eXpress Script Editor Help

6 May 2008

Table of Contents

eXpress Script Editor 1
File menu 1
New (Ctrl+N) 1
Open (Ctrl+O) 1
Save (Ctrl+S) 1
Save As... 1
Close 1
Close All 1
Print (Ctrl+P) 1
Printer Setup... 1
Clear Previous File List 1
Editor Properties 1
Exit 1
Previous File List 1
Edit menu 1
Undo (Ctrl+Z) 1
Redo (Ctrl+Shift+Z) 1
Cut (Ctrl+X) 1
Copy (Ctrl+C) 2
Paste (Ctrl+V) 2
Delete (Ctrl+D) 2
Select All (Ctrl+A) 2
Word Wrap (Ctrl+W) 2
Search menu 2
Find... (Ctrl+F) 2
Find Next (F3) 2
Replace... (Ctrl+R) 2
Go to Line... (Ctrl+G) 2
Bookmarks 2
Set Bookmark 1 through 5 (Shift+F1 through Shift+F5) 2
Go to Bookmark 1 through 5 (Ctrl+F1 through Ctrl+F5) 2
Options menu 2
Show Tool Bar 2
Syntax Highlight 2
View Permanent Declarations 2
Tools menu 2
Check Script (F4) 2
Compile Script (F5) 3
Script Mass Compiler 3
Dialog Designer (F10) 3
Help 3
Contents 3
This Window 3
About... 3
Editor Properties 5
Edit Window Font 5
Font Name 5
Size 5
Bold 5

Tab Size 5
Highlight Colors 5
Set Text Color 5
Set Background Color 5
OK 5
Cancel 5
Help 5
Color Selection 7
Languge Elements 9
Comments 9
Statements 9
Line Continuation Character 9
Numbers 9
Variable and Constant Names 9
Variables 11
Variable Types 11
Variant 11
Variants and Concatenation 11
Other Data Types 11
Scope of Variables 12
Declaration of Variables 12
Flow of Control 13
Control Structures 13
The GoTo 13
The Do Loops 13
The While Loop 13
The For ... Next Loop 13
The If and Select Statements 13
Subroutines and Functions 15
Subroutine and Function Naming Conventions 15
ByRef and ByVal 15
Calling Procedures in DLLs 16
Files 17
File Input/Output 17
Arrays 19
Arrays 19
User Defined Types 21
User Defined Types 21
Dialogs and Dialog Controls 23
Dialog Support 23
OK and Cancel 23
List Boxes and Drop-down List Boxes 24
Check Boxes 25
Text Boxes and Text 26
Option Buttons and Group Boxes 27
The Dialog Function 28
The Dialog Box Controls 28
The Dialog Function Syntax 28
OLE Automation 31
What is OLE Automation? 31
Accessing an Object 31

What is an OLE Object? 32
OLE Fundamentals 32
OLE Object 33
OLE Automation 33
Class 33
OLE Automation and Word 6.0 example 33
Making Applications Work Together 33
WIN.INI 34
The Registration Database 34
Associations 34
Shell Operations 34
OLE Object Servers 34
DDE/OLE Automation 34
Quick Reference 35
Language Reference 35
Functions, Subroutines, Statements, Reserved Words 35
Data Types, Operators and Precedence 36
Data Types 36
Arithmetic Operators 36
Operator Precedence 36
Relational Operators 36
Comparison Operators 36
Functions/Statements/Subroutines by Category 37
Functions/Statements/Subroutines – Alphabetical Listing 37
Functions, Statements and Subroutines 39
Abs Function 39
ActivateScreen Subroutine (eXpress Plus) 39
AppActivate Statement 39
Asc Function 40
Atn Function 40
Beep Statement 40
Begin Dialog Statement 40
Call Statement 42
CancelButton Statement 43
CBool Function 43
CDbl Function 44
ChDir Statement 44
ChDrive Statement 44
CheckBox Statement 45
Choose Function 45
Chr Function 45
CInt Function 46
CLng Function 46
Close Statement 46
ComboBox Statement 47
Const Statement 47
CopyToClipboard Subroutine (eXpress Plus) 48
Cos Function 48
CreateObject Function 48
CSng Function 49
CStr Function 49

CurDir Function 50
CurrentPage Function (UTS eXpress Plus) 50
CurrentScreen Function (eXpress Plus) 50
CVar Function 51
Date Function 51
DateSerial Function 51
DateValue Function 51
Day Function 52
Declare Statement 52
Dialog Function 53
Dim Statement 54
Dir Function 54
DlgText Statement 55
DlgVisible Statement 55
Do...Loop Statement 55
DropListBox Statement 56
End Statement 56
DlgEnable Statement 57
EnterText Subroutine (eXpress Plus) 58
EOF Function 58
Erase Statement 58
Exit Statement 59
Exp Function 59
FileCopy Function 59
FileLen Function 60
FileOpenDialog Function (eXpress Plus) 60
FileSaveDialog Function (eXpress Plus) 60
Fix Function 61
For Each...Next Statement 61
For...Next Statement 61
Format Function 62
FreeFile Function 67
Function Statement 68
Get Statement 68
GetCursorCol Function (eXpress Plus) 68
GetCursorRow Function (eXpress Plus) 69
GetLastMsg Function (eXpress Plus) 69
GetObject Function 69
GetPage Subroutine (UTS eXpress Plus) 69
GetScreenAttribute Function (eXpress Plus) 70
GetScreenColor Function (eXpress Plus) 72
GetScreenLine Function (eXpress Plus) 72
GetScreenText Function (eXpress Plus) 74
GetWindowState Function (eXpress Plus) 77
GetUserParam Function (eXpress Plus) 77
Global Statement 78
GoTo Statement 79
GoToPage Subroutine (UTS eXpress Plus) 79
GroupBox Statement 80
Hex Function 80
HoldMessages Subroutine (eXpress Plus) 80

Hour Function 80
If...Then...Else Statement 81
Input Function 82
InputBox Function 82
InStr Function 82
Int Function 83
IsArray Function 83
IsDate Function 83
IsEmpty Function 83
IsNull Function 84
IsNumeric Function 84
IsObject Function 84
KeyboardState Function (eXpress Plus) 85
Kill Statement 85
LBound Function 85
LCase Function 86
Left Function 86
Len Function 87
Let Statement 87
Line Input # Statement 87
ListBox Statement 88
LoadScreen Subroutine (eXpress Plus) 89
LOF Function 89
Log Function 89
MarkBlock Subroutine (eXpress Plus) 90
MessageWaitingState Function (eXpress Plus) 90
Mid Function 90
Minute Function 91
MkDir Statement 91
Month Function 91
MsgBox Function, MsgBox Statement 92
Name Statement 93
Now Function 93
Oct Function 93
OKButton Statement 94
On Error Statement 94
Open Statement 96
Option Base Statement 97
Option Explicit Statement 97
OptionButton Statement 97
OptionGroup Statement 98
PasteFromClipboard Subroutine (eXpress Plus) 98
Print # Statement 98
Print Statement 100
PrintBeginDoc Subroutine (eXpress Plus) 100
PrintEndDoc Subroutine (eXpress Plus) 101
PrintMoveTo Subroutine (eXpress Plus) 101
PrintNewPage Subroutine (eXpress Plus) 101
PrintPageHeight Function (eXpress Plus) 101
PrintPageWidth Function (eXpress Plus) 101
PrintRect Subroutine (eXpress Plus) 102

PrintSelect Function (eXpress Plus) 102
PrintSetFont Subroutine (eXpress Plus) 102
PrintSetFontSize Subroutine (eXpress Plus) 102
PrintSetFontStyle Subroutine (eXpress Plus) 103
PrintSetOrientation Subroutine (eXpress Plus) 103
PrintTextHeight Function (eXpress Plus) 104
PrintTextWidth Function (eXpress Plus) 104
PushButton Statement 104
Put Statement 105
Randomize Statement 105
Receive Function (eXpress Plus) 106
ReDim Statement 106
RefreshScreen Subroutine (eXpress Plus) 106
Rem Statement 106
Right Function 107
RmDir Statement 107
Rnd Function 108
SaveScreen Subroutine (eXpress Plus) 108
ScreenAvailable Function (eXpress Plus) 108
ScreenOpen Function (eXpress Plus) 109
Second Function 109
Seek Function 109
Seek Statement 109
Select Case Statement 110
SendKeys Statement 111
Set Statement 112
SetCursor Subroutine (eXpress Plus) 112
SetDblClickAction Subroutine (eXpress Plus) 113
SetScreenText Subroutine (eXpress Plus) 113
SetSignOnResult Subroutine 114
SetWindowState Subroutine (eXpress Plus) 114
Sgn Function 114
Shell Function 114
Sin Function 115
Space Function 115
Sqr Function 115
Static Statement 115
Stop Statement 116
StorePage Subroutine (UTS eXpress Plus) 116
Str Function 117
StrComp Function 117
String Function 118
Sub Statement 118
SwitchToolbar Subroutine (eXpress Plus) 119
T27Key Subroutine (eXpress Plus) 119
Tan Function 120
Text Statement 120
TextBox Statement 120
Time Function, Time Statement 121
Timer Event 121
TimeSerial Function 122

TimeValue Function 122
Trim, LTrim, RTrim Functions 122
Type Statement 123
UBound Function 124
UCase Function 124
UnholdMessages Subroutine (eXpress Plus) 125
UTSKey Subroutine (eXpress Plus) 125
Val Function 126
VarType Function 126
Wait Subroutine (eXpress Plus) 126
WaitForSpecificString Function (eXpress Plus) 127
WaitForString Function (eXpress Plus) 127
Weekday Function 127
While...Wend Statement 127
WinHelp Subroutine (eXpress Plus) 128
With Statement 129
Write # Statement 130
Year Function 130
Constants 131
Predefined Constants 131
UTS Control Characters, Escape Sequences and FCCs 135
UTS Control Characters 135
UTS Control and Escape Sequences 135
UTS Field Control Characters 136
UTS Special Emphasis Character Codes 140

eXpress Script Editor

This window provides the means to develop and edit eXpress Scripts. The script editor contains multiple tabs that allow more than one script to be edited at a time.

Following is a description of each eXpress Script Editor command. All the commands may be performed by making a menu selection. Toolbar buttons, shortcut keys and right-mouse-click actions are available for frequently used commands. A right mouse click anywhere in the test area will produce a pop-up menu of commands.

The menu items below show the shortcut key combination (in parentheses) where applicable.

File menu

The **File** menu contains commands to maintain script files and setup printing.

New (Ctrl+N)

Use this command to create a new script file (.BAS).

Open (Ctrl+O)

Use this command to open an existing script file or user library (USER.LIB). Note: Script Functions and Subroutines may be placed in a user library and thus become global to any script that needs to call them. The user library must reside in the SCRIPTS directory under the product installation directory.

Save (Ctrl+S)

Use this command to save the current script file.

Save As...

Use this command to save the current script file to another file name or user library (USER.LIB).

Close

Close the currently selected script.

Close All

Close all open scripts.

Print (Ctrl+P)

Use this command to print the entire script.

Printer Setup...

Allow margins to be set and allow printers and printer fonts to be selected for printing.

Clear Previous File List

Remove all file names from the list of previously accessed files.

Editor Properties

Use this command to edit the properties of the script editor: window font, highlight colors and tab stops.

Exit

Exit the Script Editor. If you entered the eXpress Script Editor from the eXpress Script Manager, you will receive a message box stating that the application was launched via Automation and should only be closed from a client. This message is normal. Simply press the **Yes** button.

Previous File List

Select (open) from the list of previous accessed script files.

Edit menu

The **Edit** menu contains commands to manage selected text between the editor and the Windows clipboard.

Undo (Ctrl+Z)

Use this command to reverse the effects of the most recent change.

Redo (Ctrl+Shift+Z)

Use this command to reverse the effects of the most recent **Undo** command.

Cut (Ctrl+X)

Use this command to place the selected text on the clipboard and delete.

Copy (Ctrl+C)

Use this command to copy the selected text to the clipboard.

Paste (Ctrl+V)

Use this command to paste the contents of the clipboard to the current cursor position.

Delete (Ctrl+D)

Use this command to delete the selected text without copying to the clipboard.

Select All (Ctrl+A)

Use this command to select all text in the script.

Word Wrap (Ctrl+W)

Use this command as a toggle. By default, long lines may only viewed/edited by first bringing the excess text into view with the horizontal scroll bar or by using the cursor keys (arrows). When **Word Wrap** is set, long lines wrap to the next line and are viewable within the confines (width) of the window.

Search menu

The **Search** menu contains commands to locate and change text within the script.

Find... (Ctrl+F)

Use this command to enable the **Find** dialog used to locate text strings.

Find Next (F3)

Use this command to find the next occurrence of the same string used on the previous find.

Replace... (Ctrl+R)

Use this command to enable the **Replace** dialog used to locate and replace text strings.

Go to Line... (Ctrl+G)

Use this command to go to a specific line.

Bookmarks

The **Bookmarks** menu contains commands to mark lines and navigate within the script.

Set Bookmark 1 through 5 (Shift+F1 through Shift+F5)

Use one of these commands to mark a line at the current text cursor position. A bookmarked line will appear with a grey background.

Go to Bookmark 1 through 5 (Ctrl+F1 through Ctrl+F5)

Use one of these commands to go to a line previously bookmarked by one of the five corresponding **Set Bookmark** commands.

Options menu

The **Option** menu contains commands to specify color, font and tab stop preferences.

Show Tool Bar

Use this command to toggle the display of the toolbar.

Syntax Highlight

Use this command to toggle the display of the script syntax. This command is affected by the settings of the **Editor Properties** command, above.

View Permanent Declarations

Use this command to show the permanent Enable declarations plus any USER.LIB code that is automatically included with all Express Enable Scripts. You browse permanent declarations in read-only mode; however, you may copy code from the browse window to the Windows clipboard using the **Ctrl+C** key.

Tools menu

The **Tools** menu contains commands to check syntax, compile scripts to binary files and initiate the Dialog Form Designer.

Check Script (F4)

Use this command to check the syntax of the entire script.

Compile Script (F5)

Use this command to compile the script and save it in encrypted form (.BAX). This option is useful for sites that wish to secure the contents of script files from general viewing (e.g., user-id/password). Either the text form (.BAS) or the compiled form of the script may be made available to the user for execution.

Script Mass Compiler...

Use this selection to compile all scripts in a selected directory.

Dialog Designer (F10)

Use this command to initiate the Dialog Form Designer. To edit an existing dialog, place the dialog on the Windows Clipboard before using this command.

Help

The **Help** menu contains commands to display on-line help and information about the product.

Contents

Use this command to display the contents of the eXpress Plus on-line help.

This Window

Use this command to receive on-line help for this window.

About...

Use this command to display copyright and product version information.

Editor Properties

This dialog is used to change the properties or appearance of a script window.

Edit Window Font

The controls in this group affect the font typeface, size and intensity used to display the script.

Font Name

From this drop-down list box, choose from the list of non-proportional, fixed fonts installed on your PC.

Size

With this spin wheel, increase or decrease the font size.

Bold

Check this box to increase the font intensity.

Tab Size

With this spin wheel, increase or decrease the number characters between tab characters.

Highlight Colors

Use this group to assign colors to different parts of the script text.

Set Text Color

To change color, select the type of text (Normal text, Strings, etc.) and select from the Set Text Color drop-down list box to change the foreground.

Set Background Color

To change the background color, select from the Set Background Color drop-down list box.

OK

Click this button to accept the changes made and exit the dialog.

Cancel

Click this button to discard the changes made and exit the dialog.

Help

Click this button to receive on-line help for this dialog.

Color Selection

The color options include 16 standard Windows colors, in addition to the default colors set in the current Windows environment.

Languge Elements

Comments

Comments are non-executed lines of code that are included for the benefit of the programmer. Comments can be included virtually anywhere in a script. Any text following an apostrophe or the word **Rem** is ignored by Enable. **Rem**, all other keywords, and most names in Enable are not case sensitive:

```
' This whole line is a comment
rem This whole line is a comment
REM This whole line is a comment
Rem This whole line is a comment
```

Comments can also be included on the same line as executed code:

```
MsgBox Msg ' Display message.
```

Everything after the apostrophe is a comment.

Statements

In Enable, there is no statement terminator. More than one statement can be put on a line if they are separated by a colon:

```
X.AddPoint( 25, 100) :  X.AddPoint(  0,  75)
```

Which is equivalent to:

```
X.AddPoint( 25, 100)
X.AddPoint(  0,  75)
```

Line Continuation Character

The underscore (_) is the line continuation character in Enable. There must be a space before and after the line continuation character.

```
X.AddPoint _
( 25, 100)
```

Numbers

Enable supports three representations of numbers: Decimal, Octal and Hexadecimal. Most of the numbers used in this manual are decimal or base 10 numbers; however, if you need to use octal (base 8) or hexadecimal (base 16) numbers, simply prefix the number with **&O** or **&H**, respectively.

Variable and Constant Names

Variable and constant names must begin with a letter. They may be comprised of uppercase letters (A through Z), lowercase letters (a through z), underscore (_) characters, and numeric digits (0 through 9). Variable and constant names can be no longer than 40 characters and cannot be reserved words. For a table of reserved words, see Language Reference**.** One exception to this rule is that object member names and property names may be reserved words.

Variables

Variable Types

As is the case with Visual Basic, when a variable is introduced in Enable, it is not necessary to declare it first (see Option Explicit for an exception to this rule).

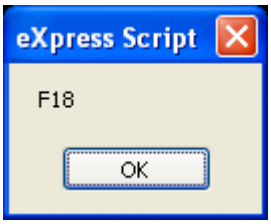
Note: Use the browse buttons (>>) (<<) to view the variable types.

Variant

When a variable is used but not declared, then it is implicitly declared as a variant data type. Variants can also be declared explicitly using **As Variant** as in **Dim x As Variant**. The variant data type is capable of storing numbers, strings, dates and times. When using a variant you do not have to convert a variable explicitly from one data type to another. This data type conversion is handled automatically.

Example:

```
Sub Main
    Dim x        'variant variable
    x = 10
    x = x + 8
    x = "F" & x
    MsgBox x, 0, "eXpress Script 'displays F18
End Sub
```



A variant variable can readily change its type and its internal representation can be determined by using the function **VarType**. **VarType** returns a value that corresponds to the explicit data types. See VarType for return values.

When storing numbers in variant variables, the data type used is always the most compact type possible. For example, if you first assign a small number to the variant it will be stored as an integer. If you then assign your variant to a number with a fractional component, it will then be stored as a double.

For doing numeric operations on a variant variable, it is sometimes necessary to determine if the value stored is a valid numeric, thus avoiding an error. This can be done with the IsNumeric function.

Variants and Concatenation

If a string and a number are concatenated, the result is a string. To be sure your concatenation works regardless of the data type involved, use the ampersand (&) operator instead of the plus (+) operator. The ampersand will not perform arithmetic on your numeric values; it will simply concatenate them as if they were strings.

Example:

```
X = String_1 & Integer_2 ' Concatenate a string and a number - OK
```

Instead of:

```
X = String_1 + Integer_2 ' Error
```

The IsEmpty function can be used to find out if a variant variable has been previously assigned.

Other Data Types

The six data types available in Enable are shown below with their declaration character suffixes:

Data Type	Suffix	Type Declaration	Size	Range
String	$	Dim StrVar As String	String of characters	0 to 65,500 characters
Integer	%	Dim IntVar As Integer	2-byte integer	-32,768 to 32,767
Long	&	Dim LongVar As	4-byte	-2,147,483,648 to 2,147,483,647

Data Type	Suffix	Type Declaration	Size	Range
		Long	integer	
Single	!	Dim SingVar As Single	4-byte floating-point number	-3.402823E38 to -1.401298E-45 (negative values) 1.401298E-45 to 3.402823E38 (positive values)
Double	#	Dim DblVar As Double	8-byte floating-point number	-1.79769313486232D308 to -4.94065645841247D-324 (negative values) 4.94065645841247D-324 to 1.79769313486232D308 (positive values)
Variant		Dim X As Variant	Date/time, floating-point number or string	Date values: January 1, 0000 through December 31, 9999; numeric values: same range as Double; string values: same range as String
Currency	(Currency data type is not supported)			

Scope of Variables

Variables can be local or global. Variables declared within a Sub or Function are local. Variables declared outside any **Sub** or **Function** are global.

Example:

```
Option Explicit
Global const Count As Integer ' (Global)
Sub BTN_1()
    Dim x As Integer          ' (Local)
        Count = count + 1
        x = count
End Sub
```

Declaration of Variables

In Enable, variables are declared with the Dim statement. To declare a variable other than a variant, the variable must be followed by **As** or suffixed by a type declaration character such as a percent character (%) for Integer type.

```
Sub Main
   Dim X As Integer
   Dim Y As Double
   Dim  Name$, Age%
    ' multiple declaration on one line
   Dim v
End Sub
```

Flow of Control

Control Structures

Enable has complete process control functionality. The control structures available are **Do** loops, **While** loops, **For** loops, **Select Case**, **If Then** and **If Then Else**. In addition, Enable has one branching statement: **GoTo.**

Note: Use the browse buttons (>>) (<<) to view each control structure.

The GoTo

The GoTo statement branches to the label specified on the **Goto** statement.

```
Goto labelx
    .
    .

labelx:
```

The program execution jumps to the part of the program that begins with the label, "Labelx:".

The Do Loops

The Do...Loop allows you to execute a block of statements an indefinite number of times. The variations of the **Do...Loop** are **Do While**, **Do Until**, **Do Loop While** and **Do Loop Until**.

```
Do While condition
        statement(s)...
Loop
Do Until condition
        statement(s)...
Loop
Do
        statement(s)...
Loop While condition
Do
        statement(s)...
Loop Until condition
```

Do While and **Do Until** check the condition before entering the loop, thus the block of statements inside the loop are only executed when those conditions are met. **Do Loop While** and **Do Loop Until** check the condition after having executed the block of statements, thereby guaranteeing that the block of statements is executed at least once.

See also, Data Types, Operators and Precedences.

The While Loop

The While...Wend loop is similar to the **Do While** loop. The condition is checked before executing the block of statements comprising the loop.

```
While condition
        statement(s)...
Wend
```

See also, Data Types, Operators and Precedences.

The For ... Next Loop

The For...Next loop has a counter variable and repeats a block of statements a set number of times. The counter variable increases or decreases with each repetition through the loop. The counter default is one if the **Step** variation is not used.

```
For counter = beginvalue To endvalue [Step
increment]
        statement(s)...
Next
```

See also, Data Types, Operators and Precedences.

The If and Select Statements

The If...Then block has a single line and multiple line syntax. The condition of an **If** statement can be a comparison or an expression, but it must evaluate to true or false.

```
If condition Then statement(s) ' single line syntax
```

```
If condition Then              ' multiple line syntax
        statement(s)...
End If
```

The other variation of the **If** statement is the If...Then...Else statement. This statement should be used when there are different statement blocks to be executed depending on the condition. A variation of the **If...Then...Else** is the **If...Then...ElseIf**.

```
If condition Then
        statement(s)...
ElseIf condition Then
        statement(s)...
Else
        statement(s)...
End If
```

The Select Case statement tests the same variable for many different values. This statement tends to be easier to read, understand and follow, and should be used in place of a complicated **If...Then...ElseIf** statement.

```
Select Case variable_to_test
        Case 1
            statement(s)...
        Case 2
            statement(s)...
        Case 3
            statement(s)...
        Case Else
            statement(s)...
End Select
```

See also, Data Types, Operators and Precedence.

Subroutines and Functions

Subroutine and Function Naming Conventions

Subroutine and function names may be comprised of uppercase letters (A through Z), lowercase letters (a through z), the underscore character (_) and numeric digits (0 through 9). Subroutine and function names must begin with a letter, must be no longer than 40 characters and must not be a reserved word. For a list of reserved words, see the list of reserved words in Language Reference.

Enable allows script developers to create their own functions or subroutines or to make DLL calls. Subroutines are created with the syntax:

```
Sub subname
.
.
End Sub
```

Functions have a similar syntax:

```
Function funcname As type
.
.
End Function
```

DLL functions are declared via the **Declare** statement.

ByRef and ByVal

ByRef gives other subroutines and functions the permission to make changes to variables that are passed as parameters. The keyword **ByVal** denies this permission and the parameters cannot be reassigned outside their local procedure. **ByRef** is the Enable default and does not need to be used explicitly. Because **ByRef** is the default, all variables passed to other functions or subroutines can be changed; the only exception is when using the **ByVal** keyword to protect the variable or using parentheses, which indicate the variable, is **ByVal**.

If the arguments or parameters are passed with parentheses around them, you will tell Enable that you are passing them **ByVal**.

```
SubOne var1, var2, (var3)
```

The var3 parameter in this case is passed by value and cannot be changed by the subroutine, SubOne.

```
Function R( X As String, ByVal n As Integer)
```

In this example, the "R" function is receiving two parameters, "X" and "n". The second parameter, "n", is passed by value and the contents cannot be changed from within the "R" function.

In the following code samples, scalar variable types and user-defined types are passed by reference:

Scalar Variables

```
Sub Main
    Dim x(5) As Integer
    Dim i As Integer
    for i = 0 to 5
        x(i) = i
    next i
    Print i
    Joe (i), x ' Parenthesis around it turn it into
               ' an expression which passes by value
    print "should be 6: "; x(2), i
End Sub
Sub Joe( ByRef j As Integer, ByRef y() As Integer )
    print "Joe: "; j, y(2)
    j = 345
    for i = 0 to 5
        print "i: "; i; "y(i): "; y(i)
    next i
    y(2) = 3 * y(2)
End Sub
```

Passing User-Defined Types by Ref to DLL's and Enable functions

```
' OpenFile() Structure
Type OFSTRUCT
    cBytes As String * 1
    fFixedDisk As String * 1
    nErrCode As Integer
```

```
    reserved As String * 4
    szPathName As String * 128
End Type
' OpenFile() Flags
Global Const OF_READ = &H0
Global Const OF_WRITE = &H1
Global Const OF_READWRITE = &H2
Global Const OF_SHARE_COMPAT = &H0
Global Const OF_SHARE_EXCLUSIVE = &H10
Global Const OF_SHARE_DENY_WRITE = &H20
Global Const OF_SHARE_DENY_READ = &H30
Global Const OF_SHARE_DENY_NONE = &H40
Global Const OF_PARSE = &H100
Global Const OF_DELETE = &H200
Global Const OF_VERIFY = &H400
Global Const OF_CANCEL = &H800
Global Const OF_CREATE = &H1000
Global Const OF_PROMPT = &H2000
Global Const OF_EXIST = &H4000
Global Const OF_REOPEN = &H8000
Declare Function OpenFile Lib "Kernel" (ByVal _
   lpFileName As String, lpReOpenBuff As OFSTRUCT, _
   ByVal wStyle As Integer) As Integer
Sub Main
    Dim ofs As OFSTRUCT
    ' Print OF_READWRITE
    ofs.szPathName = "c:\enable\openfile.bas"
    print ofs.szPathName
    ofs.nErrCode = 5
    print ofs.nErrCode
    OpenFile "t.bas", ofs
    print ofs.szPathName
    print ofs.nErrCode
End Sub
```

Calling Procedures in DLLs

DLLs or Dynamic-Link Libraries are used extensively by engineers to execute functions and subroutines located within the libraries. There are two main ways that scripting can be extended: 1) calling functions and subroutines in DLLs and 2) calling functions and subroutines located in the calling application. The mechanisms used for calling procedures in either place are similar (see the Declare Statement for more details).

To declare a DLL procedure or a procedure located in your calling application, place a declare statement in the global declaration section of the script. All declarations in Enable are global to the run and accessible by all subroutines and functions.

If the procedure does not return a value, declare it as a subroutine. If the procedure does have a return value declare it as a function.

```
Declare Function GetPrivateProfileString Lib "Kernel32" _
   (ByVal lpApplicationName As String, _
    ByVal lpKeyName As String, _
    ByVal lpDefault As String, _
    ByVal lpReturnedString As String, _
    ByVal nSize As _ Integer, _
    ByVal lpFileName As String) As Integer
Declare Sub InvertRect Lib "User" _
   (ByVal hDC AS Integer, aRect As Rectangle)
```

Notice the line extension character is the underscore (_). If a piece of code is too long to fit on one line, a line extension character can be used when needed.

Once a procedure is declared, you can call it just as you would another function.

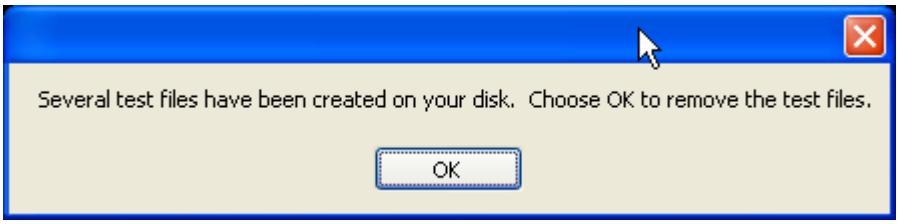
It is important to note that Enable cannot verify that you are passing correct values to a DLL procedure. If you pass incorrect values, the procedure may fail.

Files

File Input/Output

Enable supports limited file Input/Output(I/O).

```
Sub FileIO_Example()
    Dim Msg                          ' Declare variable
    Call Make3Files()                ' Create data files
    Msg = "Several test files have been created on your disk.  "
    Msg = Msg & "Choose OK to remove the test files."
    MsgBox Msg, 0, " "
    For I = 1 To 3
        Kill "TEST" & I              ' Remove data files from disk
    Next I
End Sub

Sub Make3Files ()
    Dim I, FNum, FName               ' Declare variables
    For I = 1 To 3
        FNum = FreeFile              ' Determine next file number
        FName = "TEST" & FNum
        Open FName For Output As FNum  ' Open file.
        Print #I, "This is test #" & I ' Write string to file
        Print #I, "Here is another ";  "line";  I
    Next I
    Close                            ' Close all files
End Sub
```



Arrays

Arrays

Enable supports single-dimensioned and multidimensional arrays. By using arrays, you can refer to a series of variables by the same name each with a separate index. Arrays have upper and lower bounds. Enable allocates space for each index number in the array. Arrays should not be declared larger than necessary.

All the elements in an array have the same data type. Enable supports arrays of integers, singles, double and strings.

Ways to declare a fixed size array:

- Global array – use the Global or Dim statement outside the general declarations section of a module to declare the array;
- Local array – use the Dim statement inside a procedure or function.

Enable does not support dynamic arrays.

Single-dimensioned Arrays:

When declaring an array, the array name must be followed by the upper bound (boundary) in parentheses. The upper bound must be an integer.

```
Dim ArrayName (10) As Interger
Dim Sum (20) As Double
```

To create a global array, you simply use **Global** in place of **Dim**:

```
Global Counters (12) As Integer
Global Sums (26) As Double
```

The same declarations within a procedure use **Static** or **Dim**:

```
Static Counters (12)  As Integer
Static Sums (26) As Double
```

The Counters declaration creates an array with 13 elements, with index numbers running from 0 to 12. The Sums declaration creates an array with 27 elements. To change the default lower bound to 1, place an Option Base statement in the declarations section of a module:

```
Option Base 1
```

Another way to specify lower bound is to provide it explicitly (as an integer, in the range -32,768 to 32,767) using the **To** key word:

```
Dim Counters (1 To 13) As Integer
Dim Sums (100 To 126) As String
```

In the preceding declarations, the index numbers of Counters run from 1 to 13, and the index numbers of Sums run from 100 to 126.

Note: Many other versions of Basic allow you to use an array without first declaring it. With Enable Basic, you must declare an array before using it.

Loops often provide an efficient way to manipulate arrays. For example, the following **For** loop initializes all elements in the array to a value of five (5):

```
Static Counters (1 To 20) As Integer
Dim I As Integer
    For I = 1 To 20
        Counter ( I ) = 5
    Next I
...
```

Multidimensional Arrays:

Enable supports multidimensional arrays. For example, the following example declares a two dimensional array within a procedure.

```
Static Mat(20, 20) As Double
```

Either or both dimensions can be declared with explicit lower bounds.

```
Static Mat(1 to 10, 1 to 10) As Double
```

You can efficiently process a multidimensional array with the use of **For** loops. In the following statements, the elements in a multidimensional array are set to a value.

```
Dim L As Integer, J As Integer
   Static TestArray(1 To 10, 1 to 10) As Double
   For L = 1 to 10
    For J = 1 to 10
        TestArray(L,J) = L * 10 J
    Next J
   Next L
```

Arrays can be more than two-dimensional. Enable does not have an arbitrary upper bound on array dimensions.

```
Dim ArrTest(5, 3, 2)
```

This declaration creates an array that has three dimensions with sizes 6 by 4, by 3 unless **Option Base 1** is set previously in the code. The use of **Option Base 1** sets the lower bound of all arrays to 1 instead of 0.

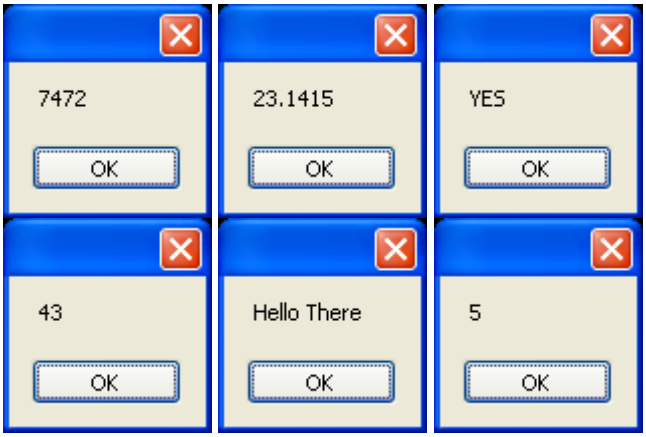
User Defined Types

User Defined Types

Users can define their own types that are composites of other built-in or user-defined types. Variables of these new user types can be declared. Member variables of the new type can be accessed using dot notation. Variables of user-defined types cannot be passed to DLL functions expecting "C" structures.

User-defined types are created using the type statement, which must be placed outside the procedure in your Enable code. User-defined types are global. The variables that are declared as user-defined types can be either global or local. User-defined types in Enable cannot contain arrays at this time.

```
Type type1
    a As Integer
    d As Double
    s As String
End Type

Type type2
    a As Integer
    o As type1
End Type

Dim type2a As type2
Dim type1a As type1

Sub TypeExample ()
    a = 5
    type1a.a = 7472
    type1a.d = 23.1415
    type1a.s = "YES"
    type2a.a = 43
    type2a.o.s = "Hello There"
    MsgBox type1a.a, 0, ""
    MsgBox type1a.d, 0, ""
    MsgBox type1a.s, 0, ""
    MsgBox type2a.a, 0, ""
    MsgBox type2a.o.s, 0, ""
    MsgBox a, 0, ""
End Sub
```



Dialogs and Dialog Controls

Dialog Support

The topics introduced here cover the code that is used to establish and maintain dialogs in a script. Written manually, this code can be very complicated; therefore, it is recommended that you use the eXpress Dialog Form Designer, from within the eXpress Script Editor whenever possible.

The syntax is similar to the syntax used in Microsoft Word Basic. The dialog syntax is not part of Microsoft Visual Basic or Microsoft Visual Basic for Applications (VBA); however, the scripting language has complete support for dialogs.

Most of the standard Windows dialog box controls are supported.

Use the browse buttons (>>) (<<) to review the controls available for custom dialog boxes and the guidelines for using them.

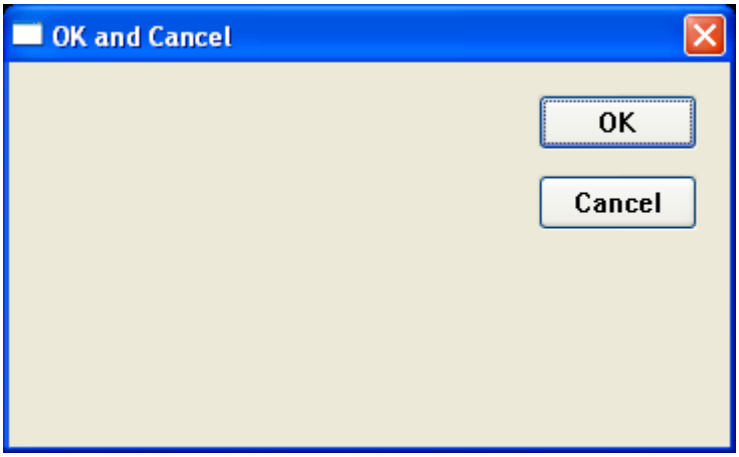
See:

- OK and Cancel
- List Boxes and Drop-down List Boxes
- Check Boxes
- Text Boxes and Text
- Option Buttons and Group Boxes
- The Dialog Function
- The Dialog Box Controls
- The Dialog Function Syntax

OK and Cancel

```
Sub Main
    Begin Dialog ButtonSample 16,32,180,96,"OK and Cancel"
        OKButton 132,8,40,14
        CancelButton 132,28,40,14
    End Dialog
        Dim Dlg1 As ButtonSample
        Button = Dialog (Dlg1)
End Sub
```

Every custom dialog box must contain at least one command button: an **OK** button or a **Cancel** button.

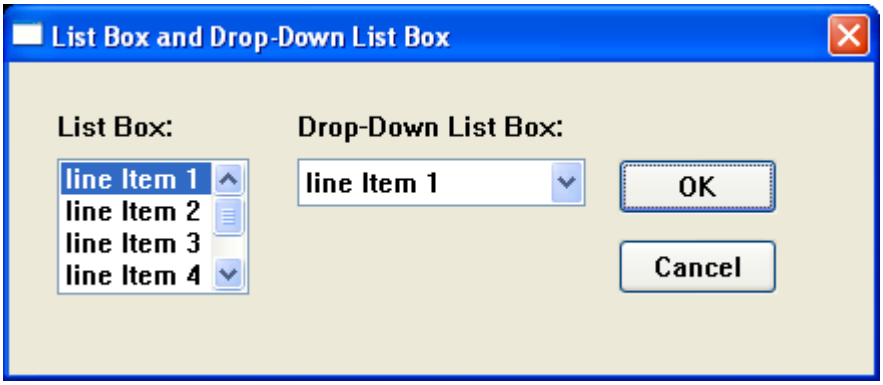


As an easier and better alternative, see the eXpress Dialog Form Designer.

List Boxes and Drop-down List Boxes

You can use a list box or drop-down list box to present a list of items from which the user can select. A drop-down list box saves space (it can drop down to cover other dialog box controls temporarily). The items displayed in a list box or drop-down list box are stored in an array that is defined before the instructions that define the dialog box.

```
Sub Main
      Dim MyList$ (5)
      MyList (0) = "line Item 1"
      MyList (1) = "line Item 2"
      MyList (2) = "line Item 3"
      MyList (3) = "line Item 4"
      MyList (4) = "line Item 5"
      MyList (5) = "line Item 6"
Begin Dialog BoxSample 159,175, 216, 78, "List Box and Drop-Down List Box"
  OKButton 152,24,40,14
  CancelButton 152,44,40,14
  ListBox 12,24,48,40, MyList$ (), .Lstbox
  DropListBox 72,24,72,40, MyList$ (), .DrpList
  Text 12,12,32,8, "List Box:"
  Text 72,12,68,8, "Drop-Down List Box:"
End Dialog

      Dim Dlg1 As BoxSample
      Button = Dialog ( Dlg1 )

End Sub
```

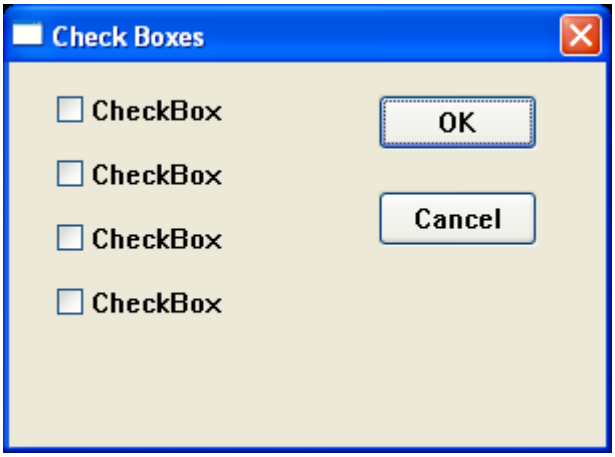


As an easier and better alternative, see the eXpress Dialog Form Designer.

Check Boxes

```
Sub Main
    Begin Dialog CheckSample 15,32,149,96,"Check Boxes"
    OKButton 92,8,40,14
    CancelButton 92,32,40,14
    CheckBox 12,8,45,8,"CheckBox",.CheckBox1
    CheckBox 12,24,45,8,"CheckBox",.CheckBox2
    CheckBox 12,40,45,8,"CheckBox",.CheckBox3
    CheckBox 12,56,45,8,"CheckBox",.CheckBox4
End Dialog
    Dim Dlg1 As CheckSample
    Button = Dialog ( Dlg1 )
End Sub
```

You use a check box to make a "yes" or "no", or "on" or "off" choice. For example, you could use a check box to display or hide a toolbar in your application.

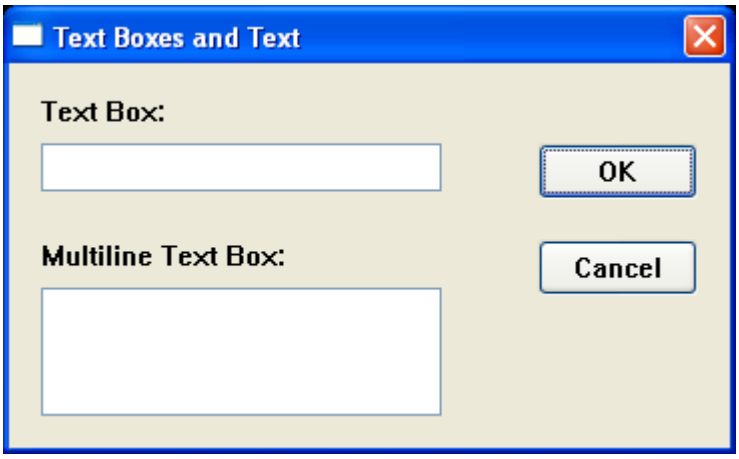


As an easier and better alternative, see the eXpress Dialog Form Designer.

Text Boxes and Text

```
Sub Main
    Begin Dialog TextBoxSample 16,30,180,96,_
        "Text Boxes and Text"
    OKButton 132,20,40,14
    CancelButton 132,44,40,14
    Text 8,8,32,8,"Text Box:"
    TextBox 8,20,100,12,.TextBox1
    Text 8,44,84,8,"Multiline Text Box:"
    TextBox 8,56,100,32,.TextBox2
End Dialog
    Dim Dlg1 As TextBoxSample
    Button = Dialog ( Dlg1 )

End Sub
```

A text box control is a box in which the user can enter text while the dialog box is displayed. By default, a text box holds a single line of text. Multiline text boxes are not supported with hand-coded dialogs in a script; however, they are supported using eQuate or the eXpress Dialog Form Designer).

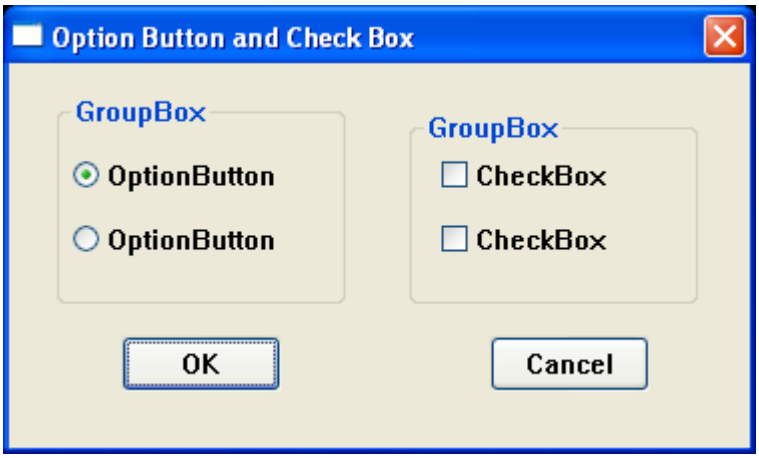


As an easier and better alternative, see the eXpress Dialog Form Designer.

Option Buttons and Group Boxes

You can have option buttons to allow the user to choose one option from several. Typically, you would use a group box to surround a group of option buttons, but you can also use a group box to set off a group of check boxes or any related group of controls.

```
Sub Main
Begin Dialog GroupSample 31,32,185,96,"Option Button and Check Box"
     OKButton 28,68,40,14
     CancelButton 120,68,40,14
     GroupBox 12,8,72,52,"GroupBox",.GroupBox1
     GroupBox 100,12,72,48,"GroupBox",.GroupBox2
     OptionGroup .OptionGroup1
     OptionButton 16,24,54,8,"OptionButton",.OptionButton1
     OptionButton 16,40,54,8,"OptionButton",.OptionButton2
     CheckBox 108,24,45,8,"CheckBox",.CheckBox1
     CheckBox 108,40,45,8,"CheckBox",.CheckBox2
 End Dialog
     Dim Dlg1 As GroupSample
     Button = Dialog (Dlg1)
 End Sub
```



As an easier and better alternative, see the eXpress Dialog Form Designer.

The Dialog Function

Enable supports the dialog function. This function is a user-defined function that can be called while a custom dialog box is displayed. The dialog function makes nested dialog boxes possible and receives messages from the dialog box while it is still active.

When the function **Dialog()** is called in Enable, it displays the dialog box, and calls the dialog function for that dialog. Enable calls the dialog function to see if there are any commands to execute. Typical commands that might be used are disabling or hiding a control. By default, all dialog box controls are enabled. If you want a control to be hidden, you must explicitly make it disabled during initialization. After initialization, Enable displays the dialog box. When an action is taken by the user, Enable calls the dialog function and passes values to the function that indicate the kind of action to take and the control that was acted upon.

The dialog box and its function are connected in the dialog definition. A function name argument (e.g., **UserDialog1**, below) is added to the **Begin Dialog** instruction, and matches the name of the dialog function located in your Enable program.

```
Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
```

As an easier and better alternative, see the eXpress Dialog Form Designer.

The Dialog Box Controls

A dialog function needs an identifier for each dialog box control upon which some action will take place. The dialog function uses string identifiers. String identifiers are the same as the identifiers used in the dialog record.

```
CheckBox 8, 56, 203, 16, "Check to display controls", .Chk1
```

The control's identifier and label are different. An identifier begins with a period and is the last parameter in a dialog box control instruction. In the sample code above, **Check to display controls** is the label and **Chk1** is the identifier.

As an easier and better alternative, see the eXpress Dialog Form Designer.

The Dialog Function Syntax

The syntax for the dialog function is as follows:

```
Function FunctionName( ControlID$, Action%, SuppValue%)
   Statement Block
   FunctionName = ReturnValue
End Function
```

All parameters in the dialog function are required.

A dialog function returns a value when the user chooses a command button. Enable acts on the value returned. The default is to return zero (0) and close the dialog box. If a non-zero is assigned, the dialog box remains open. By keeping the dialog box open, the dialog function allows the user to do more than one command from the same dialog box.

ControlID$ receives the identifier of the dialog box control.

Action identifies the action that calls the dialog function. Enable supports two actions:

Action 1	The value passed before the dialog becomes visible.
Action 2	The value passed when an action is taken (i.e.; a button pushed, checkbox checked, etc.). The ControlID$ is the same as the identifier for the control that was chosen.

SuppValue receives supplemental information about a change in a dialog box control. The information **SuppValue** receives depends upon which control calls the dialog function:

Checkbox	0 if cleared, 1 if selected.
Option Button	Number of option buttons selected, where zero is the first option button within a group.
Command Button	A value identifying the button chosen. **SuppValues** for push buttons are internal only.
OK Button	-1
Cancel Button	0

The following dialog function uses a Select Case control structure to check the value of Action. The **SuppValue** is ignored in this function.

```
' This sample file outlines dialog capabilities,
' including nesting dialog boxes.

Sub Main
```

```
    Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
        Text 8,10,73,13, "Text Label:"
        TextBox 8, 26, 160, 18, .FText
        CheckBox 8, 56, 203, 16, "Check to display controls", .Chk1
        GroupBox 8, 79, 230, 70, "This is a group box:", .Group
        CheckBox 18,100,189,16, "Check to change button text", .Chk2
        PushButton 18, 118, 159, 16, "File History", .History
        OKButton 177, 8, 58, 21
        CancelButton 177, 32, 58, 21
        End Dialog

    Dim Dlg1 As UserDialog1
    x = Dialog( Dlg1 )
End Sub


Function Enable( ControlID$, Action%, SuppValue%)

Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox",. ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
End Dialog
Dim Dlg2 As UserDialog2
Dlg2.FText = "Your default string goes here"


Select Case Action%

Case 1
    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
    DlgVisible "History", 0
Case 2
    If ControlID$ = "Chk1" Then
        DlgEnable "Group"
        DlgVisible "Chk2"
        DlgVisible "History"
    End If

    If ControlID$ = "Chk2" Then
        DlgText "History", "Push to display nested dialog"
    End If

    If ControlID$ = "History" Then
        Enable =1
        x = Dialog( Dlg2 )
    End If


Case Else

End Select
Enable =1

End Function
```

As an easier and better alternative, see the eXpress Dialog Form Designer.

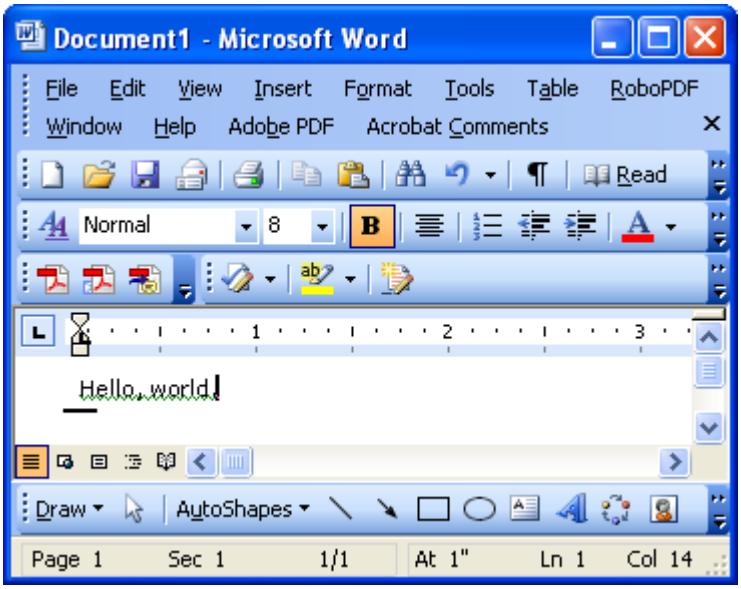
OLE Automation

What is OLE Automation?

OLE Automation is a standard, promoted by Microsoft that applications use to expose their OLE objects to development tools, Enable Basic and containers that support OLE Automation. A spreadsheet application may expose a worksheet, chart, cell or range of cells — all as different types of objects (e.g., Microsoft Excel). A word processor might expose objects such as an application, paragraph, sentence, bookmark or selection (e.g., Microsoft Word).

When an application supports OLE Automation, the objects it exposes can be accessed by Enable Basic. You can use Enable Basic to manipulate these objects by invoking methods on the object, or by getting and setting the objects properties, just as you would with the objects in Enable Basic. For example, if you created an OLE Automation object named **MyObj**, you might write code such as follows to manipulate the object:

```
Sub Main
  Dim MyObj As Object
  Set MyObj = CreateObject ("Word.Basic")
  MyObj.AppShow
  MyObj.FileNewDefault
  MyObj.Insert "Hello, world."
  MyObj.Bold 1
End Sub
```



The following syntax is supported for the GetObject function:

```
Set MyObj = GetObject ("", class)
```

The *class* parameter represents the class of the object to retrieve. The first parameter at this time must be an empty string.

The properties and methods an object supports are defined by the application that created the object. See the application's documentation for details on the properties and methods it supports.

Accessing an Object

The following functions and properties allow you to access an OLE Automation object:

Name	Description
CreateObject Function	Creates a new object of a specified type.
GetObject Function	Retrieves an object pointer to a running application.

What is an OLE Object?

An OLE Automation Object is an instance of a class within your application that you wish to manipulate programmatically, as in a script. These classes may be new classes whose sole purpose is to collect and expose data and functions in a way that is meaningful to your users.

The object becomes programmable when you expose member functions. OLE Automation defines two types of members that you may expose for an object:

- **Methods** are member functions that perform an action on an object. For example, a Document object might provide a Save method.
- **Properties** are member function pairs that set or return information about the state of an object. For example, a Drawing object might have a style property.

Microsoft suggests the following objects could be exposed by implementing the listed methods and properties for each object:

OLE Automation Object	Methods	Properties
Application	Help	ActiveDocument
	Quit	Application
	Add Data	Caption
	Repeat	DefaultFilePath
	Undo	Documents
		Height
		Name
		Parent
		Path
		Printers
		StatusBar
		Top
		Value
		Visible
		Width

OLE Automation Object	Methods	Properties
Document	Activate	Application
	Close	Author
	NewWindow	Comments
	Print	FullName
	PrintPreview	Keywords
	RevertToSaved	Name
	Save	Parent
	SaveAs	Path
		ReadOnly
		Saved
		Subject
		Title
		Value

To provide access to more than one instance of an object, expose a collection object. A collection object manages other objects. All collection objects support iteration over the objects they manage. For example, Microsoft suggests an application with a multiple document interface (MDI) might expose a document's collection object with the following methods and properties:

Collection Object	Methods	Properties
Documents	Add	Application
	Close	Count
	Item	Parent
	Open	

OLE Fundamentals

Object Linking and Embedding (OLE) is a technology that allows programmers of Windows-based applications to create applications that can display data from many different applications. OLE allows the user to edit that data from within the application in which it was created. In some cases, the user can even edit the data from within their application.

The following terms and concepts are fundamental to understanding OLE.

OLE Object

An OLE object refers to a discrete unit of data supplied by an OLE application. An application can expose many types of objects. For example, a spreadsheet application can expose a worksheet, macro sheet, chart, cell or range of cells — all as different types of objects. You use the OLE control to create linked and embedded objects. When a linked or embedded object is created, it contains the name of the application that supplied the object, its data (or, in the case of a linked object, a reference to the data) and an image of the data.

OLE Automation

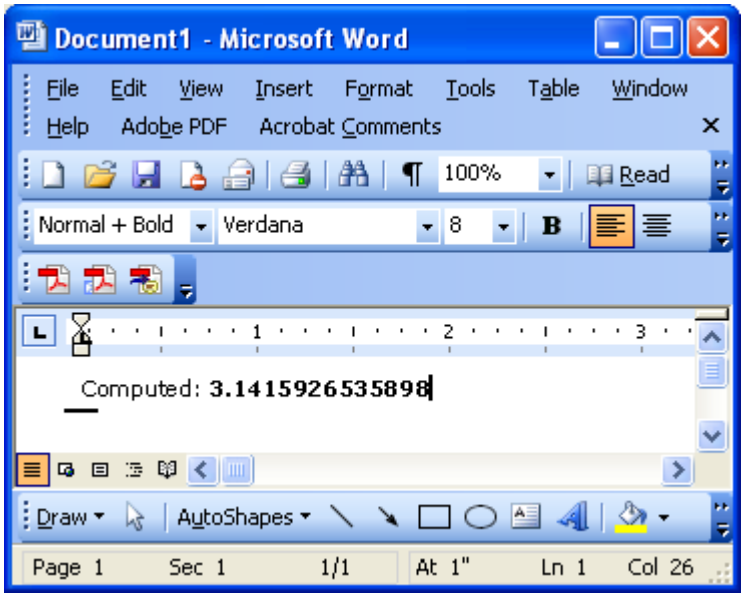
Some applications provide objects that support OLE Automation. You can use scripting to manipulate the data programmatically in these objects. Some objects that support OLE Automation also support linking and embedding. You can create an OLE Automation object by using the CreateObject function.

Class

The class of an object determines the application that provides the object's data and the type of data the object contains. The class names of some commonly used Microsoft applications include **MSGraph**, **MSDraw**, **WordDocument**, and **ExcelWorksheet**.

OLE Automation and Word 6.0 example

```
Sub OLEexample()
    Dim word6 As Object
    Dim myData As String

    myData = 4 * Atn(1)
        ' Demonstrates Automatic type conversion
    Set word6 = CreateObject("Word.Basic")
    word6.AppShow
    word6.FileNewDefault
    word6.Insert "Computed: "
    word6.Bold 1        ' Show value in boldface
    word6.Insert myData
    word6.Bold 0

    MsgBox "Done"
End Sub
```



Making Applications Work Together

Operations like linking and object embedding need applications to work together in a coordinated fashion; however, there is no way that Windows can be set up in advance to accommodate all the applications and

dynamic link libraries that can be installed. Even within an application, the user has the ability to select various components to install.

As part of the installation process, Windows requires that applications supporting DDE/OLE features register their support by storing information in several different locations. The most important of these to Enable is the registration database.

WIN.INI

The WIN.INI file contains a special section called [embedding] that contains information about each of the applications that operate as object servers.

The Registration Database

Starting with Windows 3.1, each Windows system maintains a registration database file that records details about the DDE and OLE functions supported by the installed applications. The database is stored in the file called **REG.DAT** in the **\WINDOWS** directory. The file is a database that contains information that controls a variety of activities relating to data integration using DDE and OLE. The information contained in the **REG.DAT** database can be divided into four basic categories.

Associations

The table contains information that associates files with specific extensions to particular applications. This is essentially the same function performed by the [extensions] section of the **WIN.INI**.

Shell Operations

Windows contains two programs that are referred to as Shell programs. The term, "shell," refers to a program that organizes basic operating system tasks, like running applications, opening files, and sending files to the printer. Shell programs use lists, windows, menus and dialog boxes to perform these operations. In contrast, command systems like DOS require the entry of explicit command lines to accomplish these tasks.

OLE Object Servers

The registration database maintains a highly structured database of the details needed by programs that operate as object servers. This is by far the most complex task performed by the database. There is no **WIN.INI** equivalent for this function.

DDE/OLE Automation

The registration database contains the details and the applications that support various types of DDE/OLE Automation operations.

It is useful to appreciate the difference in structure between the WIN.INI file and the **REG.DAT** database. **WIN.INI** is simply a text document. There are no special structures other than headings (simply titles enclosed in brackets) that organize the information. If you want to locate an item in the **WIN.INI** file, you must search through the file for the specific item you want to locate. The registration database is a tree-like, structured database used for storing information relating to program and file operations, in particular, those that involve the use of DDE or OLE. The tree structure makes it easier to keep the complex set of instructions, needed to implement DDE and OLE operations, organized and accessible by the applications that need to use them. This is not possible when you are working with a text document like **WIN.INI**. The **WIN.INI** file records all sorts of information about the Windows system in a simple sequential listing.

Quick Reference

Language Reference

This topic is a quick reference of the functions, subroutines, statements, Data Types, Operators and Precedences available.

Data Types, Operators and Precedences

Functions/Statements/Subroutines by Category

Functions/Statements/Subroutines - Alphabetical Listing

Functions, Subroutines, Statements, Reserved Words

Abs, ActivateScreen, AppActivate, Append, As, Asc, Atn

Base, Beep, Begin, Binary, Btm, ByVal

Call, Case, CDbl, ChDir, ChDrive, CheckBox, Chr, CInt, CLng, Cbool, Close, Col, Const, CopyToClipboard, Cos, CreateObject, CSng, CStr, Ctype, CurDir, CurrentPage, CurrentScreen, Cvar

Date, DateSerial, DateValue, Day, Declare, Dim, Dir, Do, Dialog, DlgEnable,

DlgText, DlgVisible, DropListBox

Else, ElseIf, End, EndIf, EnterText, EOF, Erase, Error, Exit, Exp

FileCopy, FileLen, FileOpenDialog, FileSaveDialog, Fix, FreeFile, FontName, FontSize, For, Format

Get, GetCursorCol, GetCursorRow, GetLastMsg, GetObject, GetScreenAttribute, GetScreenColor, GetScreenLine, GetScreenText, GetUserParam, GetWindowState, Global, GoTo, GroupBox

Hex, Hour

If, Input, InputBox, InStr, Int, Integer, IsArray, IsData, IsEmpty, IsNull, IsNumeric, IsObject

Key, KeyBoardState, Kill

LBound, LCase, Left, Len, Length, Let, Lft, Line, LineNumber, ListBox, LoadScreen, LOF, Log, Loop, LTrim

MarkBlock, MessageWaitingState, Mid, Minute, MkDir, Month, Ms, MsgBox

Name, Next, Now

Object, Oct, OKButton, On, Open, Option, OptionButton, OptionGroup

PasteFromClipboard, Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth, PushButton, Put

Randomize, ReDim, RefreshScreen, Rem, Return, Right, RmDir, Rnd, Row, Rt, RTrim

SaveScreen, ScreenAvailable, ScreenOpen, Second, Seek, Select, SendKeys, Set, SetCursor, SetDblClickAction, SetScreenText, SetSignOnResult, SetWindowState, Sgn, Shell, Sin, Space, Sqr, Static, Status, Stop, Str, StrComp, String, Style, Sub, Subroutine

T27Key, Tan,Text, TextBox, TextStr, Then, Time, Timer, TimeSerial, TimeValue,Tmp, To, Tp, Trim, Type

UBound, UCase, UTSKey

Val, Value, Variant, VarType

Wait, WinHelp, Weekday, Wend, While, Wid, With, Write

X

Y, Year

Data Types, Operators and Precedence

Data Types

Variable		Type Declaration	Size
String	$	Dim Str_Var As String	0 to 65500 char
Integer	%	Dim Int_Var As Integer	2 bytes
Long	&	Dim Long_Var As Long	4 bytes
Single	!	Dim Sing_Var As Single	4 bytes
Double	#	Dim Dbl_Var As Double	8 bytes
Variant		Dim X As Variant	
Currency		(Currency datatype is not supported)	

Arithmetic Operators

Operator	Function	Usage
^	Exponentiation	x% = y%^2
-	Negation	x% = -2
*	Multiplication	x% = 2 * 3
/	Division	x% = 10 / 2
Mod	Modulo	x% = y% Mod z%
+	Addition	x% = 2 + 3
-	Subtraction	x% = 6 - 4

*Arithmetic operators follow mathematical rules of precedence

* "+"or "&" can be used for string concatenation.

Operator Precedence

Operator	Description	Order
()	Parenthesis	Highest
^	Exponentiation	
-	Unary Minus	
/, *	Division/Multplication	
mod	Modulo	
+, -, &	Addition, subtraction, concatenation	
=, <>, >, <, <=, >=	Relational	
not	Logical negation	
and	Logical conjunction	
or	Logical disjunction	
xor	Logical exclusion	
eqv	Logical equivalence	
imp	Logical implication	Lowest

Relational Operators

Operator	Function	Usage
<	Less than	x < y
<=	Less than or equal to	x <= y
=	Equals	x = y
>=	Greater than or equal to	x >= y
>	Greater than	x > y
<>	Not equal to	x <> y

Comparison Operators

Operator	Function	True Tests	Binary Result
And	Logical And	If 9 And 8 = 8	1001 And 1000 → 1000
Eqv	Logical Equivalence	If 9 Eqv 8 = -2	1001 Eqv 1000 → 1110
Imp	Logical Implication	If 8 Imp 9 = -1	1000 Imp 1001 → 1111
Is	Compare Two Objects Refs	If Obj Is Obj	
Not	Negation	If Not 0 = -1	
Mod	Divide and Return Remainder	If 9 Mod 8 = 1	
Or	Logical Or	If 9 Or 8 = 9	1001 Or 1000 → 1001
Xor	Logical Exclusion	If 9 Xor 8 = 1	1001 Xor 1000 → 0001

Functions/Statements/Subroutines by Category

Arrays

Dim, Erase, Choose, Global, Lbound, Option Base, Option Explicit, ReDim, Static, Ubound

Flow of Control

Do. . . Loop, End, Exit Do, Exit For, Exit Function, Exit Sub, For Each...Next, For. . . Next, GoTo, If. . . Then. . . Else. . . End If, Select Case, Stop, While. . . Wend, With

Converting

Asc, CDbl, Chr, CInt, CLng, CSng, CStr, CVar, Date, DateSerial, DateValue, Day, Fix, Format, Hex, Hour, Int, Minute, Month, Now, Oct, Second, Str, Time, Timer, TimeSerial, TimeValue, Val, Weekday, Year

Data and Screens (Express)

ActivateScreen, CopyToClipboard, CurrentPage, CurrentScreen, EnterText, GetCursorCol, GetCursorRow, GetLastMsg, GetScreenAttribute, GetScreenColor, GetScreenLine, GetScreenText, GetWindowState, KeyboardState, MarkBlock, MessageWaitingState, PasteFromClipboard, RefreshScreen, ScreenAvailable, ScreenOpen, SetCursor, SetDblClickAction, SetScreenText, SetWindowState, T27Key, UTSKey, Wait, WaitForSpecificString, WaitForString

Dialog

Begin Dialog, CancelButton, CheckBox, Dialog, DlgEnable, DlgText, DlgVisible, DropListBox, End Dialog, GroupBox, InputBox, ListBox, MsgBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

File I/O

ChDir, ChDrive, Close, CurDir, Dir, EOF, FileCopy, FileLen, FreeFile, Get, Input, Kill, Line Input #, LOF, MkDir, Name, Open, Print#, Put, RmDir, Seek, Write #

Math

Abs, Atn, CBool, Cos, Exp, Log, Rnd, Sgn, Sin, Sqr, Tan

Miscellaneous

AppActivate, Beep, CreateObject, FileOpenDialog, FileSaveDialog, GetObject, GetUserParam, LoadScreen, On Error, Randomize, Rem, SaveScreen, SendKeys, Set, SetSignOnResult, Shell, Type, SwitchToolbar, WinHelp

Printing (Express)

Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Procedures

Call, Declare, End, Function, Sub, End Sub, Exit, Global

Strings

Asc, Chr, InStr, LCase, Left, Len, Let, LTrim, Mid, Right, RTrim, Space, StrComp, String, Trim, UCase

Variables and Constants

Const, Dim, IsArray, IsDate, IsEmpty, IsNull, IsNumeric,, Option Explicit, VarType

Functions/Statements/Subroutines – Alphabetical Listing

A

Abs
ActivateScreen
AppActivate
Asc
Atn

B

Beep
Begin Dialog

C

Call
CancelButton
CBool
CDbl
ChDir
ChDrive
CheckBox
Choose
Chr
CInt
CLng
Close
Const
CopyToClipboard
Cos
CreateObject
CSng
CStr
CurDir
CurrentPage
CurrentScreen
CVar Function

D

Date
DateSerial
DateValue
Day
Declare
Dialog
Dim
Dir
DlgEnable
DlgText
DlgVisible
Do…Loop
DropListBox

E

End
End Dialog
End Function

End If
End Sub
End Type
End With
EnterText
EOF
Erase
Exit
Exit Do
Exit For
Exit Function
Exit Sub
Exp

F

FileCopy
FileLen
FileOpenDialog
FileSaveDialog
Fix
For Each...Next
For…Next
Format
FreeFile
Function

G

Get
GetCursorCol
GetCursorRow
GetLastMsg
GetObject
GetScreenAttribute
GetScreenColor
GetScreenLine
GetScreenText
GetUserParam
GetWindowState
Global
GoTo
GroupBox

H

Hex
Hour

I

If. . . Then. . . Else. . . End If
Input
InputBox
InStr
Int
IsArray
IsDate
IsEmpty
IsNull
IsNumeric
IsObject

K

KeyboardState
Kill

L

Lbound
LCase
Left
Len
Let
Line Input #
ListBox
LoadScreen
LOF
Log
LTrim

M

MarkBlock
Mid
Minute
MkDir
Month
MessageWaitingState
MsgBox

N

Name
Now

O

Oct
OKButton
On Error
Open
Option Base
Option Explicit
OptionButton
OptionGroup

P

PasteFromClipboard
Print
Print #
PrintBeginDoc
PrintEndDoc
PrintMoveTo
PrintNewPage
PrintPageHeight
PrintPageWidth
PrintRect
PrintSelect
PrintSetFont
PrintSetFontSize
PrintSetFontStyle
PrintSetOrientation
PrintTextHeight
PrintTextWidth
PushButton
Put

R

Randomize
ReDim
RefreshScreen
Rem
Right
RmDir
Rnd
RTrim

S

SaveScreen
ScreenAvailable
ScreenOpen
Second
Seek
Seek Statement
Select Case
SendKeys
Set
SetCursor
SetDblClickAction
SetScreenText
SetSignOnResult
SetWindowState
Sgn
Shell
Sin
Space
Sqr
Static
Stop
Str
StrComp
String
Sub
SwitchToolbar

T

T27Key
Tan
Text
TextBox
Time
Timer
TimeSerial
TimeValue
Trim
Type

U

UBound
UCase
UTSKey

V

Val
VarType

W

Wait
WaitForSpecificString
WaitForString
Weekday
While. . . Wend
WinHelp
With
Write #

Y

Year

Functions, Statements and Subroutines

Abs Function

Return the absolute value of a *number*.

Format:

Abs(*number*)

The data type of the return value is the same as that of the *number* argument. If the *number* argument is a variant of VarType (String) and can be converted to a number, the return value will be a variant of **VarType** (Double). If the numeric expression results in a null, **Abs** returns a null.

Example:

```
Sub Main
    Dim Msg, X, Y
    X = InputBox("Enter a Number:")
    Y = Abs(X)

    Msg = "The number you entered is "  & X
    Msg = Msg + ". The Absolute value of " & X & " is " & Y
    MsgBox Msg ' Display Message.

End Sub
```

ActivateScreen Subroutine (eXpress Plus)

Activate the specified screen, if it is available.

Format:

ActivateScreen *ScreenNumber*

The *ScreenNumber* parameter is an integer expression that represents the configured screen number to be activated. The activated screen is still not available to the script. A script still open only has access to the screen from which it was started.

If an invalid *ScreenNumber* is entered, it is ignored.

Related Topics: ScreenAvailable, ScreenOpen

Example:

```
Sub Main()
' Activate a new screen
      If ScreenAvailable (2) = 1 Then
             MsgBox "Screen 2 Availavle"
             If ScreenOpen(2) = 0 Then
                    MsgBox "Screen 2 NOT Open"
                    ActivateScreen 2
             End If
      End If
End Sub
```

AppActivate Statement

Activate an application.

Format:

AppActivate "*application*"

The *application* parameter is a string expression and is the name that appears in the title bar of the application window to be activated.

Related Topics: Shell, SendKeys

Example:

```
Sub Main ()
    AppActivate "Microsoft Word"
    SendKeys "%F,%N,Enable"
    Msg = "Click OK to close Word"
    MsgBox Msg
    AppActivate "Microsoft Word"    ' Focus back to Word
```

```
    SendKeys "%F,%C,N"              ' Close Word
End Sub
```

Asc Function

Return a numeric value that is the ASCII code for the first character in a string.

Format:

Asc(*string*)

Example:

```
Sub Main ()
    Dim I, Msg                    ' Declare variables.
    For I = Asc("A") To Asc("Z") ' From A through Z.
        Msg = Msg & Chr(I)        ' Create a string.
    Next I
    MsgBox Msg                    ' Display results.
End Sub
```

Atn Function

Return the arctangent of a number.

Format:

Atn(*string*)

The *rad* argument can be any numeric expression. The result is expressed in radians.

Related Topics: Cos, Tan, Sin

Example:

```
Sub AtnExample ()
    Dim Msg, Pi            ' Declare variables.
    Pi = 4 * Atn(1)        ' Calculate Pi.
    Msg = "Pi is equal to " & Str(Pi)
    MsgBox Msg             ' Display results.
End Sub
```

Beep Statement

Sound a tone through the computer's speaker.

Format:

Beep

The frequency and duration of the beep depends on hardware, which may vary among computers.

Example:

```
Sub BeepExample ()
    Dim Answer, Msg                         ' Declare variables.
    Do
        Answer = InputBox("Enter a value from 1 to 3.")
        If Answer >= 1 And Answer <= 3 Then ' Check range.
            Exit Do                         ' Exit Do…Loop.
        Else
            Beep                            ' Beep if not in range.
        End If
    Loop
    MsgBox "You entered a value in the proper range."
End Sub
```

Begin Dialog Statement

Mark the beginning of a dialog and the overall dimensions of the dialog box.

Note: Dialogs when written manually can be very difficult; therefore, it is recommended that you use the Enable Dialog Designer, from within the eXpress Script Editor whenever possible.

Format:

Begin Dialog *name starting-x-pos*, *starting-y-pos*, *width*, *height*, "*caption*" [, . *functionname*]

Related Topics: Dialog Support, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

The following is a sign-on script was developed with the aid of eXpress Script Recorder and eXpress Script Editor. Initially, DEMAND and MAPPER sign-on sessions were recorded and saved as script files. Subsequently, the files were merged and additional code added with eXpress Script Editor. The dialog was created within the eXpress Script Editor. The sign-on script allows the user to enter all sign-on parameters on one dialog. In addition, the script illustrates the ability to sign-on to a choice of host and session types:

```
' *** Multi-system Sign-on Script ***
Sub Main()

Begin Dialog DIALOG_1 134,70, 229, 120, "Sign-on Dialog"
  Text 4,4,68,8, "Host Parameters:"
  Text 8,16,51,8, "User Id."
  Text 8,41,51,8, "Password"
  Text 8,66,51,8, "Project Id."
  Text 8,92,80,8, "TIP Transaction Code"
  Text 84,4,76,8, "MAPPER Parameters:"
  Text 88,16,51,8, "User Id."
  Text 88,41,51,8, "Department"
  Text 88,66,51,8, "Password"
  TextBox 8,24,33,11, .Host_User_Id
  TextBox 8,50,33,11, .Host_PassWd
  TextBox 8,74,66,11, .Host_Proj_Id
  TextBox 8,102,34,11, .TIP_Tx_Code
  TextBox 88,24,33,11, .Map_User_Id
  TextBox 88,50,14,11, .Map_Dept
  TextBox 88,74,66,11, .Map_PassWd
  OptionGroup .GROUP_1
    OptionButton 177,32,32,12, "Host 1"
    OptionButton 177,44,32,12, "Host 2"
  GroupBox 168,20,48,40, "Select a Host:"
  OKButton 169,80,45,15
  CancelButton 169,97,45,15
End Dialog

    Dim Dlg as DIALOG_1
    Dim OID as String        ' Open Id.

SignOnDialog:

    If Dialog(Dlg) = 0 then
        Exit Sub
    End If

    If Trim$(Dlg.Host_User_Id) = "" _
    or Trim$(Dlg.Host_PassWd) = "" Then
        MsgBox "Host User Id. and Host Password are " + _
            "required entries.", MB_ICONEXCLAMATION
        GoTo SignOnDialog
    End If

    If Trim$(Dlg.TIP_Tx_Code) = "" _
    and Trim$(Dlg.Map_User_Id) = "" Then
        If Dlg.Group_1 = 0 Then
            OID = "$$open dkmsdem"
        Else
            OID = "$$open ekmsdem"
        End If
        GoTo Demand
    Else
        If Dlg.Group_1 = 0 Then
            OID = "$$open dkmsmcb"
        Else
            OID = "$$open ekmsmcb"
```

```
            End If
        End If
        GoTo TIP

    Demand:
        Wait 1000
        EnterText OID
        UTSKey UK_TRANSMIT_KEY
        Wait 2000
        EnterText Trim$(Dlg.Host_User_Id) + "/" _
            + Dlg.Host_PassWd
        UTSKey UK_TRANSMIT_KEY
        Wait 1000
        EnterText Dlg.Host_Proj_Id
        UTSKey UK_TRANSMIT_KEY
        Exit Sub

    TIP:
        Wait 1000
        EnterText OID
        UTSKey UK_TRANSMIT_KEY
        Wait 2000
        EnterText Trim$(Dlg.Host_User_Id) + "/" _
            + Dlg.Host_PassWd
        UTSKey UK_TRANSMIT_KEY
        Wait 1000
        UTSKey UK_TRANSMIT_KEY
        Wait 1000
        If Trim$(Dlg.Map_User_Id) <> "" Then
            EnterText "mapper"
            UTSKey UK_TRANSMIT_KEY
        Else
            EnterText Dlg.TIP_Tx_Code
            UTSKey UK_TRANSMIT_KEY
            Exit Sub
        End If
        Wait 1000
        EnterText Dlg.Map_User_Id
        UTSKey UK_TAB_FORWARD
        EnterText Dlg.Map_Dept
        UTSKey UK_TAB_FORWARD
        EnterText Dlg.Map_PassWd
        UTSKey UK_TRANSMIT_KEY

    End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Call Statement

Activate a script subroutine or DLL function.

Formats:

Call *name* [(*parameters(s)*)]

or

name [*parameter(s)*]

Activate a script or DLL subroutine called *name*. The first parameter is the *name* of the function or subroutine to call, and the second is the list of *parameter(s)* to pass to the called function or subroutine.

Note: Script Functions and Subroutines may be placed in a user library (USER.LIB) and become global to any script that needs to call them. The user library must reside in the SCRIPTS directory under the installation directory.

You are never required to use the **Call** statement when calling an Enable or DLL subroutine. Parentheses must be used in the argument list if the **Call** statement is being used.

Example:

```
Sub Main ()
    Call Beep
    MsgBox "Returns a Beep"
End Sub
```

CancelButton Statement

Use to close a dialog when omitting changes.

Format:

CancelButton *starting-x-pos*, *starting-y-pos*, *width*, *height*

Related Topics: Begin Dialog, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_1()
Begin Dialog Dialog_1 0,0, 252, 136, "Dialog Title"
  OptionGroup .GRP_1
    OptionButton 32,48,80,12, "1st Radio - Group 1"
    OptionButton 32,64,80,12, "2nd Radio - Group 1"
  OptionGroup .GRP_2
    OptionButton 144,48,84,12, "1st Radio - Group 2"
    OptionButton 144,64,84,12, "2nd Radio - Group 2"
  OKButton 24,96,68,20
  CancelButton 156,96,52,20
  GroupBox 24,36,92,52, "Option Group 1"
  GroupBox 136,36,100,52, "Option Group 2"
  GroupBox 24,4,212,28, "Check Group"
  CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
  CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
End Dialog
Dim Dlg1 As Dialog_1
Dlg1.Grp_1 = 0           ' Set 1st button - group 1
Dlg1.Grp_2 = 1           ' Set 2nd button - group 2
button = Dialog ( Dlg1 )
If button = 0 Then Return
MsgBox "Grp1: " + Dlg1.Grp_1 + ", Grp2: " + Dlg1.Grp_2
Dialog Dlg1
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

CBool Function

Convert expression from one data type to Boolean.

Format:

CBool (*expression*)

The parameter *expression* must be a valid string or numeric expression.

Example:

```
Sub Main

  Dim A, B, Check

  A = 5: B = 5
  Check = CBool(A = B)
  Print Check

  A = 0
  Check = CBool(A)
```

```
    Print Check

End Sub
```

CDbl Function

Convert expressions from one data type to a double.

Format:

CDbl(*expression*)

The *expression* parameter must be a valid string or numeric expression.

Example:

```
Sub Main ()
    Dim y As Integer
    y = 25
    If  VarType(y) = 2 Then
        Print y
        x = CDbl(y)
        ' Converts integer value of y to a double value in x
        Print x
    End If
End Sub
```

ChDir Statement

Change the default directory.

Format:

ChDir *pathname*

Pathname syntax:

[*drive*:][\] *dir*[*dir*]…

The *pathname* parameter is a string limited to fewer then 128 characters. The *drive* parameter is optional. The *dir* parameter is a directory name. **ChDir** changes the default directory on the current drive if the *drive* is omitted.

Related Topics: ChDrive, CurDir, Dir, MkDir, RmDir

Example:

```
Sub Main ()
    Dim Answer, Msg, NL        ' Declare variables.
    NL = Chr(10)               ' Define newline.
    CurPath = CurDir()         ' Get current path.
    ChDir "\"                  ' Change to the root directory.
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change "
    Msg = Msg & "back to your previous default directory."
    Answer = MsgBox(Msg)       ' Get user response.
    ChDir CurPath              ' Change back to user default.
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg                 ' Display results.
End Sub
```

ChDrive Statement

Change the current drive.

Format:

ChDrive *drivename*

The *drivename* parameter is a string and must correspond to an existing drive. If *drivename* contains more than one letter, only the first character is used.

Related Topics: ChDir, CurDir, Dir, MkDir, RmDir

Example:

```
Sub Main ()
    Dim Msg, NL                ' Declare variables.
    NL = Chr(10)               ' Define newline.
    CurPath = CurDir()         ' Get current path.
```

```
    ChDir "\"                  ' Change to the root directory.
    ChDrive "G:"               ' Change to G drive.
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change back "
    Msg = Msg & "to your previous default directory."
    MsgBox Msg                 ' Get user response.
    ChDir CurPath              ' Change back to user default.
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg                 ' Display results.
End Sub
```

CheckBox Statement

Use a checkbox in a dialog for selecting one or more in a series of choices.

Format:

CheckBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, "*caption*", *.name*

Related Topics: Begin Dialog, CancelButton, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub Main ()
    Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
        TEXT 10, 10, 28, 12, "Name:"
        TEXTBOX 42, 10, 108, 12, .nameStr
        TEXTBOX 42, 24, 108, 12, .descStr
        CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
        OKBUTTON 42, 54, 40, 12
    End Dialog
    Dim Dlg1 As DialogName1
    Dialog Dlg1

    MsgBox Dlg1.nameStr
    MsgBox Dlg1.descStr
    MsgBox Dlg1.checkInt
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Choose Function

Return a value from a list of arguments.

Format:

Choose(*number*, *choice1*, [*choice2*,] [*choice3*,]…)

Choose will return a null value if number is less than one or greater than the number of choices in the list. If *number* is not an integer, it will be rounded to the nearest integer.

Example:

```
Sub Main
    number = 2
    GetChoice = Choose(number, "Choice1", "Choice2", "Choice3")
    Print GetChoice
End Sub
```

Chr Function

Return a one-character string whose ASCII number is the argument.

Format:

Chr(*integer*)

Chr returns a String

Example:

```
Sub ChrExample ()
    Dim X, Y, Msg, NL
```

```
        NL = Chr(10)
        For X = 1 to 2
            For Y = Asc("A") To Asc("Z")
                Msg = Msg & Chr(Y)
            Next Y
        Msg = Msg & NL
        Next X
        MsgBox Msg
    End Sub
```

CInt Function

Convert any valid expression to an integer.

Format:

CInt(*expression*)

Example:

```
    Sub Main ()
        Dim y As Long

        y = 25
        If VarType(y) = 2 Then
           Print y
           x = CInt(y)
           'Converts long value of y to an integer value in x
           Print x
        End If

    End Sub
```

CLng Function

Convert any valid expression into a Long.

Format:

CLng(*expression*)

Example:

```
    Sub Main ()
        Dim y As Integer

        y = 25
        If  VarType(y) = 2 Then
           Print y
           x = CLng(y)
           ' Converts integer value of y to a long value in x
           Print x
        End If

    End Sub
```

Close Statement

Close active file.

Format:

Close [*filenumber*]

If the optional *filenumber* parameter is omitted, all open files will be closed.

Related Topics: EOF, Input, Line Input, Open, Print #, Write #

Example:

```
    Sub Make3Files ()
        Dim I, FNum, FName                ' Declare variables.
        For I = 1 To 3
            FNum = FreeFile               ' Determine next file number.
            FName = "TEST" & FNum
            Open FName For Output As FNum  ' Open file.
            Print #I, "This is test #" & I ' Write string to file.
```

```
        Print #I, "Here is another ";  "line";  I
    Next I
    Close                           ' Close all files.
End Sub
```

ComboBox Statement

Use a combo box in a dialog to allow the user to make a selection either by typing text into the combo box or by selecting an item from its list. If there are more items in the list than will fit in the combo box, a scroll bar will appear allowing access to all items in the list.

Format:

ComboBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, *listsource*, *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_3()
Dim LISTSRC$(2)
LISTSRC(0) = "Item 1"
LISTSRC(1) = "Item 2"
LISTSRC(2) = "Item 3"
Begin Dialog DIALOG_2 16,16, 204, 96, "Test ComboBox"
  ComboBox 36,4,128,32, LISTSRC$(), .COMBOBOX_1
  TextBox 36,40,128,20, .TEXTBOX_2
  OKButton 8,64,76,20
  CancelButton 108,64,72,20
End Dialog
Dim Dlg2 As DIALOG_2
Dlg2.COMBOBOX_1 = "Combo List"
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.COMBOBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Const Statement

Assign a symbolic name to a constant value.

Format:

[Global] Const *name* = *expression*

A constant must be defined before it is used.

The **Global** statement must be outside the procedure section (i.e., not in a **Sub** or **Function**) of Enable. Global variables are available to all functions and subroutines.

The definition of a **Const** in Enable outside the procedure is global. The syntax, **Global Const** and **Const**, used below, outside the procedure, are identical.

A type declaration character may be used (see Other Data Types). If no type declaration character is used, Enable will automatically assign one of the following data types to the constant by evaluating *expression*:

Long (if *expression* evaluates to a long or integer),
Double (if a decimal place is present) or
String (if *expression* evaluates to a string).

Example:

```
Global Const GloConst = 142
Const MyConst = 122
                              ' Global to all procedures in a module
Sub Main ()
```

```
    Dim Answer, Msg, NL     ' Declare variables.
    Const PI = 3.14159
    NL = Chr(10)            ' Define newline.
    CurPath = CurDir()      ' Get current path.
    ChDir "\"
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change "
    Msg = Msg & "back to your previous default directory."
    Answer = MsgBox(Msg)    ' Get user response.
    ChDir CurPath           ' Change back to user default.
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg              ' Display results.
    Myvar = myConst + PI + GloConst
    Print MyVar
End Sub
```

CopyToClipboard Subroutine (eXpress Plus)

Copy the marked text to the Windows clipboard. This subroutine must be preceded by a **MarkBlock** subroutine.

Format:

CopyToClipboard

Related Topics: MarkBlock , PasteFromClipboard

Cos Function

Return the cosine of an angle.

Format:

Cos(*expression*)

The *radian* argument must be expressed in radians and must be a valid numeric expression. **Cos** will, by default, return a Double unless a Single or Integer is specified as the return value.

Example:

```
Sub Main()
    Dim J As Double
    Dim I As Single              ' Declare variables.
    Dim K As Integer
    For I =1 To 10    '
        Msg = Msg & Cos(I) & ", " 'Cos function call
        J=Cos(I)
        Print J
        K=Cos(I)
        Print K
    Next I
    MsgBox Msg                  ' Display results.
    MsgBox Msg1
End Sub
```

CreateObject Function

Create an OLE automation object.

Format:

CreateObject(*class*)

The *class* parameter has the following Format:

"*appname.objecttype*"

The *class* parameter has the following parts:

Part	Description
appname	Name of the application providing the object.
Objecttype	Type or class of object to create.

Example:

```
Sub BUTTON_BAR_02()
  Dim word6 as object
```

```
  Set word6 = CreateObject("Word.Basic")
  word6.AppShow
  word6.FileNewDefault
  word6.ViewPage
  word6.InsertPara
  word6.Insert DF_CUST_NAME + Chr$(13) + DF_ADDRESS1 + Chr$(13)
  if Left$(DF_ADDRESS2,1) <> " " then
     word6.Insert DF_ADDRESS2 + Chr$(13)
  end if
  if Left$(DF_ADDRESS3,1) <> " " Then
     word6.Insert DF_ADDRESS3 + Chr$(13)
  end if
  word6.insert  DF_CITY  + ", " + DF_STATE + " " + DF_ZIP
  word6.InsertPara
  word6.InsertPara
  word6.Insert "Dear Valued Customer:"
  word6.InsertPara
  word6.InsertPara
  word6.Insert "The following is your account number and "
  word6.Insert "telephone number as stored in our database:"
  word6.InsertPara
  word6.Bold 1
  word6.Insert "Account:" + Chr$(9) + DF_ACCOUNT + Chr$(13)
  word6.insert "Phone number:" + Chr$(9) + DF_PHONE
  word6.bold 0
  word6.InsertPara
  word6.Insert "If either of the above is incorrect, please "
  word6.Insert "contact us immediately."
  word6.InsertPara
  word6.InsertPara
  word6.Insert "Sincerely,"
  word6.InsertPara
  word6.Insert " John J. Doe"
  word6.Insert " Customer Services Rep."
  word6.InsertPara
  MsgBox "Your letter is ready to print.  Make adjustments " _
     & "in Word and print before pressing the OK button below."
End Sub
```

CSng Function

Convert any valid expression to a Single.

Format:

CSng(*expression*)

Example:

```
Sub Main ()
-  Dim y As Integer

   y = 25
   If  VarType(y) = 2 Then
     Print y
     x = CSng(y)
     'Converts the integer value of y to a single value in x
     Print x
   End If

End Sub
```

CStr Function

Convert any valid expression to a String.

Format:

CStr(*expression*)

Use **CStr** to force the result to be expressed as a String. Use the **CStr** function instead of **Str** to provide internationally aware conversions from any other data type to a String; e.g., different decimal separators are properly recognized depending on the local setting of your system.

The data in *expression* determines what is returned according to the following:

If *expression* is	CStr returns
Boolean	A string containing true (-1) or false (0).
Other numeric	A string containing the number.

Related Topics: Str

CurDir Function

Return the current path for the specified drive.

Format:

CurDir(*drive*)

CurDir returns a Variant.

Related Topics: ChDir, ChDrive, Dir, MkDir, RmDir

Example:

```
'Declare Function CurDir Lib "NewFuns.dll" ()–As String
Sub Form_Click ()
    Dim Msg, NL             ' Declare variables.
    NL = Chr(10)            ' Define newline.
    Msg = "The current directory is: "
    Msg = Msg & NL & CurDir()
    MsgBox Msg              ' Display message.
End Sub
```

CurrentPage Function (UTS eXpress Plus)

Return the number of the page currently displayed on the screen (active page).

Format:

CurrentPage()

Note: This function is only meaningful when the screen is configured for two or more pages (see, Pages, in UTS eXpress Plus Configuration or in UTS eXpress Net Administration). Furthermore, this function does not apply to page usage in T27 emulation.

CurrentScreen Function (eXpress Plus)

Return the currently opened environment (screen) number as assigned in the eXpress configuration.

Format:

CurrentScreen()

Returns an integer.

Example:

```
Sub Main()
    ' Variables
    MaxScreens = 24           ' set to your maximum to be configured.

    CurScrn = CurrentScreen   ' Get current screen, then find 1 previous
    x = CurScrn – 1
    If x = 0 Then
        x = MaxScreens
    End If

    Do
        If ScreenOpen(x) Then ' check if screen is open
            ActivateScreen(x) ' activate the previous screen found
            Exit Do
        Else
            x = x - 1         ' Screen wasn't found, so try next count.
            If x = 0 Then
```

```
                x = MaxScreens
            End If
        End If
    Loop
End Sub
```

CVar Function

Return a value from a list of arguments.

Format:

CVar (*expression*)

Example:

```
Sub Main
      Dim MyInt As Integer
      MyInt = 4534
      Print MyInt
      MyVar = CVar(MyInt & "0.23") 'makes MyInt a Variant + 0.32
      Print MyVar
End Sub
```

Date Function

Return the current system date.

Format:

Date

or

Date()

Date returns a Variant of **VarType 8** (String) containing a date.

Related Topics: Day, Format, Hour, Minute, Month, Now, Second, Weekday, Year

Examples:

```
x = Date()
Print Date
Print x
Print VarType(Date)

SysDate = Date
MsgBox Sysdate,0,"System Date"

' Returns current system date in the system-defined long
' date format.
MsgBox Format(Date, "Short Date") & " Short Date"
MsgBox Format(Date, "Long Date") & " Long Date"
```

DateSerial Function

Return a variant (Date) corresponding to the *year*, *month* and *day* specified.

Format:

DateSerial (*year*, *month*, *day*)

All three parameters for the **DateSerial** function are required and must be valid.

Related Topics: DateValue, Day, Month, TimeSerial, TimeValue, Year

Example:

```
Sub Main
      Dim MDate
      MDate = DateSerial(1959, 5, 29)
      Print MDate
End Sub
```

DateValue Function

Return a variant (Date) corresponding to the *dateexpression* specified.

Format:

DateValue(*dateexpression*)

The *dateexpression* parameter can be a string or any expression that can represent a date, time or both a date and a time.

Related Topics: DateSerial, Day, Month, TimeSerial, TimeValue, Year

Example:

```
Sub Main()
Dim v As Variant
Dim d As Double
        d = Now
        Print d
        v = DateValue("1959/05/29")
        MsgBox (VarType(v))
        MsgBox (v)
End Sub
```

Day Function

Return an integer between 1 and 31 that is the portion of the *date* parameter representing the day of the month.

Format:

Day(*date*)

The *date* parameter is any string expression.

The returned integer represents the day of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Format, Hour, Minute, Month, Now, Second, Weekday, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Day(MyDate)
    print x

End Sub
```

Declare Statement

Refer to an external procedure in a Dynamic Linking Library (DLL).

Formats:

Declare Sub *procedure* Lib "*library*" [Alias "*aliasname*"] [(*arguementlist*)]

or

Declare Function *procedure* Lib "*library*" [Alias "*aliasname*"] [(*arguementlist*)] [As *type*]

The *procedure* parameter is the name of the function or subroutine being called.

Supply the name of the DLL that contains the *procedure* on the *library* parameter.

The optional **Alias** *aliasname* clause is used to supply the procedure name in the DLL if different from the name specified on the *procedure* parameter.

When the optional *arguementlist* needs to be passed, the format is as follows:

([[ByVal] *variable* [As *type*][,[ByVal] *variable* [As *type*]]...])

The optional **ByVal** parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional **As** *type* parameter is used to specify the data type. Valid types are **String**, **Integer**, **Single**, **Double**, **Long** and **Variant** (see Other Data Types).

If a procedure has no arguments, use double parentheses () only to assure that no arguments are passed. For Example:

```
Declare Sub OneTime Lib "Check" ()
```

The following syntax is an Enable extension to the Declare statement but is not supported by Microsoft Visual Basic.

Declare Function *procedure* App [Alias "*aliasname*"] [(*argumentlist*)][As Type]

This form of the Declare statement refers to a function located in the executable file located in the application where Enable is embedded.

Related Topics: Call

Example:

```
Declare Function GetFocus Lib "User" () As Integer
Declare Function GetWindowText Lib "User" (ByVal hWnd%, _
    ByVal Mess$, ByVal cbMax%) As Integer

Sub Main
    Dim hWindow%
    Dim str1 As String * 51
    Dim str2 As String * 25

    hWindow% = GetFocus()
    print "GetWindowText returned: ", _
        GetWindowText( hWindow%, str1,51 )
    print "GetWindowText2 returned: ", _
        GetWindowText( hWindow%, str2, 25)
    print str1
    print str2
End Sub
```

Dialog Function

Return a value corresponding to the button the user chooses.

Format:

Dialog [(*dialogrecord*)]

The **Dialog()** function is used to display the dialog box specified by *dialogrecord*. The *dialogrecord* parameter is the name of the dialog and must be defined in a preceding **Dim** statement.

The return value or button:

Value	Description
-1	OK button.
0	Cancel button.
>0	A command button where 1 is the first push button in the definition of the dialog, 2 is the second, and so on.

Related Topics: Begin Dialog, CancelButton, CheckBox, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
' This sample shows all of the dialog controls on one
' dialog and how to vary the response based on which
' PushButton was pressed.

Sub Main ()
  Dim MyList$(2)
  MyList(0) = "Banana"
  MyList(1) = "Orange"
  MyList(2) = "Apple"
Begin Dialog DIALOGNAME1 59,111, 240, 147, "Test Dialog"
  OptionGroup .GRP1
    OptionButton 42,56,48,12, "Option&1"
    OptionButton 42,66,48,12, "Option&2"
  OptionGroup .GRP2
    OptionButton 42,92,48,12, "Option&3"
    OptionButton 42,102,48,12, "Option&4"
  GroupBox 132,81,70,36, "Group"
  Text 10,10,28,12, "Name:"
  TextBox 40,10,50,12, .joe
  ListBox 102,10,108,16, MyList$(), .MyList1
  DropListBox 42,32,108,36, MyList$(), .DropList1$
  CheckBox 142,56,48,12, "Check&A", .Check1
```

```
  CheckBox 142,66,48,12, "Check&B", .Check2
  CheckBox 142,92,48,12, "Check&C", .Check3
  CheckBox 142,102,48,12, "Check&D", .Check4
  CancelButton 42,124,40,12
  OKButton 90,124,40,12
  PushButton 140,124,40,12, "&Push Me 1"
  PushButton 190,124,40,12, "Push &Me 2"
End Dialog
  Dim Dlg1 As DialogName1
  Dlg1.joe  = "Def String"
  Dlg1.MyList1 = 1
  Dlg1.DropList1 = 2
  Dlg1.grp2 = 1
  ' Dialog returns -1 for OK, 0 for Cancel, button #
  ' for PushButtons
  button = Dialog( Dlg1 )
  ' MsgBox "button: " & button ' uncomment for return val
  If button = 0 Then Return
  MsgBox "TextBox: "& Dlg1.joe
  MsgBox "ListBox: " & Dlg1.MyList1
  MsgBox Dlg1.DropList1
  MsgBox "grp1: " & Dlg1.grp1
  MsgBox "grp2: " & Dlg1.grp2
  Begin Dialog DialogName2 60, 60, 160, 60, "Test Dialog 2"
      Text 10, 10, 28, 12, "Name:"
      TextBox 42, 10, 108, 12, .fred
      OkButton 42, 44, 40, 12
  End Dialog
  If button = 2 Then
      Dim Dlg2 As DialogName2
      Dialog Dlg2
      MsgBox Dlg2.fred
  ElseIf button = 1 Then
      Dialog Dlg1
      MsgBox Dlg1.MyList1
  End If
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Dim Statement

Allocate storage for, and declare the data type of, variables and arrays in a module.

Format:

Dim *variablename*[(*subscripts*)] [As *type*][,*name*] [As *type*]]

The types currently supported are **Integer**, **Long**, **Single**, **Double,** and **String** and **Variant**.

Example:

```
Sub Main
    Dim x As Long
    Dim y As Integer
    Dim z As single
    Dim a As double
    Dim s As String
    Dim v As Variant ' This is the same as Dim x or Dim x as any
```

Dir Function

Return a file/directory name that matches the given path and attributes.

Format:

Dir[(*path*,*attributes*)]

The path parameter is a string expression that contains a file name. The file name may include drive and directory specifications. In addition, standard wildcard characters, asterisk (*) and question mark (?), may be included.

The Dir function returns the first file name that matches the path specification. A Dir function with no parameters specified will return the next file name that matches the path specification. When no more files meet the specification, an empty string is returned.

The attributes parameter is a numeric expression containing a number equal to the sum of all required attributes. The attributes are:

Value	Attribute
0	Normal
2	Hidden
4	System file
8	Volume label
16	Directory

Related Topics: ChDir, ChDrive, CurDir, MkDir, RmDir

DlgText Statement

Set or change the text of a dialog control.

Format:

DlgText "*controlname*", *string*

The *controlname* parameter is the name of the control on the dialog box. The *string* parameter is the value to which it is set.

Related Topics: DlgEnable, DlgVisible

Example:

```
If ControlID$ = "Chk2" Then
    DlgText "History", "Push to display nested dialog"
End If
```

DlgVisible Statement

Hide or make visible a particular control on a dialog box.

Format:

DlgVisible "*controlname*", *value*

The *controlname* parameter is the name of the control on the dialog box. The *value* parameter is the value to which it is set: 1 = Visible, 0 = Hidden. "On" is equal to 1. If the *value* parameter is omitted, the status of the control toggles.

Related Topics: DlgEnable, DlgText

Example:

```
If ControlID$ = "Chk1" Then
          DlgEnable "Group", On
          DlgVisible "Chk2"
          DlgVisible "History"
End If
```

See the DlgEnable Statement for a complete example.

Do...Loop Statement

Repeat a group of statements while a condition is true or until a condition is met.

Formats:

Do [While | Until *condition*]
    [*statement(s)*]
    [Exit Do]
    [*statement(s)*]
Loop [While | Until *condition*]

The *condition* parameter may be a numeric or string expression.

Related Topics: While..Wend, With

See also, Data Types, Operators and Precedences.

Example:

```
Sub Main ()
    Dim Value, Msg        ' Declare variables.
    Do
        Value = InputBox("Enter a value from 5 to 10.")
        If Value >= 5 And Value <= 10 Then
            Exit Do       ' Exit Do...Loop.
        Else
            Beep          ' Beep if not in range.
        End If
    Loop
End Sub
```

DropListBox Statement

Use a drop-down list box in a dialog to allow the user to make a selection from a drop-down list. The drop-down list is useful when you wish to conserve space on the dialog.

Format:

DropListBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, *listsource*, *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_3()
Dim LISTSRC$(2)
LISTSRC(0) = "Item 1"
LISTSRC(1) = "Item 2"
LISTSRC(2) = "Item 3"
Begin Dialog DIALOG_2 0,0, 204, 96, "Test DropListBox"
  DropListBox 36,12,128,44, LISTSRC$(), .DROPLISTBOX_1
  TextBox 36,36,128,20, .TEXTBOX_2
  OKButton 8,64,76,20
  CancelButton 108,64,72,20
End Dialog
Dim Dlg2 As DIALOG_2
Dlg2.DROPLISTBOX_1 = 0
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.DROPLISTBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

End Statement

End a program or a block of statements such as a Sub procedure or a Function.

Format:

End [Dialog | Function | If | Sub | Select | Type | With]

The **End** statement, without any parameters, may be used anywhere within a procedure to close files opened with the **Open** statement and to clear variables. To suspend execution without clearing variables or closing files, use the **Stop** statement.

Related Topics: Dialog, Do...Loop, Exit, Function, If...Then...Else, Select Case, Stop, Sub, Type, With

Example:

```
Sub Main()

    Dim Var1 as String
```

```
    Var1 = "hello"
    MsgBox " Calling Test"
    Test Var1
    MsgBox Var1

End Sub

Sub Test(wvar1 as string)

    wvar1 = "goodbye"
    MsgBox "Use of End Statement"
    End

End Sub
```

DlgEnable Statement

Enable or disable a particular control on a dialog box.

Format:

DlgEnable "*controlname*", *value*

The *controlname* parameter is the name of the control on the dialog box. The *value* parameter is the value to which it is set: 1 = Enable, 0 = Disable. "On" is equal to 1 in the example below. If the *value* parameter is omitted, the status of the control toggles.

Related Topics: DlgText, DlgVisible

Example:

```
Sub Main
    Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
        Text 8,10,73,13, "Text Label:"
        TextBox 8, 26, 160, 18, .FText
        CheckBox 8, 56, 203, 16, "Check to display controls",. Chk1
        GroupBox 8, 79, 230, 70, "This is a group box:", .Group
        CheckBox 18,100,189,16, "Check to change button text", .Chk2
        PushButton 18, 118, 159, 16, "File History", .History
        OKButton 177, 8, 58, 21
        CancelButton 177, 32, 58, 21
        End Dialog

    Dim Dlg1 As UserDialog1
    x = Dialog( Dlg1 )
End Sub


Function Enable( ControlID$, Action%, SuppValue%)

Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox",. ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
End Dialog
Dim Dlg2 As UserDialog2
Dlg2.FText = "Your default string goes here"


Select Case Action%

Case 1
    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
```

```
        DlgVisible "History", 0
    Case 2
        If ControlID$ = "Chk1" Then
            DlgEnable "Group"
            DlgVisible "Chk2"
            DlgVisible "History"
        End If

        If ControlID$ = "Chk2" Then
            DlgText "History", "Push to display nested dialog"
        End If

        If ControlID$ = "History" Then
            Enable =1
            x = Dialog( Dlg2 )
        End If


    Case Else

    End Select
    Enable = 1

    End Function
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

EnterText Subroutine (eXpress Plus)

Enter the specified string at the current cursor position on the screen.

Format:

EnterText *string*

The *string* parameter is any valid string expression.

Example:

(See Begin Dialog Statement).

EOF Function

Return a value during file input that indicates whether the end of a file has been reached.

Format:

EOF(*filenumber*)

Related Topics: Close, Input, Line Input, Open, Print #, Write #

Example:

```
' This example uses the Input function to read 10 characters
' at a time from a file and display them in a MsgBox.  This
' example assumes that TESTFILE is a text file with a few
' lines of sample data.

Sub Main
    Open "TESTFILE" For Input As #1    ' Open file.
    Do While Not EOF(1)                ' Loop until end of file.
    MyStr = Input(10, #1)              ' Get ten characters.
    MsgBox MyStr
    Loop
    Close #1                           ' Close file.
End Sub
```

Erase Statement

Reinitialize the elements of a fixed array.

Format:

Erase *arrayname*[,*arrayname*] ...

Related Topics: Dim

Example:

```
' This example demonstrates some of the features of
' arrays.  The lower bound for an array is 0 unless
' option base has been set as in this Example:

Option Base 1
Sub Main
    Dim a(10) As Double
    MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
    Dim i As Integer
    For i = 0 to 3
        a(i) = 2 + i * 3.1
    Next i
    Erase( a ) ' If this line is uncommented,
               ' then the values will all be 0
    Print a(0),a(1),a(2), a(3)
End Sub
```

Exit Statement

Exit a loop or procedure.

Format:

Exit {Do | For | Function | Sub}

Related Topics: End, Stop

Example:

```
' This sample shows Do ... Loop with Exit Do to get out.
Sub Main ()
    Dim Value, Msg                ' Declare variables.
    Do
       Value = InputBox("Enter a value from 5 to 10.")
       If Value >= 5 And Value <= 10 Then    ' Check range.
          Exit Do                 ' Exit Do...Loop.
       Else
          Beep                    ' Beep if not in range.
       End If
    Loop
End Sub
```

Exp Function

Return *e* to a power (*e* ^ *number*).

Format:

Exp(*number*)

The value of the constant *e* is approximately 2.71828.

Related Topics: Log

Example:

```
Sub ExpExample ()
    ' Exp(x) is e ^x so Exp(1) is e ^1 or e.
    Dim Msg, ValueOfE    ' Declare variables.
    ValueOfE = Exp(1)    ' Calculate value of e.
    Msg = "The value of e is " & ValueOfE
    MsgBox Msg           ' Display message.
End Sub
```

FileCopy Function

Copy a file from source to destination.

Format:

FileCopy(*sourcefile*,*destinationfile*)

The *sourcefile* and *destinationfile* parameters must be valid string expressions. The *sourcefile* is the file name of the file to copy; *destinationfile* is the file name to which the *sourcefile* is to be copied.

FileLen Function

Return a Long integer that is the length of the file in bytes.

Format:

FileLen(*filename*)

Example:

```
Sub Main

       Dim MySize
       MySize = FileLen("C:\TESTFILE")      ' Returns file length
(bytes).
       Print MySize

End Sub
```

FileOpenDialog Function (eXpress Plus)

Open the File Open dialog.

Format:

FileOpenDialog (*initialdirectory*, *filename*, *defaultextension*, *filter*, *title*)

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to be displayed initially when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1|*filter-1*|*file-type-description-2*|*filter-2*|...*file-type-description-n*|*filter-n*

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the *file-type-description* and the second "*.txt" is the *filter*.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related topics: FileSaveDialog, LoadScreen, SaveScreen

Example:

```
Sub Main()
      Dim fname as String

      fname = FileOpenDialog("C:\AA_UX32_3.00\T27Scripts", "", "FRM",
"Form files(*.frm)|*.frm|All files(*.*)|*.*", "Select Load Form File")
      If fname <> "" Then
             loadScreen(fname)
      Else
             MsgBox "Load Cancelled"
      End If
End Sub
```

FileSaveDialog Function (eXpress Plus)

Open the File Save dialog.

FileSaveDialog (*initialdirectory*, *filename*, *defaultextension*, *filter*, *title*)

The *initialdirectory* is any string expression containing the initial directory to use when the dialog is opened.

The *filename* parameter is any string expression containing the file name to be displayed initially when the dialog is opened (usually left blank).

The *defaultextension* parameter is any string expression containing the default file extension to use if one is not supplied by the user.

The *filter* parameter is any string expression containing the filter or filters used to restrict file selection. A *filter* is setup as follows:

file-type-description-1|*filter-1*|*file-type-description-2*|*filter-2*|...*file-type-description-n*|*filter-n*

For example, to show all text files or all files use the following filter string:

"Text files(*.txt)|*.txt|All files(*.*)|*.*"

"Text files(*.txt)" is the *file-type-description* and the second "*.txt" is the *filter*.

The *title* parameter is any string expression containing the title to display in the file dialog.

This function returns the full file name, including path, of the selected file. If a file is not selected, the dialog is cancelled, and an empty string is returned.

Related Topics: FileOpenDialog Function (eQuate), LoadScreen , SaveScreen

Example:

```
Sub Main()
      Dim fname as String

      fname = FileSaveDialog("C:\AA_UX32_3.00\T27Scripts", "", "FRM",
"Form files(*.frm)|*.frm|All files(*.*)|*.*", "Select Save Form File")
      If fname <> "" Then
            SaveScreen(fname)
      Else
            MsgBox "Save Cancelled"
      End If
End Sub
```

Fix Function

Return the integer portion of a number.

Format:

Fix(*number*)

Related Topics: Int

For Each...Next Statement

Repeat the group of statements for each element in an array or collection.

For Each *element* In *group*

[*statement(s)*]

[Exit For]

[*statement(s)*]

Next [*element*]

For Each...Next statements can be nested if each loop element is unique. The **For Each...Next** statement cannot be used with and array of user defined types.

Example:

```
Sub Main
  dim z(1 to 4) as double
  z(1) = 1.11
  z(2) = 2.22
  z(3) = 3.33
  For Each v In z
    Print v
  Next v
End Sub
```

For...Next Statement

Repeat the execution of a block of statements for a specified number of times.

Format:

For *counter* = *expression1* To *expression2* [Step *increment*]

[*statement(s)*]

Next [*counter*]

See also, Data Types, Operators and Precedences.

Example:

```
Sub main ()
Dim x,y,z

    For x = 1 to 5
        For y = 1 to 5
            For z = 1 to 5
            Print "Looping" ,z,y,x
            Next z
        Next y
    Next x
End Sub
```

Format Function

Format a string, number or variant (date/time) data type to a format expression.

Format:

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

Format returns a string.

The **Format** has two parts:

Part	Description
expression	Expression to be formatted.
fmt	A string of characters that specifies how the expression is to be displayed, or the name of a commonly used format that has been predefined in Enable Basic. Do not mix different type format expressions in a single *fmt* parameter.

If the *fmt* parameter is omitted or is zero-length and the *expression* parameter is a numeric, **Format** provides the same functionality as the **Str** function by converting the numeric value to the appropriate return data type. Positive numbers converted to strings using **Format** lack the leading space reserved for displaying the sign of the value, whereas those converted using **Str** retain the leading space.

To format numbers, you can use the commonly used formats that have been predefined in Enable Basic or you can create user-defined formats with standard characters that have special meaning when used in a format expression.

Predefined numeric format names:

Format Name	Desctiption
General Number	Display the number as is – without thousands separators.
Fixed	Display at least one digit to the left and two digits to the right of the decimal separator.
Standard	Display the number with thousands separators, if appropriate; display two digits to the right of the decimal separator.
Percent	Display the number multiplied by 100 with a percent sign (%) appended to the right; display two digits to the right of the decimal separator.
Scientific	Use standard scientific notation.
True/False	Display False if number is 0; otherwise, display True.

The following shows the characters you can use to create user-defined number formats:

Character	Meaning
Null string	Display the number with no formatting.
0	Digit placeholder. Display a digit or a zero. If the number being formatted has fewer digits than there are zeros (on either side of the decimal) in the format expression, leading or trailing zeros are displayed. If the number has more digits to the right of the decimal separator than there are zeros to the right of the format expression's decimal separator, the number is rounded to as many decimal places as there are zeros. If the number has more digits to left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, the extra digits are displayed without modification.

Character	Meaning
#	Digit placeholder. Displays a digit or nothing. If there is a digit in the expression being formatted in the position where the # appears in the format string, displays it; otherwise, nothing is displayed.
.	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator.
%	Percentage placeholder. The percent character (%) is inserted in the position where it appears in the format string. The expression is multiplied by 100.
,	Thousand separator. The thousands separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Use of this separator as specified in the format statment contains a comma surrounded by digit placeholders (0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means, "scale the number by dividing it by 1000, rounding as needed."
E-E+e-e+	Scientific format. If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e- or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a plus sign next to positive exponents.
:	Time separator. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
/	Date separator. The actual character used as the date separator in the formatted output depends on Date Format specified in the International section of the Control Panel.
- $ ( ) space	Display a literal character. To display a character other than one of those listed, precede it with a backslash (\).
\	Display the next character in the format string. The backslash itself is not displayed. To display a backslash, use two backslashes (\\). Examples of characters that can't be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).
"String"	Display the string inside the double quotation marks. To include a string in *fmt* from within Enable, you must use the ANSI code for a double quotation mark Chr(34) to enclose the text.
*	Display the next character as the fill character. Any empty space in a field is filled with the character following the asterisk.

Unless the *fmt* argument contains one of the predefined formats, a format expression for numbers can have from one to four sections. Each section must be separated by semicolons.

If you use	The result is
One section only	The format expression applies to all values.
Two	The first section applies to positive values, the second to negative values.
Three	The first section applies to positive values, the second to negative values, and the third to zeros.
Four	The first section applies to positive values, the second to negative values, the third to zeros, and the fourth to Null values.

The following example has two sections: the first defines the format for positive values and zeros; the second section defines the format for negative values:

"$#,##0;($#,##0)"

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero:

"$#,##0;;\Z\e\r\o"

Sample format expressions for numbers are shown below (examples assume the Country is set to the United States in the International section of the Control Panel). The first column contains the format strings. The other columns contain the output that results if the formatted data has the value given in the column headings:

Format (*fmt*)	Positive 3	Negative 3	Decimal 3	Null
Null string	3	-3	0.3	
0	3	-3	1	
0.00	3.00	-3.00	0.30	
#,##0	3	-3	1	
#,##0.00;;;Nil	3.00	-3.00	0.30	Nil
$#,##0;($#,##0)	$3	($3)	$1	
$#,##0.00;($#,##0.00)	$3.00	($3.00)	$0.30	
0%	300%	-300%	30%	
0.00%	300.00%	-300.00%	30.00%	
0.00E+00	3.00E+00	-3.00E+00	3.00E-01	
0.00E-00	3.00E00	-3.00E00	3.00E-01	

Numbers can also be used to represent date and time information. You can format date and time serial numbers using date and time formats or number formats because date/time serial numbers are stored as floating-point values.

To format dates and times, you can use either the commonly used formats that have been predefined in Enable or create user-defined date/time formats using standard characters that have special meaning when used in a format expression.

The following table shows the predefined data format names you can use and the meaning of each:

Format Name	Description
General	Display a date and/or time. For real numbers, display a date and time (e.g. 4/3/93 03:34 PM); if there is no fractional part, display only a date (e.g. 4/3/93); if there is no integer part, display time only (e.g. 03:34 PM).
Long Date	Display a Long Date, as defined in the International section of the Control Panel.
Medium Date	Display a date in the same form as the Short Date, as defined in the international section of the Control Panel, except spell out the month abbreviation.
Short Date	Display a Short Date, as defined in the International section of the Control Panel.
Long Time	Display a Long Time, as defined in the International section of the Control panel. Long Time 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 (e.g. 17:45)

This table shows the characters you can use to create user-defined date/time formats:

Character	Meaning
c	Display the date as ddddd and display the time as ttttt, in that order.
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).
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).
ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If mm immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the day of the year as a two-digit number (00-99)
yyyy	Display the day of the year as a four-digit number (0000-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
ttttt	Display a time serial number as a complete time (including hour, minute, and second) formatted using the time separator defined by the Time Format in the International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.
AM/PM	Use the 12-hour clock and display an uppercase AM/PM.
am/pm	Use the 12-hour clock display a lowercase am/pm.

Character	Meaning
A/P	Use the 12-hour clock display a uppercase A/P.
a/p	Use the 12-hour clock display a lowercase a/p
AMPM	Use the 12-hour clock and display the contents of the 11:59 string (s1159) in the WIN.INI file with any hour before noon; display the contents of the 23:59 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of user-defined date and time formats:

Format	Display
m/d/yy	2/26/65
d-mmmm-yy	26-February-65
d-mmmm	26-February
mmmm-yy	February-65
hh:mm AM/PM	06:45 PM
h:mm:ss a/p	6:45:15 p
h:mm:ss	18:45:15
m/d/yy h:mm	2/26/65 18:45

Strings can also be formatted with **Format**. A format expression for strings can have one section or two sections separated by a semicolon.

If you use	The result is
One section only	The format expression applies to all string data.
Two	The first section applies to string data, the second to Null values and zero-length strings.

The following characters can be used to create a format expression for strings:

Character	Meaning
@	Character placeholder. Displays a character or a space. Placeholders are filled from right to left unless there is an exlamation character (!) in the format string.
&	Character placeholder. Display a character or nothing.
<	Force lowercase.
>	Force uppercase.
!	Force placeholders to fill from left to right instead of right to left.

Related Topic: Str

Example:

```
' This example shows various uses of the Format function to
' format values using both named and user-defined formats.
' For the date separator (/), time separator (:), and AM/
' PM literal, the actual formatted output displayed by your
' system depends on the locale settings on which the code
' is running.  When times and dates are displayed in the
' development environment, the short time and short date
' formats of the code locale are used.  When displayed by
' running code, the short time and short date formats of
' the system locale are used, which may differ from the code
' locale.  For this example, English/United States is
' assumed.

' MyTime and MyDate are displayed in the development
' environment using current system short time and short
' date settings.

Sub Main
```

```
MyTime = "08:04:23 PM"
MyDate = "January 27, 1993"

MsgBox Now
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"

MsgBox Day( MyDate ) & " Days"
MsgBox Month( MyDate ) & " Months"
MsgBox Year( MyDate ) & " Years"

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

' Returns current system date in the system-defined long
' date format.
MsgBox Format (Date, "Short Date")
MsgBox Format (Date, "Long Date")

MyStr = Format (MyTime, "h:n:s")      ' Returns "17:4:23".
MyStr = Format (MyTime, "hh:nn:ss AMPM")
                                      ' Returns "05:04:23 PM".
MyStr = Format (MyDate, "dddd, mmm d yyyy")
                         ' Returns "Wednesday, Jan 27 1993".

' If format is not supplied, a string is returned.
MsgBox Format (23)                    ' Returns "23".

' User-defined formats.
MsgBox Format (5459.4, "##,##0.00")   ' Returns "5,459.40".
MsgBox Format (334.9, "###0.00")      ' Returns "334.90".
MsgBox Format (5, "0.00%")            ' Returns "500.00%".
MsgBox Format ("HELLO", "<")          ' Returns "hello".
MsgBox Format ("This is it", ">")     ' Returns "THIS IS IT".
End Sub
```

FreeFile Function

Return the next valid unused file number.

Format:

FreeFile[()]

Example:

```
Sub Make3Files ()
    Dim I, FNum, FName                  ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile                 ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum   ' Open file.
        Print #I, "This is test #" & I  ' Write string to file.
        Print #I, "Here is another "; "line"; I
    Next I
    Close                               ' Close all files.
End Sub
```

Function Statement

Declare and define a procedure that can receive arguments and return a value of a specified data type.

Format:

Function *functionname* [(*argumentlist*)] [As type]
[*statement(s)*]
functionname = expression
[Exit Function]
[*statement(s)*]
functionname = expression
End Function

When the optional *argumentlist* needs to be passed, the format is as follows:

([ByVal] *variable* [As *type*][,[ByVal] *variable* [As *type*]]...)

The optional **ByVal** parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional **As** *type* parameter is used to specify the data type. Valid types are **String**, **Integer**, **Single**, **Double**, **Long** and **Variant** (see Other Data Types).

Related Topics: Dim, End, Exit, Sub

Example:

```
Sub Main
    For I = 1 to 10
        Print GetColor2(I)
    Next I
End Sub
Function GetColor2( c% ) As Long
   GetColor2 = c% * 25
   If c% > 2 Then
      GetColor2 = 255       ' 0x0000FF - Red
   End If
   If c% > 5 Then
      GetColor2 = 65280     ' 0x00FF00 - Green
   End If
   If c% > 8 Then
      GetColor2 = 16711680  ' 0xFF0000 - Blue
   End If
End Function
```

Get Statement

Read from a disk file into a variable.

Format:

Get [#] *filenumber*, [*recordnumber*,] *variablename*

The Get statement has three parts:

Parameter	Description
filenumber	The number used to open the file.
recordnumber	For files opened in Binary mode, *recordnumber* is the byte position where reading starts.
variablename	The name of the variable used to receive the data from the file.

Related Topics: Open, Put

GetCursorCol Function (eXpress Plus)

Retrieve the column of the current cursor position within the logical screen.

Format:

GetCursorCol()

Related Topics: GetCursorRow, SetCursor

GetCursorRow Function (eXpress Plus)

Retrieve the row of the current cursor position within the logical screen.

Format:

GetCursorRow()

Related Topics: GetCursorCol, SetCursor

GetLastMsg Function (eXpress Plus)

In the UTS environment, this function retrieves the first 80 characters of the last message received (including any control sequences) from the host or communication system. In the T27 environment, this function retrieves the first 50 significant characters of the screen.

Format:

GetLastMsg()

The communication system may return multiple messages before control is returned to the script; therefore, only the last message is accessible by this function.

Since the message may contain control sequences, this function may not be very useful unless you are familiar with the handling of control sequences by the communications system. Consider using the GetScreenLine or GetScreenText functions.

Related Topics: GetScreenLine, GetScreenText, WaitForSpecificString, WaitForString

GetObject Function

Retrieve an OLE Automation object from a file.

Format:

GetObject("*filename*"[,"*class*"])

The **GetObject** function has these parts:

Part	Description
filename	Full path and name of file containing the object. If filename is an empty string (""), class is required, and the function returns the currently active object of the specified type.
class	String representing the class of the object.

The optional *class* parameter has the following Format:

"*appname.objecttype*"

The *class* parameter has the following parts:

Part	Description
appname	Name of the application providing the object.
objecttype	Type or class of object to get.

If the optional *class* is not supplied, the OLE2 DLLs determine the application based on the filename provided. Use the optional *class* parameter when the file supports more than one class.

Example:

```
Dim WordObject as Object
Set WordObject = GetObject ("C:\WINWORD\LETTERS\OLETST.DOC")
```

GetPage Subroutine (UTS eXpress Plus)

Set the specified page to the current screen. The current page is overwritten.

Format:

GetPage(*PageNumber*)

The *PageNumber* is any integer expression.

Note: This subroutine is only meaningful when the screen is configured for 2 or more pages (see, Pages, in UTS eXpress Plus Configuration or in UTS eXpress Net Administration). Furthermore, this subroutine does not apply to page usage in T27 emulation.

Related topics: GoToPage, StorePage

Example:

The following is a general-purpose script to perform **GoToPage**, **StorePage** and **GetPage** subroutines. The script can be used in conjunction with a custom tool bar or script menu. The script uses the first two character of the button or menu item caption to determine the action to be performed.

```
Sub Main()
' General paging action script

' The following dialog will prompt the user for a page number.  The user can
' click a button or just press a number key to select the page.
Begin Dialog PAGEDLG 1,58, 233, 29, "Select page (Click button or press key)"
      PushButton 4,8,17,12, "&1", .PushButton_1
      PushButton 24,8,17,12, "&2 ", .PushButton_2
      PushButton 44,8,16,12, "&3 ", .PushButton_3
      PushButton 64,8,16,12, "&4", .PushButton_4
      PushButton 84,8,16,12, "&5", .PushButton_5
      PushButton 104,8,16,12, "&6 ", .PushButton_6
      PushButton 124,8,16,12, "&7", .PushButton_7
      PushButton 144,8,16,12, "&8 ", .PushButton_8
      PushButton 164,8,16,12, "&9", .PushButton_9
      CancelButton 184,8,40,12
End Dialog


Dim D as PAGEDLG

      Cap = GetUserParam(3) ' Get caption of calling button or menu item

      Btn = Dialog(D) ' Prompt for page number

      If Btn >= 1 and Btn <= 9 Then
            Select Case UCase$(Left$(Cap, 2))     ' Perform action based on button or
                                                  ' menu item caption (1st 2 chars)
                  Case "ST"
                        StorePage(Btn)
                  Case "GE"
                        GetPage(Btn)
                  Case "GO"
                        GoToPage(Btn)
            End Select
      End If
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

GetScreenAttribute Function (eXpress Plus)

Return Protected, Blink and Video Off attribute states and the specified column and row position.

Format:

GetScreenAttribute (*column*, *row*)

The *column* and *row* parameters are any integer expressions.

The attribute is a numeric expression containing a number equal to the sum of all required attributes.

The attribute may be checked using an Integer or Constant:

Attribute	Integer	Constant
Normal	0	ATTR_NORMAL
Start of Field	1	ATTR_FIELD
Tab Stop	2	ATTR_TAB
Data Field Changed	4	ATTR_CHANGED
Protected	8	ATTR_PROTECTED
Video Off	16	ATTR_VIDEO_OFF
Numeric Only Input	32	ATTR_NUMERIC
Alphabetic Only Input	64	ATTR_ALPHA
Blinking	128	ATTR_BLINK
Right Justified Data	256	ATTR_RIGHT
UTS Low Intensity	512	ATTR_LOWINT
Reverse Video	1024	ATTR_REV

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topic: GetScreenColor

See also, Data Types, Operators and Precedences

Example:

```
x = GetScreenAttribute(12, 3)
If (x and 128) <> 0 then ' Mask out other attributes
      MsgBox "It's blinking"
End If
```

Example2:

```
'This script selects all typed characters in the current field.
'
Sub Main()
      Dim CRow as Integer
      Dim CCol as Integer
      Dim SCol as Integer
      Dim ECol as Integer
      Dim FoundBlank as Integer
      Dim s as Integer
      Dim l as Integer
      Dim EndDel as String

      ' Get the cursor row and column.
      CRow = GetCursorRow()
      CCol = GetCursorCol()

      If (GetScreenAttribute(CCol, CRow) and 8) = 8 Then ' Cursor must be in
                                                         ' an unprotected area.
                                                         ' Logical AND (and 8)
                                                         ' is used to mask out
                                                         ' other attributes.
            Exit Sub
      End If

      ' Find the end of the field (or end of the line)
      s = CCol
      ECol = 80
      Do
            If (GetScreenAttribute(s, CRow) and 8) = 8 Then
                  s = s - 1
                  ECol = s
                  Exit Do
            End If
            If s >= 80 Then
                  Exit Do
            End If
            s = s + 1
      Loop
      ' s now points to character before next protected region
      ' Work backward to the first non space (or beginning of
      ' the field) then get the end of the selection
      SCol = 1
      FoundBlank = False
      Do
            If (GetScreenAttribute(s, CRow) and 8) = 8 Then
                  SCol = s + 1
                  Exit Do
            End If
            If FoundBlank = False and GetScreenText(s, CRow, 1) <> " " Then
                  FoundBlank = True
                  ECol = s
            End If
```

```
                If s = 1 Then
                        SCol = s
                        Exit Do
                End If
                s = s - 1
        Loop
        ' Move the cursor the start of the field.
        SetCursor SCol, CRow
        ' Mark the selection
        MarkBlock SCol, CRow, ECol, CRow
        RefreshScreen
        ' Done.
End Sub
```

GetScreenColor Function (eXpress Plus)

Return a 2-digit hex number indicating the background and foreground color at the specified column and row position.

Format:

GetScreenColor(*column*, *row*)

The *column* and *row* parameters are any integer expressions.

Colors are expressed as an index in the range 0 to 7. The first digit is the background color index and the second is foreground color index.

Colors are: 0 = black, 1 = Red, 2 = Green, 3 = Yellow, 4 = Blue, 5 = Magenta, 6 = Cyan, 7 = white.

For example, white text on a red background is 17 hexadecimal or 23 decimal.

Related Topic: GetScreenAttribute

Example:

```
c = GetScreenColor(4, 12)
' Display the background and foreground color index
MsgBox "Background:" & Str$((c and &h70) / 16) & " Foreground:" & Str$(c and &h07)
```

GetScreenLine Function (eXpress Plus)

Retrieve one logical line of the mapped terminal screen buffer. This function will retrieve any text within the specified area including protected and video off.

Format:

GetScreenLine(*linenumber*)

This function returns a String.

The *linenumber* parameter is any integer expression, but must be within the range of 1 through the total number of lines of the terminal screen.

Related Topics: GetScreenText

Example:

This script downloads a 2200 host file to the PC. The host file must be accessible by the @ED processor (i.e., not a binary file). The script downloads the file one line at a time and is, therefore, slow and not suitable for large file transfers. Consider an FTP transfer as shown in an example of the eXpress Dialog Form Designer.

```
Sub Main()
 Dim ScreenLine as String
 Dim HFileElt as String
 Dim fn as Integer
 Dim TextLine as String
 Dim ThisLine as String
 Dim PrevLine as String
 Dim MsgLine as String

 Begin Dialog DOWNLOAD 149,47, 200, 120, "Download"
   OKButton 96,100,32,16
   CancelButton 136,100,32,16
   Text 12,4,100,8, "File Name:"
   TextBox 12,16,108,12, .HostFileBox
```

```
  Text 12,36,64,8, "Element name:"
  TextBox 12,48,108,12, .ElementNameBox
  Text 12,68,64,8, "PC file name:"
  TextBox 12,80,148,12, .PCFileBox
End Dialog

Dim DLDlg as DOWNLOAD
If dialog(DLDlg) = 0 then
     Exit Sub
End If
MBOK = MB_OK
DbugTitle = "Variable Textline"
fn = FreeFile
Open Trim$(DLDlg.PCFileBox) for output as fn
HFileElt = DLDlg.HostFileBox
If Right$(HFileElt,1) <> "." then
     HFileElt = HFileElt + "."
End If

If Trim$(DLDlg.ElementNameBox) <> "" then
     HFileElt = HFileElt + Trim$(DLDlg.ElementNameBox)
End If

EnterText chr$(30) & "@ed,rb " & HFileElt
UTSKey UK_TRANSMIT_KEY
Wait 1000
ThisLine = GetCursorRow() - 1
PrevLine = ThisLine - 1
TextLine = GetScreenLine(ThisLine)
If Left(TextLine,31) = chr$(30) + "END ED. NO CORRECTIONS APPLIED" then
     MsgLine = "Cannot read '" + HFileElt + "'"
     MsgBox MsgLine,MB_ICONEXCLAMATION,"Error"
     Exit Sub
End If
EnterText "N"
UTSKey UK_TRANSMIT_KEY
Wait 1000
TextLine = GetScreenLine(ThisLine)
While Left(TextLine,5) <> chr$(30) + "EOF:"
     TextLine = GetScreenLine(ThisLine)
     If Left(TextLine,1) <> chr$(30) then
            ScreenLine = GetScreenLine(PrevLine)
            TextLine = Mid(ScreenLine,2,79) + RTrim(TextLine)
     Else
            TextLine = RTrim(Mid(TextLine,2,79))
     End If
     Print #fn, TextLine
     EnterText "N"
     UTSKey UK_TRANSMIT_KEY
     Wait 400
     TextLine = GetScreenLine(ThisLine)
WEnd
EnterText "OMIT"
UTSKey UK_TRANSMIT_KEY
Close fn
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

GetScreenText Function (eXpress Plus)

Retrieve a text string from the specified positions within the logical screen. This function will retrieve any text within the specified area including protected and video off.

Format:

GetScreenText(*col*, *row*, *len*)

The *col*, *row* and *len* parameters are any integer expression.

If *row* is specified as -1, then *col* is assumed to be the offset from the beginning of the logical screen buffer. For Example:

```
GetScreenText (5, 2, 5)
```

Is the same as:

```
GetScreenText (85, -1, 5)
```

The above example assumes the screen has 80 columns.

Related Topics: GetScreenLine, SetScreenText

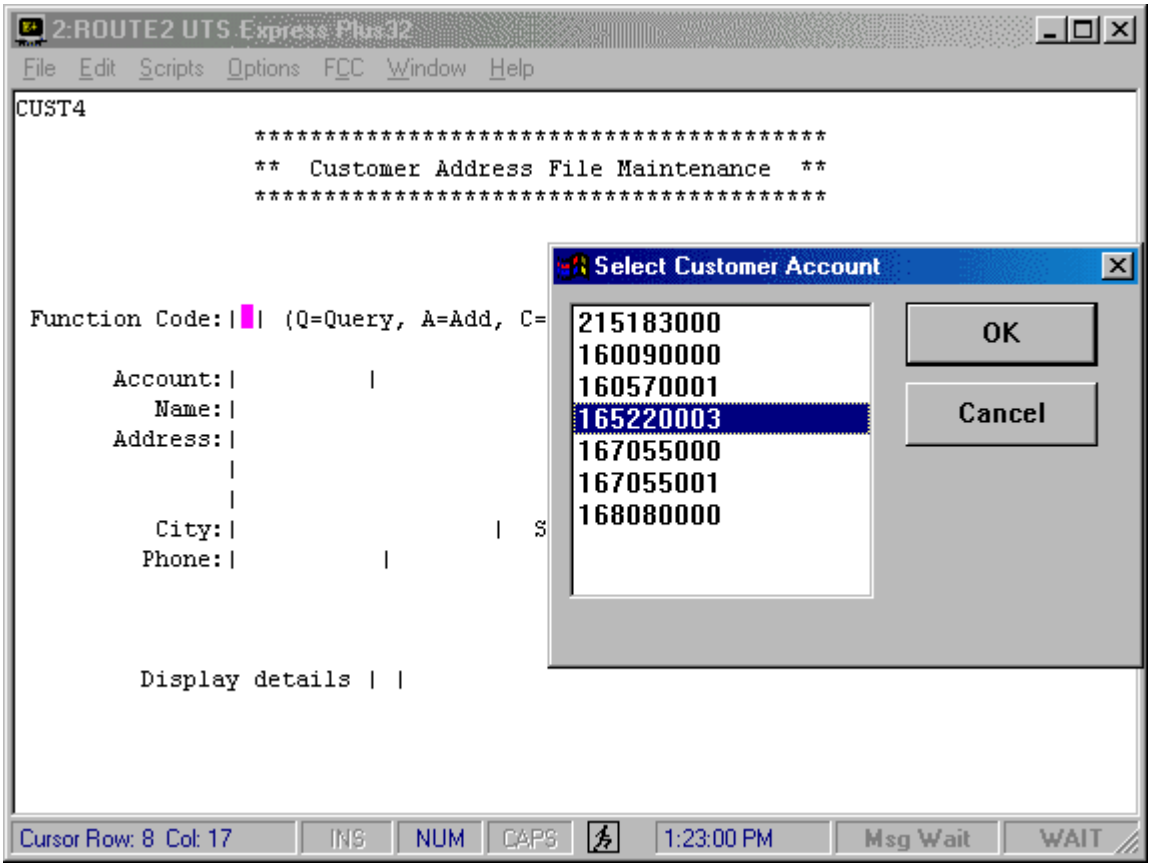
Example:

In this example, a host legacy transaction (cust3) is executed and a dialog of existing accounts is displayed. Upon selecting an account, the account number is passed to the next host transaction (cust4) where detailed information (line items) is produced on the screen. The script then accumulates line item totals and displays a message box containing the results.

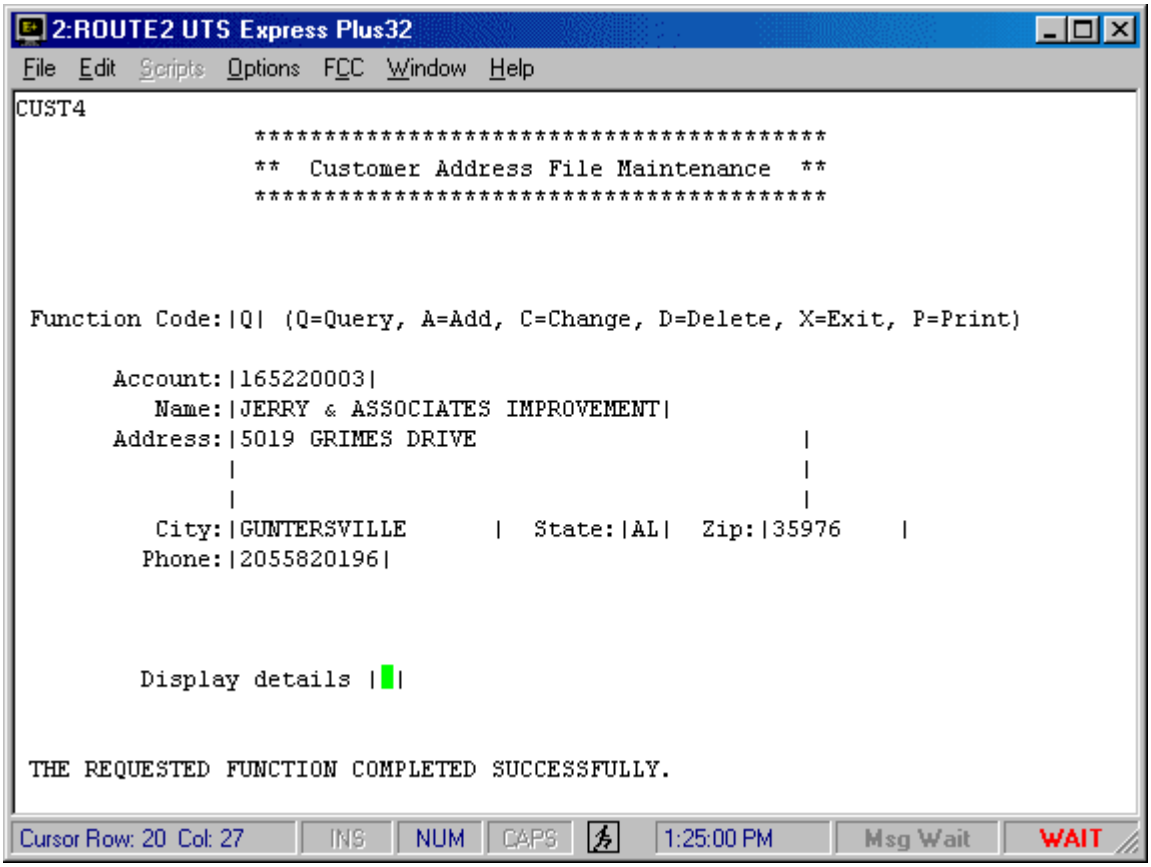
```
Sub Main()
    dim tmp as string
    dim x as integer
    dim AccLst$(6)

    Begin Dialog ACCDLG -1,41, 146, 94, "Select Customer Account"
        ListBox 4,4,76,80, AccLst(), .AccLstBox
        OKButton 88,4,48,16
        CancelButton 88,24,48,16
    End Dialog

    Dim Dlg1 as ACCDLG

    AccLst$(0) = "215183000"
    AccLst$(1) = "160090000"
    AccLst$(2) = "160570001"
    AccLst$(3) = "165220003"
    AccLst$(4) = "167055000"
    AccLst$(5) = "167055001"
    AccLst$(6) = "168080000"

    UTSKey UK_CURSOR_TO_HOME
    UTSKey UK_ERASE_DISPLAY
    EnterText "cust3"
    UTSKey UK_TRANSMIT_KEY
    If not WaitForSpecificString(3, 22, 33 ,_
    "Customer Address File Maintenance") Then Exit Sub
    if Dialog(Dlg1) = 0 then
        Exit Sub
    End If
    x = Dlg1.AccLstBox
    Acc$ = AccLst$(x)
    EnterText "Q" & Acc$
    UTSKey UK_TRANSMIT_KEY
    for x = 1 to 20
        wait 500
        Tmp = GetScreenText(2, 23, 7)
        If Tmp = "THE REQ" then    ' Found account
            Exit For
        End If
        if Tmp = "UNABLE " then    ' Did not find account
            Exit Sub
```

```
            End If
        Next x
        If x > 20 then
            Exit Sub
        End If
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TAB_FORWARD
        UTSKey UK_TRANSMIT_KEY
        If not WaitForSpecificString(2,25,20, _
            "Customer Detail List") Then Exit Sub
        Tot = 0
        for x = 1 to 13
            Tmp = GetScreenText(47, 7 + x, 10)
            Tot = Tot + Val(Tmp)
        Next x
        MsgBox "Total of Orders on this screen: " + Format$(Tot, _
            "########.00"), MB_ICONINFORMATION, "Order Total"
        UTSKey UK_TRANSMIT_KEY
    End Sub
```

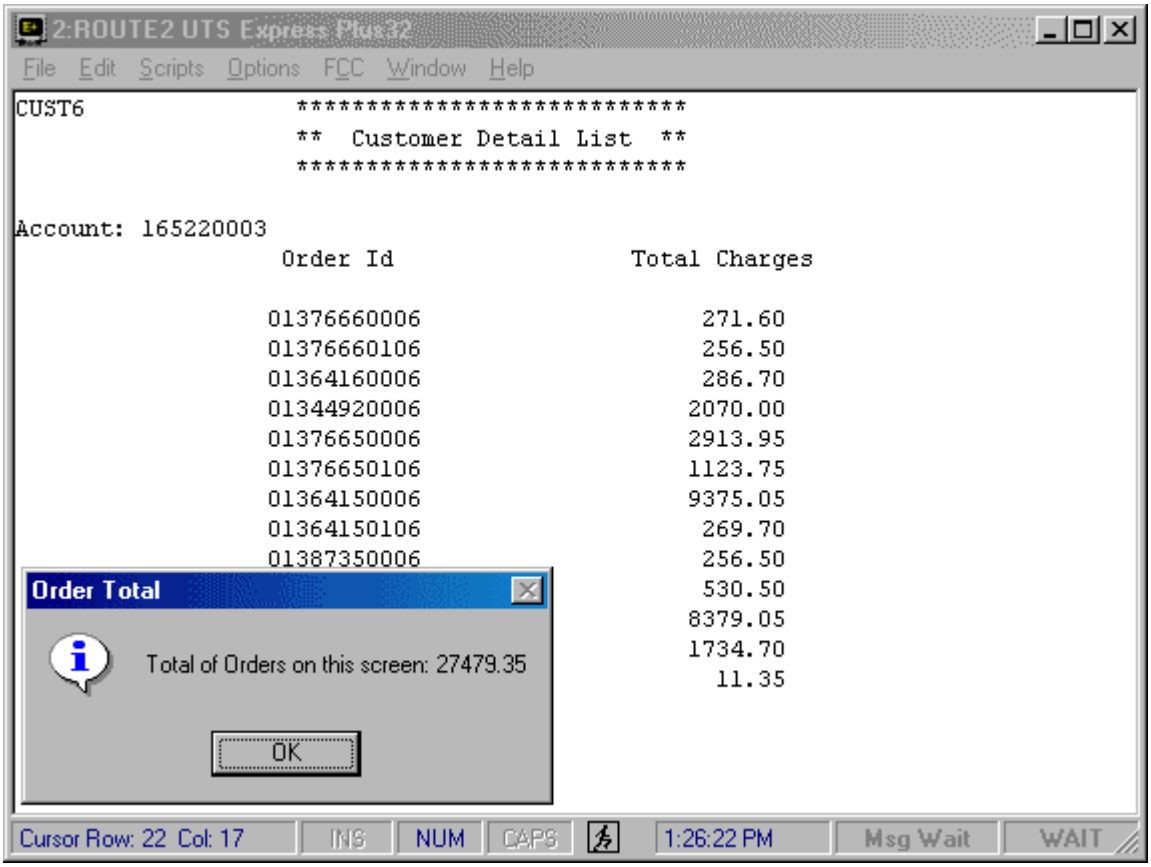
Select account:



Display general account information:



Display account detail:



For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

GetWindowState Function (eXpress Plus)

Retrieve the Window State of the current screen.

Format:

GetWindowState ()

Returns an Integer.

The function returns one of the following:

Constant	Integer	State
WSNORMAL	0	Window Normal
WSMINIMIZED	1	Window Minimized
WSMAXIMIZED	2	Window Maximized

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topic: SetWindowState

GetUserParam Function (eXpress Plus)

Retrieve user information for a calling script.

Format:

GetUserParam(*ParamNumber*)

Return a string.

Whenever a script is run, three parameters will be passed. Parameter 1 will always be the screen number. Parameter 2 will indicate whether this is a sign-on script (1 = Sign-on, 0 = other). The third parameter will be the toolbar button caption, menu item caption or an empty string if the script was started some other way.

The menu item caption will be passed as parameter 3 when a script is run from a menu. When the script is started from a toolbar button, the button caption will be passed.

For a script started by an action key sequence, parameter 3 will be set as follows:

KEY_*vvvsac*

Where:

vvvv is the virtual key code.

s is a Y or N indicating the SHIFT key was used.

a is a Y or N indicating the ALT key was used.

c is a Y or N indicating the CTRL key was used.

For example, <Ctrl> + A would be "KEY_065NNY", <Alt> + Enter would be "KEY_013NYN".

Parameter 3 will be an empty string when a script is run as an automatic sign-on script.

Examples:

```
ButtonCap = GetUserParam(3)
ScreenNumber = GetUserParam(1)
```

Global Statement

Declare a global variable and allocate storage space.

Format:

[Global] Const *constant*

A constant must be defined before it is used.

The **Global** statement must be outside the procedure section (i.e., not in a **Sub** or **Function**) of Enable. Global variables are available to all functions and subroutines in your program.

The definition of a **Const** in Enable, outside the procedure, is global. The syntax, **Global Const** and **Const** (used below, outside the **Sub**) are identical.

A type declaration character may be used (see Other Data Types). If no type declaration character is used, Enable will automatically assign one of the following data types to the constant by evaluating *expression*:

Long (if *expression* evaluates to a long or integer),
Double (if a decimal place is present) or
String (if *expression* evaluates to a string).

Related Topics: Const, Dim, Type

Example:

```
Global Const GloConst = 142
Const MyConst = 122         ' Global to all procedures in a module
Sub Main ()
    Dim Answer, Msg, N      ' Declare variables
    Const PI = 3.14159
    NL = Chr(10)            ' Define newline
    CurPath = CurDir()      ' Get current path
    ChDir "\"
    Msg = "The current directory has been changed to "
    Msg = Msg & CurDir() & NL & NL & "Press OK to change "
    Msg = Msg & "back to your previous default directory."
    Answer = MsgBox(Msg)    ' Get user response
    ChDir CurPath           ' Change back to user default
    Msg = "Directory changed back to " & CurPath & "."
    MsgBox Msg              ' Display results
    myvar =myConst + PI + GloConst
    Print MyVar
End Sub
```

GoTo Statement

Branch unconditionally and without return to a specified label in a procedure.

Format:

GoTo *linelabel*

Example:

```
Sub main ()

    Dim x,y,z

    For x = 1 to 5
        For y = 1 to 5
            For z = 1 to 5
            Print "Looping" ,z,y,x
                If y > 3 Then
                    GoTo Label1
                End If
            Next z
        Next y
    Next x
Label1:

End Sub
```

GoToPage Subroutine (UTS eXpress Plus)

Move the specified inactive page to the current screen.

Format:

GoToPage(*PageNumber*)

The *PageNumber* is any integer expression.

Note: This subroutine is only meaningful when the screen is configured for 2 or more pages (see, Pages, in UTS eXpress Plus Configuration or in UTS eXpress Net Administration). Furthermore, this subroutine does not apply to page usage in T27 emulation.

Related topics: GetPage, StorePage

Example:

The following is a general-purpose script to perform **GoToPage**, **StorePage** and **GetPage** subroutines. It can be used in conjunction with a custom tool bar or script menu. It uses the first two character of the button or menu item caption to determine the action to be performed.

```
Sub Main()
' General paging action script

' The following dialog will prompt the user for a page number.  The user can
' click a button or just press a number key to select the page.
Begin Dialog PAGEDLG 1,58, 233, 29, "Select page (Click button or press key)"
      PushButton 4,8,17,12, "&1", .PushButton_1
      PushButton 24,8,17,12, "&2 ", .PushButton_2
      PushButton 44,8,16,12, "&3 ", .PushButton_3
      PushButton 64,8,16,12, "&4", .PushButton_4
      PushButton 84,8,16,12, "&5", .PushButton_5
      PushButton 104,8,16,12, "&6 ", .PushButton_6
      PushButton 124,8,16,12, "&7", .PushButton_7
      PushButton 144,8,16,12, "&8 ", .PushButton_8
      PushButton 164,8,16,12, "&9", .PushButton_9
      CancelButton 184,8,40,12
End Dialog


Dim D as PAGEDLG

      Cap = GetUserParam(3) ' Get caption of calling button or menu item
```

```
        Btn = Dialog(D) ' Prompt for page number

        If Btn >= 1 and Btn <= 9 Then
               Select Case UCase$(Left$(Cap, 2))     ' Perform action based on button or
                                                     ' menu item caption (1st 2 chars)
                      Case "ST"
                             StorePage(Btn)
                      Case "GE"
                             GetPage(Btn)
                      Case "GO"
                             GoToPage(Btn)
               End Select
        End If
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

GroupBox Statement

Use a group box in a dialog to logically group controls.

Format:

GroupBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, "*caption*"

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
GroupBox 24,4,212,28, "Check Group"
CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
```

Hex Function

Return the hexadecimal value of a decimal parameter.

Format:

Hex(*number*)

Hex returns a string.

The *number* parameter can be any valid number. It is rounded to the nearest whole number before evaluation.

Related Topics: Oct

Example:

```
Sub Main ()

    Dim Msg As String, x%
    x% = 10
    Msg =Str( x%) &  " decimal is "
    Msg = Msg & Hex(x%) & " in hex "
    MsgBox Msg
End Sub
```

HoldMessages Subroutine (eXpress Plus)

Force messages from the host to be held until a **Receive** or **UnholdMessages** is issued. Messages will be automatically released at the end of script execution. Note: This subroutine only applies to UTS emulation.

Format

HoldMessages

Related Topics: Receive; UnholdMessages

Hour Function

Return an integer between 0 and 23 that is the portion of the *time* parameter representing the hour of the day.

Format:

Hour(*time*)

The *time* parameter is any string expression that can represent a time.
Related Topics: Date, Day, Format, Minute, Month, Now, Second, Weekday, Year
Exxample:

```
MyTime = "08:04:23 PM"
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

If...Then...Else Statement
Allow conditional statements to be executed in the code.
Formats:

If *condition* Then
[*statement(s)*]
[ElseIf *condition* Then
[*statement(s)*]]
[Else
[*statement(s)*]]
End If

or

If *condition* Then *statement* [Else *statement*]

Related Topics: Select Case
See also, Data Types, Operators and Precedences
Example:

```
Sub IfTest
    Dim msg as String
    Dim nl as String
    Dim someInt as Integer

    nl = Chr(10)
    msg = "Less"
    someInt = 4

    If 5 > someInt Then msg = "Greater" : Beep
    MsgBox(msg)

    If 3 > someInt Then
        msg = "Greater"
        Beep
    Else
        msg = "Less"
    End If
    MsgBox(msg)

    If someInt = 1 Then
        msg = "Spring"
    ElseIf someInt = 2 Then
        msg = "Summer"
    ElseIf someInt = 3 Then
        msg = "Fall"
    ElseIf someInt = 4 Then
        msg = "Winter"
    Else
        msg = "Salt"
    End If
    MsgBox(msg)

End Sub
```

Input Function

Return characters from a sequential file.

Format:

Input(*n*,[#]*filenumber*)

The **Input** function has two parameters: *n* and *filenumber*. The *n* parameter is the number of bytes to be read from a file, and *filenumber* is the number used in the open statement when the file was opened.

Related Topics: Close, EOF, Line Input, Open, Print #, Write #

Example:

```
Sub Main
     Open "TESTFILE" For Input As #1 ' Open file.
    Do While Not EOF(1)              ' Loop until end of
file.
    MyStr = Input(10, #1)            ' Get ten characters.
    MsgBox MyStr
    Loop
    Close #1                         ' Close file.
End Sub
```

InputBox Function

Display a prompt in a dialog box.

Format:

InputBox(*prompt*[,[*title*][,[*default*][,*x-pos*,*y-pos*]]])

InputBox returns a String.

The **Input** function has these parts:

Part	Description
prompt	String expression displayed as the message in the dialog box.
title	String expression displayed in the title bar of the dialog.
default	String expression displayed in the textbox as the default response if no other input is provided.
x-pos	Numeric expression that specifies, in twips, the horizontal distance of the left edge of the dialog from the left edge of the screen. If omitted, *y-pos* must also be omitted. If *x-pos* and *y-pos* are omitted, the dialog box is horizontally centered and vertically positioned approximately one-third the way down the screen.
y-pos	Numeric expression that specifies, in twips, the vertical distance of the upper edge of the dialog from the top edge of the screen.

Note: A twip is 1/20 of a printer's point (1,440 twips equal an inch and 567 twips equal a centimeter).

Example:

```
MyTime = InputBox$ ("Enter Time <hh:mm:ss>." + Chr(13) + _
            Chr(10) + "(Not military time).", "Time Entry")
MyDate = InputBox$ ("Enter Date.", "Date Entry")
MsgBox Format(MyDate, "Long Date") + " at " + _
       Format(MyTime, "Long Time")
```

InStr Function

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

Format:

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

Return the character position of the first occurrence of *string2* within *string1*.

The *numbegin* parameter is not optional and sets the starting point of the search within *string1*. The *numbegin* parameter must be a valid positive integer, no greater than 65,535.

The *string1* parameter is the string being searched and *string2* is the string for which we are looking.

The function returns the following values:

If:	InStr returns:
string2 is found within *string1*	Position at which match is found
string2 is not found	0
string2 is zero-length	*numbegin*

If:	InStr returns:
string2 is Null	Null
string1 is zero-length	0
string1 is Null	Null
numbegin > *string2*	0

Related Topics: Len

Example:

```
Sub Main ()
    B$ = "Good Bye"
    A% = InStr(2, B$, "Bye")
    C% = Instr(3, B$, "Bye")
End Sub
```

Int Function

Return the integer portion of a number.

Format:

Int(*number*)

Related Topics: Fix

IsArray Function

Return a Boolean value True or False indicating whether the *vaiablename* parameter is an array.

Format:

IsArray(*variablename*)

Related Topics: IsEmpty, IsNumeric, VarType, IsObject

Example:

```
Sub Main

      Dim MArray(1 To 5) As Integer, MCheck

      MCheck = IsArray(MArray)
      Print MCheck

End Sub
```

IsDate Function

Return a value that indicates if a variant parameter can be converted to a date.

Format:

IsDate(*variant*)

The **IsDate** function returns True if the variant can be converted to a date; otherwise, it returns False.

Related Topics: IsEmpty, IsNull, IsNumeric, VarType

Example:

```
If IsDate(inputvar)
    MsgBox Format(inputvar, "Long Date")
Else
    MsgBox "Input not a valid date."
EndIf
```

IsEmpty Function

Return a value that indicates if a variant parameter has been initialized.

Format:

IsEmpty(*variant*)

The **IsEmpty** function returns True if the variant is empty; otherwise, it returns False.

Related Topics: IsDate, IsNull, IsNumeric, VarType

Example:

```
Sub BTN_5()
    Dim new, answer
```

```
    If IsEmpty(new) Then
        answer = MsgBox ("Start at the beginning?", 36)
        If answer = 6 Then
            new = 1
            MsgBox "Starting at the beginning now."
        End If
    End If
End SUB
```

IsNull Function

Return a value that indicates if a variant contains the NULL value.

Format:

IsNull(*variant*)

IsNull returns a True if *variant* contains NULL; otherwise, it returns False.

The NULL value is special because it indicates that the *variant* parameter contains no data. This is different from a null-string, which is a zero-length string and an empty string that has not yet been initialized.

Related Topics: IsDate, IsEmpty, IsNumeric, VarType

IsNumeric Function

Return a value that indicates if a variant contains numerics.

Format:

IsNumeric(*variant*)

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

The *variant* parameter can be any variant, numeric value, date or string (if the string can be interpreted as a numeric).

Related topics: IsDate, IsEmpty, IsNull, VarType

Example:

```
Sub Form_Click ()
    Dim TestVar                              ' Declare variable
    TestVar = InputBox("Please enter a number or a letter:")
    If IsNumeric(TestVar) Then               ' Evaluate variable
        MsgBox "Entered data is numeric."    ' Message if number
    Else
        MsgBox "Entered data is not numeric." ' Message if not
    End If
End Sub
```

IsObject Function

Return a Boolean value True or False indicating whether the *objectname* parameter is an object.

Format:

IsObject(*objectname*)

Related Topics: IsEmpty, IsNumeric, VarType, IsArray

Example:

```
Sub Main

    Dim MyInt As Integer, MyCheck
    Dim MyObject As Object
    Dim YourObject As Object
     Set MyObject = CreateObject("Word.Basic")

    Set YourObject = MyObject
    MyCheck = IsObject(YourObject)

    Print MyCheck

End Sub
```

KeyboardState Function (eXpress Plus)

Returns the state of the keyboard: 1 – locked or 0 – unlocked.

Format:

KeyboardState()

The state may be checked using an Integer or Constant:

State	Integer	Constant
Keyboard Unlocked	0	KEYBOARDUNLOCKED
Keyboard Locked	1	KEYBOARDLOCKED

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Example:

```
If KeyboardState() = KEYBOARDLOCKED
    UTSKey UK_KEYBOARD_UNLOCK
    Msg1 = GetLastMsg
End If
```

Kill Statement

Delete files from a disk. Kill will only delete files. To remove a directory, use the **RmDir** Statement.

Format:

Kill *filename*

Related Topics: RmDir

Example:

```
Const NumberOfFiles = 3

Sub Main ()
    Dim Msg                 ' Declare variable.
    Call MakeFiles()        ' Create data files.
    Msg = "Several test files have been created on your "
    Msg = Msg & "disk. You may see them by switching tasks."
    Msg = Msg & " Choose OK to remove the test files."
    MsgBox Msg
    For I = 1 To NumberOfFiles
        Kill "TEST" & I     ' Remove data files from disk.
    Next I
End Sub

Sub MakeFiles ()
    Dim I, FNum, FName      ' Declare variables.
    For I = 1 To NumberOfFiles
        FNum = FreeFile     ' Determine next file number.
        FName = "TEST" & I
        Open FName For Output As FNum    ' Open file.
        Print #FNum, "This is test #" & I
                            ' Write string to file.
        Print #FNum, "Here is another ";  "line";  I
    Next I
    Close                   ' Close all files.
    Kill FName
End Sub
```

LBound Function

Return the smallest available subscript for the dimension of the indicated array.

Format:

LBound(*array* [,*dimension*])

Related Topics: Dim, Global, Option Base, Static, UBound

Example:

```
' This example demonstrates some of the features of
' arrays.  The lower bound for an array is 0 unless it is
' specified or an Option Base is set as in this example.
```

```
Option Base 1

Sub Main
    Dim a(10) As Double
    MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
    Dim i As Integer
    For i = 0 to 3
        a(i) = 2 + i * 3.1
    Next i
    Print a(0),a(1),a(2), a(3)
End Sub
```

LCase Function

Return a string in which all letters of the string parameter have been converted to lower case.

Format:

LCase(*string*)

Related Topics: UCase

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable.  It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls.

Sub Main
    MyString = "  <-Trim->  "            ' Initialize string
    TrimString = LTrim(MyString)         ' TrimString = "<-Trim->  "
    MsgBox "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString))  ' TrimString = "  <-trim->"
    MsgBox "|" & TrimString & "|"
    TrimString = LTrim(RTrim(MyString))  ' TrimString = "<-Trim->"
    MsgBox "|" & TrimString & "|"        ' Using the Trim function
                                         ' alone achieves the same
                      ' result.
    TrimString = UCase(Trim(MyString))   ' TrimString = "<-TRIM->"
    MsgBox "|" & TrimString & "|"
End Sub
```

Left Function

Return the left most *number* of characters of a string parameter.

Format:

Left(*string*, *number*)

The *string* parameter is the string expression from which the leftmost characters are returned.

The *number* parameter is the numeric expression indicating the number of characters that will be returned.

Related Topics: Len, Mid, Right

Example:

```
Sub Main ()
    Dim LWord, Msg, RWord, SpcPos, UsrInp ' Declare variables.
    Msg = "Enter two words separated by a space."
    UsrInp = InputBox(Msg)                ' Get user input.
    print UsrInp
    SpcPos = InStr(1, UsrInp, " ")        ' Find space.
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1)  ' Get left word.
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos)
                                          ' Get right word.
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    Else
```

```
        Msg = "You didn't enter two words."
    End If
    MsgBox Msg                          ' Display message.
    MidTest = Mid("Mid Word Test", 4, 5)
    Print MidTest
End Sub
```

Len Function

Return the number of characters in a string.

Format:

Len(*string*)

Related Topics: InStr

Example:

```
Sub Main ()
    A$ = "Enable"
    StrLen% = Len(A$)    ' the value of StrLen is 6
    MsgBox StrLen%
End Sub
```

Let Statement

Assign a value to a variable.

Format:

[Let] *variablename* = *expression*

The **Let** keyword is optional and only rarely used. The **Let** keyword is required in older versions of basic.

Example:

```
Sub Form_Click ()
    Dim Msg, Pi            ' Declare variables.
    Let Pi = 4 * Atn(1)    ' Calculate Pi.
    Msg = "Pi is equal to " & Str(Pi)
    MsgBox Msg             ' Display results.
End Sub
```

Line Input # Statement

Read a line from a sequential file into a String or Variant variable.

Format:

Line Input # *filenumber*, *name*

The parameter *filenumber* is used in the open statement to open the file. The *name* parameter is the name of a variable used to hold the line of text from the file.

Related Topics: Close, Input, EOF, Open, Print #, Write #

Example:

This script uploads a PC file to a 2200 host. The host file must be accessible by the @ELT processor (i.e., not a binary file). The script uploads the file one line at a time and is, therefore, slow and not suitable for large file transfers. Consider an FTP transfer as shown in an example of the eXpress Dialog Form Designer.

```
' *** Quick and dirty text upload script.
' *** UTS eXpress Plus or UTS eXpress Net example.
Sub Main()
    Dim InsRow as Integer
    Dim HFileEle as String
    Dim fn as integer
    Dim Inp as String
    Dim Outp as String

    Begin Dialog FDLG 10,10, 165, 110, "Upload parameters"
          Text 4,4,52,8, "Host file name"
          TextBox 4,12,112,12, .Ed_HostFile
          Text 4,28,112,8, "Host symbolic element name"
          TextBox 4,36,112,12, .Ed_Element
```

```
            Text 4,64,56,8, "PC file name"
            TextBox 4,72,152,12, .Ed_PCFile
            OKButton 84,92,32,12
            CancelButton 124,92,32,12
        End Dialog

        Dim DLG as FDLG

        If Dialog(DLG) = 0 then
            Exit Sub
        End If

        fn = FreeFile
        Open Trim$(DLG.Ed_PCFile) for input as fn
        HFileEle = DLG.Ed_HostFile
        If Right$(HFileEle,1) <> "." then
            HFileEle = HFileEle + "."
        End If
        If Trim$(DLG.Ed_Element) <> "" then
            HFileEle = HFileEle + Trim$(DLG.Ed_Element)
        End If
        EnterText "@elt,iq " & HFileEle
        UTSKey UK_TRANSMIT_KEY
        Wait 2000
        InsRow = GetCursorRow()
        while not Eof(fn)
            Outp = ""
            Line Input #fn, Inp
            For x = 1 to len(Inp)
                If Mid$(Inp, x, 1) = Chr$(9) Then
                    Outp = Outp + "    "
                ElseIf Mid$(Inp, x, 1) <> Chr$(13) Then
                    Outp = Outp + Mid$(Inp, x, 1)
                End If
            Next x
            If Not WaitForSpecificString(InsRow, 1, 1, Chr$(30)) then
                Exit Sub
            End If
            EnterText Outp
            UTSKey UK_TRANSMIT_KEY
        Wend

        If Not WaitForSpecificString(InsRow, 1, 1, Chr$(30)) then
            Exit Sub
        End If
        EnterText "@eof"
        UTSKey UK_TRANSMIT_KEY
    End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

ListBox Statement

Use a list box in a dialog to allow the user to make a selection from a list box. If there are more items in the list than will fit in the list box, a scroll bar will appear allowing access to all items in the list.

Format:

ListBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, *listsource*, *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, OKButton, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
    Sub BTN_3()
    Dim LISTSRC$(2)
    LISTSRC(0) = "Item 1"
```

```
LISTSRC(1) = "Item 2"
LISTSRC(2) = "Item 3"
Begin Dialog DIALOG_2 0,0, 204, 96, "Test ListBox"
  ListBox 36,8,128,16, LISTSRC$(), .LISTBOX_1
  TextBox 36,36,128,20, .TEXTBOX_2
  OKButton 8,64,76,20
  CancelButton 108,64,72,20
End Dialog
Dim Dlg2 As DIALOG_2
Dlg2.LISTBOX_1 = 1
button = Dialog(Dlg2)
If button = 0 Then Return
x = Dlg2.LISTBOX_1
Dlg2.TEXTBOX_2 = LISTSRC(x)
MsgBox "Text box is set to: " + Dlg2.TEXTBOX_2
Dialog Dlg2
End SUB
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

LoadScreen Subroutine (eXpress Plus)

Load an entire screen/form from a file.

Format:

LoadScreen *filename*

The *filename* parameter is any string expression containing the file name of a previously saved screen/form file. Specific form loads can be assigned to function keys.

Related topic:FileOpenDialog, FileSaveDialog, SaveScreen

Example:

```
Sub Main()
      Dim fname as String

      fname = FileOpenDialog("C:\AA_UX32_3.00\T27Scripts", "", "FRM",
"Form files(*.frm)|*.frm|All files(*.*)|*.*", "Select Load Form File")
      If fname <> "" Then
            loadScreen(fname)
      Else
            MsgBox "Load Cancelled"
      End If
End Sub
```

LOF Function

Return a long number for the number of bytes in the open file.

Format:

LOF(*filenumber*)

The parameter *filenumber* is required and must be an integer.

Related Topics: FileLen

Example:

```
Sub Main

      Dim FileLength
      Open "TESTFILE" For Input As #1
      FileLength = LOF(1)
Print FileLength
      Close #1

End Sub
```

Log Function

Return the natural log of a number.

Format:

Log(*number*)

The *number* parameter must be greater than zero, and must be a valid number.

Related Topics: Cos, Exp, Sin, Tan

Example:

```
Sub Form_Click ( )
    Dim I, Msg, NL
    NL = Chr(13) & Chr(10)
    Msg = Exp(1) & NL
    For I = 1 to 3
        Msg = Msg & Log(Exp(1) ^ I ) & NL
    Next I
    MsgBox Msg
End Sub
```

MarkBlock Subroutine (eXpress Plus)

Mark a block of text on the screen to be subsequently copied to the Windows clipboard by the **CopyToClipboard** subroutine.

Format:

MarkBlock *StartCol, StartRow, EndCol, EndRow*

StartCol, StartRow, EndCol and *EndRow* are numeric expressions indicating the boundaries of the block of screen text to be copied to the Windows clipboard.

Related Topics: CopyToClipboard , PasteFromClipboard

MessageWaitingState Function (eXpress Plus)

Return the state of the message waiting facility: 1 – message waiting or 0 – message not waiting.

Format:

MessageWaitingState

The state may be checked utilizing an Integer or Constant:

State	Integer	Constant
Message Not Waiting	0	NOMESSAGEWAITING
Message Waiting	1	MESSAGEWAITING

Note: Since variables do not have to be declared prior to first reference, use an Option Explicit statement (external to the procedure) when using constants to assure that they are spelled correctly.

Example:

```
If MessageWaitingState = MESSAGEWAITING
    UTSKey UK_MSG_WAIT
    Msg1 = GetLastMsg
    UTSKey UK_SOE
    EnterText "OK"
    UTSKey UK_TRANSMIT_KEY
End If
```

Mid Function

Return a substring within a string.

Format:

Mid(*string*,*begin*,*length*)

Mid returns a String.

The **Mid** function has these parts:

Part	Description
string	String expression from which another string is created.
begin	The *begin* argument is a long expression that indicates the character position in *stringexpression* at which the part to be taken begins.
length	The *length* is a long expression that indicates the number of characters to return.

Related Topics: Left, Len, Right

Example:

```
Sub BTN_6()
    Dim MidWord, Msg, TstStr, SpcPos1, SpcPos2, WordLen
    TstStr = "Mid Function Demo"
    SpcPos1 = InStr(1, TstStr, " ")             ' Find 1st space
    SpcPos2 = InStr(SpcPos1 + 1, TstStr, " ")   ' Find 2nd space
    WordLen = (SpcPos2 - SpcPos1) - 1           ' Get 2nd word length
    MidWord = Mid(TstStr, SpcPos1 + 1, WordLen) ' Get 2nd word
    Msg = "the word in the middle of Title is '" & MidWord & "'."
    MsgBox Msg, 0, TstStr
End SUB
```

Minute Function

Return an integer between 0 and 59 that is the portion of the *time* parameter representing the minute of the hour.

Format:

Minute(*time*)

The *time* parameter is any string expression that can represent a time.

Related Topics: Date, Day, Format, Hour, Month, Now, Second, Weekday, Year

Example:

```
MyTime = "08:04:23 PM"
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

MkDir Statement

Create a new directory.

Format:

MkDir *path*

The *path* parameter is a string expression that must contain fewer than 128 characters.

Related Topics: ChDir, ChDrive, CurDir, Dir, RmDir

Example:

```
Sub Main
   Dim DST As String
   DST = "t1"
   mkdir DST
   mkdir "t2"
End Sub
```

Month Function

Return an integer between 1 and 12 that is the portion of the *date* parameter representing the month of the year.

Format:

Month(*date*)

The *date* parameter is any string expression that can represent a date.

The returned integer represents the month of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Now, Second, Weekday, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Month(MyDate)
    print x

End Sub
```

MsgBox Function, MsgBox Statement

Display a message in a dialog box and waits for the user to choose a button.

MsgBox Function returns a value indicating which button the user has chosen; the **MsgBox** statement does not.

Function Format:

MsgBox(*msg*[,[*type*][,*title*]])

Statement Format:

MsgBox *msg*[,[*type*][,*title*]]

The *msg* parameter is the string displayed in the dialog box as the message. The second and third parameters are optional and respectively designate the type of buttons and the title displayed in the dialog box.

The *type* is the sum of the values specifying the type of buttons to display, the icon style to use, the identity of the default button and the modality. The following illustrates the values and meaning of each group:

Constant	Value	Meaning
MB_OK	0	Display OK button only.
MB_OKCANCEL	1	Display OK and Cancel buttons.
MB_ABORTRETRYIGNORE	2	Display Abort, Retry and Ignore buttons.
MB_YESNOCANCEL	3	Display Yes, No and Cancel buttons.
MB_YESNO	4	Display Yes and No buttons.
MB_RETRYCANCEL	5	Display Retry and Cancel buttons.
————————	——	————————
MB_ICONSTOP	16	 Display:
MB_ICONQUESTION	32	 Display:
MB_ICONEXCLAMATION	48	 Display:
MB_ICONINFORMATION	64	 Display:
————————	——	————————
MB_DEFBUTTON1	0	First button is default.
MB_DEFBUTTON2	256	Second button is default.
MB_DEFBUTTON3	512	Third button is default.
————————	——	————————
MB_APPLMODAL	0	Application modal. The user must respond to the message box before continuing work in the current application.
MB_SYSTEMMODAL	4096	System modal. All applications are suspended until the user responds to the message box.

The first group of values (0-5) describes the number and type of buttons displayed in the dialog box. The second group (16, 32, 48, and 64) describes the icon style. The third group (0, 256 and 512) determines which button is the default. The fourth group (0 and 4096) determines the modality of the message box. When adding numbers to create a final value for the argument type, use only one number from each group. If omitted, the default value for type is zero.

The *title* parameter is a string expression displayed in the title bar of the dialog box. If you omit the argument title, **MsgBox** has no default title.

The value returned by the **MsgBox** function indicates which button has been selected, as shown below:

Constant	Value	Meaning
IDOK	1	OK button selected.
IDCANCEL	2	Cancel button selected.
IDABORT	3	Abort button selected.
IDRETRY	4	Retry button selected.
IDIGNORE	5	Ignore button selected.
IDYES	6	Yes button selected.
IDNO	7	No button selected.

If the dialog box displays a **Cancel** button, pressing the **Esc** key has the same effect as choosing **Cancel**.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Example:

This example uses **MsgBox** to display a "close without saving" message in a dialog box with a **Yes** button, a **No** button and a **Cancel** button. The **Yes** button is the default response. The **MsgBox** function returns a value based on the button chosen by the user. The **MsgBox** statement uses that value to display a message that indicates which button was chosen.

```
Sub Main
    Dim DgDef, Msg, Response, Title
    Title = "MsgBox Sample Question"
    Msg = "This is a sample of Close Without Saving?."
    Msg = Msg & " Do you want to save changes?"
    DgDef = MB_YESNOCANCEL + MB_ICONQUESTION + MB_DEFBUTTON1
    Response = MsgBox(Msg, DgDef, Title)
    If Response = IDYES Then
        Msg = "You chose Yes or pressed Enter."
    ElseIf Response = IDCANCEL
        Msg = "You chose Cancel or pressed Esc."
    Else
        Msg = "You chose No."
    End If
    MsgBox Msg
End Sub
```

Name Statement

Change the name of a directory or a file.

Format:

Name *oldname* As *newname*

The *oldname* and *newname* parameters are strings expressions that can optionally contain a path.

Related Topics: ChDir, Kill

Example:

```
Sub Main

    Name "testfile" As "newtest"

End Sub
```

Now Function

Return a date that represents the current date and time according to the settings in the computer's system date and time.

Format:

Now

The **Now** function returns a Variant data type containing a date and time that are stored internally as a double. The number is a date and time from January 1, 100 through December 31, 9999, where January 1, 1900 is 2. Numbers to the left of the decimal point represent the date and numbers to the right represent the time.

Related Topics: Date, Day, Format, Hour, Minute, Month, Second, Weekday, Year

Example:

```
Sub Main ()
    Dim Today
    Today = Now
End Sub
```

Oct Function

Return the octal value of the decimal parameter.

Format:

Oct(*number*)

Oct returns a string.

Related Topics: Hex, Hex$

Example:

```
Sub Main ()
    Dim Msg, Num    ' Declare variables.
    Num = InputBox("Enter a number.")    ' Get user input.
    Msg = Num & " decimal is &O"
    Msg = Msg & Oct(Num) & " in octal notation."
    MsgBox Msg    ' Display results.
End Sub
```

OKButton Statement

Use to close a dialog when accepting changes.

Format:

OKButton *starting-x-pos*, *starting-y-pos*, *width*, *height*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OptionButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub Main ()
    Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
        TEXT 10, 10, 28, 12, "Name:"
        TEXTBOX 42, 10, 108, 12, .nameStr
        TEXTBOX 42, 24, 108, 12, .descStr
        CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
        OKBUTTON 42, 54, 40, 12
    End Dialog
    Dim Dlg1 As DialogName1
    Dialog Dlg1

    MsgBox Dlg1.nameStr
    MsgBox Dlg1.descStr
    MsgBox Dlg1.checkInt
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

On Error Statement

Enable error-handling routine and specifies the line label of the error-handling routine.

Format:

On Error { GoTo *line* | Resume Next | GoTo 0 }

The *line* parameter refers to a label. That label must be present in the code or an error is generated.

Errors can be raised with the syntax:

Err.Raise x

The list below shows the corresponding descriptions for the defined values of x.

5	Invalid procedure call
6	Overflow
7	Out of memory
9	Subscript out of range
10	Array is fixed or temporarily locked
11	Division by zero
13	Type mismatch
14	Out of string space
16	Expression too complex
17	Can't perform requested operation
18	User interrupt occurred
20	Resume without error
28	Out of stack space
35	Sub, Function or Property not defined
47	Too many DLL application clients

48	Error in loading DLL
49	Bad DLL calling convention
51	Internal error
52	Bad file name or number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
59	Bad record length
60	Disk full
62	Input past end of file
63	Bad record number
67	Too many files
68	Device unavailable
70	Permission denied
71	Disk not ready
74	Can't rename with different drive
75	Path/File access error
76	Path not found
91	Object variable or With block variable not set
92	For loop not initialized
93	Invalid pattern string
94	Invalid use of Null

OLE Automation Messages:

429	OLE Automation server cannot create object
430	Class doesn't support OLE Automation
432	File name or class name not found during OLE Automation operation
438	Object doesn't support this property or method
440	OLE Automation error
443	OLE Automation object does not have a default value
445	Object doesn't support this action
446	Object doesn't support named arguments
447	Object doesn't support current local setting
448	Named argument not found
449	Argument not optional
450	Wrong number of arguments
451	Object not a collection

Miscellaneous Messages:

444	Method not applicable in this context
452	Invalid ordinal
453	Specified DLL function not found
457	Duplicate Key
460	Invalid Clipboard format
461	Specified format doesn't match format of data
480	Can't create AutoRedraw image
481	Invalid picture
482	Printer error
483	Printer driver does not support specified property
484	Problem getting printer information from from the system – make sure the printer is setup correctly
485	invalid picture type
520	Can't empty Clipboard
521	Can't open Clipboard

Example:

```
On Error GoTo dude
  Dim x as object
  x.draw        ' Object not set
  jpe           ' Undefined function call
```

```
  print 1/0     ' Division by zero
  Err.Raise 6   ' Generate an "Overflow" error
  MsgBox "Back"
  MsgBox "Jack"
  Exit Sub
dude:
  MsgBox "HELLO"
  Print Err.Number, Err.Description
  Resume Next
  MsgBox "Should not get here!"
  MsgBox "What?"
End Sub
```

Open Statement

Open a file for input and output operations.

Format:

Open *file* [For *mode*] [Access *access*] As [#]*filenumber*

You must open a file before any I/O operation can be performed on it. The **Open** statement has these parts:

Part	Description
file	The *file* parameter is a file name and path specification.
mode	Specify the reserved word that specifies the file *mode*: **Append**, **Input**, **Output**.
access	Specify the reserved word that specifies which operations are permitted on the open file: **Read**, **Write**.
filenumber	The *filenumber* is any integer expression with a value between 1 and 255, inclusive. When an **Open** statement is executed, *filenumber* is associated with the file as long as it is open. Other I/O statements can use the number to refer to the file.

If the file does not exist, it is created when the file is opened for **Append** or **Output** modes.

The *mode* argument is a reserved word that specifies one of the following file modes.

Mode	Description
Input	Sequential input mode.
Output	Sequential output mode.
Append	Sequential output mode. Append sets the file pointer to the end of the file. A subsequent **Print #** or **Write #** statement extends (appends to) the file.

The *access* argument is a reserved word that specifies the operations that can be performed on the opened file. If the file is already opened by another process and the specified type of access is not allowed, the Open operation fails and a permission-denied error occurs. The *access* argument can be one of the following reserved words:

Access	Description
Read	Open the file for reading only.
Write	Open the file for writing only.
Read Write	Open the file for both reading and writing. This access is valid only for files opened for Append mode.

Related Topics: Close, EOF, Input, Line Input, Open, Print #, Write

The following example writes data to a test file and reads it back:

```
Sub Main ()
    Dim FileData, Msg, NL            ' Declare variables.
    NL = Chr(10)                     ' Define newline.
    Open "TESTFILE" For Output As #2
                                     ' Open to write file.
    Print #2, "This is a test of the Print # statement."
    Print #2                         ' Print blank line to file.
    Print #2, "Zone 1", "Zone 2"
                                     ' Print in two print zones.
    Print #2, "With no space between"  ;  "."
                                      'Print two strings together.
```

```
    Close
    Open "TESTFILE" for Input As #2  ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData      ' Read a line of data.
        Msg = Msg & FileData & NL    ' Construct message.
        MsgBox Msg
    Loop
    Close                            ' Close all open files.
    MsgBox "Testing Print Statement" ' Display message.
    Kill "TESTFILE"                  ' Remove file from disk.
End Sub
```

Option Base Statement

Declare the default lower bound for array subscripts.

Format:

Option Base *number*

The **Option Base** statement is never required. If used, the **Option Base** statement can appear only once in a module, can occur only in the Declarations section and must be used before you declare the dimensions of any arrays.

The value of *number* must be either 0 or 1. The default base is 0.

The **To** clause in the Dim, Global, and Static statements provides a more flexible way to control the range of an array's subscripts. If you do not explicitly set the lower bound with a **To** clause, you can use **Option Base** to change the default lower bound to 1.

Related Topics: Dim, Global, Lbound, Static, UBound

The following example uses the **Option Base** statement to override the default base array subscript value of 0:

```
Option Base 1                    ' Module level statement.
Sub Main
  Dim A(20), Msg, NL             ' Declare variables.
  NL = Chr(10)                   ' Define newline.
  Msg = "The lower bound of array A is " & LBound(A) & "."
  Msg = Msg & NL & "The upper bound is " & UBound(A) & "."
  MsgBox Msg                     ' Display message.
End Sub
```

Option Explicit Statement

Force explicit declaration of all variables.

Format:

Option Explicit

The **Option Explicit** statement is used outside of the script in the Declarations section.

It is highly recommended that an Option Explicit statement be used in all Enable Actions.

Related Topics: Const, Global

Example :

```
Option Explicit
Sub Main
    Print y        ' because y is not explicitly
                   ' dimmed, an error will occur.

End Sub
```

OptionButton Statement

Use an option button (radio button) in a dialog for selecting one, and only one, option from a group of options.

Format:

OptionButton *starting-x-pos*, *starting-y-pos*, *width*, *height*, "*caption*"

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionGroup, PushButton, Text, TextBox

Example:

```
Sub BTN_1()
Begin Dialog Dialog_1 0,0, 252, 136, "Dialog Title"
  OptionGroup .GRP_1
    OptionButton 32,48,80,12, "1st Radio - Group 1"
    OptionButton 32,64,80,12, "2nd Radio - Group 1"
  OptionGroup .GRP_2
    OptionButton 144,48,84,12, "1st Radio - Group 2"
    OptionButton 144,64,84,12, "2nd Radio - Group 2"
  OKButton 24,96,68,20
  CancelButton 156,96,52,20
  GroupBox 24,36,92,52, "Option Group 1"
  GroupBox 136,36,100,52, "Option Group 2"
  GroupBox 24,4,212,28, "Check Group"
  CheckBox 36,16,76,12, "Check_Box_1", .CHECKBOX_1
  CheckBox 124,16,76,12, "Check_Box_1", .CHECKBOX_2
End Dialog
Dim Dlg1 As Dialog_1
Dlg1.Grp_1 = 0           ' Set 1st button - group 1
Dlg1.Grp_2 = 1           ' Set 2nd button - group 2
button = Dialog ( Dlg1 )
If button = 0 Then Return
MsgBox "Grp1: " + Dlg1.Grp_1 + ", Grp2: " + Dlg1.Grp_2
Dialog Dlg1

End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

OptionGroup Statement

Use an option group in a dialog for grouping mutually exclusive option buttons.

Format:

OptionGroup *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, PushButton, Text, TextBox

Example:

(See OptionButton.)

PasteFromClipboard Subroutine (eXpress Plus)

Paste the contents of the Windows clipboard to the current cursor position of the screen. This subroutine is normally preceded by the **SetCursor** subroutine.

Format:

PasteFromClipboard

Related Topics: CopyToClipboard, MarkBlock, SetCursor

Print # Statement

Write data to a sequential file.

Format:

Print # *filenumber*, [[{*Spc*(*n*)|*Tab*(*n*)}][*expressionlist*][{ **;** | **,** }]]

The **Print** statment consists of the following parts:

Part	Description
filenumber	The filenumber is the number used in an **Open** statement to open a sequential file. This parameter can be any numeric expression that evaluates to the number of an open file. Note that the number sign (#) preceding *filenumber* is mandatory.
Spc(*n*)	*Spc* is the name of the script function optionally used to insert *n* spaces into the printed output. Multiple uses are permitted.

Part	Description
Tab(*n*)	*Tab* is the name of the Basic function optionally used to tab to the *n*th column before printing expressionlist. Multiple uses are permitted.
expressionlist	The expressionlist is the numeric and/or string expressions to be written to the file.
{;|,}	Choose the character that determines the position of the next character printed. A semicolon means the next character is printed immediately after the last character. A comma means the next character is printed at the start of the next print zone. Print zones begin every 14 columns. If neither character is specified, the next character is printed on the next line.

If you omit *expressionlist*, the Print # statement prints a blank line in the file, but you must include the comma. Because Print # writes an image of the data to the file, you must delimit the data so it is printed correctly. If you use commas as delimiters, Print # also writes the blanks between print fields to the file.

The Print # statement usually writes Variant data to a file the same way it writes any other data type; however, there are some exceptions:

- If the data being written is a Variant of VarType 0 (Empty), Print # writes nothing to the file for that data item.
- If the data being written is a Variant of VarType 1 (Null), Print # writes the literal #NULL# to the file.
- If the data being written is a Variant of VarType 7 (Date), Print # writes the date to the file using the Short Date format defined in the WIN.INI file. When either the date or the time component is missing or zero, Print # writes only the part provided to the file.

Related Topics: Close, EOF, Input, Line Input, Open, Write #

The following example writes data to a test file:

```
Sub Main
    Dim I, FNum, FName                ' Declare variables.
    For I = 1 To 3
        FNum = FreeFile               ' Determine next file number.
        FName = "TEST" & FNum
        Open FName For Output As FNum  ' Open file.
        Print #I, "This is test #" & I ' Write string to file.
        Print #I, "Here is another ";  "line";  I
    Next I
    Close                             ' Close all files.
End Sub
```

The following example writes data to a test file and reads it back.

```
Sub Main ()
    Dim FileData, Msg, NL             ' Declare variables.
    NL = Chr(10)                      ' Define newline.
    Open "TESTFILE" For Output As #1   ' Open to write file.
    Print #2, "This is a test of the Print # statement."
    Print #2
                                      ' Print blank line to file.
    Print #2, "Zone 1", "Zone 2"       ' Print in two print zones.
    Print #2, "With no space between"  ;  "."
                                      ' Print two strings together.
    Close
    Open "TESTFILE" for Input As #2    ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData        ' Read a line of data.
        Msg = Msg & FileData & NL      ' Construct message.
        MsgBox Msg
    Loop
    Close                             ' Close all open files.
    MsgBox "Testing Print Statement"   ' Display message.
    Kill "TESTFILE"                   ' Remove file from disk.
End Sub
```

Print Statement

Print a string to the default printer (specified by the user in the Windows Control Panel).

Format:

Print *expression*

Print text on the current printer at the current x, y coordinates. The x and y coordinates are set using the **PrintMoveTo** subroutine.

Use the following technique for printing text and graphics to the default printer:

- Send text and graphics to the Printer Object and print them using the **PrintNewPage** and **PrintEndDoc** subroutines.

The printer object is a device-independent drawing space that supports **Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight** and **PrintTextWidth** subroutines. When you finish placing the information on the printer object, you use the **PrintEndDoc** or **PrintNewPage** subroutines to send the output to the printer.

Related Topics: PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect , PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

```
Option Explicit
Sub Main()
    dim x as integer
    dim LineHeight as integer
    dim Margin as integer

    PrintBeginDoc
    ' Set printer font
    PrintSetFont "Courier New"
    PrintSetFontStyle fsFontBold
    ' Calculate margin as one inch
    Margin = PrintPageHeight / 11
    ' Calculate line height based on current font
    LineHeight = PrintTextHeight("X")

    'Print a title
    PrintMoveTo Margin, Margin - LineHeight
    Print "List of Orders for Account " + GetScreenText(10, 5, 9)

    'Print order ids and amounts
    PrintSetFontStyle fsNormal
    PrintMoveTo Margin, Margin
    For x = 1 to 13
        PrintMoveTo Margin, Margin + x * LineHeight
        Print GetScreenText(18, 7 + Str(x), 11), _
            GetScreenText(44, 7 + Str(x), 12)
    Next x

    ' Print a box around orders ids and amounts
    PrintRect Margin, Margin, Margin + (Margin * 6), _
                Margin + (LineHeight * (x + 1)), 5
    PrintEndDoc
    MsgBox "Printing Complete", MB_ICONINFORMATION
End Sub
```

PrintBeginDoc Subroutine (eXpress Plus)

Initialize the printer object (page) context.

Format:

PrintBeginDoc

Related Topics: Print, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintEndDoc Subroutine (eXpress Plus)

Terminate printing. If print is in the current printer context, it is printed.

Format:

PrintEndDoc

Related Topics: Print, PrintBeginDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintMoveTo Subroutine (eXpress Plus)

Move the current x- and y-coordinates of the print object. The new x- and y-coordinates will be used as the upper left position of text printed using the **Print** statement.

Format:

PrintMoveTo *x-coordinate*, *y-coordinate*

The *x-coordinate* and *y-coordinate* parameters are any integer expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintNewPage Subroutine (eXpress Plus)

Cause the current content of the printer object to be printed immediately.

Format:

PrintNewPage

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintPageHeight Function (eXpress Plus)

Retrieve the current page height in pixels. The pixel height and width of the page will vary depending on the currently selected printer.

Format:

PrintPageHeight()

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintPageWidth Function (eXpress Plus)

Retrieve the current page width in pixels. The pixel height and width of the page will vary depending on the currently selected printer.

Format:

PrintPageWidth()

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintRect Subroutine (eXpress Plus)

Print a rectangle using the specified left, top, right and bottom coordinates, and line width.

Format:

PrintRect left, *top*, *right*, *bottom*, *width*

The *left*, *top*, *right*, *bottom* and *width* parameters are any integer expression and are specified in pixels.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSelect Function (eXpress Plus)

Change to the selected printer from within a script. The selection is only valid for the duration of the eXpress Plus session (not the individual screen session).

Format:

PrintSelect (*PrinterName*)

The *PrinterName* is the name of any printer available to Windows. Because Windows printer names often are very long, this function allows you to enter just enough of the name to uniquely identify the desired printer. For example "hp deskjet 990c" can be entered as just "hp" as long as there is no other printer that starts with the same characters. The name is not case sensitive.

If an empty string is given instead of a printer name, the Printer Setup dialog will be displayed. The user may then select a printer and press the OK button, leave it set to the default and press the OK button or Cancel the dialog. In either case when the OK button is pressed, the Printer Name from the dialog will be used.

PrintSelect returns an integer. A True (-1) is returned if successful, else False (0) is returned.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

```
if PrintSelect("HP") then
      UTS_Key_PrintScreenAll
      PrintSelect("")
else
      MsgBox "Invalide printer name selected"
end if
```

Example 1: Select a printer from the dialog.

```
PrintSelect("")
```

Example 2: Select a specific printer.

```
PrintSelect("Lexmark Optra plus PS2")
```

PrintSetFont Subroutine (eXpress Plus)

Set the current print font.

Format:

PrintSetFont *name*

The *name* parameter is any string expression containing a valid font name. If the font does not exist on the user's system, Windows will substitute a default font of the selected printer.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetFontSize Subroutine (eXpress Plus)

Set the size of the current print font in points.

Format:

PrintSetFontSize *size*

The *size* parameter is any integer expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetFontStyle Subroutine (eXpress Plus)

Set the style of the current print font.

Format:

PrintSetFontStyle *style*

The *style* parameter is a numeric expression containing a number equal to the sum of all required attributes.

The *style* parameter may be stated as an Integer or a Constant:

Effect	Integer	Constant
Normal	0	fsNormal
Bold	1	fsFontBold
Italic	2	fsFontItalic
Underline	4	fsFontUnderline
Strikethrough	8	fsFontStrikeThru

Styles may be ORed together to create combined effects. For example, to set bold and italic, use "fsFontBold OR fsFontItalic".

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetOrientation, PrintTextHeight, PrintTextWidth

Example:

(See Print Statement).

PrintSetOrientation Subroutine (eXpress Plus)

Set the print page orientation to portrait or landscape.

Format:

PrintSetOrientation *orientation*

The *orientation* parameter is stated as an Integer. Use 0 for portrait or 1 for landscape.

Note: The **PrintSetOrientation** should be done the **PrintBeginDoc** subroutine.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintTextHeight, PrintTextWidth

Example:

```
Sub Main()
'Printer orientation test

      PrintSelect("")
      PrintSetOrientation 0 ' Protrait
      PrintBeginDoc
      Print "This is a line of text in protrait mode"
      PrintEndDoc
      PrintSetOrientation 1 ' Landscape
      PrintBeginDoc
      Print "This line show be in landscape mode"
      PrintEndDoc
End Sub
```

PrintTextHeight Function (eXpress Plus)

Retrieve the height in pixels of a specified text string. The text height can be used to determine vertical spacing between lines.

Format:

PrintTextHeight(*textstring*)

The *textstring* parameter is any string expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextWidth

Example:

(See Print Statement).

PrintTextWidth Function (eXpress Plus)

Retrieve the width in pixels of a specified text string. The text width can be used to determine horizontal spacing.

Format:

PrintTextWidth(*textstring*)

The *textstring* parameter is any string expression.

Related Topics: Print, PrintBeginDoc, PrintEndDoc, PrintMoveTo, PrintNewPage, PrintPageHeight, PrintPageWidth, PrintRect, PrintSelect, PrintSetFont, PrintSetFontSize, PrintSetFontStyle, PrintSetOrientation, PrintTextHeight

Example:

(See Print Statement).

PushButton Statement

Use a push button in a dialog for assigning a button to a command.

Format:

PushButton *starting-x-pos*, *starting-y-pos*, *width*, *height*, "*caption*" *.name*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, Text, TextBox

Example:

```
Sub Main
    Begin Dialog UserDialog1 60,60, 260, 188, "3", .Enable
        Text 8,10,73,13, "Text Label:"
        TextBox 8, 26, 160, 18, .FText
        CheckBox 8, 56, 203, 16, "Check to display controls",. Chk1
        GroupBox 8, 79, 230, 70, "This is a group box:", .Group
        CheckBox 18,100,189,16, "Check to change button text", .Chk2
        PushButton 18, 118, 159, 16, "File History", .History
        OKButton 177, 8, 58, 21
        CancelButton 177, 32, 58, 21
        End Dialog

    Dim Dlg1 As UserDialog1
    x = Dialog( Dlg1 )
End Sub

Function Enable( ControlID$, Action%, SuppValue%)

Begin Dialog UserDialog2 160,160, 260, 188, "3", .Enable
    Text 8,10,73,13, "New dialog Label:"
    TextBox 8, 26, 160, 18, .FText
    CheckBox 8, 56, 203, 16, "New CheckBox", .ch1
    CheckBox 18,100,189,16, "Additional CheckBox", .ch2
    PushButton 18, 118, 159, 16, "Push Button", .but1
    OKButton 177, 8, 58, 21
    CancelButton 177, 32, 58, 21
End Dialog
Dim Dlg2 As UserDialog2
```

```
Dlg2.FText = "Your default string goes here"

Select Case Action%

Case 1
    DlgEnable "Group", 0
    DlgVisible "Chk2", 0
    DlgVisible "History", 0
Case 2
    If ControlID$ = "Chk1" Then
        DlgEnable "Group"
        DlgVisible "Chk2"
        DlgVisible "History"
    End If

    If ControlID$ = "Chk2" Then
        DlgText "History", "Push to display nested dialog"
    End If

    If ControlID$ = "History" Then
        Enable =1
        x = Dialog( Dlg2 )
    End If

Case Else

End Select
Enable =1

End Function
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Put Statement

Write to a disk file from a variable.

Format:

Put [#] *filenumber*, [*recordnumber*,] *variablename*

The **Put** statement has three parts:

Parameter	Description
filenumber	The number used to open the file.
recordnumber	For files opened in Binary mode, *recordnumber* is the byte position where writing starts.
variablename	The name of the variable containing the data to be written to the file.

Related Topics: Open, Get

Randomize Statement

Initialize the random number generator.

Format:

Randomize [*number*]

The **Randomize** statement has one optional parameter: *number*. This parameter can be any valid number and is used to initialize the random number generator. If you omit the parameter, then the value returned by the **Timer** function is used as the default parameter to seed the random number generator.

Example:

```
Sub Main
      Dim MValue
      Randomize   ' Initialize random-number generator.
      MValue = Int((6 * Rnd) + 1)
      Print MValue
End Sub
```

Receive Function (eXpress Plus)

Receive a message from the host. This function is used in conjunction with **HoldMessages**. When the Receive function is called, one message will be accepted and processed (moved to the screen, etc.). If no messages are received from the host within the specified *TimeOut* value (in milliseconds), the function will return a False (zero). If a message is received, the function returns as True (-1).

Format:

Receive (*TimeOut*)

This function returns an Integer. The *TimeOut* parameter is valid integer expression.

This function is useful when a stream of messages is expected from the host and the script needs to process each one separately. Receiving each message explicitly guarantees that no messages will be missed due to the asynchronous nature of message receipt from the host.

Related Topics: HoldMessages; UnholdMessages

ReDim Statement

Declare dynamic arrays and reallocate storage space.

Format:

ReDim *varname*(*subscripts*)[As *type*][,*varname*(*subscripts*)]

The **ReDim** statement is used to size or resize a dynamic array that has already been declared using the **Dim** statement with empty parentheses. You can use the **ReDim** statement to change the number of elements in an array repeatedly, but not to change the number of dimensions in an array or the type of the elements in the array.

Related Topics: Dim, Option Base, Set, Static

Example:

```
Sub Main

Dim TestArray() As Integer
Dim I
ReDim TestArray(10)
For I = 1 To 10
      TestArray(I) = I + 10
      Print TestArray(I)
Next I

End Sub
```

RefreshScreen Subroutine (eXpress Plus)

Repaint the screen in its entirety. The **RefreshScreen** subroutine should be used after one or more **SetScreenText** subroutine calls in order to see all information painted on the screen.

Format:

RefreshScreen

Note: Normally, **RefreshScreen** should only be used after the last **SetScreenText** call since **RefreshScreen** has to paint the entire screen and, as such, takes longer to execute than other screen handling commands.

Related Topics: SetScreenText

Rem Statement

Include explanatory remarks in a program.

Format:

Rem *remark*

Or any characters after a single quote (') on a line:

' *remark*

The *remark* parameter is the text of any comment you wish to include in the code.

Example:

```
Rem This is a remark

Sub Main()

    Dim Answer, Msg               ' Declare variables.
```

```
    Do
        Answer = InputBox("Enter a value from 1 to 3.")
        Answer = 2
        If Answer >= 1 And Answer <= 3 Then    ' Check range.
            Exit Do                ' Exit Do...Loop.
        Else
            Beep                   ' Beep if not in range.
        End If
    Loop
    MsgBox "You entered a value in the proper range."
End Sub
```

Right Function

Return the right-most *number* characters of the string parameter.

Format:

Right(*string*, *number*)

The *string* parameter is the string expression from which the rightmost characters are returned.

The *number* parameter is the numeric expression indicating the number of characters that will be returned.

Related Topics: Left, Len, Mid

Example:

```
' The example uses the Right function to return the first
' of two words input by the user.

Sub Main ()
    Dim LWord, Msg, RWord, SpcPos, UsrInp ' Declare variables
    Msg = "Enter two words separated by a space."
    UsrInp = InputBox(Msg)                ' Get user input
    SpcPos = InStr(1, UsrInp, " ")        ' Find space
    If SpcPos Then
        LWord = Left(UsrInp, SpcPos - 1)  ' Get left word
        RWord = Right(UsrInp, Len(UsrInp) - SpcPos)
                                          ' Get right word
        Msg = "The first word you entered is <" & LWord & ">"
        Msg = Msg & RWord & "."
    Else
        Msg = "You didn't enter two words."
    End If
    MsgBox Msg                            ' Display message
End Sub
```

RmDir Statement

Remove an existing directory.

Format:

RmDir *path*

The *path* parameter is a string that is the name of the directory to be removed.

Related Topics: ChDir, ChDrive, CurDir, Dir, MkDir

Example:

```
' This sample shows the functions mkdir (Make Directory)
' and rmdir (Remove Directory)

Sub Main
   Dim dirName As String
   dirName = "t1"
   mkdir dirName
   mkdir "t2"
   MsgBox "Directories: t1 and t2 created. Press OK to  " _
            & "remove them"
   rmdir "t1"
   rmdir "t2"
End Sub
```

Rnd Function

Returns a random number.

Format:

Rnd [(*number*)]

The *number* parameter must be a valid numeric expression.

The **Rnd** function returns a Single value less than 1 but greater than or equal to 0.

The value of *number* determines how **Rnd** generates a random number:

Value of *number*:	Number returned:
< 0	The same number every time as determined by *number*.
> 0	The next random number in the sequence.
= 0	The number most recently generated.
number omitted	The next random number in the sequence.

Example:

```
' The example uses the Rnd function to simulate rolling a
' pair of dice by generating random values from 1 to 6.

Sub Main ()
    Dim Dice1, Dice2, Msg          ' Declare variables.
    Dice1 = CInt(6 * Rnd() + 1)    ' Generate first die value.
    Dice2 = CInt(6 * Rnd() + 1)    ' Generate second die value.
    Msg = "You rolled a " & Dice1
    Msg = Msg & " and a " & Dice2
    Msg = Msg & " for a total of "
    Msg = Msg & Str(Dice1 + Dice2) & "."
    MsgBox Msg                     ' Display message.
End Sub
```

SaveScreen Subroutine (eXpress Plus)

Save an entire screen/form to a file.

Format:

SaveScreen *filename*

The *filename* parameter is any string expression containing the file name to receive the screen/form. This capability is commonly used to save forms to files in the T27 environment and reload them (specific form loads can be assigned to function keys) from files rather than from the host. The file format is binary and only usable by the emulator.

Related topic: FileOpenDialog, FileSaveDialog, LoadScreen

Example:

```
Sub Main()
      Dim fname as String

      fname = FileSaveDialog("C:\AA_UX32_3.00\T27Scripts", "", "FRM",
"Form files(*.frm)|*.frm|All files(*.*)|*.*", "Select Save Form File")
      If fname <> "" Then
            SaveScreen(fname)
      Else
            MsgBox "Save Cancelled"
      End If
End Sub
```

ScreenAvailable Function (eXpress Plus)

Returns a 1 (true) is the specified screen number is available (configured with a route).

Format:

ScreenAvailable (*ScreenNumber*)

The *ScreenNumber* parameter is an integer expression. If an invalid screen number is entered, it is ignored.

Related Topics: ActivateScreen, ScreenOpen

Example: See ActivateScreen.

ScreenOpen Function (eXpress Plus)

Returns a 1 (true) if the specified screen number is currently open.

ScreenOpen (*ScreenNumber*)

The *ScreenNumber* parameter is an integer expression. If an invalid screen number is entered, it is ignored.

Related Topics: ActivateScreen, ScreenAvailable

Example: See ActivateScreen.

Second Function

Return an integer between 0 and 59 that is the portion of the *time* parameter representing the second of the minute.

Format:

Second(*time*)

The *time* parameter is any string expression that can represent a time.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Weekday, Year

Example:

```
MyTime = "08:04:23 PM"
MsgBox MyTime

MsgBox Second( MyTime ) & " Seconds"
MsgBox Minute( MyTime ) & " Minutes"
MsgBox Hour( MyTime ) & " Hours"
```

Seek Function

Return a number that represents the byte position where the next operation is to take place. The first byte in the file is at position 1.

Format:

Seek(*filenumber*)

The *filenumber* parameter is used on an open statement and must be a valid numeric expression.

Related Topics: Open

Example:

```
Sub Main
    Open "TESTFILE" For Input As #  ' Open file for reading.
    Do While Not EOF(1)             ' Loop until end of file.
    MyChar = Input(1, #1)           ' Read next character of data.
    Print Seek(1)                   ' Print byte position  .
    Loop
    Close #1                        ' Close file.
End Sub
```

Seek Statement

Set the position in a file for the next read or write.

Format:

Seek *filenumber*, *position*

The *filenumber* parameter is used in the open statement and must be a valid numeric expression. The *position* parameter is the number that indicates where the next read or write is to occur. In Enable, *position* is the byte position relative to the beginning of the file.

Related Topics: Open

Example:

```
Sub Main
    Open "TESTFILE" For Input As #1 ' Open file for reading.
    For i = 1 To 24 Step 3          ' Loop until end of file.

    Seek #1, i                      ' Seek to byte position
    MyChar = Input(1, #1)           ' Read next character of data.
    Print MyChar                    ' Print character of data.
```

```
    Next i
    Close #1                        ' Close file.
End Sub
```

Select Case Statement

Execute one of the sets of statement(s) in the case, based on the test variable.

Format:

Select Case *testexpression*
[Case *expressionlist1*
[*statement(s)*]]
[Case *expressionlist2*
[*statement(s)*]]
[Case Else
[*statement(s)*]]
End Select

The **Select Case** statement has these parts:

Part	Description
Select Case	Begin the **Select Case** decision control sequence.
testexpression	The *testexpression* is any numeric or string expression. If *testexpression* matches the *expressionlist* associated with the **Case** clause, the *statement(s)* following the **Case** clause are executed.
Case	Set apart a group of Enable statements to be executed if an expression in *expressionlist* matches the *testexpression*.
expressionlist	The *expressionlist* consists of a comma-delimited list of one or more of the following forms: *expression* *expression* **To** *expression* **Is** *compare-operator expression*
statement(s)	Any number of Enable statements on one or more lines.
Case Else	Begin the statement(s) to be executed if no match is found between the *testexpression* and an *expressionlist* in any of the other **Case** selections.
End Case	Ends the **Select Case**.

The *expression* parameter may be any numeric or string expression; however, it must be compatible with the type of *testexpression*.

The *compare-operator* may be any valid comparison operator, except **Is** and **Like**.

Related Topics: If...Then...Else

See also, Data Types, Operators and Precedence.

Example:

```
Sub Test ()
    For x = 1 to 5
        print x
        Select Case x
        Case 2
            Print "Outer Case Two"
        Case 3
            Print "Outer Case Three"
'            Exit For
            Select Case x
            Case 2
                Print "Inner Case Two"
            Case 3
                Print "Inner Case Three"
'                Exit For
            Case Else    ' Must be something else.
                Print "Inner Case Else:", x
```

```
            End Select

            Print "Done with Inner Select Case"
        Case Else    ' Must be something else.
            Print "Outer Case Else:",x
        End Select
    Next x
    Print "Done with For Loop"
End Sub
```

SendKeys Statement

Send one or more keystrokes to the active window as if they had been entered at the keyboard.

Format:

SendKeys *keys*

The *keys* parameter is a string and is sent to the active window.

To send a single keyboard character, use the character itself. To send the letter A, use "A". To send multiple keyboard characters, one behind the other, include them in the string in the order you want them sent. To send a D followed by an E and then followed by an F, use "DEF".

Ten keyboard characters have special significance when used with the **SendKeys** statement:

Character(s)	Usage
Braces { }	Braces are used to enclose a special character or key name being sent. For example, {F4} sends function key 4.
Plus sign +	The plus sign is the SHIFT key.
Caret ^	The caret is the CTRL key.
Percent sign %	The percent sign is the ALT key.
Tilde ~	The tilde is the ENTER key.
Parentheses ( )	Parentheses are used to enclose multiple keystrokes in combination with the SHIFT, CTRL and ALT keys. For example, "%(EF)" would be the same as holding down the ALT key while pressing E followed by F.
Brackets [ ]	No special significance but must be enclosed in braces when sent; e.g., "{[} and {]}".

To send any special character, enclose it in braces. For example, "{{}" sends an open brace and "{+}" sends a plau sign.

To send keys that do not display when you press them, use the following substitution codes:

Key	Substitution Code
BACKSPACE	{BACKSPACE}
BREAK	{BREAK}
CAPS LOCK	{CAPSLOCK}
DEL	{DELETE} or {DEL}
DOWN ARROW	{DOWN}
END	{END}
ENTER	{ENTER} or ~
ESC	{ESC}
HELP	{HELP}
HOME	{HOME}
INS	{INSERT}
LEFT ARROW	{LEFT}
NUM LOCK	{NUMLOCK}
PAGE DOWN	{PGDN}
PAGE UP	{PGUP}
RIGHT ARROW	{RIGHT}
SCROLL LOCK	{SCROLLLOCK}
TAB	{TAB}
UP ARROW	{UP}

Key	Substitution Code
F1	{F1}
F2	{F2}
:	:
F16	{F16}

To repeat a key, follow the key by the number of times to repeat the keystroke. For example, "{UP 10}" is the same as pressing the UP ARROW 10 times. Note: A space is required between the key and the number.

Example:

```
Sub Main ()
    Dim I, X, Msg               ' Declare variables.
    X = Shell("Calc.exe", 1)    ' Shell Calculator.
    For I = 1 To 5              ' Set up counting loop.
        SendKeys I & "{+}"
                                ' Send keystrokes to Calculator.
    Next I                      ' to add each value of I.

    Msg = "Choose OK to close the Calculator."
    MsgBox Msg                  ' Display OK prompt.
    AppActivate "Calculator"    ' Return focus to Calculator.
    SendKeys "%{F4}"            ' Alt+F4 to close Calculator.
End Sub
```

Set Statement

Assign an object to an object variable.

Format:

Set *objectvar* = {[New] *objectexpression* | Nothing}

Related Topics: Dim, Global, Static

Example:

```
Sub Main
    Dim visio As Object
    Set visio = CreateObject( "visio.application" )
    Dim draw As Object
    Set draw = visio.Documents
    draw.Open "c:\visio\drawings\Sample1.vsd"
    MsgBox "Open docs: " & draw.Count
    Dim page As Object
    Set page = visio.ActivePage
    Dim red As Object
    Set red = page.DrawRectangle (1, 9, 7.5, 4.5)
    red.FillStyle = "Red fill"

    Dim cyan As Object
    Set cyan = page.DrawOval (2.5, 8.5, 5.75, 5.25)
    cyan.FillStyle = "Cyan fill"

    Dim green As Object
    Set green = page.DrawOval (1.5, 6.25, 2.5, 5.25)
    green.FillStyle = "Green fill"

    Dim DarkBlue As Object
    set DarkBlue = page.DrawOval (6, 8.75, 7, 7.75)
    DarkBlue.FillStyle = "Blue dark fill"

    visio.Quit
End Sub
```

SetCursor Subroutine (eXpress Plus)

Set the column and row position of the cursor within the logical screen.

Format:

SetCursor *col*, *row*

Related Topics: GetCursorCol, GetCursorRow

SetDblClickAction Subroutine (eXpress Plus)

Set the action to be taken when the user double clicks the mouse on a screen. The subroutine also allows a script name to be assigned when the action is set to run a script.

Format:

SetDblClickAction *action*, *script*

The *action* parameter is an integer expression specifying the action to be taken upon a double click.

The *action* parameter may be expressed as an Integer or Constant:

Action	Integer	Constant
None	0	DBLCLICKNONE
Transmit	1	DBLCLICKTRANSMIT
Run Script	2	DBLCLICKRUNSCRIPT

The *script* parameter is any string expression containing the script name (including file extension) to be executed when the user double clicks the mouse on the screen. Setting the script name to blank has the same effect as setting the action to DBLCLICKNONE.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Example:

```
Sub Main()
' *** Automatic Sign On Script
    Wait 2000
    If not WaitForString("LOGON - Menu-Assisted Resource Control            ") Then Exit Sub
    EnterText "U1ID"
    T27Key TK_TAB
    EnterText "U1PW"
    T27Key TK_TRANSMIT
    SetDblClickAction DBLCLICKRUNSCRIPT, "T27Marc2.bas"
' *** End of recorded script
End Sub
```

SetScreenText Subroutine (eXpress Plus)

Set the string value of an area within the logical screen. This subroutine will set text regardless of protected FCCs in the screen.

Format:

SetScreenText *col*, *row*, *len*, *text*

The *col*, *row* and *len* parameters are any integer expression. The *text* parameter is any string expression.

If *row* is specified as -1, then *col* is assumed to be the offset from the beginning of the logical screen buffer. For Example:

```
SetScreenText 5, 2, 5, "text"
```

Is the same as:

```
SetScreenText 85, -1, 5, "text"
```

The above example assumes the screen has 80 columns.

Related Topics: GetScreenText, GetScreenLine, RefreshScreen

Example:

```
' Put the trans code in the screen
SetScreenText 1, 1, 6, "CUST1 "
```

Example 2:

The **SetScreenText** can be used to issue UTS control sequences (Unisys ClearPath 2200 Servers only) in a script at the beginning of the screen allowing control page updates or setting FCCs in the screen.

Control characters are always entered as symbolic names enclosed within angle brackets *<>". If a "<" character is needed, it can be entered as

The following Control Sequence moves the cursor to the home position, clears the entire screen and then sets up an FCC field with video off to allow hidden entry of a user id and password:

<ESC>e<ESC>m<US> E@

The user id and password would be entered with another **SetScreenText**.

The following sequence would restore video on:

<ESC>e<ESC>m<US> D@

SetSignOnResult Subroutine

Set the result of sign-on script.

Format:

SetSignOnResult *Result*

Result is True (-1) or False (0).

Example:

```
SetSignOnResult True
```

SetWindowState Subroutine (eXpress Plus)

Set the specified state of the current screen window.

Format:

SetWindowState *State*

The *State* parameter is a integer expression specifying the property type to which the window of the current screen is to be set. The *State* parameter may be expressed as an Integer or Constant:

State	Integer	Constant
Window Normal	0	WSNORMAL
Window Minimized	1	WSMINIMIZED
Window Maximized	2	WSMAXIMIZED

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Related Topic: GetWindowState

Sgn Function

Return an integer indicating the sign (+, –, 0) of a number.

Format:

Sgn(*number*)

The number parameter can be any valid numeric expression. The **Sgn** function returns the following values:

Value	Condition
1	*number* > 0
0	*number* = 0
-1	*number* < 0

Shell Function

Run an executable program.

Format:

Shell(*app* [,*style*])

The **Shell** function has two parameters. The first one, *app*, is the name of the program to be executed. The name of the program in *app* must include a .PIF, .COM, .BAT or .EXE file extension or an error will occur. The second argument, *style*, is the number corresponding to the style of the window. The second argument is also optional, and if omitted, the program is opened minimized with focus.

Value	Window Style
1, 5, 9	Normal with focus.
2	Minimized with focus (default).
3	Maximized with focus.
4, 8	Normal without focus.
6, 7	Minimized without focus.

Return value: ID, the task ID of the started program.

Example:

```
' This example uses Shell to leave the current application
' and run the Calculator program included with Microsoft
' Windows; it then uses the SendKeys statement to send
```

```
' keystrokes to add some numbers.
Sub Main ()
  Dim I, X, Msg               ' Declare variables
  X = Shell("Calc.exe", 1)    ' Shell Calculator
  For I = 1 To 5              ' Set up counting loop
    SendKeys I & "{+}"        ' Send keystrokes to Calculator
  Next I                      ' to add each value of I
  Msg = "Choose OK to close the Calculator."
  MsgBox Msg                  ' Display OK prompt
  AppActivate "Calculator"    ' Return focus to Calculator
  SendKeys "%{F4}"            ' Alt+F4 to close Calculator
End Sub
```

Sin Function

Return the sine of an angle that is expressed in radians.

Format:

Sin(*radian*)

Example:

```
Sub Main ()
    pi = 4 * Atn(1)
    rad = 90 * (pi/180)
    x = Sin(rad)
    print x
End Sub
```

Space Function

Skip a specified number of spaces in a **Print #** statement.

Format:

Space(*number*)

The *number* parameter can be any valid integer and determines the number of blanks.

Example:

```
Sub Main
    MsgBox "Hello" & Space(20) & "There"
End Sub
```

Sqr Function

Return the square root of a number.

Format:

Sqr(*number*)

The *number* parameter must be a valid number greater than or equal to zero.

Example:

```
Sub Form_Click ()
    Dim Msg, Number           ' Declare variables.
    Msg = "Enter a non-negative number."
    Number = InputBox(Msg)    ' Get user input.
    If Number < 0 Then
        Msg = "Cannot determine the square root of a " _
            & "negative number."
    Else
        Msg = "The square root of " & Number & " is "
        Msg = Msg & Sqr(Number) & "."
    End If
    MsgBox Msg                ' Display results.

End Sub
```

Static Statement

Declare variables and allocate storage space. These variables will retain their value through the program run.

Format:

Static *variable*

Related Topics: Dim, Function, Sub

Example:

```
' This example shows how to use the static keyword to
' retain the value of the variable i in sub Joe.  If Dim is
' used instead of Static then i is empty when printed on
' the second call as well as the first.

Sub Main
    For i = 1 to 2
        Joe 2
    Next i
End Sub
Sub Joe( j as integer )
    Static i
    print i
    i = i + 5
    print i
End Sub
```

Stop Statement

End execution of the program.

Format:

Stop

The **Stop** statement can be placed anywhere in your code.

Related Topics: End, Exit

Example:

```
Sub main ()
    Dim x,y,z

    For x = 1 to 5
        For y = 1 to 5
            For z = 1 to 5
            Print "Looping" ,z,y,x
            Next z
        Next y
          Stop
    Next x
End Sub
```

StorePage Subroutine (UTS eXpress Plus)

Store the current screen to the specified page. The specified page is overwritten.

Format:

StorePage(*PageNumber*)

The *PageNumber* is any integer expression.

Note: This subroutine is only meaningful when the screen is configured for 2 or more pages (see, Pages, in UTS eXpress Plus Configuration or in UTS eXpress Net Administration). Furthermore, this subroutine does not apply to page usage in T27 emulation.

Related topics: GetPage, GoToPage

Example:

The following is a general-purpose script to perform the **GoToPage**, **StorePage** and **GetPage** subroutines. It can be used in conjunction with a custom tool bar or script menu. It uses the first two character of the button or menu item caption to determine the action to be performed.

```
Sub Main()
' General paging action script

' The following dialog will prompt the user for a page number.  The user can
```

```
' click a button or just press a number key to select the page.
Begin Dialog PAGEDLG 1,58, 233, 29, "Select page (Click button or press key)"
      PushButton 4,8,17,12, "&1", .PushButton_1
      PushButton 24,8,17,12, "&2 ", .PushButton_2
      PushButton 44,8,16,12, "&3 ", .PushButton_3
      PushButton 64,8,16,12, "&4", .PushButton_4
      PushButton 84,8,16,12, "&5", .PushButton_5
      PushButton 104,8,16,12, "&6 ", .PushButton_6
      PushButton 124,8,16,12, "&7", .PushButton_7
      PushButton 144,8,16,12, "&8 ", .PushButton_8
      PushButton 164,8,16,12, "&9", .PushButton_9
      CancelButton 184,8,40,12
End Dialog


Dim D as PAGEDLG

      Cap = GetUserParam(3) ' Get caption of calling button or menu item

      Btn = Dialog(D) ' Prompt for page number

      If Btn >= 1 and Btn <= 9 Then
             Select Case UCase$(Left$(Cap, 2))     ' Perform action based on button or
                                                   ' menu item caption (1st 2 chars)
                    Case "ST"
                           StorePage(Btn)
                    Case "GE"
                           GetPage(Btn)
                    Case "GO"
                           GoToPage(Btn)
             End Select
      End If
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Str Function

Return the value of a numeric expression.

Format:

Str(*numericexpression*)

Str returns a String.

Use the **Format** function to convert numeric values you want formatted as dates, times or in other user-defined formats.

The **Str** function recognizes only the period (.) as a valid decimal separator. When a possibility exists that different decimal separators may be used (e.g., in international applications), you should use **CStr** to convert a number to a string.

Related topics: CStr, Format, Val

Example:

```
Sub main ()
    Dim msg
    a = -1
    msgBox "Num = " & Str(a)
    MsgBox "_Abs(Num) =" & Str(_Abs(a))

End Sub
```

StrComp Function

Return a variant that is the result of the comparison of two strings.

Format:

StrComp(*string1*,*string2*, [*compare*])

Example:

```
Sub Main

Dim MStr1, MStr2, MComp
      MStr1 = "ABCD": MStr2 = "today"       ' Define variables.
      print MStr1, MStr2
      MComp = StrComp(MStr1, MStr2) ' Returns -1.
      print MComp
      MComp = StrComp(MStr1, MStr2) ' Returns -1.
      print MComp
      MComp = StrComp(MStr2, MStr1) ' Returns 1.
      print MComp
End Sub
```

String Function

String is used to create a string that consists of one character repeated repeatedly.

Formats:

String(*numeric*, *charcode*)

or

String(*numeric*, *string*)

String returns a string.

Related topics: Space

Example:

```
Sub Main

    Dim MString
    MString = String(5, "*")  ' Returns "*****".
    MString = String(5, 42)   ' Returns "44444".
    MString = String(10, "Today")     ' Returns "TTTTTTTTTT".
    Print MString
End Sub
```

Sub Statement

Declare and define a **Sub** procedure name, parameters and code.

Format:

Sub *subname* [(*argumentlist*)]
[*statement(s)*]
subname = *expression*
[Exit Sub]
[*statement(s)*]
subname = *expression*
End Sub

When the optional *argumentlist* needs to be passed, the format is as follows:

([ByVal] *variable* [As *type*][,[ByVal] *variable* [As type]]...)

The optional **ByVal** parameter specifies that the *variable* is passed by value instead of by reference (see ByRef and ByVal).

The optional **As** *type* parameter is used to specify the data type. Valid types are **String**, **Integer**, **Single**, **Double**, **Long** and **Variant** (see Other Data Types).

Related Topics: Call, Dim, Function

Example:

```
Sub Main
   Dim DST As String
   DST = "t1"
   mkdir DST
   mkdir "t2"
End Sub
```

SwitchToolbar Subroutine (eXpress Plus)

Change to a different toolbar.

Format:

SwitchToolbar *ToolbarFile*, *ShowIt*

ToolbarFile is any string expression comprising just the file name and extension (no path) of a toolbar file. *ShowIt* indicates weather or not the toolbar is visible (**True** or **False**). If **True**, the toolbar will be shown immediately upon executing the script; if **False**, the toolbar will be hidden. In addition, the new toolbar and shown state will be used the next time the screen is opened.

Examples:

```
SwitchToolBar "MAPPER.TBR", True   ' Show the mapper toolbar
SwitchToolBar "", False            ' Use the default toolbar and hide it
SwitchToolBar "", True             ' Show the default toolbar
SwitchToolBar "MYTOOL.TBR", False  ' Select a custom toolbar and hide it
```

T27Key Subroutine (eXpress Plus)

Issue any of the supported T27 keystrokes.

Format:

T27Key *key*

The *key* parameter is an integer expression representing the specific T27 key to be issued. The *key* may be specified as an Integer or Constant:

Constant	Integer
TK_ARROWDN	249
TK_ARROWLEFT	247
TK_ARROWRIGHT	248
TK_ARROWUP	246
TK_BACKSPACE	8
TK_BACKTAB	196
TK_BOUND	218
TK_CARRIAGERTN	13
TK_CLRALLVTAB	16442
TK_CLREOL	134
TK_CLREOP	135
TK_CLRFORMS	159
TK_CLRHOME	128
TK_COPY	16432
TK_CTRL	164
TK_CUT	16431
TK_DBLZERO	234
TK_DELCHAR	132
TK_DELCHARPAGE	16425
TK_DELLINE	133
TK_HOME	174
TK_INSCHAR	130
TK_INSCHARPAGE	16424
TK_INSLINE	131
TK_LOCAL	168
TK_LOCKCTRL	165
TK_LOGICALEOL	16415
TK_MARK	217
TK_MOVELINEDOWN	138
TK_MOVELINEUP	139
TK_NEXTPAGE	253
TK_PASTE	16434
TK_PREVPAGE	252
TK_PRINTALL	157
TK_PRINTUNPROT	156
TK_RECALL	214
TK_RECEIVE	170
TK_ROLLDN	136
TK_ROLLUP	137
TK_SETFORMS	158
TK_SPECIFY	166
TK_STORE	213

Constant	Integer
TK_TAB	198
TK_TOGGLEFORMS	141
TK_TOGGLETAB	16441
TK_TRANSMIT	172
TK_TRANSMITLINE	16428
TK_TRIPZERO	236
TK_UPPERONLYON	210
TK_UPPERONLYOFF	211
TK_WRITEESC	16426
TK_WRITEETX	3
TK_WRITEGS	16427

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Tan Function

Return the tangent of an angle as a double.

Format:

Tan(*angle*)

The *angle* parameter must be a valid angle expressed in radians.

Related Topic: Atn, Cos, Sin

Example:

```
Sub Main ()
    Dim Msg, Pi         ' Declare variables.
    Pi = 4 * Atn(1)     ' Calculate Pi.
    Msg = "Pi is equal to " & Pi
    MsgBox Msg          ' Display results.
    x = Tan(Pi/4)
    MsgBox  x & " is the tangent of Pi/4"
End Sub
```

Text Statement

Create a text field for titles and labels.

Format:

Text *starting-x-pos*, *starting-y-pos*, *width*, *height*, *label*

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, TextBox

Example:

```
Sub Main ()
    Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
        TEXT 10, 10, 28, 12, "Name:"
        TEXTBOX 42, 10, 108, 12, .nameStr
        TEXTBOX 42, 24, 108, 12, .descStr
        CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
        OKBUTTON 42, 54, 40, 12
    End Dialog
    Dim Dlg1 As DialogName1
    Dialog Dlg1

    MsgBox Dlg1.nameStr
    MsgBox Dlg1.descStr
    MsgBox Dlg1.checkInt
End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

TextBox Statement

Create a Text Box for typing in numbers and text.

Format:

TextBox *starting-x-pos*, *starting-y-pos*, *width*, *height*, .*default_string*, [32]

The optional string, "32", instructs enable to provide password protection characters as the user types into the text box. The password protection character is the asterisk (*).

Related Topics: Begin Dialog, CancelButton, CheckBox, Dialog, DropListBox, GroupBox, ListBox, OKButton, OptionButton, OptionGroup, PushButton, Text

Example:

```
Sub Main ()
    Begin Dialog DialogName1 60, 60, 160, 70, "ASC - Hello"
        TEXT 10, 10, 28, 12, "Name:"
        TEXTBOX 42, 10, 108, 12, .nameStr
        TEXTBOX 42, 24, 108, 12, .descStr
        CHECKBOX 42, 38, 48, 12, "&CHECKME", .checkInt
        OKBUTTON 42, 54, 40, 12
    End Dialog
    Dim Dlg1 As DialogName1
    Dialog Dlg1

    MsgBox Dlg1.nameStr
    MsgBox Dlg1.descStr
    MsgBox Dlg1.checkInt

End Sub
```

For an easier and better alternative to designing dialogs, see the eXpress Dialog Form Designer. For eQuate users, use the Form Designer in the eQuate Applications Manager program.

Time Function, Time Statement

Returns the current system time or sets the system time.

Time function returns a value; the **Time** statement does not.

Function Format:

Time[()]

Statement Format:

Time = *time*

The time parameter is any numeric or string expression that represents a time.

```
x = Time$(Now)
Print x

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

To set the system time, use the TIME statement:

```
SysTime = "8:00:00 AM"
Time = SysTime
```

Timer Event

Timer Event is used to track elapsed time or can be displayed as a stopwatch in a dialog. The timer's value is the number of seconds from midnight.

Format:

Timer

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeSerial, TimeValue.

Example:

```
Sub Main

    Dim TS As Single
    Dim TE As Single
    Dim TEL As Single
```

```
        TS = Timer

        MsgBox "Starting Timer"

        TE = Timer

        TT = TE - TS
        Print TT

End Sub
```

TimeSerial Function

Return the time serial for the supplied parameters *hour, minute, second*.

Format:

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

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeValue

Example:

```
Sub Main

        Dim MTime
        MTime = TimeSerial(12, 25, 27)
        Print MTime

End Sub
```

TimeValue Function

Return a double precision serial number based of the supplied string parameter.

Format:

TimeValue (*timestring*)

Midnight = TimeValue("23:59:59")

Related topics: DateSerial, DateValue, Hour, Minute, Now, Second, TimeSerial

Example:

```
Sub Main

        Dim MTime
        MTime = TimeValue("12:25:27 PM")
        Print MTime

End Sub
```

Trim, LTrim, RTrim Functions

Return a copy of a string with leading, trailing or both leading and training spaces removed.

Format:

[L | R]Trim(*string*)

LTrim removes leading spaces. **RTrim** removes trailing spaces. **Trim** removes leading and trailing spaces.

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable.  It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls

Sub Main
    MyString = "  <-Trim->  "           ' Initialize string
    TrimString = LTrim(MyString)        ' TrimString = "<-Trim->  "
    MsgBox "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString)) ' TrimString = "  <-trim->"
    MsgBox "|" & TrimString & "|"
```

```
    TrimString = LTrim(RTrim(MyString)) ' TrimString = "<-Trim->"
    MsgBox "|" & TrimString & "|"       ' Using the Trim function
                                        ' alone achieves the same
                                        ' result.
    TrimString = UCase(Trim(MyString))  ' TrimString = "<-TRIM->"
    MsgBox "|" & TrimString & "|"
End Sub
```

Type Statement

Define a user-defined data type containing one or more elements.

Format:

Type *usertype elementname* [(*subscripts*)] As *typename*
[*elementname* [(*subscripts*)] As *typename*]
. . .
End Type

The Type statement has these parts:

Part	Description
Type	Marks the beginning of a user-defined type.
usertype	Name of a user-defined data type. It follows standard variable naming conventions.
elementname	Name of an element of the user-defined data type. It follows standard variable-naming conventions.
subscripts	Dimensions of an array element.
typename	One of these data types: Integer, Long, Single, Double, String (for variable-length strings), String * length (for fixed-length strings), Variant or another user-defined type. The argument *typename* cannot be an object type.
End Type	Marks the end of a user-defined type.

Once you have declared a user-defined type using the **Type** statement, you can declare a variable of that type anywhere in your script. Use **Dim** or **Static** to declare a variable of a user-defined type. Line numbers and line labels are not allowed in **Type...End Type** blocks.

User-defined types are often used with data records because data records frequently consist of a number of related elements of different data types. Arrays cannot be an element of a user-defined type in Enable.

Example:

```
' This sample shows some of the
' features of user defined types.
Type type1
    a As Integer
    d As Double
    s As String
End Type

Type type2
    a As String
    o As type1
End Type
Type type3
    b As Integer
    c As type2
End Type

Dim type2a As type2
Dim type2b As type2
Dim type1a As type1
Dim type3a as type3

Sub Form_Click ()
    a = 5
    type1a.a = 7472
    type1a.d = 23.1415
```

```
    type1a.s = "YES"
    type2a.a = "43 - forty three"
    type2a.o.s = "Yaba Daba Doo"
    type3a.c.o.s = "COS"
    type2b.a = "943 - nine hundred and forty three"
    type2b.o.s = "Yogi"
    MsgBox type1a.a
    MsgBox type1a.d
    MsgBox type1a.s
    MsgBox type2a.a
    MsgBox type2a.o.s
    MsgBox type2b.a
    MsgBox type2b.o.s
    MsgBox type3a.c.o.s
    MsgBox a
End Sub
```

UBound Function

Return the value of the largest usable subscript for the specified dimension of an array.

Format:

Ubound(*arrayname*[,*dimension*])

Related Topics: Dim, Global, Lbound, Option Base, Static

Example:

```
' This example demonstrates some of the features of
' arrays.  The lower bound for an array is 0 unless it is
' specified or Option Base has set it as is done in this
' example.

Option Base 1

Sub Main
    Dim a(10) As Double
    MsgBox "LBound: " & LBound(a) & " UBound: " & UBound(a)
    Dim i As Integer
    For i = 1 to 3
        a(i) = 2 + i
    Next i
    Print a(1),a(1),a(2), a(3)
End Sub
```

UCase Function

Return a copy of a string in which all lowercase characters have been converted to uppercase.

Format:

UCase(*string*)

Related Topics: LCase

Example:

```
' This example uses the LTrim and RTrim functions to strip leading
' and trailing spaces, respectively, from a string variable.  It
' uses the Trim function alone to strip both types of spaces.
' LCase and UCase are also shown in this example as well as the
' use of nested function calls

Sub Main
    MyString = "  <-Trim->  "            ' Initialize string
    TrimString = LTrim(MyString)        ' TrimString = "<-Trim->  "
    MsgBox "|" & TrimString & "|"
    TrimString = LCase(RTrim(MyString))    ' TrimString = "  <-trim->"
    MsgBox "|" & TrimString & "|"        ' TrimString = "<-Trim->"
    MsgBox "|" & TrimString & "|"        ' Using the Trim function
                            ' alone achieves the same
                            ' result.
```

```
    TrimString = UCase(Trim(MyString))    ' TrimString = "<-TRIM->"
    MsgBox "|" & TrimString & "|"
End Sub
```

UnholdMessages Subroutine (eXpress Plus)

Releases messages stopped by the **HoldMessages** statement. Messages will be received and processed in the normal manner.

Format:

UnholdMessages

Related Topics: HoldMessages; Receive

UTSKey Subroutine (eXpress Plus)

Issue any of the supported UTS keystrokes.

Format:

UTSKey *key*

The *key* parameter is an integer expression representing the specific UTS key to be issued. The *key* may be specified as an Integer or Constant:

Constant	Integer
UK_BACK_SPACE	95
UK_CURSOR_DOWN	6
UK_CURSOR_LEFT	7
UK_CURSOR_RETURN_KEY	32
UK_CURSOR_RIGHT	8
UK_CURSOR_TO_END_LINE	66
UK_CURSOR_TO_HOME	23
UK_CURSOR_TO_START_LINE	65
UK_CURSOR_UP	9
UK_DELETE_IN_DISPLAY	11
UK_DELETE_IN_LINE	12
UK_DELETE_LINE	10
UK_ERASE_CHAR	67
UK_ERASE_DISPLAY	14
UK_ERASE_TO_END_DISPLAY	15
UK_ERASE_TO_END_FIELD	16
UK_ERASE_TO_END_LINE	17
UK_FKEY_1	43
UK_FKEY_2	44
UK_FKEY_3	45
UK_FKEY_4	46
UK_FKEY_5	47
UK_FKEY_6	48
UK_FKEY_7	49
UK_FKEY_8	50
UK_FKEY_9	51
UK_FKEY_10	52
UK_FKEY_11	53
UK_FKEY_12	54
UK_FKEY_13	55
UK_FKEY_14	56
UK_FKEY_15	57
UK_FKEY_16	58
UK_FKEY_17	59
UK_FKEY_18	60
UK_FKEY_19	61
UK_FKEY_20	62
UK_FKEY_21	63
UK_FKEY_22	64
UK_INSERT_IN_DISPLAY	25
UK_INSERT_IN_LINE	26

Constant	Integer
UK_INSERT_LINE	24
UK_KEYBOARD_UNLOCK	27
UK_LINE_DUP	28
UK_MSG_WAIT	29
UK_PRINT_KEY	30
UK_PRINT_ENTIRE_SCREEN	69
UK_SOE	3
UK_TAB_BACK	33
UK_TAB_FORWARD	34
UK_TAB_SET	35
UK_TRANSMIT_KEY	36

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Val Function

Return the numeric value of a string of characters.

Format:

Val(*string*)

Example:

```
Sub main
    Dim Msg
    Dim YourVal As Double
    YourVal = Val(InputBox$("Enter a number"))
    Msg = "The number you enered is: " & YourVal
    MsgBox Msg
End Sub
```

VarType Function

Return a value that indicates how the parameter *varname* is stored internally.

Format:

VarType(*varname*)

The *varname* parameter is a variant data type.

VarType	Return Values	
Empty	0	
Null	1	
Integer	2	
Long	3	
Single	4	
Double	5	
Currency	6	(not available at this time)
Date/Time	7	(mapped to a string)
String	8	

Related Topics: **IsNull, IsNumeric**

Example:

```
If VarType(x) = 5 Then Print "Vartype is Double"
    ' Display variable type
```

Wait Subroutine (eXpress Plus)

Wait for a fixed amount of time. A call to this subroutine causes the script to wait for a period before continuing on to the next script statement, subroutine or function.

Format:

Wait *time*

The *time* parameter is an integer expression containing the amount of time to wait expressed in milliseconds.

Example:

(see Begin Dialog).

WaitForSpecificString Function (eXpress Plus)

Cause the script to wait for the specified *string* at the specified location on the screen.

Format:

WaitForSpecificString(*row*, *col*, *len*, *string*)

The *col*, *row* and *len* parameters are any integer expression. The *string* parameter is any string expression.

If the *len* parameter is zero (0), the length of the specified *string* will be used.

Related Topics: GetLastMsg , GetScreenLine, GetScreenText, WaitForString

Example:

(see GetScreenText).

WaitForString Function (eXpress Plus)

Cause the script to wait for the specified *string*. Like the **GetLastMsg** function, **WaitForString**, only retrieves the first 80 characters of the last message received (including any control sequences) from the host or communication system.

Format:

WaitForString(*string*)

The communication system may return multiple messages before control is returned to the script; therefore, only the last message is accessible by this function.

Since the message may contain control sequences, this function may not be very useful unless you are familiar with the handling of control sequences by the communications system. Consider using the WaitForSpecificString function.

Related Topics: GetLastMsg , GetScreenLine, GetScreenText, WaitForSpecificString

Weekday Function

Return an integer between 1 (Sunday) and 7 (Saturday) that represents the day of the week for a date argument.

Format:

Weekday(*date*)

The *date* parameter is any string expression that can represent a date.

The returned integer represents the day of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Second, Year

Example:

```
Sub Main
    MyDate = "03/03/96"
    print MyDate
    x = Weekday(MyDate)
    print x

End Sub
```

While...Wend Statement

Execute a series of statements as long as a condition is true.

Format:

While *condition*

[*statement(s)*]

Wend

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

Part	Description
While	Begins the **While...Wend** flow of control structure.
condition	The *condition* is any numeric or expression that evaluates to true or false. If the *condition* is true, the statements are executed.
statement(s)	Any number of valid script statements.
Wend	Ends the **While...Wend** flow of control structure.

Related Topics: Do...Loop Statement, With

See also, Data Types, Operators and Precedences

Example:

```
Sub Main
    Const Max = 5
    Dim A(5) As String
    A(1) = "Programmer"
    A(2) = "Engineer"
    A(3) = "President"
    A(4) = "Tech Support"
    A(5) = "Sales"
    Exchange = True

    While Exchange
        Exchange = False
        For I = 1 To Max
            MsgBox A(I)
        Next I
    Wend

End Sub
```

WinHelp Subroutine (eXpress Plus)

Start Windows Help (WINHELP.EXE) and ask for specific information.

Format:

WinHelp *HFile*, *Cmd*, *Dta*, *KWord*

The *HFile* parameter is any string expression containing the full file name including path and .HLP extension.

The *Cmd* parameter is any integer expression that represents a number equating to one of the help API commands (see below).

The *Dta* parameter is any integer expression containing a context id. of a topic in a help system. The parameter is only used on the CONTEXT, CONTEXTPOPUP and SETCONTENTS commands. For other commands, it is zero (0).

The *KWord* parameter is any string expression containing the keyword for the search. It is only used on the COMMAND, KEY and PARTIALKEY commands.

Following is a list of supported WinHelp commands and their corresponding *Cmd* value:

Command:	*Cmd* value:	Action:
COMMANDS	258	Executes a Help macro linked to the keyword (index) string in *Dta*.
CONTENTS	3	Displays the Help contents topic as defined by the Contents option in the [OPTIONS] section of the .HPJ file (this is a 16-bit convention and is usually replaced by FINDER for 32-bit help).
CONTEXT	1	Displays Help for a particular topic identified by a context number that has been defined in the [MAP] section of the .HPJ file.
CONTEXTPOPUP	8	Displays in a pop-up window a particular Help topic identified by a context number that has been defined in the [MAP] section of the .HPJ file.
FINDER	11	Displays the **Help Topics** dialog box for the file named in *HFile*. Requires a .CNT file with the same file name be located in the same directory as the .HLP file.
FORCEFILE	9	Ensures that WinHelp is displaying the correct Help file. If the correct Help file is currently displayed, there is no action. If the incorrect Help file is displayed, WinHelp opens the correct file.
HELPONHELP	4	Displays the **Help Topics** dialog of the Windows Help-on-Help file.
KEY	257	Displays the topic found in the keyword list that matches the keyword passed in the *Dta* parameter, if there is one

Command:	*Cmd* value:	Action:
		exact match. If there is more than one match or no match, displays the Index tab.
PARTIAL KEY	261	Displays the topic found in the keyword list that matches the partial keyword passed in the *Dta* parameter, if there is one exact match. If there is more than one match or no match, displays the Index tab.
QUIT	2	Informs the Help application that Help is no longer needed. If no other applications have asked for Help, Windows closes the Help application.
SETCONTENTS	5	Selects the contents topic. Windows will display this topic if the user selects the Contents command and no .CNT file exists.

Examples:

```
' Display the topic mapped to context id. 400
WinHelp "c:\Program Files\AppL\cnf.hlp", 1, 400, ""
' Display the Help Topics dialog for the help file.
WinHelp "c:\My Documents\Help Projects\T27Plus32\T27Plus32.hlp", 11, 0, ""
' Display the Help Topics dialog of the Windows Help-on-Help file.
WinHelp "", 4, 0, ""
' Search for and display the "edit route" topic in the help file.
WinHelp "c:\Program Files\KMSystems\UTSPlus32\1.0\qp-cnf.hlp", 257, 0, "edit route"
```

With Statement

Execute a series of statements on a single object or user-defined type.

Format:

With *object*
[*statement(s)*]
End With

The **With** statement allows you to perform a series of commands or statements on a particular object without referring to the name of that object again. **With** statements can be nested by putting one **With** block within another **With** block. You will need to fully specify any object in an inner **With** block to any member of an object in an outer **With** block.

Related Topics: Do...Loop, While...Wend

Example:

```
' This sample shows some of the features of
' user defined types and the with statement.

Type type1
    a As Integer
    d As Double
    s As String
End Type

Type type2
    a As String
    o As type1
End Type

Dim type1a As type1
Dim type2a As type2

Sub Main ()

    With type1a
      .a = 65
      .d = 3.14
    End With
```

```
    With type2a
      .a = "Hello, world"
      With .o
        .s = "Goodbye"
      End With
    End With
    type1a.s = "YES"
    MsgBox type1a.a
    MsgBox type1a.d
    MsgBox type1a.s
    MsgBox type2a.a
    MsgBox type2a.o.s

End Sub
```

Write # Statement

Write and format data to a sequential file that must be opened in output or append mode.

Format:

Write # *filenumber* [,*parameterlist*]

A comma-delimited list of the supplied parameters is written to the indicated file. If no parameters are present, the "newline" character is all that will be written to the file.

Related Topics: Close, EOF, Input, Line Input, Open, Print #

Example:

```
Sub Main ()

    Open "TESTFILE" For Output As #1 ' Open to write file.
    userData1$ = InputBox ("Enter your own text here")
    userData2$ = InputBox ("Enter more of your own text here")
    Write #1, "This is a test of the Write # statement."
    Write #1,userData1$, userData2
    Close #1

    Open "TESTFILE" for Input As #2  ' Open to read file.
    Do While Not EOF(2)
        Line Input #2, FileData      ' Read a line of data.
        Print FileData               ' Construct message.

    Loop
    Close #2                         ' Close all open files.
    MsgBox "Testing Print Statement" ' Display message.
    Kill "TESTFILE"                  ' Remove file from disk.
End Sub
```

Year Function

Return an integer between 100 and 9999 that is the portion of the *date* parameter representing a year.

Format:

Year(*date*)

The *date* parameter is any string expression that can represent a date.

The returned integer represents the year of the *date* parameter.

If *date* is a Null, this function returns a Null.

Related Topics: Date, Day, Format, Hour, Minute, Month, Now, Second, Weekday

Example:

```
ThisYear = Year(Now)
```

Constants

Predefined Constants

This topic contains all constants and their equivalent values that are predefined when a script is invoked.

Note: Since variables do not have to be declared prior to first reference, use an **Option Explicit** statement (external to the procedure) when using constants to assure that they are spelled correctly.

Double-Click Action:

Constant	Value	Action
DBLCLICKNONE	0	None
DBLCLICKTRANSMIT	1	Transmit
DBLCLICKRUNSCRIPT	2	Run Script

Keyboard States:

Constant	Value	Description
KEYBOARDUNLOCKED	0	Keyboard Unlocked
KEYBOARDLOCKED	1	Keyboard Locked

Message Box Constants:

Constant	Value	Description
MsgBox Buttons:		
MB_OK	0	OK button only.
MB_OKCANCEL	1	OK and Cancel buttons.
MB_ABORTRETRYIGNORE	2	Abort, Retry and Ignore buttons.
MB_YESNOCANCEL	3	Yes, No and Cancel buttons.
MB_YESNO	4	Yes and No buttons.
MB_RETRYCANCEL	5	Retry and Cancel buttons
MsgBox Icons:		
MB_ICONSTOP	16	Critical message.  For Windows 95, display:
MB_ICONQUESTION	32	Warning query.  For Windows 95, display:
MB_ICONEXCLAMATION	48	Warning message.  For Windows 95, display:
MB_ICONINFORMATION	64	Information message.  For Windows 95, display:
MsgBox Defaults:		
MB_APPLMODAL	0	Application Modal Message Box. The user must respond to the message before continuing work in the current application (Default).
MB_DEFBUTTON1	0	First button is default.
MB_DEFBUTTON2	256	Second button is default.
MB_DEFBUTTON3	512	Third button is default.
MB_SYSTEMMODAL	4096	System Modal. All applications are suspended until the user responds to the message box.
MsgBox return values:		
IDOK	1	OK button pressed.
IDCANCEL	2	Cancel button pressed.
IDABORT	3	Abort button pressed.
IDRETRY	4	Retry button pressed.
IDIGNORE	5	Ignore button pressed.

Constant	Value	Description
IDYES	6	Yes button pressed.
IDNO	7	No button pressed.

Message Waiting States:

Constant	Value	Description
NOMESSAGEWAITING	0	Message Not Waiting
MESSAGEWAITING	1	Message Waiting

Print Font Style Types:

Constant	Value	Description
fsNormal	0	Normal font
fsFontBold	1	Bold font
fsFontItalic	2	Italic font
fsFontUnderline	4	Underlined font
fsFontStrikeThru	8	Strikethrough font

Screen Attributes:

Constant	Value	Description
ATTR_NORMAL	0	Normal
ATTR_FIELD	1	Start of Field (set on first position of field)
ATTR_TAB	2	Tab Stop (at start of field only)
ATTR_CHANGED	4	Data Field Changed Flag
ATTR_PROTECTED	8	Protected
ATTR_VIDEO_OFF	16	Video Off
ATTR_NUMERIC	32	Numeric Only Input
ATTR_ALPHA	64	Alphabetic Only Input
ATTR_BLINK	128	Blinking
ATTR_RIGHT	256	Right Justified Data
ATTR_LOWINT	512	UTS Low Intensity
ATTR_REV	1024	Reverse Video

T27 Keys:

Constant	Integer
TK_ARROWDN	249
TK_ARROWLEFT	247
TK_ARROWRIGHT	248
TK_ARROWUP	246
TK_BACKSPACE	8
TK_BACKTAB	196
TK_BOUND	218
TK_CARRIAGERTN	13
TK_CLRALLVTAB	16442
TK_CLREOL	134
TK_CLREOP	135
TK_CLRFORMS	159
TK_CLRHOME	128
TK_COPY	16432
TK_CTRL	164
TK_CUT	16431
TK_DBLZERO	234
TK_DELCHAR	132
TK_DELCHARPAGE	16425
TK_DELLINE	133
TK_HOME	174
TK_INSCHAR	130
TK_INSCHARPAGE	16424

Constant	Integer
TK_INSLINE	131
TK_LOCAL	168
TK_LOCKCTRL	165
TK_LOGICALEOL	16415
TK_MARK	217
TK_MOVELINEDOWN	138
TK_MOVELINEUP	139
TK_NEXTPAGE	253
TK_PASTE	16434
TK_PREVPAGE	252
TK_PRINTALL	157
TK_PRINTUNPROT	156
TK_RECALL	214
TK_RECEIVE	170
TK_ROLLDN	136
TK_ROLLUP	137
TK_SETFORMS	158
TK_SPECIFY	166
TK_STORE	213
TK_TAB	198
TK_TOGGLEFORMS	141
TK_TOGGLETAB	16441
TK_TRANSMIT	172
TK_TRANSMITLINE	16428
TK_TRIPZERO	236
TK_UPPERONLYON	210
TK_UPPERONLYOFF	211
TK_WRITEESC	16426
TK_WRITEETX	3
TK_WRITEGS	16427

UTS Keys:

Constant	Value
UK_BACK_SPACE	95
UK_CURSOR_DOWN	6
UK_CURSOR_LEFT	7
UK_CURSOR_RETURN_KEY	32
UK_CURSOR_RIGHT	8
UK_CURSOR_TO_END_LINE	66
UK_CURSOR_TO_HOME	23
UK_CURSOR_TO_START_LINE	65
UK_CURSOR_UP	9
UK_DELETE_IN_DISPLAY	11
UK_DELETE_IN_LINE	12
UK_DELETE_LINE	10
UK_ERASE_CHAR	67
UK_ERASE_DISPLAY	14
UK_ERASE_TO_END_DISPLAY	15
UK_ERASE_TO_END_FIELD	16
UK_ERASE_TO_END_LINE	17
UK_FKEY_1	43
UK_FKEY_2	44
UK_FKEY_3	45
UK_FKEY_4	46
UK_FKEY_5	47
UK_FKEY_6	48
UK_FKEY_7	49
UK_FKEY_8	50
UK_FKEY_9	51
UK_FKEY_10	52
UK_FKEY_11	53
UK_FKEY_12	54
UK_FKEY_13	55

Constant	Value
UK_FKEY_14	56
UK_FKEY_15	57
UK_FKEY_16	58
UK_FKEY_17	59
UK_FKEY_18	60
UK_FKEY_19	61
UK_FKEY_20	62
UK_FKEY_21	63
UK_FKEY_22	64
UK_INSERT_IN_DISPLAY	25
UK_INSERT_IN_LINE	26
UK_INSERT_LINE	24
UK_KEYBOARD_UNLOCK	27
UK_LINE_DUP	28
UK_MSG_WAIT	29
UK_PRINT_KEY	30
UK_PRINT_ENTIRE_SCREEN	69
UK_SOE	3
UK_TAB_BACK	33
UK_TAB_FORWARD	34
UK_TAB_SET	35
UK_TRANSMIT_KEY	36

Window States:

Constant	Value	Description
WSNORMAL	0	Window Normal
WSMINIMIZED	1	Window Minimized
WSMAXIMIZED	2	Window Maximized

UTS Control Characters, Escape Sequences and FCCs

UTS Control Characters

Code	Meaning
ACK	Acknowledge
BEL	Audible alarm (processor message)
BS	Backspace
CAN	Cancel (search/rewind)
CR	Carriage Return (cursor return)
DLE	Data link escape (control character code)
DC1	Device control 1 (transmit)
DC2	Device control 2 (Print)
DC3	Device control 3
DC4	Device control 4 (lock keyboard)
EM	End of medium
ENQ	Enquiry (status poll)
EOT	End of transmission
ESC	Escape
ETB	End-of-transmission-block (more to come)
ETX	End of text
FF	Form feed
FS	File separator (start blink marker)
GS	Group separator (end blink marker)
HT	Horizontal tab
LF	Line feed
NAK	Negative acknowledgment
NUL	Null
RS	Record separator (start of entry)
SI	Shift In
SO	Shift Out
SOH	Start of header
STX	Start of text
SUB	Substitute character (protected fields)
SYN	Synchronizing character
US	Unit separator
VT	Vertical tab (vertical tab/return address)

UTS Control and Escape Sequences

The following table contains a listing of control and escape sequences:

Sequence	Description
DC3 DC3	Offline Load
EM M N	Basic FCC Sequence without Address
EM O M N C1 C2	Expanded Color FCC Sequence without Address
ESC "	Select Loadable Character Set
ESC a	Erase Unprotected Data
ESC b	Erase to End of Line
ESC c	Delete In Line
ESC C	Delete In Display
ESC d	Insert In Line
ESC D	Insert In Display
ESC DC1	Transmit All Fields
ESC DC2	Print Transparent
ESC DC4	Lock Keyboard
ESC e	Cursor-to-Home
ESC E	Transfer Changed Fields
ESC f	Scan Up
ESC F	Transfer Variable Fields
ESC g	Scan Left

Sequence	Description
ESC G	Transfer All Fields
ESC h	Scan Right
ESC H	Print Form
ESC HT	Tab Stop Set
ESC i	Scan Down
ESC j	Insert Line
ESC k	Delete Line
ESC K	Erase to End of Field
ESC L	Keyboard Unlock
ESC M	Erase Display
ESC o	Control Page Access
ESC P	Call Error Log
ESC Q	Initiate Confidence Test
ESC R	Clear Error Log
ESC SO 0410	Character Set Load (UTS40)
ESC SO 0411 through 0419, 041:, 041;	Kanji Set Load
ESC SO 041?	Character Set Load (UTS 30)
ESC SO 1400	Recall Request
ESC SO 9000	Successful Load (Terminal Response)
ESC SO 904*	Load Error Response (Terminal Response)
ESC SO :003	Block Response (Terminal Response)
ESC SO ;007	ID Response
ESC SO 0210	Remote Load Block
ESC SP	End Character Set Selection/Emphasis
ESC 20 through 2F	Create or Replace Emphasis
ESC t	Transmit Changed Fields
ESC T	Send Cursor Address
ESC u	Clear Changed Bits
ESC U	Return Loadable Character Set ID
ESC VT Y X SI	Cursor Positioning
ESC w	Clear FCC
ESC W	Send Graphic Pointer Address
ESC W X, Y	Graphic Pointer Address (Terminal Response)
ESC y	Line Duplication
ESC Y	Add Emphasis
ESC z	Backward Tab
ESC Z a	Enter Graphic Mode
ESC Z b	Enter Alphanumeric Mode
ESC Z c	Enter Merge Mode
ESC Z f through o	Graphic Color Control
SO DC3 p	Kanji Shift In
SO DC3 q	Kanji Shift Out
US Y X M N	Basic FCC Sequence with Address
US Y X O M N C1 C2	Expanded Color FCC Sequence with Address

UTS Field Control Characters

ASCII Characters Used as M in the UTS 400-Compatible FCC Sequence (EM M N and US Y X M N):

Bit 3 Tab Status	Bit 2 Changed Data	Bit 1,0 Intensity	Character
Tab Stop	Field Changed	Normal Intensity	0
Tab Stop	Field Changed	Video Off	1
Tab Stop	Field Changed	Low Intensity/Reverse Video	2
Tab Stop	Field Changed	Blinking	3
Tab Stop	Field Not Changed	Normal Intensity	4
Tab Stop	Field Not Changed	Video Off	5
Tab Stop	Field Not Changed	Low Intensity/Reverse Video	6
Tab Stop	Field Not Changed	Blinking	7
No Tab Stop	Field Changed	Normal Intensity	8
No Tab Stop	Field Changed	Video Off	9
No Tab Stop	Field Changed	Low Intensity/Reverse Video	:
No Tab Stop	Field Changed	Blinking	;

Bit 3 Tab Status	Bit 2 Changed Data	Bit 1,0 Intensity	Character
No Tab Stop	Field Not Changed	Normal Intensity	<
No Tab Stop	Field Not Changed	Video Off	=
No Tab Stop	Field Not Changed	Low Intensity/Reverse Video	>
No Tab Stop	Field Not Changed	Blinking	?

ASCII Characters Used as N in the UTS 400-Compatible FCC Sequence (EM M N and US Y X M N):

Bit 2 Justification	Bit 1, 0 Type of Data Entry	Character
Normal Field	Any	0
Normal Field	Alpha	1
Normal Field	Numeric	2
Normal Field	Protected	3
Right-Justified	Any	4
Right-Justified	Alpha	5
Right-Justified	Numeric	6
Right-Justified	Protected	7

ASCII Characters Used as M in the Expanded FCC Sequence (EM O M N C1 C2 and US Y X O M N C1 C2):

Bit 3 Tab Status	Bit 2 Changed Data	Bit 1 Intensity	Bit 0 Video	Bit 5 Emphasis Not Protected	Bit 5 Emphasis Protected
Tab Stop	Field Changed	Normal Intensity	Video On	@	`
Tab Stop	Field Changed	Normal Intensity	Video Off	A	a
Tab Stop	Field Changed	Low Intensity	Video On	B	b
Tab Stop	Field Changed	Low Intensity	Video Off	C	c
Tab Stop	Field Not Changed	Normal Intensity	Video On	D	d
Tab Stop	Field Not Changed	Normal Intensity	Video Off	E	e
Tab Stop	Field Not Changed	Low Intensity	Video On	F	f
Tab Stop	Field Not Changed	Low Intensity	Video Off	G	g
No Tab Stop	Field Changed	Normal Intensity	Video On	H	h
No Tab Stop	Field Changed	Normal Intensity	Video Off	I	i
No Tab Stop	Field Changed	Low Intensity	Video On	J	j
No Tab Stop	Field Changed	Low Intensity	Video Off	K	k
No Tab Stop	Field Not Changed	Normal Intensity	Video On	L	l
No Tab Stop	Field Not Changed	Normal Intensity	Video Off	M	m
No Tab Stop	Field Not Changed	Low Intensity	Video On	N	n
No Tab Stop	Field Not Changed	Low Intensity	Video Off	O	o

ASCII Characters Used as N in the Expanded Sequence (EM O M N C1 C2 and US Y X O M N C1 C2):

Bit 3 Blinking	Bit 2 Justification	Bit 1, 0 Type of Data Entry	Bit 4 Normal Intensity	Bit 4 Reverse Video
No Blink	Normal Field	Any	@	P
No Blink	Normal Field	Alpha	A	Q
No Blink	Normal Field	Numeric	B	R
No Blink	Normal Field	Protected	C	S
No Blink	Right-Justified	Any	D	T
No Blink	Right-Justified	Alpha	E	U
No Blink	Right-Justified	Numeric	F	V

Bit 3 Blinking	Bit 2 Justification	Bit 1, 0 Type of Data Entry	Bit 4 Normal Intensity	Bit 4 Reverse Video
No Blink	Right-Justified	Protected	G	W
Blinking	Normal Field	Any	H	X
Blinking	Normal Field	Alpha	I	Y
Blinking	Normal Field	Numeric	J	Z
Blinking	Normal Field	Protected	K	[
Blinking	Right-Justified	Any	L	\
Blinking	Right-Justified	Alpha	M	]
Blinking	Right-Justified	Numeric	N	^
Blinking	Right-Justified	Protected	O	__

ASCII Characters User as O in the Expanded Color FCC Sequence (EM O M N C1 C2 and US Y X O M N C1 C2):

Hex Code	ASCII	Meaning
20	SP	One color operator (C1) with the character color in the lease significant 3 bits and the background color in the next 3 bits.
21	!	One color operator (C1) with character color only.
22	"	One color operator (C1) with background color only.
23	#	Two color operators. The first (C1) has character color (4 bits), and the second (C2) has background color (4 bits).

For C1, bits 0-2 define the character color; bits 3-5 define the background color; bit 6 is always 1 for ASCII; and bit 7 is always 0. Bits 0 and 3 are used for red, bits 1 and 4 for green, and bits 2 and 5 for blue, as shown:

Bits 0 & 3	= 0	No red
	= 1	Red
Bits 1 & 4	= 0	No green
	= 1	Green
Bits 2 & 5	= 0	No blue
	= 1	Blue

These three bits combine to provide eight colors.

Color bit options:

Bits 5 4 3 or 2 1 0	Color
0 0 0	Black
0 0 1	Red
0 1 0	Green
0 1 1	Yellow
1 0 0	Blue
1 0 1	Magenta
1 1 0	Cyan
1 1 1	White

Example:
Yellow character on black background (000011) plus the ASCII bit (01000011) is a hex 43 or an ASCII C.

Characters Used for C1 in the Expanded Color FCC Sequence:

Background—-> Character	Black	Red	Green	Yellow	Blue	Magenta	Cyan	White
Black	@	H	P	X	`	h	p	x
Red	A	I	Q	Y	a	i	q	y
Green	B	J	R	Z	b	j	r	z
Yellow	C	K	S	[	c	k	s	{
Blue	D	L	T	\	d	l	t	*
Magenta	E	M	U	]	e	m	u	}
Cyan	F	N	V	^	f	n	v	~
White	G	O	W	_	g	o	w	?

Decimal characters representing Y (row) and X (column) coordinates:

Row/Column	Character
1	SP
2	!
3	"
4	#
5	$
6	%
7	&
8	'
9	(
10	)
11	*
12	+
13	,
14	-
15	.
16	/
17	0
18	1
19	2
20	3
21	4
22	5
23	6
24	7
25	8
26	9
27	:
28	;
29	<
30	=
31	>
32	?
33	@
34	A
35	B
36	C
37	D
38	E
39	F
40	G
41	H
42	I
43	J
44	K
45	L
46	M

Row/Column	Character
47	N
48	O
49	p
50	Q
51	R
52	S
53	T
54	U
55	V
56	W
57	X
58	Y
59	Z
60	[
61	\
62	]
63	^
64	_
65	`
66	a
67	b
68	c
69	d
70	e
71	f
72	g
73	h
74	i
75	j
76	k
77	l
78	m
79	n
80	o

UTS Special Emphasis Character Codes

ASCII Character	Hex Code	Special Emphasis
SP	20	No Emphasis
!	21	Column Separator
"	22	Loadable Character Set
#	23	Loadable Character Set + Column Separator
$	24	Underscore
%	25	Column Separator + Underscore
&	26	Underscore + Loadable Character Set
'	27	Underscore + Loadable Character Set + Column Separator
(	28	Strikethrough
)	29	Column Separator + Strikethrough
*	2A	Strikethrough + Loadable Character Set
+	2B	Strikethrough + Loadable Character Set +Column Separator
'	2C	Underscore + Strikethrough
-	2D	Column Separator + Underscore + Strikethrough
.	2E	Strikethrough + Underscore + Loadable Character Set
/	2F	Strikethrough + Underscore + Loadable Character Set + Column Separator

Index

A

Abs Function 39
Accessing an Object 31
ActivateScreen Subroutine 39
AppActivate Statement 39
Arithmetic Operators 35
  summary 35
Arrays 19
Asc Function 40
Atn Function 40
Automation 31

B

Beep Statement 40
Begin Dialog Statement 40
ByRef and ByVal 15
ByVal 15

C

Call Statement 42
Calling Procedures in DLLs 16
Cancel button 23
  Dialog 23
CancelButton Statement 43
CBool Function 43
CDbl Function 44
ChDir Statement 44
ChDrive Statement 44
Check Boxes 25
CheckBox Statement 45
Choose Function 45
Chr Function 45
CInt Function 46
Class 32
CLng Function 46
Close Statement 46
Color Selection 7
ComboBox Statement 47
Comments 9
Concatenation 11
Const Statement 47
Constant Names 9
Continuation 9
Control and Escape Sequences 135
Control Characters 135
Control Structures 13
CopyToClipboard Subroutine 48
Cos Function 48
CreateObject Function 48
CSng Function 49
CStr Function 49
CurDir Function 50
CurrentPage Function 50
CurrentScreen Function 50
CVar Function 51

D

Data Types 11, 36
  summary 35
Data Types Operators and Precedence 36
Date Function 51
DateSerial Function 51
DateValue Function 51
Day Function 52
Declaration of Variables 12
Declare Statement 52
Dialog Box Controls 28
Dialog Function 28, 53
  Syntax 28
Dialog Support 23
Dim 19
Dim Statement 54
Dir Function 54
DlgEnable Statement 57
DlgText Statement 55
DlgVisible Statement 55
Do Loops 13
Do...placeLoop Statement 55
Double 11, 35
download 72
Drop-down List Boxes 24
DropListBox Statement 56

E

Editor 1
Editor Properties 5
End Statement 56
EnterText Subroutine 58
EOF Function 58
Erase Statement 58
Exit Statement 59
Exp Function 59
eXpress Plus Script Editor 1

F

Field Control Characters 136
File Input/Output 17
FileCopy Function 59
FileLen Function 60
FileOpenDialog Function 60
FileSaveDialog Function 60
Fix Function 61
For ... Next placeLoop 13

For Each...Next Statement 61
For...Next Statement 61
Format Function 62
FreeFile Function 67
Function Statement 68
Functions/Statements/Subroutines - Alphabetical Listing 37
Functions/Statements/Subroutines by Category 37
G
Get Statement 68
GetCursorCol Function 68
GetCursorRow Function 69
GetLastMsg Function 69
GetObject Function 69
GetPage Subroutine 69
GetScreenAttribute Function 70
GetScreenColor Function 72
GetScreenLine Function 72
GetScreenText Function 74
GetUserParam Function 77
GetWindowState Function 77
Global array 19
Global Statement 78
GoTo 13
GoTo Statement 79
GoToPage Subroutine 79
Group boxes 27
GroupBox Statement 80
H
Hex Function 80
HoldMessages Subroutine 80
Hour Function 80
I
If and Select Statements 13
If...Then...Else Statement 81
Input Function 82
InputBox Function 82
InStr Function 82
Int Function 83
Integer 11, 35
IsArray Function 83
IsDate Function 83
IsEmpty Function 83
IsNull Function 84
IsNumeric Function 84
IsObject Function 84
K
KeyboardState Function 85
Kill Statement 85
L
Language Reference 35
LBound Function 85
LCase Function 86
Left Function 86
Len Function 87
Let Statement 87
Line Continuation Character 9
Line Input # Statement 87
List Boxes 24
ListBox Statement 88
LoadScreen Subroutine 89
Local array 19
LOF Function 89
Log Function 89
Logical Operators 35
Long 11, 35
LTrim Function 122
M
Making Applications Work Together 33
MarkBlock Subroutine 90
Message Box Constants 131
MessageWaitingState Function 90
Methods 32
Mid Function 90
Minute Function 91
MkDir Statement 91
Month Function 91
MsgBox Function 92
MsgBox Statement 92
N
Name Statement 93
Naming Conventions 15
Now Function 93
Numbers 9
O
Object 31
Accessing 31
Class 32
What is 32
Oct Function 93
OK button 23
Dialog 23
OKButton Statement 94
OLE Automation 31, 32
OLE Automation and Word 6.0 example 33
OLE Fundamentals 32
OLE Object 32
On Error Statement 94
Open Statement 96
Operators 36
Option Base 19
Option Base Statement 97

Option buttons 27
Option Explicit Statement 97
OptionButton Statement 97
OptionGroup Statement 98
Other Data Types 11
P
PasteFromClipboard Subroutine 98
Precedences 36
Predefined Constants 131
Print # Statement 98
Print Font Style Types 131
Print Statement 100
PrintBeginDoc Subroutine 100
PrintEndDoc Subroutine 101
PrintMoveTo Subroutine 101
PrintNewPage Subroutine 101
PrintPageHeight Function 101
PrintPageWidth Function 101
PrintRect Subroutine 102
PrintSelect Function 102
PrintSetFont Subroutine 102
PrintSetFontSize Subroutine 102
PrintSetFontStyleSubroutine 103
PrintSetOrientation Subroutine 103
PrintTextHeight Function 104
PrintTextWidth Function 104
Properties 32
PushButton Statement 104
Put Statement 105
Q
Qperator Precedence 35
R
Randomize Statement 105
Receive Function 106
ReDim Statement 106
RefreshScreen Subroutine 106
Relational Operators 35
Rem 9
Rem Statement 106
Right Function 107
RmDir Statement 107
Rnd Function 108
RTrim Function 122
S
SaveScreen Subroutine 108
Scope of Variables 12
Screen Activate 39
ScreenAvailable Function 108
ScreenOpen Function 109
Script Editor 1
Second Function 109
Seek Function 109
Seek Statement 109
Select Case Statement 110
Select Statements 13
SendKeys Statement 111
Set Statement 112
SetCursor Subroutine 112
SetDblClickAction Subroutine 113
SetScreenText Subroutine 113
SetSignOnResult Subroutine 114
SetWindowState Subroutine eXpress Plus 114
Sgn Function 114
Shell Function 114
Sin Function 115
Single 11, 35
Space Function 115
Special Emphasis Character Codes 140
Sqr Function 115
Statements 9
Static array 19
Static Statement 115
Stop Statement 116
StorePage Subroutine 116
Str Function 117
StrComp Function 117
String 11, 35
String Function 118
Sub Statement 118
Subroutine and Function Naming Conventions 15
SwitchToolbar Subroutine 119
T
T27 Keys 131
T27Key Subroutine 119
Tan Function 120
Text 26
Text Boxes 26
Text Statement 120
TextBox Statement 120
Time Function 121
Time Statement 121
Timer Event 121
TimeSerial Function 122
TimeValue Function 122
Trim Function 122
Type Statement 123
U
UBound Function 124
UCase Function 124
UnholdMessages Subroutine 125
upload 87
User Defined Types 21

UTS Control and Escape Sequences 135
UTS Control Characters 135
UTS Field Control Characters 136
UTS Keys 131
UTS Special Emphasis Character Codes 140
UTS States 131
UTSKey Subroutine 125
V
Val Function 126
Values 15
  Passing 15
Variable and Constant Names 9
Variable Types 11
Variables 12, 21
  Decalration of 12
  Scope of 12
Variant 11, 35
Variants and Concatenation 11
VarType Function 126
W
Wait Subroutine 126
WaitForSpecificString Function 127
WaitForString Function 127
Weekday Function 127
What is an OLE Object? 32
What is OLE Automation? 31
While placeLoop 13
While...Wend Statement 127
WinHelp Subroutine 128
With Statement 129
Write # Statement 130
Y
Year Function 130