up]	3270 and 5250
Cursor left fast	[fastleft]	3270 and 5250
Cursor right fast	[fastright]	3270 and 5250

Description	Mnemonics IBM Personal Communications™ compatibility mode	Emulation
Cursor Select	[crsel]	3270 and 5250
Delete char	[delete]	3270 and 5250
Duplicate	[dup]	3270 and 5250
Edit-Select All		3270 and 5250
Edit-Cut		3270 and 5250
Edit-Copy		3270 and 5250
Edit-Paste		3270 and 5250
Edit-Paste Continue		3270 and 5250
Edit-Copy Append		3270 and 5250
Edit-Undo Paste		3270 and 5250
Enter	[enter]	3270 and 5250
Erase EOF	[eraseeof]	3270 and 5250
Erase Input	[erinp]	3270 and 5250
Erase Field		3270 and 5250
End of Field	[eof]	3270 and 5250
Field Exit	[fldext]	5250
Field Minus	[field+]	5250
Field Plus	[field-]	5250
Field Mark	[fieldmark]	3270 and 5250
Forward Word Tab	[wordright]	3270 and 5250
Help	[help]	5250
Home	[home]	3270 and 5250
Host Print	[print]	5250
Insert Toggle	[insert]	3270 and 5250
Jump to next session	[jump]	3270 and 5250
Last Field		3270 and 5250
New Line	[newline]	3270 and 5250
Num Lock	[numlock]	3270 and 5250
Page Down or Roll Up	[pagedn]	5250
Page Up or Roll Down	[pageup]	5250
Print Screen - Default	[printps]	3270 and 5250

Description	Mnemonics IBM Personal Communications™ compatibility mode	Emulation
Print Screen - File		3270 and 5250
Print Screen - Printer		3270 and 5250
Reset	[reset]	3270 and 5250
Ruler Toggler		3270 and 5250
Scroll Lock	[scrlock]	3270 and 5250
System Request	[sysreq]	3270 and 5250
Tab	[tab]	3270 and 5250
Pa1	[pa1]	3270 and 5250
Pa2	[pa2]	3270 and 5250
Pa3	[pa3]	3270 and 5250
Pf1 to Pf24	[pf1] - [pf24]	3270 and 5250

## Sample

```
autECLSession.autECLPS.SendKeys "[enter]"
autECLPS.SearchText
```

## Description

Searches for the first occurrence of text in the presentation space. The search is case-sensitive. If text is found, the function returns a TRUE value. It returns a FALSE value if no text is found.

## Syntax

**autECLSession.autECLPS.SearchText** (*text* [, *Dir* [, *Row* [, *Col*]])

## Parameters

Part	Type	Description
Text	String	Required. Text to set in field.
Dir	Integer	Optional. Direction in which to search. Only value 1, for search forward, is supported now.
Row	Integer	Optional. Row position at which to start the search in the presentation space.
Col	Integer	Optional. Column position at which to start the search in the presentation space.

## Remarks

Added in version 4.37

macros  
autECLPS.GetText

**Description**

Returns characters from the presentation space.

**Syntax**

**autECLSession.autECLPS.GetText** ([Row [, Col [, Len]])

**Parameters**

Part	Type	Description
Row	Integer	Optional. Row at which to begin the retrieval in the presentation space. Default value is 1.
Col	Integer	Optional. Column at which to begin the retrieval in the presentation space. Default value is 1.
Len	Integer	Optional. Number of characters to retrieve from the presentation space. Default value is the number of characters of the presentation space.

**Remarks**

Null and non-printable characters are replaced by spaces.  
autECLPS.SetText

**Description**

Send the given text to the current field.  
Unlike SendKeys, mnemonics are not supported.

**Syntax**

**autECLSession.autECLPS.SetText** *str*

**Parameter**

Part	Type	Description
str	String	Required. Text to set in field.

**Remarks**

Added in version 4.30

autECLPS.GetTextRect

**Description**

Returns characters from a rectangular area in the presentation space.  
No wrapping takes place in the text retrieval; only the rectangular area is retrieved.

**Syntax**

**autECLSession.autECLPS.GetTextRect** (*StartRow, StartCol, EndRow, EndCol*)

## Parameters

Part	Type	Description
StartRow	Integer	Required. Row at which to begin the retrieval in the presentation space.
StartCol	Integer	Required. Column at which to begin the retrieval in the presentation space.
EndRow	Integer	Required. Row at which to end the retrieval in the presentation space
EndCol	Integer	Required. Column at which to end the retrieval in the presentation space..

## Remarks

Null and non-printable characters are replaced by spaces.

Added in version 4.31

autECLPS.Wait

## Description

The Wait method waits for the number of milliseconds specified by the milliseconds parameter.

## Syntax

**autECLSession.autECLPS.Wait** (*Delay*)

## Parameter

Part	Type	Description
Delay	Integer	The number of milliseconds to wait.

## Sample

```
autECLSession.autECLPS.Wait 4860
```

```
autECLPS.WaitForCursor
```

## Description

Waits until the cursor is at the specified screen location.

## Syntax

**autECLSession.autECLPS.WaitForCursor**(*row, col, [optional TimeOut]*)

## Parameters

Part	Type	Description
row	Integer	Required. Row position of the cursor.
col	Integer	Required. Column position of the cursor.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is Infinite

## Returns

## macros

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.  
autECLPS.WaitWhileCursor

**Description**

The WaitWhileCursor method waits while the cursor is located at a specified position.

**Syntax**

**autECLSession.autECLPS.WaitWhileCursor**(row, col, [optional TimeOut])

**Parameters**

Part	Type	Description
row	Integer	Required. Row position of the cursor.
col	Integer	Required. Column position of the cursor.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is Infinite

**Returns**

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.  
autECLPS.WaitForString

**Description**

Waits until the specified text appears on the specified screen location.  
Search string is case-sensitive.

**Syntax**

**autECLSession.autECLPS.WaitForString**(str, row, col, [optional] TimeOut, [optional] WaitForIr, [optional] bCaseSens)

**Parameters**

Part	Type	Description
str	String	Required. The text string that you want to wait for.
row	Integer	Required. The row where you expect the string to appear. Unlike with IBM Personal Communications, this parameter is not optional and value 0 for entire screen searches is not supported
col	Integer	Required. The column where you expect the string to appear. Unlike with IBM Personal Communications this parameter is not optional and value 0 for entire screen searches is not supported.
TimeOut	Integer	Optional. Specifies the maximum length of time in milliseconds to wait. The default is 30 second or the value set by SetTimeoutValue.
WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False. This parameter is ignored by Quick3270
bCaseSens	Boolean	Optional. If this value is True, the wait condition is verified as case-sensitive. Default value is False. This parameter is ignored by Quick3270

# Support for IBM® Personal Communications™ macros

# 13

## Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.  
`autECLPS.WaitForAttrib`

## Description

Waits until the specified attribute is displayed at the specified screen location.  
Returns True if the Attribute value is found,

## Syntax

`autECLSession.autECLPS.WaitForAttrib(row, col, Waitdata, [optional] MaskData, [optional] plane, [optional] TimeOut, [optional] WaitForIr)`

## Parameters

Part	Type	Description
row	Integer	Required. Row position of the attribute.
col	Integer	Required. Column position of the attribute.
Waitdata	Integer	Required. Specifies the value of the attribute to wait for.
MaskData	Integer	Optional. Specifies the value to use as a mask with the attribute. The default value is &HFF.
plane	Integer	Optional. Specifies the plane of the attribute to get. The default value is 3.
TimeOut	Integer	Optional. Specifies the maximum length of time in Milliseconds to wait. Default value is Infinite.
WaitForIr	Boolean	Optional. If this value is true, after meeting the wait condition the function will wait until the session is ready to accept input. Default value is False

## Returns

The function returns **True** if the condition is met or **False** if the Timeout value is exceeded.

## Remarks

The plane can have the following values:

- 1 Text Plane
- 2 Color Plane
- 3 Field Plane
- 4 Extended Field Plane

## Sample

```
autECLSession.autECLPS.WaitForAttrib 19,28,"00","3c",3,1000  
autECLFieldList
```

The `autECLFieldList` object performs operations on fields in an emulator presentation space. This object does not stand on its own. It is contained by `autECLPS`, and can only be accessed through an `autECLPS` object.

## macros

- [autECLFieldList](#)

## autECLXfer

The autECLXfer object provides file transfer services.

autECLXfer Methods

- [ReceiveFile](#)
- [SendFile](#)

autECLXfer.ReceiveFile

**Description**

This function allows receiving a file from a host session.

**Syntax**

**autECLSession.autECLXfer.ReceiveFile** *PcFile HostFile Options*

The **ReceiveFile** syntax has these arguments:

Part	Type	Description
PcFile	String	Required. Pc file name.
HostFile	String	Required. Host file name
Options	String	Optional. File transfer options

**Remarks**

Added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the options in EHLLAPI Receive File function.

autECLXfer.SendFile

**Description**

This function allows sending a file to a host session.

**Syntax**

**autECLSession.autECLXfer.SendFile** *PcFile HostFile Options*

The **SendFile** syntax has these arguments:

Part	Type	Description
------	------	-------------

## macros

PcFile	String	Required. Pc file name.
HostFile	String	Required. Host file name
Options	String	Optional. File transfer options

**Remarks**

Added in version 3.98

Following options are supported

- ASCII
- CRLF
- APPEND
- LRECL
- RECFM
- CLEAR/NOCLEAR
- PROGRESS
- QUIET
- VM/CMS/MVS/TSO/CICS

Other supported options are the same as the options in EHLLAPI Send File function.

**autECLWinMetrics**

The autECLWinMetrics object performs operations on an emulator window. It allows you to perform window rectangle and position manipulation.

Properties

- [autECLWinMetrics.WindowTitle](#)
- [autECLWinMetrics.Visible](#)
- [autECLWinMetrics.Minimized](#)
- [autECLWinMetrics.Maximized](#)
- [autECLWinMetrics.Restored](#)
- [autECLWinMetrics.Name](#)

autECLWinMetrics.WindowTitle

**Description**

This property may be change and retrieve the title bar text. WindowTitle is a String data type.

**Syntax**

```
autECLSession.autECLWinMetrics.WindowTitle = "Macro is running..."
autECLWinMetrics.Visible
```

**Description**

The visibility state of the emulator window. This property may be both changed and retrieved. Visible is a Boolean data type and is read/write enabled..

**Syntax**

```
autECLSession.autECLWinMetrics.Visible = True
```

## macros

autECLWinMetrics.Minimized

**Description**

The minimized state of the emulator window. This property may be both changed and retrieved. Minimized is a Boolean data type and is read/write enabled..

**Syntax**

```
autECLSession.autECLWinMetrics.Minimized = True
autECLWinMetrics.Maximized
```

**Description**

The maximized state of the emulator window. This property may be both changed and retrieved. Maximized is a Boolean data type and is read/write enabled..

**Syntax**

```
autECLSession.autECLWinMetrics.Maximized = True
autECLWinMetrics.Restored
```

**Description**

The restored state of the emulator window. This property may be both changed and retrieved. Restored is a Boolean data type and is read/write enabled.

**Syntax**

```
autECLSession.autECLWinMetrics.Restored = True
autECLWinMetrics.Name
```

**Description**

This property returns the short name of the session, a character ID (A-Z).

**Syntax**

```
Name = autECLSession.autECLWinMetrics.Name
autECLOIA
```

The autECLOIA object retrieves the status from the Host Operator Information Area.

autECLOIA Methods

- [WaitForInputReady](#)
- [WaitForAppAvailable](#)
- [WaitForSystemAvailable](#)

```
autECLOIA.WaitForInputReady
```

**Description**

The WaitForInputReady method waits the connection is able to accept keyboard input. This function is similar to the Quick3270 WaitForKbdUnlock function

**Syntax**

```
WaitForInputReady ([optional] TimeOut)
```

## Parameter

Part	Type	Description
TimeOut	Integer	Optional. The maximum length of time in milliseconds to wait, The default is Infinite.

## Sample

```
bResult = autECLSession.autECLOIA.WaitForInputReady(1000)
autECLOIA.WaitForAppAvailable
```

## Description

For compatibility only, this function is simply ignored by Quick3270.

## Syntax

**WaitForAppAvailable** (*[optional] TimeOut*)

## Parameter

Part	Type	Description
TimeOut	Integer	Optional. The maximum length of time in milliseconds to wait, The default is Infinite.

## Sample

```
bResult = autECLSession.autECLOIA.WaitForAppAvailable(1000)
autECLOIA.WaitForSystemAvailable
```

## Description

The WaitForSystemAvailable method waits until the OIA of the connection associated with the autECLOIA object indicates that the connection is connected to a host system.

## Syntax

**WaitForInputReady** (*[optional] TimeOut*)

## Parameter

Part	Type	Description
TimeOut	Integer	Optional. The maximum length of time in milliseconds to wait, The default is Infinite.

## Sample

```
bResult = autECLSession.autECLOIA.WaitForInputReady(1000)
autECLFieldList
```

The autECLFieldList object performs operations on fields in an emulator presentation space. This object does not stand on its own. It is contained by autECLPS, and can only be accessed through an autECLPS object.

## macros

## Properties

- [Count](#)
- [StartRow](#)
- [StartCol](#)
- [EndRow](#)
- [EndCol](#)
- [Length](#)
- [Modified](#)
- [Protected](#)
- [Numeric](#)
- [HighIntensity](#)
- [PenDetectable](#)
- [Display](#)

## Methods

- [Refresh](#)
- [FindFieldByRowCol](#)
- [FindFieldByText](#)
- [GetText](#)
- [SetText](#)

autECLFieldList.Refresh

**Description**

This function is ignored by Quick3270. Quick3270 automatically updates the field list.

**Syntax**

```
autECLFieldList.Refresh()
autECLFieldList.Count
```

**Description**

Returns the number of fields present in the autECLFieldList collection.

**Syntax**

```
autECLFieldList.Count
autECLFieldList.FindFieldByRowCol
```

**Description**

This function searches the autECLFieldList object for a field containing the given row and column coordinates.

**Syntax**

```
autECLFieldList.FindFieldByRowCol(Row, Col)
```

**Parameters**

Par t	Type	Description
Row	Intege r	Required. Field row to search for.

## macros

Col	Integer	Required. Field column to search for.
-----	---------	---------------------------------------

**Remarks**

Added in version 6.40

`autECLFieldList.FindFieldByText`

**Description**

This function searches the `autECLFieldList` object for a field containing the string passed in as `Text`.

**Syntax**

`autECLFieldList.FindFieldByText(Text)`

**Parameters**

Part	Type	Description
Text	String	Required. The text string to search for.
StartRow*	Integer	Optional. Row position in the presentation space at which to begin the search.
StartCol*	Integer	Optional. Column position in the presentation space at which to begin the search.
Direction*	Integer	Direction in which to search. Values are 1 for search forward, 2 for search backward.

\*These parameters are not supported in current version of Quick3270

**Remarks**

Added in version 6.40

`autECLFieldList(FieldIndex).Display`

**Description**

Indicates if a given field in the `autECLFieldList` collection is visible.

**Syntax**

`autECLFieldList(FieldIndex).Display`  
`autECLFieldList(FieldIndex).GetText`

**Description**

Retrieves the characters of a given field in an `autECLFieldList` collection.

**Syntax**

`Str = autECLFieldList(FieldIndex).GetText()`

## macros

`autECLFieldList(FieldIndex).SetText`**Description**

Populates a given field in an `autECLFieldList` item with the character string passed in as text. If the text exceeds the length of the field, the text is truncated.

**Syntax**

```
autECLFieldList(FieldIndex).SetText(Text)
```

**Parameters**

Par t	Type	Description
Text	String	Required. String to set in field.

```
autECLFieldList(FieldIndex).StartCol
```

**Description**

Returns the column position of the first character in a given field in the `autECLFieldList` collection.

**Syntax**

```
StartCol = autECLFieldList(FieldIndex).StartCol  
autECLFieldList(FieldIndex).StartRow
```

**Description**

Returns the row position of the first character in a given field in the `autECLFieldList` collection.

**Syntax**

```
StartRow = autECLFieldList(FieldIndex).StartRow  
autECLFieldList(FieldIndex).EndCol
```

**Description**

Returns the column position of the last character in a given field in the `autECLFieldList` collection.

**Syntax**

```
EndCol = autECLFieldList(FieldIndex).EndCol  
autECLFieldList(FieldIndex).EndRow
```

**Description**

Returns the row position of the last character in a given field in the `autECLFieldList` collection.

**Syntax**

```
EndRow = autECLFieldList(FieldIndex).EndRow
```

**macros**`autECLFieldList(FieldIndex).Length`**Description**

Returns the length of a given field in the autECLFieldList collection.

**Syntax**

```
Len = autECLFieldList(FieldIndex).Length  
autECLFieldList(FieldIndex).Modified
```

**Description**

Indicates if a given field in the autECLFieldList collection has a modified attribute.

**Syntax**

```
autECLFieldList(FieldIndex).Modified  
autECLFieldList(FieldIndex).Protected
```

**Description**

Returns True if the field in the autECLFieldList collection is Protected.

**Syntax**

```
Protected = autECLFieldList(FieldIndex).Protected  
autECLFieldList(FieldIndex).HighIntensity
```

**Description**

Returns True if the field in the autECLFieldList collection has a high intensity attribute.

**Syntax**

```
Numeric = autECLFieldList(FieldIndex).HighIntensity  
autECLFieldList(FieldIndex).PenDetectable
```

**Description**

Returns True if the field in the autECLFieldList collection has a pen detectable attribute.

**Syntax**

```
PenDetectable = autECLFieldList(FieldIndex).PenDetectable  
autECLFieldList(FieldIndex).Numeric
```

**Description**

Returns True if the field in the autECLFieldList collection has a numeric input only attribute.

**Syntax**

```
Numeric = autECLFieldList(FieldIndex).Numeric  
autSystem.Inputnd
```

**Description**

**macros**

Displays a popup input box to the user with a no-display text box so that when the user types in data only asterisks(\*) are displayed.

**Syntax**

**autSystem.Inputnd()**

**Returns**

The characters typed into the input box.

**Remarks****Added in version 4.30**

The possibility to run Attachmate's Extra!™ macros has been added in Quick3270 version **3.96**.

The Attachmate's Extra!™ compatibility mode allows Quick3270 to run directly macros recorded by Extra!™.

The following conditions must be verified to recognize an Extra!™ macro:

- File extension must be .ebm or .txt
- For .ebm files, the macro must have the standard Extra! macro header. Quick3270 will make some pointer and keyword check to identify an Extra! macro header
- The macro **must not be encrypted**
- For .txt files, no checks are made. Quick3270 assumes that all .txt files are Attachmate's Extra!™ macros

**Compatibility restrictions:**

- Quick3270 can only run macros that uses functions used by the Extra!™ macro recorder.
- System and Sess0 objects are default names created by the Extra! Macro recorder. Quick3270 uses the same name (not modifiable) for compatibility. The user should not change the default name of the objects or create Extra! Objects with other names: This is not supported by Quick3270.
- SendFile and ReceiveFile statements are supported, however Extra! file transfer schemes are not supported.

**List of supported methods**

- Sess0.Screen.Copy
- Sess0.Screen.CopyAppend
- Sess0.Screen.GetString
- Sess0.Screen.MoveTo
- Sess0.Screen.PutString
- Sess0.Screen.Rows
- Sess0.Screen.SelectAll
- Sess0.Screen.Sendkeys
- Sess0.Screen.Updated
- Sess0.Screen.WaitForCursor
- Sess0.Screen.WaitForCursorMove
- Sess0.Screen.WaitForHostQuiet
- Sess0.Screen.WaitForString
- Sess0.Visible
- Sess0.FileTransferHostOS
- Sess0.SendFile
- Sess0.ReceiveFile
- System.TimeoutValue

**List of ignored lines (the objects are created by default)**

- Dim Sessions As Object

## macros

- Dim Sess0 As Object
- Dim System As Object
- Set System = CreateObject("EXTRA.System")
- Set Sessions = System.Sessions
- Set Sess0 = System.ActiveSession
- Sess0.FileTransferScheme

**EMReadScreen****Description**

Returns data from the host screen.

**Syntax**

**EMReadScreen** *BufferStr, Len, Row, Col*

**Parameters**

Part	Type	Description
BufferStr	String	Required. Variable to contain host screen data.
Len	Integer	Required. Number of characters to retrieve..
Row	Integer	Required. Row at which to begin the retrieval in the presentation space.
Col	Integer	Required. Column at which to begin the retrieval in the presentation space.

**Remarks**

Null and non-printable characters are replaced by spaces.

**EMSendKey****Description**

Send keystrokes at the current cursor location.

**Syntax**

**EMSendKey** *KeyStr*

**Parameters**

Part	Type	Description
KeyStr	String	Required. Text that you want to put on the screen.

**Remarks**

Null and non-printable characters are replaced by spaces.

**EMWaitCursor****Description**

Suspends script execution until the host screen is ready for keyboard input and the cursor is at the specified location.

**Syntax**

**EMWaitCursor** *Timeout, Row, Col [,ExtraWait]*

# Support for Micro Focus Chameleon™/Rumba™ macros

## Parameters

Part	Type	Description
Timeout	Integer	Required. Timeout value in seconds.
Row	Integer	Required. Specifies the cursor row position in the presentation space
Col	Integer	Required. Specifies the cursor column position in the presentation space
ExtraWait	Integer	Optional. The number of milliseconds to validate for a keyboard unlocked status

## Remarks

Null and non-printable characters are replaced by spaces.

## FileWrite

### Description

Writes data to a sequential file.

### Syntax

**FileWrite** *FileName, Data*

## Parameters

Part	Type	Description
FileName	String	Required. File path and name.
Data	String	Required. String to write to file.

## FileAppend

### Description

Writes data to a sequential file.

### Syntax

**FileAppend** *Data*

## Parameters

Part	Type	Description
Data	String	Required. String to write to file.

TN3270 Plus™ macros (.txt files), Quick3270 uses the same in-built macro interpreter has for the Quick3270 macros.

Following the extensions specific for TN3270 Plus™ macros

1. Built-in variables

\$COLS	The number of screen columns.
\$COMPUTERNAME	The computer name.
\$CRLF	Carriage return and line feed.
\$CURSOR	Current cursor position on the screen relative to 1 (row 1 column 1).
\$OIA	The text in the OIA (Operator Information Area) line on the terminal screen.
\$ROWS	The number of screen rows.
\$SCRIPTFOLDER	The full path of the currently active script.
\$SCRIPTNAME	The name of currently active script.
\$SESSIONNAME	The short name of the session.
\$USERNAME	Current logged on user name.
\$SCREEN[(start[,length])]	The text at the specified location on the terminal screen.
\$SCREEN[(top,left,length)]	
\$SCREEN[(top,left,bottom,right)]	

## 2. List of specific functions or functions with a different syntax

- [Chr](#)
- [DDEExecute](#)
- [DDEInitiate](#)
- [DDEPoke](#)
- [DDERequest](#)
- [DDETerminate](#)
- [Command](#)
- [CursorTo](#)
- [Key](#)
- [MsgBox](#)
- [Replace](#)
- [Run](#)
- [Type](#)
- [Wait](#)
- [WaitFor](#)

## Chr

### Description

Convert an ANSI character code to a character.

### Syntax

\$character = **Chr**(charcode)

**Parameters****Remarks**

## DDEExecute

**Description**

Performs actions or runs commands in another application through dynamic data exchange (DDE).

**Syntax**

**DDEExecute** *\$variable,item[,onErrorLabel:]*

Or

**DDEExecute**(*\$variable,item[,onErrorLabel:]*)

**Parameters**

Part	Type	Description
\$variable	String	Required. the variable containing the DDE channel number. The DDE channel is established by the DDEInitiate command.
item	Integer	Required. the command sent to the DDE server application.
onErrorLabel	Integer	Optional. A label that execution transfers to if an error occurs.

**Remarks**

## DDEInitiate

**Description**

The DDEInitiate command begins a Dynamic Data Exchange (DDE) conversation between.

**Syntax**

**DDEInitiate** *\$variable,service,topic[,onErrorLabel:]*

Or

*\$variable* = **DDEInitiate**(*service,topic[,onErrorLabel:]*)

**Parameters**

Part	Type	Description
\$variable	Integer	Required. Variable that receives the DDE channel number.
Service	String	Required. The name of the DDE application for this conversation.
Topic	String	Required. The name of the DDE topic for this conversation.
onErrorLabel	Label	Optional. A label that execution transfers to if an error occurs.

**Remarks**

## DDEPoke

**Description**

The DDEPoke command sends text to a Dynamic Data Exchange (DDE) server application.

**Syntax**

**DDEPoke** *\$variable,item,data[,onErrorLabel:]*

Or

**DDEPoke**(*\$variable,item,data[,onErrorLabel:]*)

**Parameters**

Part	Type	Description
\$variable	Integer	Required. Variable containing the DDE channel number.
item	String	Required. The name of the item to be updated.
data	String	Required. Text to be sent to the DDE server application.
onErrorLabel*	Label	Optional. A label that execution transfers to if an error occurs.

**Remarks**

## DDERequest

**Description**

The DDERequest command receives text from a Dynamic Data Exchange (DDE) server application.

**Syntax**

**DDERequest** *\$variable,item,\$result[,onErrorLabel:]*

Or

*\$return* = **DDERequest**(*\$variable,item[,onErrorLabel:]*)

**Parameters**

Part	Type	Description
\$variable	Integer	Required. Variable containing the DDE channel number.
item	String	Required. The name of the item to be updated.
\$result	String	Required. variable that receives the result of the request.
onErrorLabel	Label	Optional. A label that execution transfers to if an error occurs.

**Remarks**

## DDETerminate

**Description**

The DDETerminate command closes the Dynamic Data Exchange (DDE) channel.

**Syntax**

**DDETerminate** *\$variable*

Or

**DDETerminate**(*\$variable*)

**Parameters**

Part	Type	Description
\$variable	Integer	Required. Variable containing the DDE channel number.

**Remarks**

## Command

**Description**

Returns data from the host screen.

**Syntax**

**command** *command*

Or

**command**(*command*)

**Parameters**

Part	Type	Description
command	String	Required. Command to be issued in the Windows command environment. Enclose the command in double quotation marks (") if it contains embedded spaces, tabs or commas. This parameter may be a variable containing the command to be issued.

**Examples**

```
command("rename test.txt test1.txt")
```

```
command("delete c:\test1.txt")
```

```
command($COMMAND).
```

## CursorTo

**Description**

Moves the cursor to the specified position or the specified row and column. The upper left-hand corner of the screen is position 1 or row 1 column 1. The lower right-hand corner of a 24 by 80 screen is position 1920 or row 24 column 80..

**Syntax**

**CursorTo**(*position*) or **CursorTo** *position*

**CursorTo**(*row,column*) or **CursorTo** *row,column*

#### Parameters

Part	Type	Description
Position	Integer	Required. The screen position relative to one.
Row	Integer	Required. Row number.
Col	Integer	Required. Column number.

#### Examples

CursorTo(1)

CursorTo(81)

CursorTo(1,1)

CursorTo(24,80)

CursorTo(\$POSITION)

CursorTo(\$ROW,\$COLUMN).

#### Key

#### Description

Send keystrokes at the current cursor location.

#### Syntax

**Key** *Keyname*

#### Parameters

Part	Type	Description
Keyname	Key ID	Required. The name of the key.

#### Examples

key(enter)

key(tab)

key(PA2)

key(PF1)

#### MsgBox

#### Description

Displays a message in a dialog box and waits for the user to click the OK or Cancel button. Click the OK button to continue script processing. Click the Cancel button to cancel the script.

#### Syntax

**MsgBox**(*prompt*[[, *icon*][, *title*]])

or

**MsgBox** *prompt*[[, *icon*][, *title*]]**Parameters**

Part	Type	Description
prompt	String	String expression displayed as the message in the dialog box. The maximum length of prompt is approximately 1024 characters, depending on the width of the characters used. If prompt consists of more than one line, you can separate the lines using a carriage return character (Chr(13)), a linefeed character (Chr(10)), or carriage return–linefeed character combination (Chr(13) & Chr(10)) between each line. You can even use the system defined constants like vbCR, vbLf, vbCrLf...
icon	String	Optional. Message box icon id. ICONEXCLAMATION is the default
title	String	Optional. String expression displayed in the title bar of the dialog box. If you omit title, the application name is displayed in the title bar.

**MsgBox Constants**

The following constants are used with the MsgBox function to identify what icon is displayed.

Constant	Description
ICONNONE	No icon is displayed.
ICONSTOP	Display Stop (white X in a circle with a red background.) icon.
ICONQUESTION	Display Warning Query icon.
ICONEXCLAMATION	Display Warning Message icon.
ICONINFORMATION	Display Information Message icon.

**Replace****Description**

Returns a string in which a specified substring has been replaced with another substring.

**Syntax**

*\$return* = **replace**(*string*,*oldString*,*newString*)<sup>1</sup>

or

**replace** *\$variable*,*oldString*,*newString*<sup>2</sup>

**Parameters<sup>1</sup>**

Part	Type	Description
<i>\$return</i>	String	Required. The string with the specified characters replaced.
<i>string</i>	String	Required. Variable containing text to be replaced..
<i>oldString</i>	String	Required. The string to be replaced.
<i>newString</i>	String	Required. The replacement string.

**Parameters<sup>2</sup>**

Part	Type	Description
<i>\$variable</i>	String	Required. Variable containing text to be replaced.
oldString	String	Required. The string to be replaced.
newString	String	Required. The replacement string.

## Run

### Description

Runs the specified application or batch file.

### Syntax

**run**(*filename*[,*windowstyle*])

or

**run** *filename*[,*windowstyle*]

### Parameters

Part	Type	Description
filename	String	Required. The full file name, including drive letter and path, of an application or batch file.
windowstyle	String	Optional. String corresponding to the style of the window in which the program is to be run.

The windowstyle named argument has these values:

Constant	Description
HIDDEN	Window is hidden.
MINIMIZED	Window is displayed as an icon.
MAXIMIZED	Window is maximized.

## Type

### Description

The type command enters the string at the current cursor location.

### Syntax

**type**("string")

or

**type** "string"

### Parameters

Part	Type	Description
string	String	Required. String expression to set in field..

**Remarks**

Null and non-printable characters are replaced by spaces.

**Wait****Description**

Pauses script processing for the specified number of milliseconds.

**Syntax**

**wait**(*milliseconds*)  
or  
**wait** *milliseconds*

**Parameters**

Part	Type	Description
milliseconds	Integer	Required. The number of milliseconds script processing should wait before continuing with the next command.

**WaitFor****Description**

Wait until one of the specified string(s) is displayed. When a string is found, script processing continues at the position indicated by the label.

When the WaitFor command is used with a single string parameter with no label, script processing continues with the next command.

**Syntax**

**WaitFor**(*string1[,label1:] [,string2,label2:...,string10,label10:]* [,*minimized | hidden*][*WindowPos*])  
or  
**WaitFor** *string1[,label1:] [,string2,label2:...,string10,label10:]* [,*minimized | hidden*][*WindowPos*]

**Parameters**

Part	Type	Description
stringxx	String	Required. Variable to contain host screen data.
labelxx	String	Optional. The name of the label in the script When the string is found, script processing continues with the next command.
minimized   hidden	String	Optional. Specifies how the Wait dialog box is displayed. With hidden, no dialog box is displayed. With the wait dialog box, the user has the option to cancel the wait
WindowPos		This parameter is not supported in Quick3270.

## Select an Excel file using GetOpenFileName

```
;Create objects
WshShell = CreateObject("WScript.Shell")

strFileName = GetOpenFileName("Select File", "", "Excel Files *.xls|*.xls")
WshShell.popup(strFileName)
WshShell = Nothing
End
```

## Creating a Reference to an Excel Object

```
;Create objects
appExcel = CreateObject("Excel.Application")
appExcel.Visible = True
Set iLine = 1
appExcel.activesheet.cells(2,1).value = GetString(iLine+6, 6, 8)
...
appExcel = Nothing
End
```

## File I/O

```
iFile = Freefile
sFile = "C:dummy.txt"
sTmp = "      "
result = MsgBox ("sFile=" & sFile , vbOK ,"Continue")
result = MsgBox ("iFile=" & str(iFile) , vbOK ,"Continue")
Open sFile For Input as iFile
result = MsgBox ("About to prime read",vbOK,"Continue")
LineInput #iFile, sTmp
result = MsgBox ("First read done",vbOK,"Continue")
While NOT EOF (iFile)
    MsgBox sTmp, vbOK ,"Processing"
    LineInput #iFile, sTmp
Wend
Close #iFile
result = MsgBox ("Finished",vbOK,"Continue")
End
```

## Connecting to a data source and executing SQL Queries

```
connectSQL = CreateObject("ADODB.Connection")
recordsetSQL = CreateObject("ADODB.RecordSet")
recordsetSQL.ActiveConnection = connectSQL
recordsetSQL.Source = "Select * from myRange1"
recordsetSQL.Open
Adrs_L = recordsetSQL.Fields.Item("A1").value
...
End
```

## Find Values from screen and use the WScript object

```
;Create objects and init variables
WshShell = CreateObject("WScript.Shell")
Screen=""
sFind=0
FIN=""
;Find Values from screen
Screen = GetString(1,1,1920)
sFind = inStr(Screen,"LSFYFND")
FIN=MID(Screen,(sFind + 9),5)
WshShell.popup(sFind)
if sFind > 0 then
    sFind = inStr(Screen,"IVGFINAL")
    FIN = MID(Screen,(sFind + 9),5)
endif

;Show Popup Value
WshShell.popup(FIN)

;Destroy Open Objects and init Variables
WshShell = Nothing
Screen=""
FIN=""
sFind=0
End
```

## Simple logon script

```
HostSettleTime = 3000 ; milliseconds
Result = WaitForCursor(19, 64)
SendKeys "logon<Enter>"
Result = WaitForCursor(5, 40)
SetWindowTitle "Macro" + " Quick3270"
sPassword = PasswordBox("Enter Password", "Logon" )
SendKeys "USERNAME"
b = MoveTo(6,40)
SendKeys sPassword
SendKeys "<Enter>"
End
```

## Using LDAP to retrieve the user name and password and start logon script

```
Set QuickExit = False
Set jRoot = GetObject("LDAP://RootDSE")
DomainPath = jRoot.Get("DefaultNamingContext")
Set Domain = GetObject("LDAP://" + DomainPath)
DomainPath = "LDAP://sl-mes.intra.company.com/DC=intra,DC=company,DC=com"
Set con = CreateObject("ADODB.Connection")
Set Com = CreateObject("ADODB.Command")
con.Provider = "AdsDSOObject"
con.Open "Active Directory Provider"
Set Com.ActiveConnection = con
Com.CommandText = "select HostUser, HostPassword from '" + DomainPath + "' WHERE
objectClass='user' AND sAMAccountName='" + sAMAccountName + "'"
Set rs = Com.Execute
if Not rs.EOF then
    HostUserName = rs.Fields("HostUserName")
    HostPassword = rs.Fields("HostPassword")
    if Len(HostUserName) = 0 Then
        rc = msgbox("User name unknown" , vbokonly + vbExclamation, "Error")
    else
        ; We have the user name and password. Start the logon process
        Result = WaitForString("Welcome", 1, 38)
        if Result = True then
            Result = MoveTo(29,47)
            SendKeys HostUserName.Value
            Result = MoveTo(30,47)
            SendKeys HostPassword.Value
            SendKeys "<Enter>"
            Result = WaitForKbdUnlock()
            r = GetString(28, 2, 8)
            if r = "rejected" then
                rc = msgbox("User already logged in" , vbokonly + vbExclamation,
"Error")
                Set QuickExit = True
            endif
        endif
    endif
endif

con.Close
rs = Nothing
Com = Nothing
con = Nothing
Domain = Nothing
jRoot = Nothing
End
```

## IBM Personal Communications VBScript sample supported by Quick3270

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=VBScript
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
autECLSession.SetConnectionByName(ThisSessionName)

REM This line calls the macro subroutine
subSub1_

sub subSub1_()
    autECLSession.autECLOIA.WaitForAppAvailable

    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "a"
    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[enter]"

    autECLSession.autECLPS.WaitForAttrib 24,5,"00","3c",3,10000

    autECLSession.autECLPS.Wait 2937

    autECLSession.autECLOIA.WaitForAppAvailable

    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[pf1]"

    autECLSession.autECLPS.WaitForAttrib 24,80,"08","3c",3,10000

    autECLSession.autECLPS.WaitForCursor 1,1,10000

    autECLSession.autECLOIA.WaitForAppAvailable

    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "/rcl"
    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[enter]"

    autECLSession.autECLPS.WaitForAttrib 24,5,"00","3c",3,10000

    autECLSession.autECLPS.Wait 4860

    autECLSession.autECLOIA.WaitForAppAvailable

    autECLSession.autECLOIA.WaitForInputReady
    autECLSession.autECLPS.SendKeys "[pf3]"
end sub
```

## IBM Personal Communications Macro sample supported by Quick3270

```
Description =Test Macro
run /min notepad.exe
[wait app]
[wait inp inh]
"a
[enter]
[wait inp inh]
wait 10 sec until FieldAttribute 0000 at (24,5)
wait 10 sec until cursor at (24,6)
[wait app]
[wait inp inh]
"sos
[enter]
[wait inp inh]
wait 10 sec until FieldAttribute 0008 at (24,80)
wait 10 sec until cursor at (1,1)
[wait app]
[wait inp inh]
"logon
[enter]b
```

## Attachmate's Extra! macro sample supported by Quick3270

```

'-----
' This macro was created by the Macro Recorder.
' Session Document: "DISPLAY LOCALHOST.EDP"
' Date: Thursday, November 08, 2009 15:19:52
' User: Denis
'-----

' Global variable declarations
Global g_HostSettleTime%

Sub Main()
'-----
' Get the main system object
Dim Sessions As Object
Dim System As Object
Set System = CreateObject("EXTRA.System")      ' Gets the system object
If (System is Nothing) Then
    MsgBox "Could not create the EXTRA System object."
    STOP
End If
Set Sessions = System.Sessions

If (Sessions is Nothing) Then
    MsgBox "Could not create the Sessions collection object."
    STOP
End If
'-----
' Set the default wait timeout value
g_HostSettleTime = 3000      ' milliseconds

OldSystemTimeout& = System.TimeoutValue
If (g_HostSettleTime > OldSystemTimeout) Then
    System.TimeoutValue = g_HostSettleTime
End If

' Get the necessary Session Object
Dim Sess0 As Object
Set Sess0 = System.ActiveSession
If (Sess0 is Nothing) Then
    MsgBox "Could not create the Session object.  Stopping macro playback."
    STOP
End If
If Not Sess0.Visible Then Sess0.Visible = TRUE
Sess0.Screen.WaitHostQuiet(g_HostSettleTime)

' This section of code contains the recorded events
Sess0.Screen.Sendkeys("F<Enter>")
Sess0.Screen.WaitHostQuiet(g_HostSettleTime)
Sess0.Screen.Sendkeys("<Pf3>")
Sess0.Screen.WaitHostQuiet(g_HostSettleTime)
System.TimeoutValue = OldSystemTimeout
End Sub

```

# Index

## A

Attachmate 73  
 autECLFieldList 68, 69, 70, 71, 72  
 autECLOIA 67, 68  
 autECLPS 56, 57, 58, 60, 61, 62, 63, 64  
 autECLSession 54, 55  
 autECLWinMetrics 66, 67  
 autECLXfer 65  
 autSystem 72

## B

Beep 30  
 Build-In 6

## C

Call 8  
 Chr 13, 76  
 Close 18  
 ClosePrinter 43  
 Command 79  
 ConnType 57  
 Const 6  
 Constants 6  
 ConverCol 35  
 ConvertRow 36  
 Count 69  
 CreateObject 28  
 CSIDL\_APPDATA 31  
 CSIDL\_COMMON\_APPDATA 31  
 CSIDL\_COMMON\_DOCUMENTS 31  
 CSIDL\_PROGRAM\_FILES 31  
 CStr 13  
 CursorPosCol 57  
 CursorPosRow 57  
 CursorTo 79

## D

Date 21, 22  
 Date\$ 22  
 DateAdd 24  
 DateSerial 25  
 DateValue 25  
 Day 23  
 DDEExecute 77

DDEInitiate 77  
 DDEPoke 78  
 DDERequest 78  
 DDETerminate 79  
 Decrypt 30  
 Dim 5  
 Display 70  
 Do 8

## E

EditCopy 36  
 EditCopyAdd 36  
 EditPaste 36  
 EditSelectAll 37  
 Else 10  
 EMReadScreen 74  
 Encrypt 30  
 End 8, 11, 12  
 EndCol 71  
 Endif 10  
 EndRow 71  
 Environ 30  
 EOF 20  
 Exit 8, 9  
 EXTRA 73

## F

FindFieldByRowCol 69  
 FindFieldByText 70  
 For 9  
 Format 26  
 FreeFile 21

## G

General Date 26  
 Get 20  
 GetCol 37  
 GetColor 38  
 GetCols 37  
 GetIpAddress 39  
 GetIpAddressV6 39  
 GetMacroKeystroke 39  
 GetObject 29  
 GetOIA 40  
 GetOpenFileName 31  
 GetRow 37

GetRowCount 38  
 GetRows 38

GetSpecialFolderLocation 31  
 GetString 41  
 GetText 41, 61, 70  
 GetTextRect 41, 61  
 GetTimeoutValue 42  
 GetVisible 42  
 GetWindowTitle 42  
 GoSub 9  
 GoTo 10

## H

HighIntensity 72

## I

if 10  
 Input 18  
 InputBox 31  
 Inputnd 72  
 InStr 13  
 Is 6  
 IsEmpty 6  
 IsNull 7  
 IsNumeric 7

## K

Key 80  
 Kill 21

## L

LCase 14  
 Left 14  
 Len 14  
 Length 72  
 Line 18  
 LineInput 18  
 literals 21  
 Loc 21  
 LOF 21  
 Long Date 26  
 Long Time 26  
 Loop 8

## M

Maximized 67  
 Medium Date 26  
 Medium Time 26  
 Mid 15  
 Minimized 67  
 Modified 72  
 Month 23  
 MoveRelative 43  
 MoveTo 42  
 MsgBox 32, 80

## N

Name 54, 67  
 Next 9  
 Now 27  
 NumCols 56  
 Numeric 72  
 NumRows 56

## O

Open 17  
 OpenPrinter 43

## P

PasswordBox 34  
 PenDetectable 72  
 Print 19  
 PrintScreen 44  
 PrintScreenFontSize 43  
 Protected 72  
 Put 20  
 PutString 44

## Q

Quit 11

## R

ReceiveFile 45, 65  
 Refresh 69  
 Rem 34  
 Replace 15, 81

Restored 67  
Return 11

Right 15  
Run 82

## S

Search 45  
SearchPS 46  
SearchText 60  
Seek 19  
SendDarkKeys 46  
SendFile 46, 65  
SendKeys 47, 58  
Session 9  
SetConnectionByName 55  
SetCursorPos 57  
SetText 49, 61, 71  
SetTimeoutValue 50  
SetVisible 50  
SetWindowTitle 50  
Shell 34  
Short Date 26  
Short Time 26  
Space 16  
StartCol 71  
StartCommunication 55  
StartRow 71  
Stop 11  
StopCommunication 55  
Str 16  
StrComp 16  
Sub 9, 11  
Symbolic 6

## T

Then 10  
Time 21, 23  
Time\$ 23  
TimeSerial 28  
TimeValue 28  
TN3270Plus 75, 76, 77, 78, 79, 80, 81, 82, 83  
Trim 17  
Type 82

## U

UCase 17

Updated 51

## V

Val 17  
VarType 7  
vbAbort 32  
vbAbortRetryIgnore 32  
vbApplicationModal 32  
vbCancel 32  
vbCritical 32  
vbDefaultButton1 32  
vbDefaultButton2 32  
vbDefaultButton3 32  
vbDefaultButton4 32  
vbExclamation 32  
vbIgnore 32  
vbInformation 32  
vbMsgBoxHelpButton 32  
vbMsgBoxRight 32  
vbMsgBoxRtlReading 32  
vbMsgBoxSetForeground 32  
vbNo 32  
vbOK 32  
vbOKCancel 32  
vbOKOnly 32  
vbQuestion 32  
vbRetry 32  
vbRetryCancel 32  
vbSystemModal 32  
vbYes 32  
vbYesNo 32  
vbYesNoCancel 32  
Visible 66

## W

Wait 62, 83  
WaitFor 83  
WaitForAppAvailable 68  
WaitForAttrib 51, 64  
WaitForCursor 51, 62  
WaitForCursorMove 52  
WaitForInputReady 67  
WaitForKbdUnlock 52  
WaitForString 53, 63  
WaitForSystemAvailable 68  
WaitHostQuiet 53  
WaitWhileCursor 63

Wend 12  
While 8, 12  
WindowTitle 66  
With 12  
Write 19

## Y

Year 24