Welcome to Formula One

Welcome to the **Borland Edition of Formula One** from Visual Components. Visual Components develops and markets a full range of tools for the component-based developer. Our product portfolio includes best of class tools for data analysis, charting, rich text and spell checking. We offer a royalty-free runtime license for all of our OCX products.

The **Professional Edition of Formula One** supports features not found in this version of Formula One. Attempting to use these features through the user interface, or by using OCX properties and methods will result in an error.

For additional information regarding the **Professional Edition of Formula One**, including Upgrade information, contact Visual Component Sales at 800-884-8665.

You can also contact Visual Components Sales any of the following ways:

- **By FAX**. You can contact us by FAX at (913)599-6597.
- **On the Internet.** Contact us at:

  World Wide Web - http://www.visualcomp.com

  Electronic Mail - sales@visualcomp.com
- **Via BBS.** You can contact us through our 24-hour bulletin board service at (913)599-6713.
- **Via CompuServe.** You can contact us through CompuServe - 74774,443.
  Visual Components also maintains a section in the MS Windows Components A+ Forum on CompuServe. These sections are used for peer to peer support and the distribution of example projects, maintenance releases, etc. To reach the Visual Components section, type:

  GO VISTOOLS

  When communicating with Visual Components via the CompuServe forums, include our account number with all messages. This assures that your message receives prompt attention.
- **By mail.** Address your correspondence to:
  Visual Components, Inc.
  15721 College Blvd.
  Lenexa, Kansas 66219
  or
  Visual Components Europe
  Lenexa House
  11 Eldon Way
  Paddock Wood, Kent
  England TN12 6BE
  Tel: +44 1892 834343
  Fax: +44 1892 835843
  BBS: +44 1892 835579

Reaching Our Technical Support Department

You can receive support directly from Visual Components Technical Support engineers by purchasing a support plan.

- You can call the support line directly at (913) 599-6500 and pay $10 per call.
- You can call (800) 884-8665 to purchase an annual support contract for $249 per developer per year. This Gold Support plan gives you access to a number of on-line information sources as well as the following:

1. Unlimited technical support calls.
2. When calling technical support, your call is placed in a priority queue for faster service.
3. Expedited open case resolution. For calls that cannot be resolved immediately, an action plan is developed within 24 hours of the customer's inquiry, and the customer is updated every 48 hours on continuing open cases.
4. Automatic beta program enrollment.

New Features in the Professional Edition of Formula One 3.0

Version 3.0 of Formula One provides a variety of new features. Among the new features provided by this version:

- **OCX Support.** Formula One functionality is available through OCX properties and methods.
- **Enhanced Excel Support.** Version 3.0 provides a number of new and improved features that make Formula One even more Excel-compatible such as:
- **Excel 5.0 support.** Formula One can now read and write Excel 5.0 files.
- **Workbooks.** Support for Excel-type multiple sheet workbooks. This means that Formula One supports 3-D cell referencing (Sheet1:Sheet5!A1:C10.) Additionally, many worksheet editing methods such as **InsertRange**, **DeleteRange**, **MoveRange**, etc. can work on multiple worksheets at the same time. This time-saving feature keeps you from having to perform the same over and over in your code to modify a series of worksheets.
- **Cell References.** Formula One can now parse range references with . or . . in addition to :(A1:B2 or A1.B2 or A1..B2)
- **Entering Text.** You can now use a leading apostrophe (') to enter numbers as text.
- **New Worksheet Functions.** The worksheet functions SUMIF, COUNTIF, CONCATENATE, ROUNDUP and ROUNDDOWN have been added to this version in enhance Excel 5 compatibility.
- **Connection to ODBC databases.** By using the new **ODBCConnect**, **ODBCQuery**, and **ODBCDisconnect** methods, you can retrieve information from an ODBC database and place it in your Formula One workbook.
- **New Built-In Chart Object.** You can now select a range of data, click on the chart tool, and draw a First Impression chart on your worksheet.
- **Built-In DLL**. In this version of Formula One, the DLL is built directly into the OCX. The only file you have to distribute with your application is VCF132.OCX. (MFCANS32.DLL and OC30.DLL must also be present.)
- **Validation Rules.** You can now create validation rules to ensure users enter appropriate data in cells. Validation rules include a formula against which to test the cell entry and a message to be displayed if validation fails. There is also a new **ValidationFailed** event to allow you to program a response when validation fails.
- **AutoFill Feature.** Formula One now provides the ability to autofill worksheet cells with common lists such as month names and days of the week. You can also add your own autofill lists.
- **Larger Color Palette.** Formula One now offers a default color palette of 56 colors instead of 16. You also have the power to customize any color on the palette.
- **Column Widths.** You can now specify column widths in twips or character units. Twips are a standard unit of measure that equal 1/1440th of an inch.
- **Hiding Columns and Rows.** You can now control the display status of columns and rows.
- **Autosizing.** Autosizing an entire column now affects the column header as well.
- **Displaying Type Markers.** You can now display markers on your worksheet that identify the type of data in each cell.
- **New Events.** Version 3.0 now provides additional events for working with objects: **ObjValueChanged**, **ObjGotFocus**, and **ObjLostFocus**. There are also additional events for capturing user interaction with a mouse: **MouseDown**, **MouseMove** and **MouseUp**. Finally, a new **Modified** event provides you additional means for responding to user changes.
- **Scroll Bar Improvements.** You can now set the scroll bars to scroll to the last row or column in a worksheet.
- **New Modified Property.** Used with the new Modified event, provides more flexibility in determining and responding to user changes.
- **Enhanced Printing Support.** Formula One can now read and write the Windows API PrintDevMode structure.

- **User-defined Names.** You can now enumerate the uer-defined names in a workbook.
- **Dropdown List Box Enhancements.** A dropdown list box can now set a cell value to the text of a selection rather than just the selection index.
- **Customizable Mousepointer.** You can now control the shape and behavior of the mouse pointer when it is over the Formula One control.

Adding the OCX to Your Application

The process you use to add an OCX to your application varies slightly from one development environment to another. In most cases it consists of:

- Adding the OCX control to your project.
- Selecting the control's tool from the tool bar and drawing the control on a form or in a window.

Consult your development environment documentation for specific steps to add a control to your application.

Distributing Formula One Applications

Please read the license agreement that was shipped with this package. You are bound by the licensing restrictions contained in that document.

Redistributing Files

You can use all the files accompanying this product for development of an application. You can redistribute the run time version of the software according to the terms of the license agreement.

You can ship the following files with your application:

File
VCF132.OCX
MFCANS32.DLL
OC30.DLL
MSVCRT20.DLL

Note These DLLs must be present on a system for Formula One to function correctly. In addition if you intend to connect to an ODBC database, ODBC32.DLL must be present. This file is not distributed with Formula One.

The Formula One Control

Each time you create a Formula One workbook control in your application, you create an object that gives you a specific view into a workbook.

The workbook object has properties and methods you can use to change its appearance and behavior. Properties allow you to change the attributes of the object. For example, you can set properties to hide the horizontal scroll bar, or to change the height of a row or the width of a column.

Methods allow you to control the activities of the object. For example, you can use methods to recalculate the workbook, or invoke one of the Formula One dialog boxes for user input.

There are times when methods change the value of a property. For example, when you use the **DeleteSheets** method to delete one or more worksheets from a workbook, the value of the **NumSheets** property is changed to reflect the new number of worksheets.

Using Properties

Using Methods

Handling Errors

Using Properties

With Formula One properties, you can perform a variety of tasks, such as hiding and displaying elements of a worksheet, selecting cells and ranges, and setting print margins. Many properties allow you to perform complex tasks with very little coding.

When you use properties in your code, you can either set, or change the value of, the property, or retrieve the property's current value. Most properties are read-write. This means you can set and retrieve them. However, some properties are read-only. This means you can retrieve their current value, but you cannot set the property to change the value.

When you refer to a property, you must list the object name first, followed by a period, and then the name of the property. In the following example, the **AllowArrows** property of the object called F1Book1 is set to True. This means the user can use the arrow keys to reposition the active cell.

```
F1Book1.AllowArrows = True
```

Property Data Types

Setting Properties

Retrieving Property Values

Properties that require an index

Property Data Types

Before setting or retrieving the value of a property, you must know its data type. Generally, a property's value can be a numeric value, a character string, or a boolean (True|False) value. To determine the correct data type for a particular property, consult the OCX Property and Method Reference.

Setting Properties

Retrieving Property Values

Properties that require an index

Setting Properties

When you set a property, you assign it a new value. To accomplish this you use the equal sign to set the property equal to an expression that describes the new value. In the following example, the **Col** property is used to set the current column to column 5.

```
F1Book1.Col = 5
```

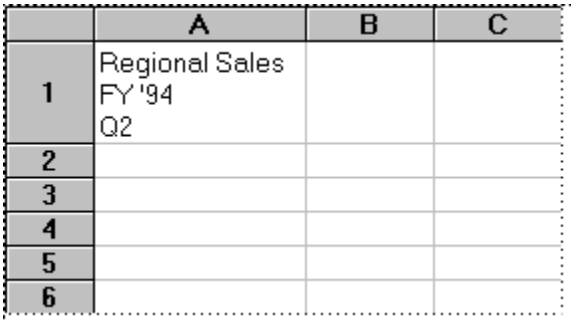
The following example shows the code required to read an Excel worksheet from disk using the **Read** method:

```
F1Book1.Read = "c:\excel\examples\amortize.xls"
```

The following example uses the **Row**, **Col**, **Number**, **Formula**, and **Text** properties to enter data in a worksheet. Numbers, 1 and 2, are entered in A1 and A2. A formula, SIN(A1) + COS(A2), is entered in A3. A text string is entered in B4.

```
F1Book1.Row = 1
F1Book1.Col = 1
F1Book1.Number = 1
F1Book1.Row = 2
F1Book1.Number = 2
F1Book1.Row = 3
F1Book1.Formula = "sin(A1) + cos(A2)"
F1Book1.Row = 4
F1Book1.Col = 2
F1Book1.Text = "The End!"
```

The following illustration shows the result of the preceding example.



Property Data Types
Retrieving Property Values
Properties that require an index

Retrieving Property Values

Retrieve a property value to determine the condition of an object before your procedure performs additional actions.

To accomplish this, use the equal sign to assign the value of the property to a variable. For example, in the following code, the data variable receives the current value of the **Entry** property, which describes the contents of the active cell.

```
Dim data As Integer

data = F1Book1.Entry
```

You can also use the current value of a property directly in an expression. In the following example, the code retrieves the value for two check box objects that ask the user whether they want to print row and column titles with their worksheet. If the result is 0, both check boxes are unchecked and the **PrintTitles** property is set to null. No titles are printed with the worksheet.

```
If F1Book1.ObjValue (1) + F1Book1.ObjValue (2) = 0 Then

    F1Book1.PrintTitles = " "

    ....

End If
```

Property Data Types

Setting Properties

Properties that require an index

Properties that require an index

Some properties take an index to identify the specific entity the represent. For example, you can use the **ColHidden** property to hide a column, but you must give it an index number to identify which column to act on. The following code hides the fourth column in the active sheet.

```
F1Book1.ColHidden(4) = True
```

Property Data Types

Setting Properties

Retrieving Property Values

Using Methods

There are two types of methods, those that take arguments, and those that do not. Methods that take no arguments initiate an action and return no value. For example, you can use the **CalculationDlg** method to display the Calculation dialog box:

```
F1Book1.CalculationDlg
```

If a method does take arguments, you must be aware of whether the method returns a value. If the method does not return a value, or you don't wish to save the returned value, the method arguments appear without parentheses as in the following example:

```
F1Book1.AddPageBreak 10
```

If you do wish to save the value returned by a method you must use the brackets. The following example returns the row number of the next row page break after row 15:

```
nextRow = F1Book1.NextRowPageBreak (15)
```

Using Properties

Handling Errors

Handling Errors

Formula One errors that occur during program execution are handled like other errors. You must provide you own error handling routines to intercept and manage errors.

Note Formula One adds 20000 to all error numbers except F1ErrorNone before reporting them to the container so that they do not conflict with the container's error numbers. To retrieve the Formula One error number you would use the following expression: F1Error = Err - 20000.

Error Constant	Number	Description
F1ErrorNone	0	Function succeeded.
F1ErrorGeneral	20001	Function failed with a non-specific error.
F1ErrorBadArgument	20002	One of the function arguments was invalid.
F1ERRORNOMEMORY	20003	Not enough memory to complete the task.
F1ErrorBadFormula	20004	The formula syntax is incorrect.
F1ErrorBufTooShort	20005	The returned result is longer than the return buffer size. A NULL string is placed in the buffer.
F1ErrorNotFound	20006	Cannot find item for which function is looking.
F1ErrorBadRC	20007	The row/column reference is invalid.
F1ErrorBadHSS	20008	Invalid view handle passed.
F1ErrorTooManyHSS	20009	Unable to create additional view handles.
F1ErrorNoTable	20010	No worksheet attached to the view.
F1ErrorUnableToOpenFile	20011	Cannot open the specified file.
F1ErrorInvalidFile	20012	Cannot read invalid file.
F1ErrorInsertShiftOffTable	20013	Insert pushes cells outside of worksheet bounds.
F1ErrorOnlyOneRange	20014	Specified command expects only one selected range.
F1ErrorNothingToPaste	20015	Nothing to paste when a paste operation was requested.
F1ErrorBadNumberFormat	20016	Invalid custom format string.
F1ErrorTooManyFonts	20017	Cannot add fonts to the table.
F1ErrorTooManySelectedRanges	20018	Cannot add selected ranges.
F1ErrorUnableToWriteFile	20019	An error occurred while writing the file.
F1ErrorNoTransation	20020	TransactCommit or TransactRollback was called without first calling TransactStart.
F1ErrorNothingToPrint	20021	No data to print in the table or selected range.
F1ErrorPrintMarginsDontFit	20022	Print margins are out of range.
F1ErrorCancel	20023	Returned if the user presses Cancel in a built-in dialog box.
F1ErrorUnableToInitializePrinter	20024	Cannot initialize the printer.
F1ErrorStringTooLong	20025	An argument to a C function specified a string that was too long.
F1ErrorFormulaTooLong	20026	Specified formula is too long.
F1ErrorUnableToOpenClipboard	20027	Cannot open the Windows clipboard.
F1ErrorPasteWouldOverflowSheet	20028	The paste operation extends beyond the last

		row or last column of the worksheet.
F1ErrorLockedCellsCannotBeModified	20029	Attempted to modify cells that are locked with protection enabled.
F1ErrorLockedDocumentCannotBeModified	20030	Attempted to modify a document that has protection enabled.
F1ErrorInvalidName	20031	Specified a user defined name that is invalid.
F1ErrorCannotDeleteNameInUse	20032	Attempted to delete a user defined name that is currently in use by a formula.
F1ErrorUnableToFindName	20003	Could not find specified user defined name.
F1ErrorNoWindow	20034	Invalid request for a worksheet that is not attached to a window.
F1ErrorSelection	20035	Invalid request for the current selection.
F1ErrorTooManyObjects	20036	Unable to create more objects.
F1ErrorInvalidObjectType	20037	The type of object selected is invalid for the operation that is attempted.
F1ErrorObjectNotFound	20038	The specified object cannot be found.
F1ErrorInvalidRequest	20039	The attempted operation is currently invalid.
F1ErrorBadValidationRule	20040	Formula One is unable to parse the validation rule.
F1ErrorBadInputMask	20041	Formula One is unable to parse the input mask.
F1ErrorValidationFailed	20042	The cell entry failed to pass the validation rule.
F1ErrorNoODBCConnection	20043	This error occurs when ODBCQuery is called without first executing a successful ODBCConnect call.
F1ErrorUnableToLoadODBC	20044	This error occurs when ODBCConnectcannot load the ODBC library.
F1ErrorUnsupportedFeature	20045	The version of Formula One you are using does not support the requested feature. Contact Visual Components to obtain a version that does support this feature.

Using Views and Workbooks

When you create a Formula One control, you are actually creating a workbook and a view into that workbook. By manipulating this connection between views and workbooks, you can add tremendous power and flexibility to your application. First, it is important to know the difference between a workbook and a view.

- Workbooks are objects that are maintained by the Formula One engine.
- A view is a window into a specific workbook.

Attaching Views to Workbooks

Controlling What is Displayed in a View

One View That Can Display Multiple WorkBooks

One Workbook Displayed in Multiple Views

Working with Views

Working with Workbooks

Working with Workbooks

A workbook is a collection of individual worksheets. The worksheet is a familiar tool you and your end-users use to store and manipulate data.

Workbooks store:

- cell data
- cell formulas
- workbook formatting information
- workbook-specific information such as printing attributes and calculation attributes.

Multiple workbooks can be open simultaneously. Formulas in one workbook can refer to cells in other workbooks. The Formula One engine manages all open workbooks.

Attaching Views to Workbooks

Using Views and Workbooks

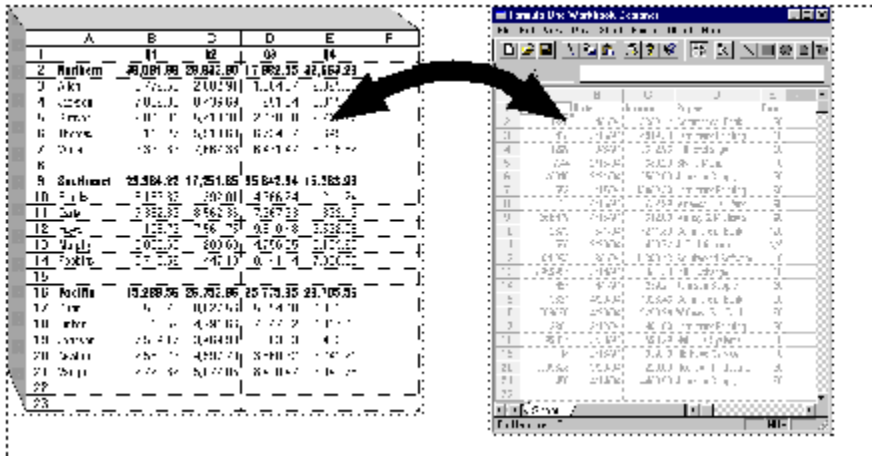
Working with Views

Working with Views

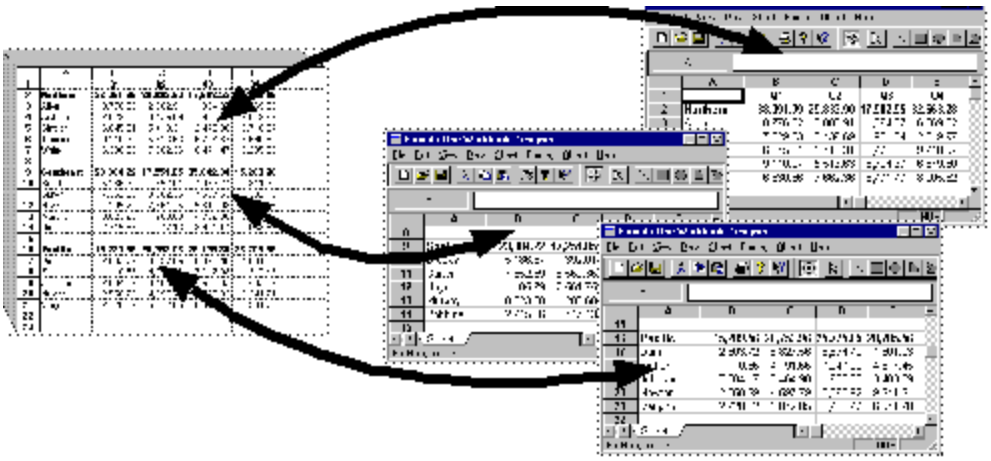
A view controls what part of the workbook is displayed in your application and when. Without a view, you cannot observe the work that you have performed in a workbook.

- Each view can only be attached to one workbook at a time.
- Multiple views can be attached to the same workbook.

After a view is created, you can change the workbook to which it is attached at any time using the **Attach** or **AttachToSS** method. This method attaches the view to the new workbook and severs the view's attachment to its previous workbook.



This figure illustrates the concept of one view attached to one workbook.



This figure illustrates the concept of multiple views into one workbook. Notice that each view displays a different area of the workbook.

Attaching Views to Workbooks

Controlling What is Displayed in a View

One View That Can Display Multiple WorkBooks

One Workbook Displayed in Multiple Views

Using View Information

Using Views and Workbooks

Using View Information

Views store information that describes how the workbook is displayed. Views contain information about:

- grid line display
- column and row heading display
- fixed row and column specifications
- maximum workbook viewing size

Views also contain information about user permissions such as whether the user is allowed to select cells, enter or edit data, or resize rows and columns.

Saving View Information

When a workbook is saved, the settings from the view that requested the save operation are saved with the workbook. When a view is attached to a workbook, the view settings are retrieved from the workbook.

Attaching Views to Workbooks

Controlling What is Displayed in a View

Working with Views

Attaching Views to Workbooks

The requirements of your application may require that you alter the view to which a workbook is attached, and vice versa. For example, some applications may have only one view, but work with multiple workbooks. Other applications may have more than one view connected to only one workbook.

When interchanging views and workbooks, there are several important rules to remember.

- A view can be connected to only one workbook at a time.
- A workbook can have multiple views to which it is attached.
- A workbook must be attached to at least one view. A workbook ceases to exist if it is not attached to a view.

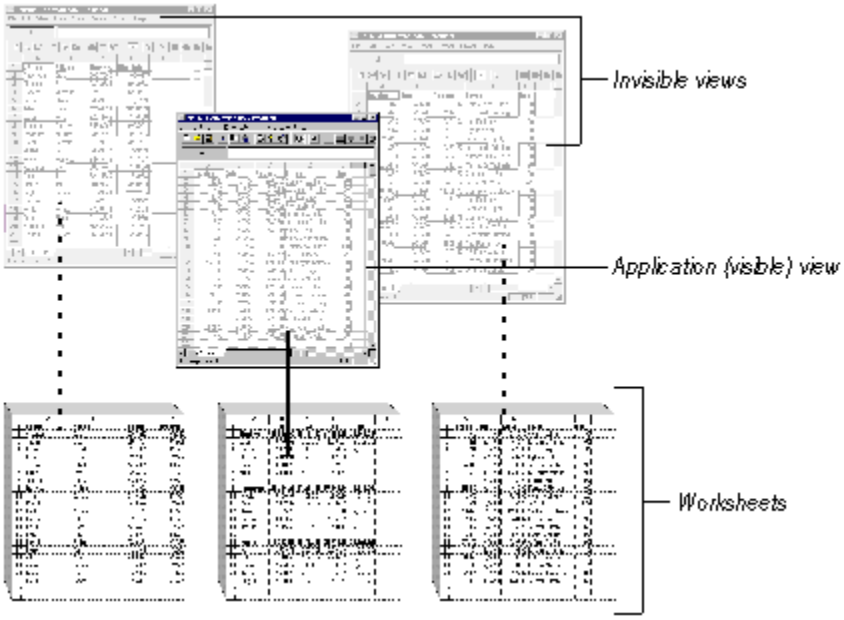
One View That Can Display Multiple WorkBooks

One Workbook Displayed in Multiple Views

One View That Can Display Multiple WorkBooks

Because a view can be connected to only one workbook at a time, you must employ invisible views to accommodate an application that uses one view that displays multiple workbooks.

To accomplish this, use the appropriate method or property to hide the view controls in your application. Hide all views in your application except the view you want displayed. Then, use the Attach method to change which workbook is connected to the visible view.



The application illustrated by this figure uses one view and three workbooks. The workbooks not attached to the application view are attached to invisible views.

One Workbook Displayed in Multiple Views

One Workbook Displayed in Multiple Views

When multiple views are attached to the same workbook, any change made in one view is reflected in the other views. The views are independent, so you can view different parts of the same workbook.

After a workbook is created dynamically, you can use the Attach or AttachToSS method to attach additional views.

One View That Can Display Multiple WorkBooks

Controlling What is Displayed in a View

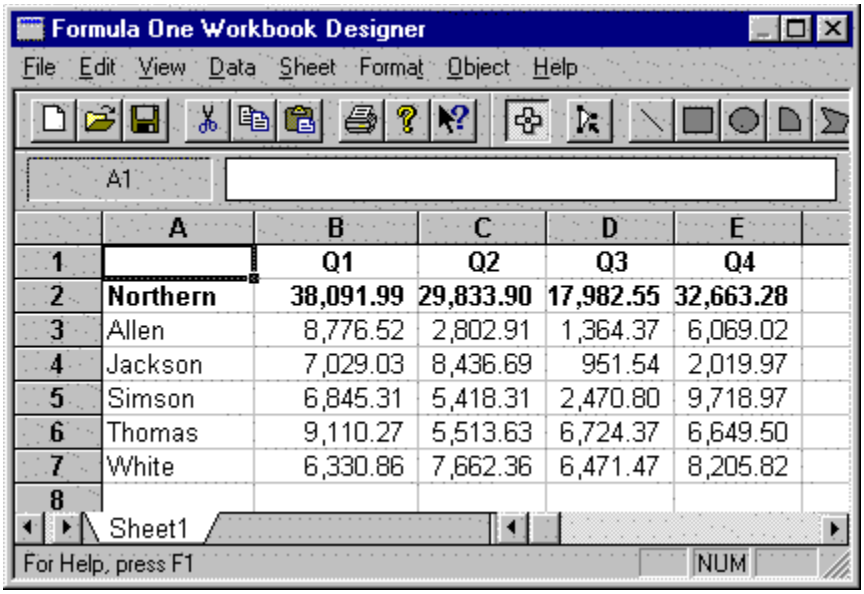
There are a number of properties you can set to determine which area of the workbook is displayed in a view. You must identify the worksheet and range of cells to appear in the current view.

If the workbook contains multiple worksheets, you can specify which worksheet you want to display in the view. This is accomplished by setting the **Sheet** property. Set **Sheet** to the index number of the worksheet you want to display. Sheets are indexed from left to right beginning with 1. Do not confuse the index with the worksheet sheet name that appears on the sheet tab.

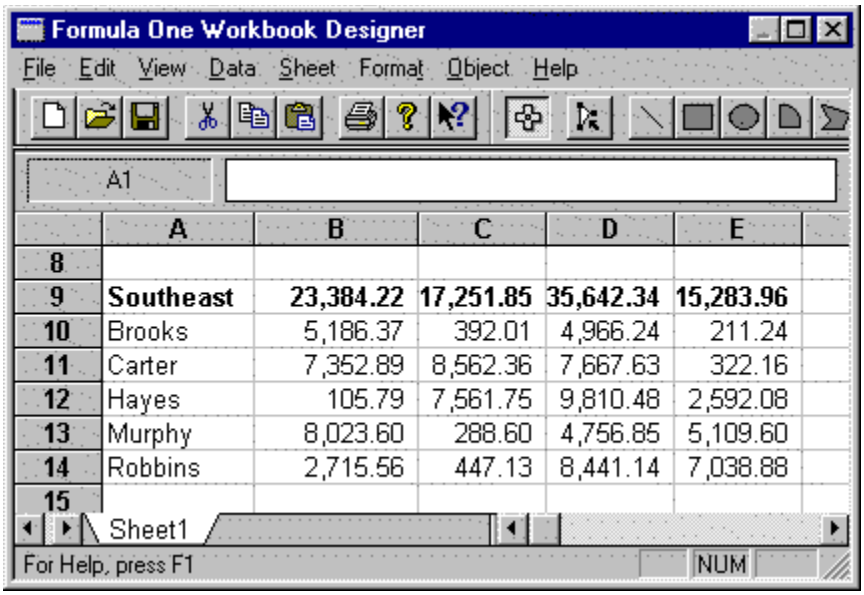
To prevent users from going to another worksheet in the workbook, you can set **ShowTabs** to off. This hides the sheet tabs, preventing the user from changing sheets. Alternatively, you could write code within the SelChange event to prevent the user from changing worksheets.

You can limit the area of each worksheet that can be seen within a view by setting the **MinRow**, **MinCol**, **MaxRow**, and **MaxCol** properties for each worksheet. This is particularly useful when you want to use multiple views to display different parts of the worksheet

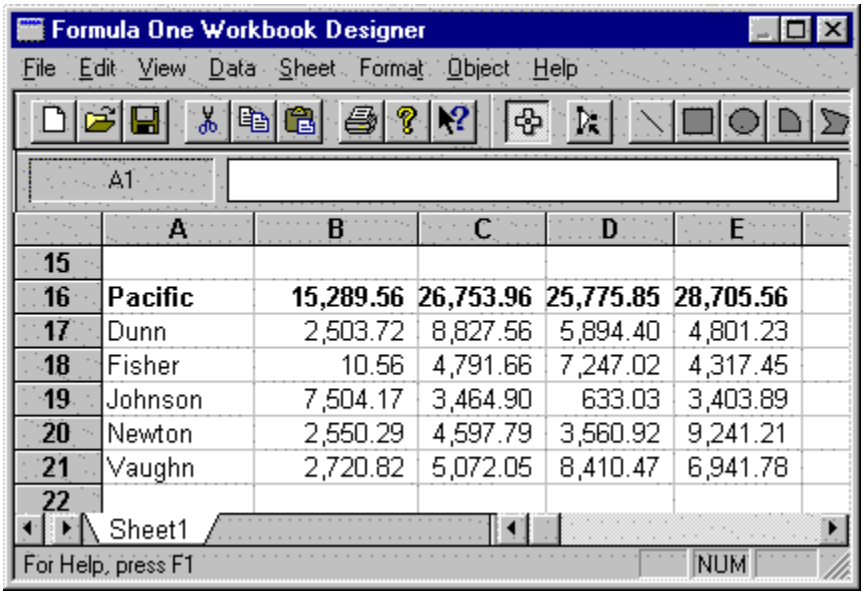
The following illustration shows the property settings used to limit the number of rows that can be displayed in each view. The data displayed in all three views is contained in one worksheet. Notice that none of the views have vertical scroll bars because the end-users are prevented from scrolling beyond the rows they already see.



MinCol = 1, MinRow = 1, MaxCol = 256, MaxRow = 8



MinCol = 1, MinRow = 8, MaxCol = 256, MaxRow = 15,



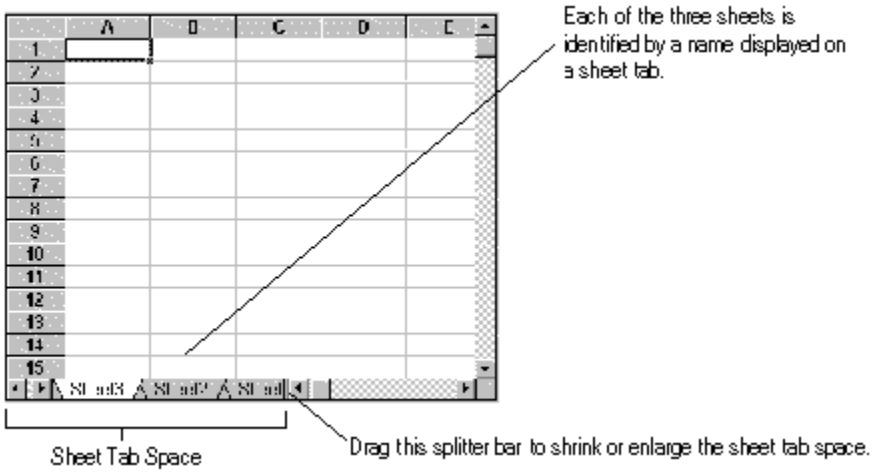
MinCol = 1, MinRow = 15, MaxCol = 256, MaxRow = 22

Attaching Views to Workbooks

The Formula One Workbook

When you open or create a file in Formula One 3.0, you are creating or opening a workbook. The workbook is where all your data is stored. A workbook can consist of one or more individual worksheets. Worksheets are useful for organizing information into separate groups. For example, you might have the year-end sales figures for each sales region on a different worksheet within the same workbook. Having all the information in one worksheet would be cumbersome; splitting it into separate files would be too inconvenient to work with.

The following illustration shows a Formula One workbook with three worksheets:



Inserting Worksheets
Selecting Worksheets
Deleting Worksheets
Renaming Worksheets
Navigating Through Worksheets
Selecting Cells
Selecting Rows and Columns

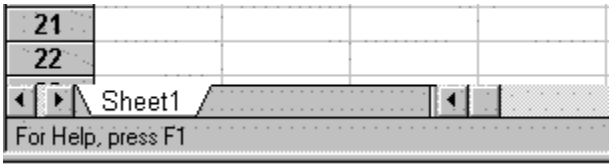
Inserting Worksheets

By default, a workbook contains only one worksheet. However, you can easily insert additional worksheets through the Workbook Designer, or with program code.

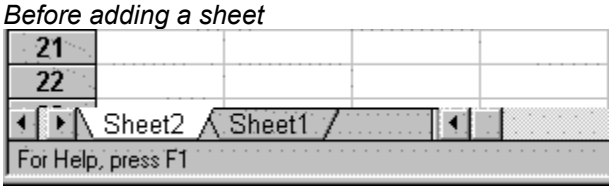
To add worksheets using the Workbook Designer:

1. Right click on the Workbook control to display the shortcut menu.
2. Select Workbook Designer from the shortcut menu.
3. Select Insert Sheet from the Sheet menu in the Workbook Designer.

One new worksheet is inserted to the left of the selected worksheet as shown in the following illustration:



Before adding a sheet



After adding a sheet

Sheet Index List
Selecting Worksheets
Deleting Worksheets
Renaming Worksheets

Sheet Index List

Each workbook maintains an indexed list of the worksheets it contains. Worksheets are indexed from left to right beginning with 1. As you add worksheets, the index is adjusted. Most methods and properties reference worksheets by index rather than name. It is important to remember that the sheet index is different from name that appears on the sheet tab.

Inserting Worksheets

Selecting Worksheets

Deleting Worksheets

Renaming Worksheets

Selecting Worksheets

Usually, you do most of your work in one worksheet at a time. This is called the active worksheet. When you have multiple worksheets in a workbook, you can use the mouse to click on a worksheet's tab to make it the active sheet. The tab is highlighted and moves on top of the other tabs.

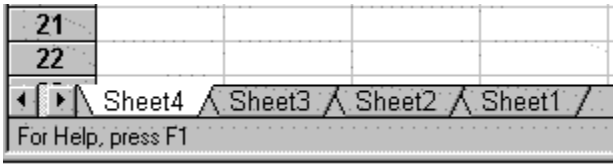
You can save time and effort by performing some tasks on several sheets at once. For example, if you want all three worksheets in your workbook to have the same title information, you can select all three worksheets and enter the titles on the active worksheet. The titles are automatically entered in the corresponding cells in the other selected worksheets as well.

To select multiple worksheets in the Workbook Designer:

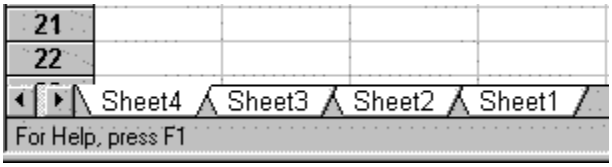
1. Use one of the following key\mouse combinations, depending on whether you want to select adjacent or non-adjacent worksheets:

Action	Result
CTRL-Click on sheet tab	Selects or deselects non-adjacent sheets. Any other selected worksheets remain selected.
SHIFT-Click on sheet tab	Selects all adjacent worksheets between the active worksheet and the worksheet you clicked on. All other worksheets are deselected.

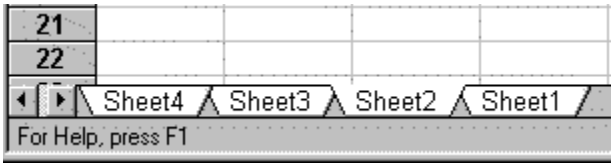
The following illustration shows various groupings of selected worksheets:



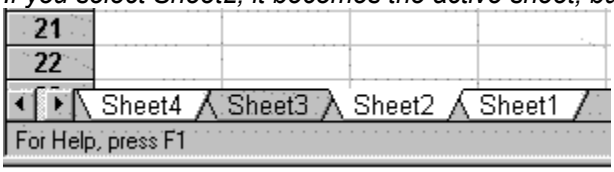
Sheet4 is the active sheet. All other sheets are deselected.



If you hold down the Shift key and select Sheet1, all sheets between the active sheet (Sheet4 and Sheet1 are selected.



If you select Sheet2, it becomes the active sheet, but all sheets remain selected.



If you hold down the Control key and select Sheet3, it is deselected, but the other sheets remain selected.

To select multiple sheets programmatically:

- Use the **SheetSelected** property to toggle an individual worksheet's selection status to on.

The following example selects the second and third worksheets in the workbook:

```
SheetSelected (2) = True

SheetSelected (3) = True
```

Working with a Group of Worksheets

When you have multiple worksheets selected, you can think of them as a group of worksheets. When you perform some actions, execute some methods, or refer to some properties, they affect all worksheets in the group. Other actions affect only the active worksheet, regardless of how many worksheets are selected.

In the Workbook Designer, the following actions work on all selected worksheets:

- Changing cell selection.
- Entering value via the edit bar.
- Inserting rows, columns, or ranges of cells
- Deleting rows, columns, or ranges of cells
- Clearing rows, columns, or ranges of cells
- Setting TopLeft/Row/Column header text.
- Setting column width.
- Setting row height.
- Moving & Copying with mouse.

Within your application code, the following methods and setting of properties affect all selected worksheets:

Methods	Setting Properties
ColWidthDlg (except Auto)	ColText
ClearRange	ColWidth
CopyRange	Entry
DeleteRange	EntryRC
EditInsert	Formula
EditDelete	FormulaRC
EditClear	HdrWidth
EnableProtection	HdrHeight
FormatAlignmentDlg	Logical
FormatBorderDlg	LogicalRC
FormatFontDlg	Number
FormatPatternDlg	NumberFormat
FormatNumberDlg	NumberRC
InsertRange	RowText
MoveRange	RowHeight
ProtectionDlg	Text
RowHeightDlg (except Auto)	TextRC
SetAlignment	TopLeftText
SetBorder	
SetColWidth	
SetFont	
SetPattern	
SetProtection	

SetRowHeight

Inserting Multiple Worksheets

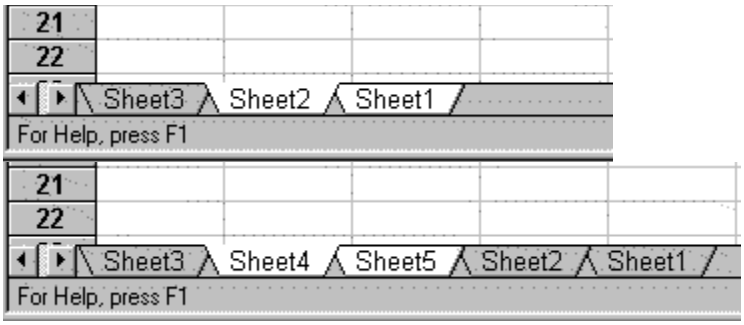
You can insert more than one sheet at a time and at any point in the sheet tab index. The number and position of the inserted sheets depends on number and position of the selected sheets in the workbook.

To insert worksheets in the Workbook Designer:

1. Select the worksheet immediately to the right of where you want to insert the new worksheets.
2. Select as many worksheets to the right of that worksheet as the number of worksheets you want to insert.

   For example, if you want to insert two worksheets, a total of two worksheets must be selected.
3. Select Insert Sheet from the Sheet menu.

The following illustration shows this process:



Since Sheet2 and Sheet1 are selected, two additional worksheets are inserted to the left of Sheet2. Notice that the newly inserted worksheets are given the next available sheet names, regardless of their position in the sheet index list.

To insert sheets in your code:

- Use **NumSheets** to increase the number of worksheets in the workbook. Additional worksheets are added to the right of all existing worksheets.
- Use the **InsertSheets** method as shown in the following example which inserts 2 worksheets to the left of the third worksheet in the workbook:

```
F1Book1.InsertSheets 3, 2
```

- Use **SheetSelected** to select the number of worksheets you want and then use the **EditInsertSheets** method. The following example selects the third and fourth worksheets in the workbook and inserts two worksheets before the third worksheet.

```
F1Book1.SheetSelected (3)

F1Book1.SheetSelected (4)

F1Book1.EditInsertSheets
```

Inserting Worksheets

Renaming Worksheets

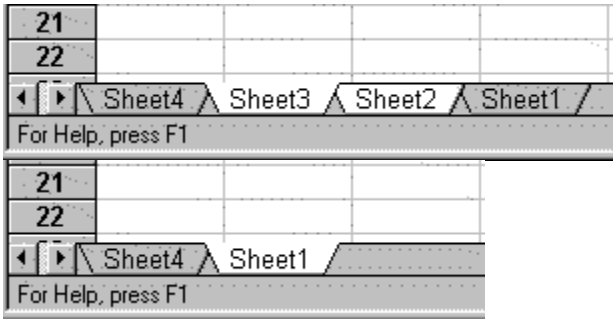
Deleting Worksheets

You can delete one or more worksheets from the sheet index list through the Workbook Designer or through application code.

To delete worksheets in the Workbook Designer:

1. Select the worksheets you want to delete.
2. Select Delete Sheets from the Sheet menu.

The following illustration shows this process:



Two methods delete worksheets; **DeleteSheets** and **EditDeleteSheets**. **DeleteSheets** takes arguments that define the position and number of worksheets to be deleted. EditDeleteSheets deletes the currently selected worksheets.

To delete worksheets in your code:

- Use **NumSheets** to decrease the total number of worksheets. Worksheets are deleted from the right.
- Use the **DeleteSheets** method as shown in the following example which deletes the third and fourth worksheets from the workbook:

```
F1Book1.DeleteSheets 3, 2
```

- Use **SheetSelected** to select sheets and then use the EditDeleteSheets method. The following example selects the first and fourth worksheets and then deletes them:

```
F1Book1.SheetSelected (1)
F1Book1.SheetSelected (4)
F1Book1.EditDeleteSheets
```

Inserting Worksheets

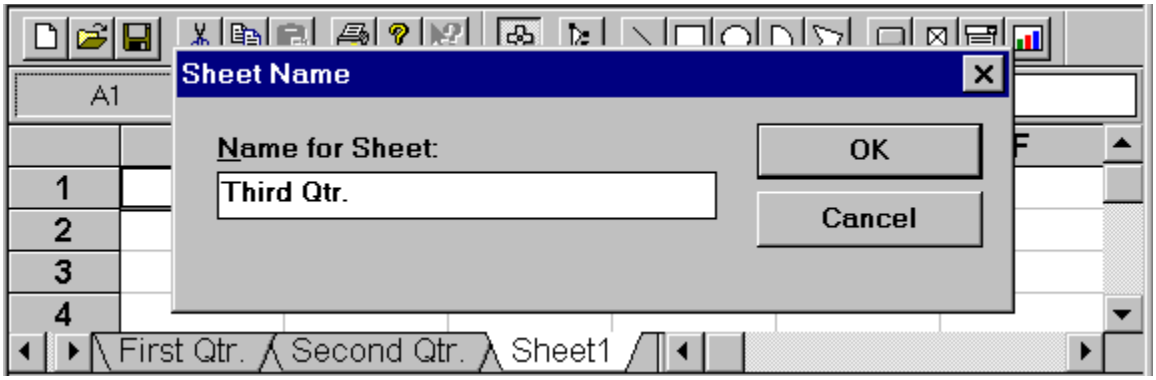
Sheet Index List

Selecting Worksheets

Renaming Worksheets

Renaming Worksheets

Formula One provides each worksheet with a default name. You can change those names to more meaningfully describe the sheets contents. For example, the sheet names in the following illustration are far more descriptive than the worksheets default names.



To edit a sheet name in the Workbook Designer:

1. Double-click on the sheet tab to display the Sheet Name dialog box.
2. Enter the new name and click OK.

To edit a sheet name in code:

- Use the **SheetName** property to rename a worksheet identified by index. The following code changes the name of the second worksheet in the workbook to QTR 2 Sales.

```
F1Book1.SheetName (2) = "QTR 2 Sales"
```

Inserting Worksheets

Sheet Index List

Selecting Worksheets

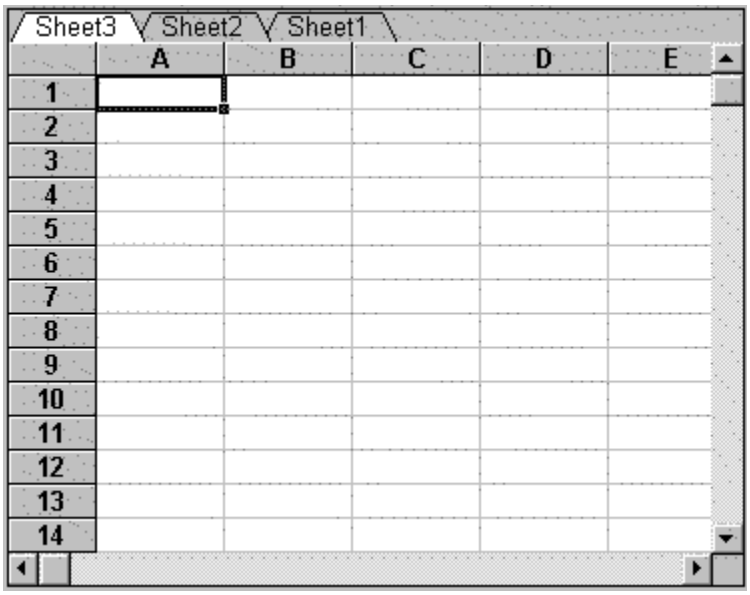
Positioning Worksheet Tabs

When designing your application, you can use the property pages or the ShowTabs property to control the appearance and position of the sheet tabs.

To format the sheet tabs using property pages:

1. Right click on the Formula One control while in design mode to display the shortcut menu.
2. Select Properties from the shortcut menu.
3. Select the Show tab.
4. In the Tabs list box, select Top to display tabs at the top of the workbook, Bottom to display tabs at the bottom of the workbook, or Off to hide the worksheet tabs.
5. Click OK.

The following illustration shows sheet tabs displayed at the top of the workbook.



You can also set the **ShowTabs** property to reposition the sheet tabs. This can be useful if your application includes multiple worksheets, but you want to limit users to interacting with a single sheet.

To limit user access to a single sheet:

1. Use the **Sheet** property to make the sheet you want users to access the active sheet.
2. Set the ShowTabs property to the constant F1TabsOff.

Navigating Through Worksheets

When working in the Workbook Designer or in a workbook at run time, you can navigate within individual worksheets using keyboard commands or mouse actions. In addition to navigating through worksheets, keyboard commands allow you to perform a variety of other tasks.

Keyboard commands allow you to:

- position the active cell in the worksheet
- page through a worksheet
- enter data typed in a cell
- move the active cell within a selected range
- enter and exit edit mode
- recalculate a workbook
- delete data from a selected cell or range

Using Keyboard Commands

Performing Mouse Actions

Selecting Cells

Selecting Rows and Columns

Using Keyboard Commands

The tables in this section list the keyboard commands you can use when working in the Workbook Designer or a workbook at run time. The following table lists action keys that allow you to enter and edit data, move the active cell within a selected range, and recalculate the workbook.

Key	Description
ENTER	When in edit mode, accepts the current entry. When a range is selected, and if the EnterMovesDown property is set to True, accepts the current entry and moves active cell vertically to next cell in selection.
SHIFT + ENTER	When in edit mode, accepts the current entry. When a range is selected, and if the EnterMovesDown property is set to True, accepts the current entry and moves active cell vertically to previous cell in selection.
TAB	When in edit mode, accepts the current entry and moves the active cell horizontally to right.
SHIFT +TAB	When in edit mode, accepts the current entry and moves the active cell horizontally to left.
F2	Enters edit mode. While in editing mode, F2 displays the Cell Text dialog box, in which you can enter multi-line data entries.
F9	Recalculates workbook.
DEL	May clear current selection depending on the setting of the **AllowDelete** property**.**
Escape	Cancels current data entry or editing operation.

Important Some development environments, such as Visual Basic override the use of these keys in design mode. This can cause unexpected results. For example, pressing the DEL key in Visual Basic's edit mode deletes the control instead of the current selection in the worksheet. If this happens, select Undo from the Edit menu to restore the control.

The following table lists the movement keys that allow you to move the active cell within a worksheet and display different sections of the worksheet.

Key	Description
Up Arrow	Moves active cell up one row.
Down Arrow	Moves active cell down one row.
Left Arrow	Moves active cell left one column.
Right Arrow	Moves active cell right one column.
CTRL Up/Down/Left/Right	Moves to the next range of cells containing data. If there is no additional data in the direction in which you are moving, moves to the edge of the worksheet.
Page Up	Moves up one screen.
Page Down	Moves down one screen.
CTRL Page Up	Moves left one screen.
CTRL Page Down	Moves right one screen.
Home	Goes to first column of current row.
End	Goes to last column of current row that contains data.
CTRL Home	Goes to row 1 column 1.

CTRL End	Goes to last row and column that contains data.

The following table lists the keys that modify the action of the movement keys.

Key	Description
Scroll lock	Causes the view window to scroll without changing current selection with all movement keys except Home, End, CTRL Home, and CTRL End.
SHIFT plus any movement key	Extends the current selection.

Performing Mouse Actions

Selecting Cells

Selecting Rows and Columns

Performing Mouse Actions

Primarily the mouse is used to select items in a worksheet at run time. The following table lists the mouse actions you can perform in a worksheet at run time or in the Workbook Designer.

Action	Description
Left Click	Moves the active cell to the pointer position.
Right Click	In the container's design mode, brings up the shortcut menu.
Left Click in Row or Column Headings	Selects entire row or column.
Left Click in Top Left Corner	Selects entire sheet.
Left Double Click in Top Left Corner, Row Headings, Column Headings, or Worksheet tabs	Displays a dialog box that allows you to enter a label for the top left corner or the column or row heading, or a new name for the worksheet that was double clicked.
Left Double Click	In the Workbook Designer, invokes in-cell editing.
	At run time, a DblClick event is fired.
Right Double Click	In the Workbook Designer, does nothing.
	At run time, the Workbook Designer is launched if **DoRDblClick** is False.
Left Click and Drag	Selects a range. If other ranges are selected, the previously selected ranges are unselected.
CTRL + Left Click and Drag	Selects a range. If other ranges are selected they remain selected.
SHIFT + Left Click and Drag	Extends the current selection.
CTRL + SHIFT Click on Row Headings, Column Headings, or Top Left Corner	Selects the row headings, column headings, or top left corner of the sheet.
Drag a Selection's Copy Handle	Copies the selection into the newly selected area.
Drag a Selection's Border	Moves the selection to a new location.
ALT + Click and Drag an Object or Object's Selection Handles	Repositions or resizes an object and aligns object sides with the cell grid.

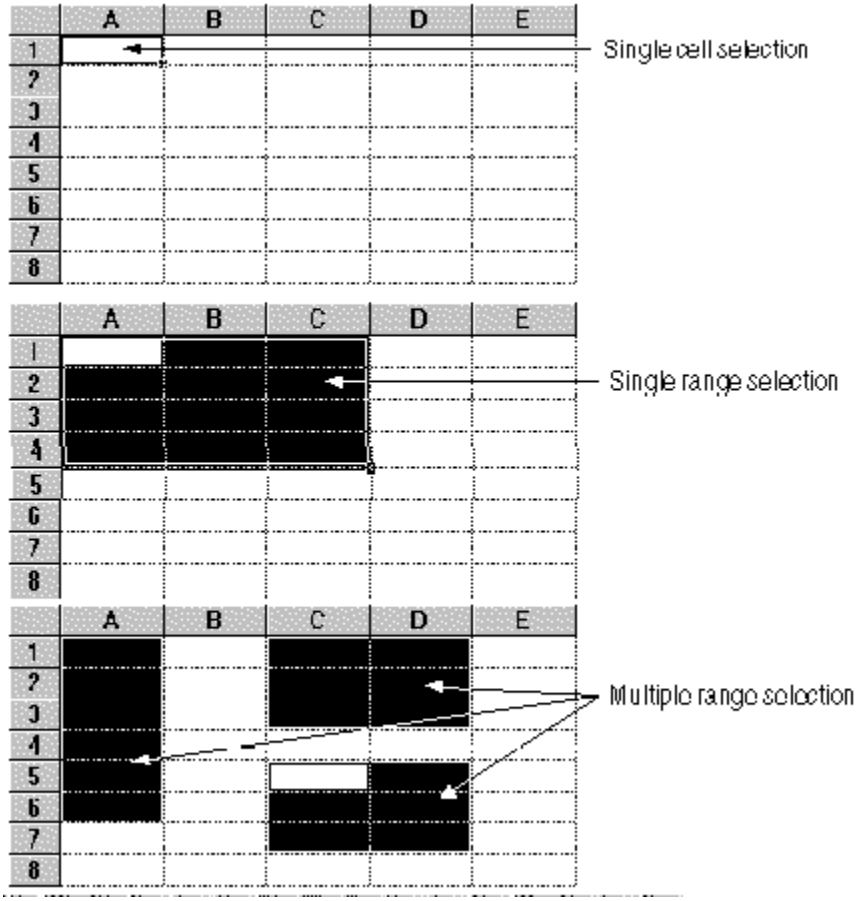
Using Keyboard Commands

Selecting Cells

Selecting Rows and Columns

Selecting Cells

Many operations require one or more cells to be selected. There are three kinds of selections: a single cell, a range of cells, and multiple ranges of cells (non-adjacent). The following illustration shows the three types of selections.



Selecting Cells with the Mouse
Selecting Cells with Properties and Methods
Selecting Rows and Columns

Selecting Cells with the Mouse

The worksheet cursor is always located on a cell. The cell on which the worksheet cursor is located is called the active cell. The active cell is also a selection or part of a selection. Any data the user enters is always placed in the active cell.

- To select a range of cells, click and hold the left mouse button and drag through the range you want to select. When a range is selected, It becomes highlighted.
- To select multiple ranges, press the CTRL key while selecting a range with the mouse. Any previously selected ranges remain selected.

Once a range is selected, you can move the active cell within the range using the ENTER, SHIFT + ENTER, TAB, and SHIFT + TAB keys. When you use these keys to move the active cell, the range remains selected.

Selecting Cells

Selecting Cells with Properties and Methods

Selecting Rows and Columns

Selecting Cells with Properties and Methods

The following properties and methods can select ranges.

- Setting the **Selection** property removes all current selections and selects a range.
- The **AddSelection** method adds a selection to the current selection list. Continue calling **AddSelection** to create multiple selections.

The following example selects two ranges, A1:D4 and E5:H8.

```
F1Book1.Selection' "A1:D4" 'Select A1:D4

F1Book1.AddSelection 5, 5, 8, 8 'Add E5:H8
```

In addition, the following properties retrieve information about multiple selections.

- The value of the **SelectionCount** property tells you the number of selections. You can use this if a selection is made by the user and you need to determine how many ranges are selected.
- The value of the **Selection** property gives you all current the selections in the form of a formula (e.g. A1:D4,E5:H8).

Selecting Cells

Selecting Cells with the Mouse

Selecting Rows and Columns

Selecting Rows and Columns

Entire rows and columns can be selected in the worksheet at run time or in the Workbook Designer using the mouse. To select a row or column, position the pointer on the header of the row or column you want to select. When you click the header, the row or column is selected.

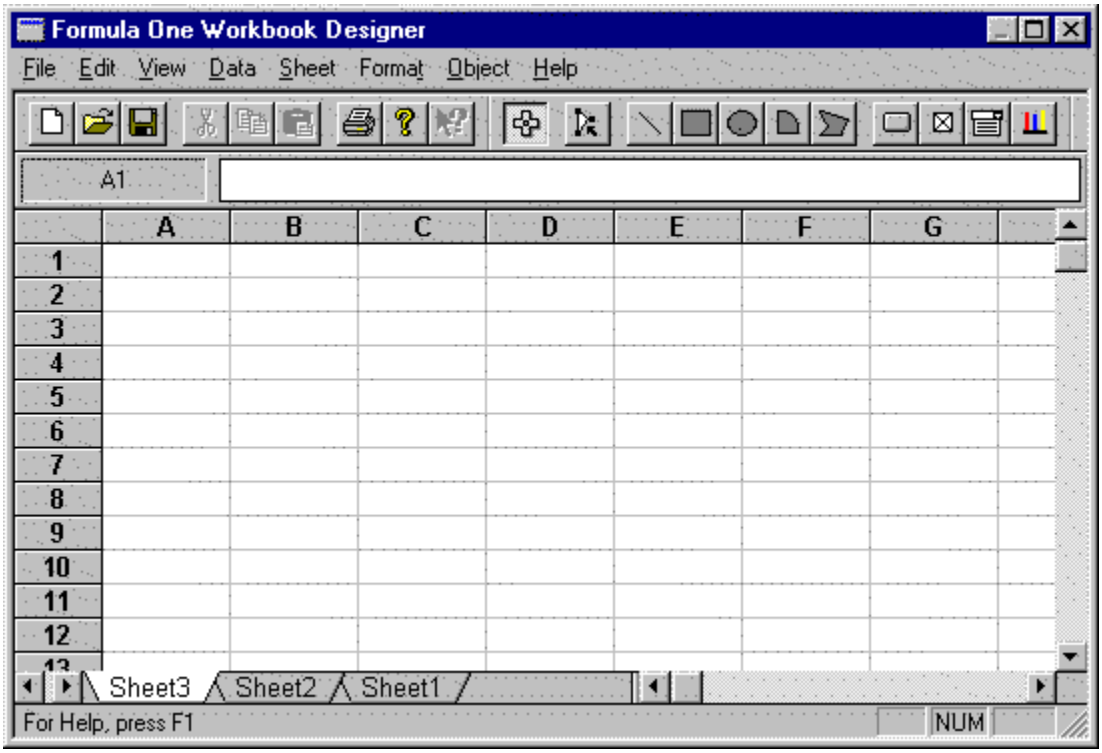
You can also select all rows and columns in the worksheet. To do this, position the pointer on the top left header and click.

Selecting Cells

Using the Workbook Designer

The Workbook Designer is a Windows application that uses the Formula One workbook engine and allows you to open, manipulate, and save a Formula One workbook. The designer allows you to visually design workbooks for your application. With the designer, you can:

- add, insert, delete, or name worksheets
- enter data and formulas in worksheet cells
- size rows and columns
- format data
- set the font attributes for data and headers
- set column and row header text
- format worksheet cells with colors and patterns
- specify cell borders and border types
- define names
- protect and hide cells
- set user permissions
- select items to be shown and hidden



To launch the Workbook Designer:

1. Right click on the Formula One control to display the shortcut menu.
2. Select Workbook Designer from the shortcut menu.

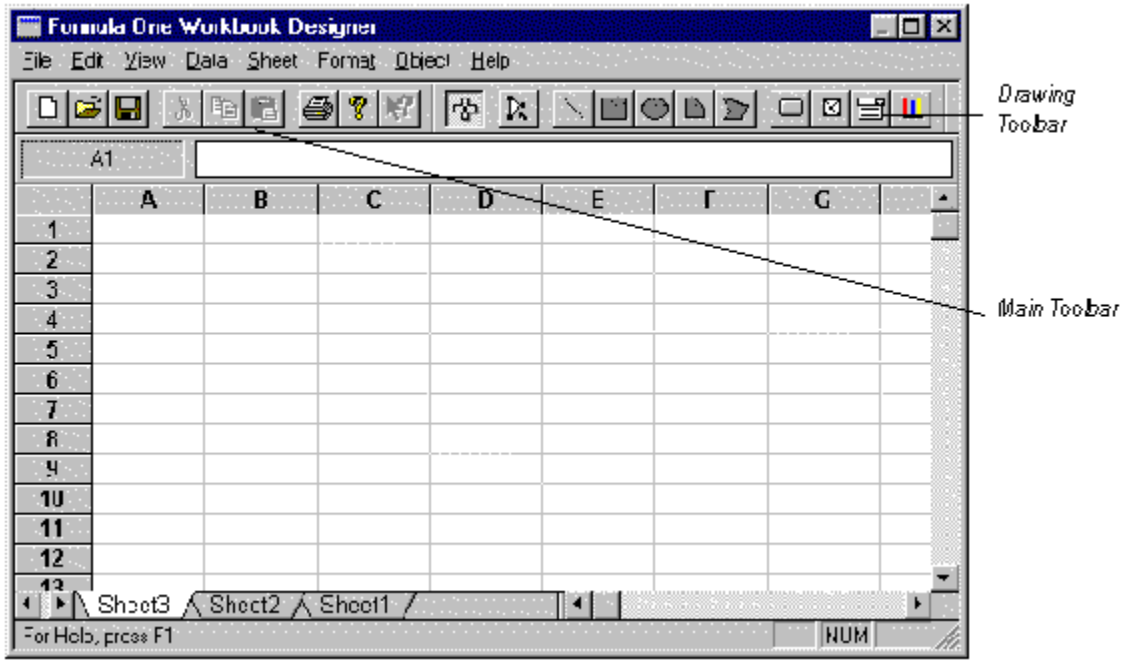
Note If the Formula One control is in edit mode, you can launch the Workbook designer by double-clicking the right mouse button on the control.

Workbook Designer Overview

Workbook Designer Overview

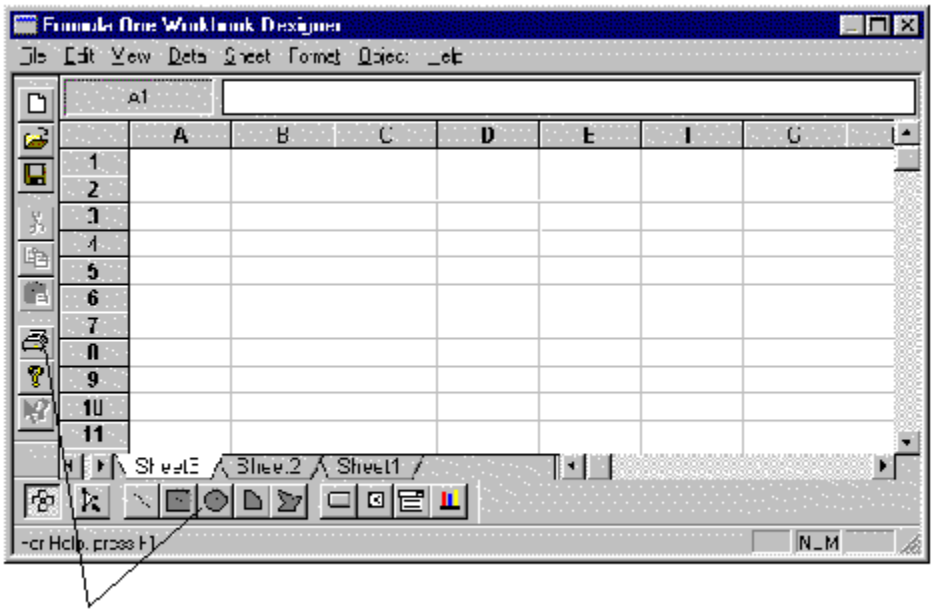
The Workbook Designer is an interactive program, available in Design mode, that allows you to design and format a workbook for your application by pointing and clicking, and choosing format commands from menus. The Workbook Designer allows you to manipulate a workbook control just like it was a part of spreadsheet application.

The following illustration shows the default appearance of the Workbook Designer. The toolbars in the Workbook Designer are dockable. This means they can be dragged to a new location within the Designer, or made to float on top of the Designer.

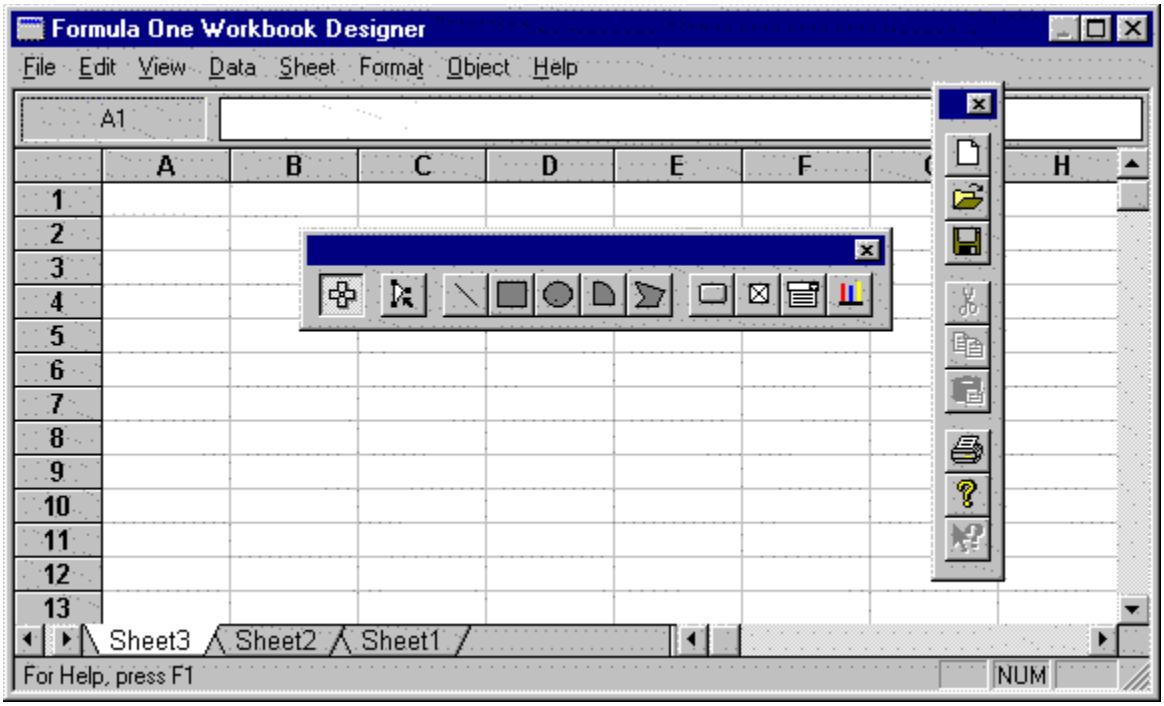


To move the dockable toolbars:

1. Select the toolbar you want to move.
2. Drag it to the new location
3. If you drag it to an edge of the Designer, the toolbar is docked on that side of the Designer. If you drag and release the toolbar on top of the designer, the toolbar is left floating.

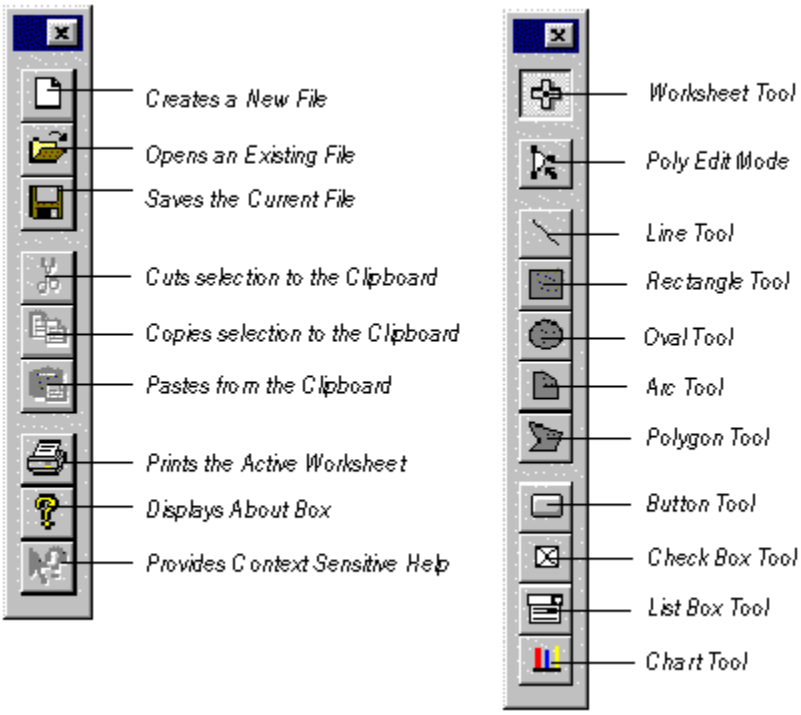


Notice the toolbars have both been docked in new locations.



In this case, both toolbars have been left to float on top of the Designer.

The following illustration identifies the icons on each toolbar.



File Menu
Edit Menu
View Menu
Data Menu
Sheet Menu
Format Menu
Object Menu

File Menu

Command	Description
New	Creates a new file. This action deletes any information currently in the Workbook Designer.
Read	Opens a workbook file from disk. Files saved in Formula One format (.VTS files), Excel 4.0 and 5.0 format (.XLS files), and tabbed text (.TXT) can be opened.
Write	Saves the current worksheet. Files can be saved in Formula One format (.VTS files), Excel 4.0 or 5.0 format (.XLS files), or tabbed text (.TXT.)
Print	Prints the active worksheet.
Page Setup	Displays the Page Setup dialog box. This dialog box allows you to define header and footer text, page margins, page print order, page centering, worksheet-related print options.
Print Setup	Displays the standard Windows Print Setup dialog box. This dialog box allows you to select the printer to which the worksheet is sent, the page orientation, and paper size.

Edit Menu

Command	Description
Cut	Cuts the current worksheet selection to the clipboard.
Copy	Copies the current worksheet selection to the clipboard.
Paste	Pastes the contents of the clipboard to the current worksheet selection.
Paste Values	Pastes values from the clipboard to the current worksheet selection. Any formatting applied to the values is ignored. In addition, only formula results are pasted; formulas are ignored.
Clear	Displays a dialog box that allows you to clear data or objects from the current selection. You can clear only formats, only values (including formulas), or both formats and values.
Insert	Inserts cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells. If you use the keyboard shortcut CTRL + I, the selected cells are shifted right to make room for the inserted cells. If you use SHIFT + CTRL + I, the selected cells are shifted down.
Delete	Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells. f you use the keyboard shortcut CTRL + K, cells to the right of the selected cells are shifted left to fill the space left by the vacated cells. If you use SHIFT + CTRL + K, cells below the selected cells

	are shifted up.
Copy Right	Data in the leftmost cell of the selected range is copied right to fill the range.
Copy Down	Data in the top cell of the selected range is copied down to fill the range.
Goto	Displays the Goto dialog box. This dialog box allows you to specify a cell to display in the worksheet window. The specified cell is made the active cell.

View Menu

Command	Description
Main Toolbar	Toggles the display of the main toolbar.
Drawing Toolbar	Toggles the display of the drawing toolbar.
Edit Bar	Toggles the display of the edit bar.
Status Bar	Toggles the display of the status bar.

Data Menu

Command	Description
Recalc	Recalculates the workbook.
Calculation	Displays the Calculation dialog box. This dialog box allows you to enable and disable automatic recalculation and specify iteration values for calculating circular references.
Define Name	Displays the Define Name dialog box. This dialog box allows you to add and delete user defined names.
Sort	Displays the Sort dialog box. This dialog box allows you to set the sorting method and sort keys for data sorting.
AutoFill List	Displays the AutoFill List dialog box. This dialog box allows you to add, delete, or edit autofill lists.

Sheet Menu

Command	Description
Set Print Area	Defines the currently selected range in the active worksheet as the Print_Area user-defined name.
Set Print Titles	Defines the currently selected range in the active worksheet as the Print_Titles user-defined name.
Set Page Breaks	Places a horizontal page break adjacent to the top edge of the active cell and a vertical page break adjacent to the left edge of the active cell. If a row or column is selected, a page break is placed adjacent to the selected row or column.
Remove Page Breaks	This command replaces Set Page Breaks if page breaks are adjacent to the active cell. Removes page breaks adjacent to the top edge and left edge of the active cell.
Insert Sheet	Inserts one or more worksheets. The number and position of inserted worksheets is determined by the number of selected worksheets. This command fails if non-contiguous sheets are selected.

Delete Sheet	Deletes the selected worksheets.
Color Palette	Displays the Color Palette dialog box. This dialog box allows you to edit colors in the color palette, specify a default color, and use the default color palette.
Default Font	Displays the Default Font dialog box. This dialog box allows you to set the default font used to display data in worksheets. In addition to setting the font and font size used to display data in a worksheet, the default font affects the widths of worksheet columns. Column widths can be set in units equal to 1/256th of the character 0 (zero) in the default font.
Enable Protection	Enables protection for protected cells in the worksheet. A check next to this command means that protection is enabled. Select the command again to disable protection.
Fix Row + Columns	Fixes the selected columns or rows. Fixed columns and rows do not scroll and cannot be edited.
Unfix Rows + Columns	These commands replace Fix Rows or Fix Columns when any rows or columns are currently fixed.

Format Menu

Command	Description
Alignment	Displays the Alignment dialog box. This dialog box allows you to specify the horizontal and vertical alignment of data in the selected range. In addition, you can enable and disable word wrapping.
Font	Displays the Font dialog box. This dialog box allows you to specify the font, point size, font style, and color of data in the selected range.
Border	Displays the Border dialog box. This dialog box allows you to specify the placement of borders in the selected range. In addition, you can specify the border line style and color. The check boxes in the Border dialog box are three-state check boxes, allowing "as is" selections to be made.
Pattern	Displays the Pattern dialog box. This dialog box allows you to specify the fill pattern and foreground and background colors for the selected range.
Cell Protection	Displays the Cell Protection dialog box. This dialog box allows you to specify whether the cells in the selected range are locked and hidden.
General	Formats data in the selected range with the General format.
Currency (0)	Formats data in the selected range with the Currency format and a decimal precision of 0.
Currency (2)	Formats data in the selected range with the Currency format and a decimal precision of 2.
Fixed	Formats data in the selected range with the Fixed format.
Percent	Formats data in the selected range with the Percent format. Numbers with this format are displayed as

	percentages with a trailing percent sign (%).
Fraction	Formats data in the selected range with the Fraction format. Numbers with this format are displayed as fractions.
Scientific	Formats data in the selected range with the Scientific format.
M/D/YY	Formats data in the selected range with the M/D/YY date format. Numbers with this format are displayed as dates.
H:MM AM/PM	Formats data in the selected range with the H:MM AM/PM time format. Numbers with this format are displayed as times.
Custom Number	Displays the Custom Number dialog box. This dialog box allows you to define custom number formats for data in the selected range.
Validation Rule	Displays the Validation Rule dialog box. This dialog box allows you to create or edit the validation rule for the current selection.
Column Width	Displays the Column Width dialog box. This dialog box allows you to set the width of the selected columns, specify default column widths, and specify automatic column width. In addition, you can specify whether the selected columns are shown or hidden.
Row Height	Displays the Row Height dialog box. This dialog box allows you to set the height of the selected rows, specify default row heights, and specify automatic row height. In addition, you can specify whether the selected rows are shown or hidden.

Object Menu

Command	Description
Pattern	Sets the fill pattern and colors for the selected objects.
Line Style	Sets the line style, width, and colors for the selected line object or the border surrounding the selected arcs, ovals, polygons, and rectangles.
Name	Displays the Object Name dialog box. This dialog box allows you to set a name for the selected object.
Options	Displays the Object Option dialog box. This dialog box allows you to set the input value cell for selected check boxes and list boxes, the text displayed by check boxes and buttons, and the list of items contained by list boxes.
Bring to Front	Places the selected objects in front of other objects in the worksheet.
Send to Back	Places the selected objects behind other objects in the worksheet.
Select All	Selects all the objects in the worksheet.

Worksheet Data Entry

One of the basic tasks encountered when working with a workbook is data entry. Formula One provides several methods for entering data.

- **Direct Entry**. This is the most direct method of data entry. Data can be entered directly in a worksheet at run time. Or, you can enter data in the Workbook Designer at design time.
- **Properties and methods**. Several properties and methods allow you to enter data in the active cell or a specified cell.

Adding the edit bar

Entering Data with Properties

Using AutoFill Lists

Validating Data

Worksheet Data Types

Cell References

Worksheet Errors

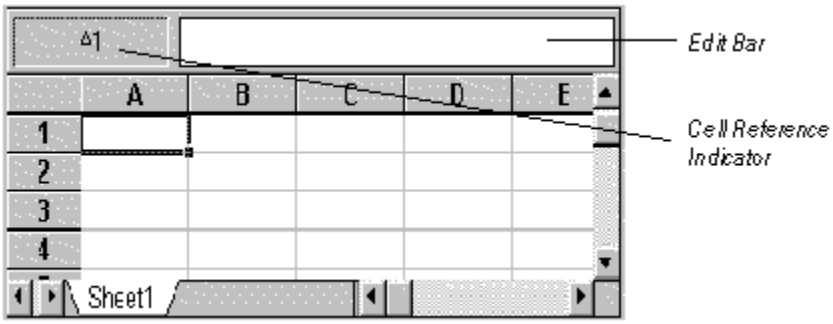
Understanding Functions

Using Names

Calculating Worksheets

Adding the edit bar

You can enter data into a worksheet by typing directly into a cell, or by typing into the edit bar. The edit bar appears by default in the Workbook designer, and you can turn on or off the display of the edit bar and cell reference indicator in your workbook control.



To control the display of the edit bar in the Workbook designer:

1. Select Edit Bar from the View menu to toggle the display of the edit bar.

To add an edit bar to your control:

- Set the **ShowEditBar** property to True.
- If the **ShowEditBar** property is True, you can also set the **ShowEditBarCellReference** to True to display the cell reference indicator with the edit bar.

Important The edit bar does not appear on the control unless the container makes it UI Active (provides a window for it.)

Entering Data with Properties

Formula One provides a full complement of properties for entering data. In addition to entering data in the active cell, a number of methods allow you to enter data in a cell you specify.

The following table lists the properties involved in entering data.

Property/ Method	Description
Entry	Sets or returns the value of the current cell in edit mode format.
EntryRC	Sets or returns the value of the specified cell in edit mode format.
Formula	Sets or returns the formula in the active cell.
FormulaRC	Sets or returns the formula in the specified cell.
Logical	Sets or returns the formula in the active cell.
LogicalRC	Sets or returns the logical value of the specified cell.
Number	Sets or returns the numeric value of the active cell.
NumberRC	Sets or returns the numeric value of the specified cell.
Text	Sets or returns the text value of the active cell.
TextRC	Sets or returns the text in the specified cell.

Entering Multi-Line Data

Limiting Data Entry

Entering Multi-Line Data

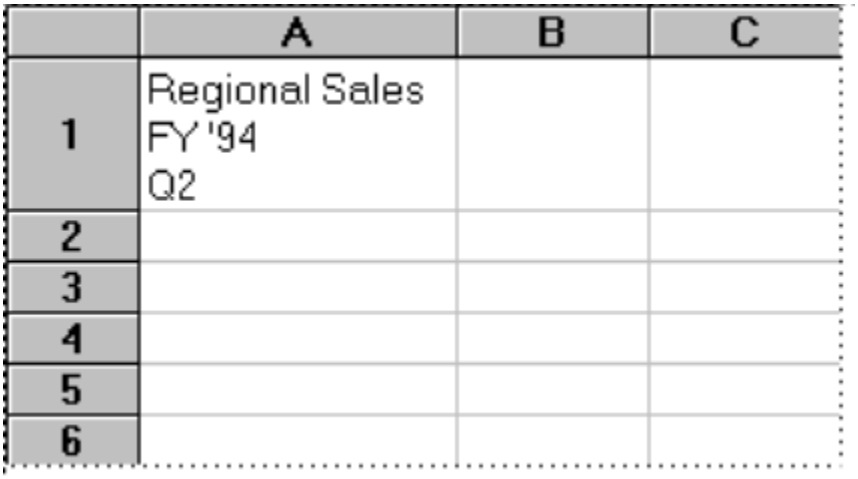
A single cell can contain as many as nine lines or 256 characters of data. When entering the data, new lines of data are specified within the cell by entering carriage return/line feeds.

- When entering data interactively in the Workbook Designer or in a worksheet at run time, press F2 when editing a cell. The Cell Text dialog box is displayed in which you can enter the cell data. To enter a line feed, press RETURN. Click the OK button to accept the entry and return to normal worksheet editing.
- When entering data with properties or methods, you should specify line feeds with the ANSI character codes for a carriage return/line feed pair (ANSI characters 13 and 10).

After entering multi-line data, you may need to resize the row and column in which the entry is placed in order to view all the data. The following example uses the TextRC property to enter three lines of data in cell A1.

```
F1Book1.TextRC( 1, 1) = "Regional Sales (Chr(13) & Chr(10)) FY '94 (Chr(13) & Chr(10)) Q2"
```

The following illustration shows the results of this example.



You can also enter multi-line row and column headers.

Formatting Row and Column Headings

Entering Data with Properties

Limiting Data Entry

Some applications may require that the user not be allowed to enter or edit data. To prevent data entry, set the **AllowInCellEditing** property to False, and set **ShowEditBar** to False. Data and formula entry and editing is thus prevented. Any data manipulation must be performed through program code.

Adding the edit bar

Limiting Formula Entry

Locking Cells

Validating Data

Limiting Formula Entry

If you only want to prevent the entering and editing of formulas, set the **AllowFormulas** property to False. Setting this property to FALSE does not affect the entry and editing of constant values.

Limiting Data Entry

Locking Cells

Displaying Formulas

Entering Formulas

Formula Operators

Locking Cells

You can lock cells in the Workbook Designer, or through code:

To set editing permissions on a per cell basis:

1. Select the cells you want to lock.
2. Select Cell Protection from the Format menu.
3. Uncheck the Locked check box and click OK.
4. Select Enable Protection from the Sheet menu to enable protection for the entire worksheet.

   This final step actually prevents the cells from being modified.

To set editing permissions on selected cells in code:

- Use the **SetProtection** method to change the locked status of the currently selected cells.
- Use **EnableProtection** to enable protection for the entire worksheet.

When a worksheet contains locked cells, ENTER, SHIFT-ENTER, TAB, and SHIFT-TAB advance the selection to the next unlocked cell.

Limiting Data Entry

Limiting Formula Entry

Displaying Formulas

It is often convenient to display formula text instead of the values they produce. Setting **ShowFormulas** to True, causes the worksheet to display formula text instead of formula results. Displaying formula text can help you debug formula- related problems.

The following example enables and disables the display of formulas.

```
F1Book1.ShowFormulas = TRUE 'Displays formulas

F1Book1.ShowFormulas = FALSE 'Displays formula text
```

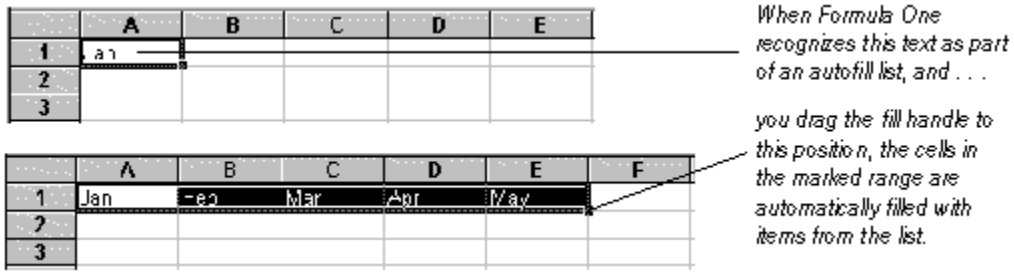
Entering Formulas

Formula Operators

Using AutoFill Lists

If you frequently use lists of names, months, or days of the week in your worksheet, you can let Formula One do some of the work for you by using the autofill feature.

Formula One's default autofill lists contain frequently used series of text such as months, and days of the week. When you enter one of the elements in these lists and drag the autofill handle, Formula One enters the rest of the data from the list as needed to fill the range you mark.



The multi-line entry is placed in cell A1. Row 1 and column A have been resized so the entry is not cropped.

Once Formula One has recognized the text as an item from an autofill list, pressing Tab puts the next list item in the next cell to the right, or Enter puts the next list item in the next cell below.

Using AutoFill Lists

Adding AutoFill Lists

Deleting AutoFill Lists

Worksheet Data Types

Entering Constant Values

Adding AutoFill Lists

You can also add your own autofill lists through the Workbook Designer, or through code.

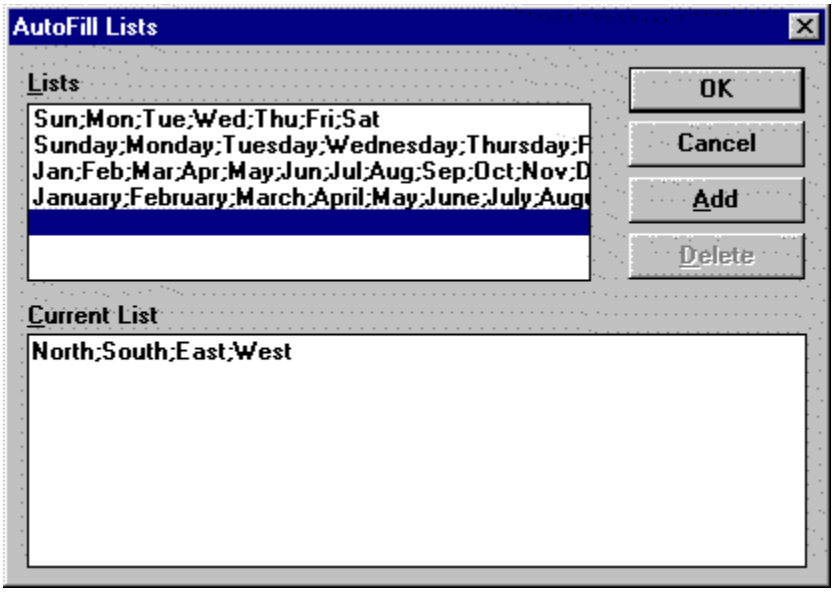
To add a new autofill list in the Workbook Designer:

1. Select Auto Fill Lists from the Data menu to display the AutoFill Lists dialog box.

   By default, a new, empty line is selected in the Lists list box.

2. Type your new autofill list in the Current List list box, separating each item with a semi-colon.

3. Click Add.



Select the empty row at the end of the list and then type your new list in the Current List list box.

To add a new autofill list in code:

- Use the **AutoFillItemsCount** property to determine how many autofill lists there are. Then increment by one and use the **AutoFillItems** property to specify the new list. The following example illustrates:

```
F1Book1.AutoFillItems (F1Book1.AutoFillItemsCount+1) = "First Grade, Second Grade, Third
                Grade"
```

Note Values recognized as data cannot be used as autofill list items, such as 1973;1974, 1a;1b, 1%;2%.

Using AutoFill Lists

Deleting AutoFill Lists

Deleting AutoFill Lists

You can also delete an existing autofill list.

To delete an autofill list in the Workbook Designer:

1. Select AutoFill Lists from the Data menu to display the AutoFill Lists dialog box.
2. Select the list you want to edit or delete from the Lists list box.
3. Click Delete.

To manipulate an autofill list in code:

- Get the current string of text that makes up a list by returning the value of the **AutoFillItems** property.

```
list = F1Book1.AutoFillItems(2)
```

- Replace a list by setting the **AutoFillItems** property.

```
list = "1st Qtr;2nd Qtr;3rd Qtr;4th Qtr"
F1Book1.AutoFillItems(2) = list
```

- Delete a list with the **DeleteAutofillItems** method.

```
F1Book1.DeleteAutoFillItems (F1Book1.AutoFillItemsCount -2)
```

Using AutoFill Lists

Adding AutoFill Lists

Validating Data

Another means of limiting data entry is to specify a validation rule for a cell. A validation rule consists of a formula to test, and text to display if the validation fails. If the formula returns True, the value is entered. If the formula returns a text string, the string is displayed and the value is not entered. If the formula returns False, the value is not entered and the validation text is displayed in an error dialog box.

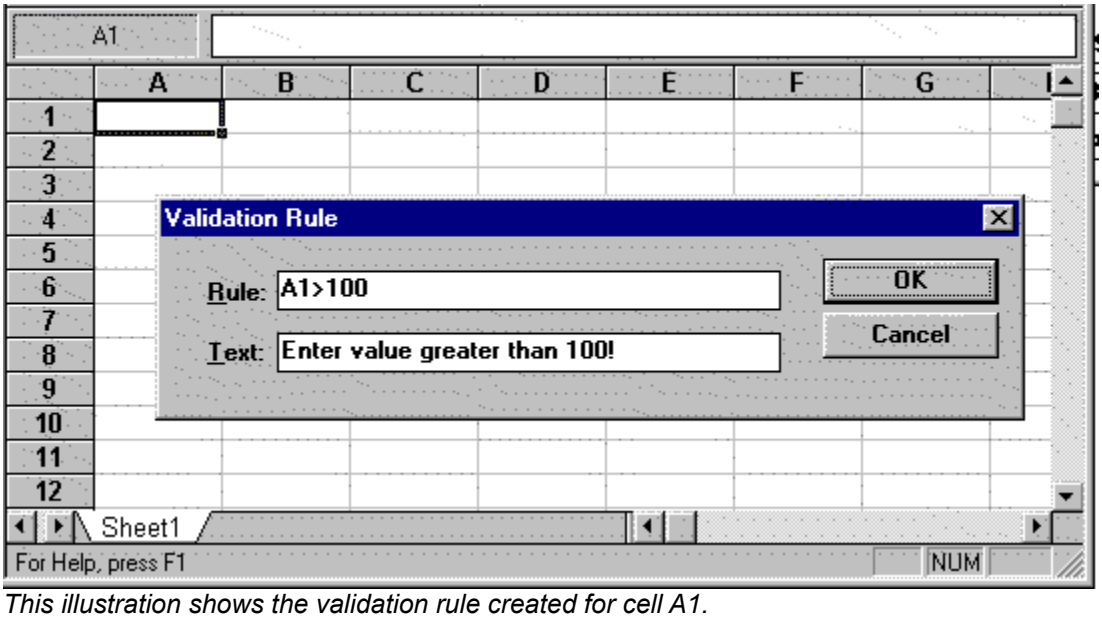
For example, you can limit the range of values a user can enter in a cell by creating a rule that fails if the user enters a number under 100 and displays the message "Enter a value greater than 100."

You can use relative references in validation rules. These references are considered to be relative to the active cell. This allows a validation rule to be properly applied to an entire range.

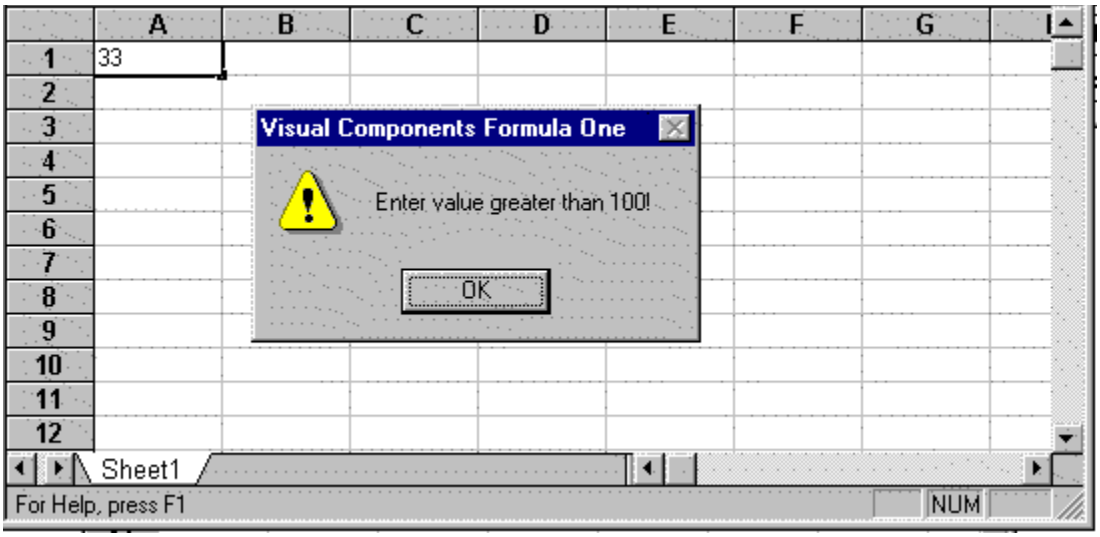
Note Validation rules are only checked data is entered manually, or through use of the **Entry** or **EntryRC** property. Any other way of entering data, such as selecting a value from a check box, bypasses validation.

To create a validation rule in the Workbook Designer:

1. Select the cell for which you want to create a validation rule.
2. Select Validation Rule from the Format menu to display the Validation Rule dialog box.
3. Enter a validation formula in the Rule text box.
4. Enter text to be displayed if the data fails the validation test in the Text box.
5. Click OK.



This illustration shows the validation rule created for cell A1.



This illustration shows the message displayed when the data entered in cell A1 fails the validation rule.

To work with validation rules in code:

- Use the **GetValidationRule** method to return the validation rule associated with the specified cell.
- Use the **SetValidationRule** method to create, or edit the validation rule associated with a cell.
- Use the **ValidationRuleDlg** method to display the Validation Rule dialog box.
- Use the **ValidationFailed** event to respond progammatically when data fails, you can even change the value of the data within the **ValidationFailed** event and attempt to revalidate.

Limiting Data Entry

The Validation Formula

Worksheet Data Types

The Validation Formula

The validation formula follows basically the same rules as those for defined names. It must also be a worksheet formula that evaluates to True or False. Following are a number of examples of validation formulas.

```
SUM(A6:A7) > A5

AND(A6>1, A6 <100)

IF (A7>1,A7<100,A7>0)

OR(ISLOGICAL (A7), A7=1,A7=0)
```

Validating Data

Worksheet Data Types

Cells can contain two types of information - constant values and formulas.

- Constant values are numbers, including dates and times, logical values, error values, and text.
- Formulas are groups of constant values, cell references, names, functions, and operators that result in a new value when calculated or evaluated.

Entering Constant Values

Entering Formulas

Entering Functions

Entering Constant Values

Numbers.Numeric entries can contain numeric characters (e.g., 1, 2, 3, 4, 5, 6, 7, 8, 9, and 0) and the special characters (e.g., +, -, (,), /, $, %, ., E, and e).

- Negative numbers can be preceded by a minus sign or enclosed in parentheses.
- Commas can be included in numeric entries as thousands separators.
- Numeric entries containing leading dollar signs are formatted as currency.
- Numeric entries containing trailing percent signs are formatted as percentages.

Formula One accepts numeric entries as fractions. If the fraction contains a leading integer (e.g., 1 1/3) it can be entered directly. If there is no leading integer, the fraction should be preceded by a zero (e.g., 0 2/3).

Numbers larger than the cell in which they are entered are converted to scientific notation unless a specific format is applied.

Use the **SetColWidthAuto** method to automatically set the column width to the correct size for all data in the column. The following code automatically sets the widths of columns 1 through 10.

```
F1Book1.SetColWidthAuto -1, 1, -1, 10, False
```

Dates and Times. Dates and times are automatically recognized by Formula One. They are entered in the cell as values and automatically formatted. The following date and time formats are automatically recognized.

Entered	Format Assigned
3/15/94	m/d/yy
15-Mar-94	d-mmm-yy
15-Mar	d-mmm
Mar-94	mmm-yy
9:55 PM	h:mm AM/PM
9:55:33 PM	h:mm:ss AM/PM
21:55	h:mm
21:55:33	h:mm:ss
3/15/94 21:55	m/d/yy h:mm

Text. Text is any set of characters that Formula One does not recognize as a number, date, or time.

Text that is wider than a cell ordinarily spills over into the cell immediately to the right. You can specify that text should wrap within the cell by enabling word wrap in your alignment format settings.

Logical and Error Values. Logical and error values are not normally entered directly in cells; they are usually the result of a formula. However, entering these values can be useful for testing formulas.

The logical values that can be entered are TRUE and FALSE. The error values that can be entered are #N/A, #VALUE!, #REF!, #NULL!, #DIV/0!, #NUM!, and #NAME?.

Worksheet Data Types

Entering Formulas

Cell References

Entering Formulas

Formulas are the basic building blocks for analyzing and calculating worksheet data. A formula is a string containing numbers, operators, worksheet functions, cell references, and names. A formula can contain as many as 1024 characters.

- When you manually enter a formula in a worksheet, you must begin the entry with an equal sign (=). Formula One recognizes this entry as a formula.
- When entering a formula using the **Formula** and **FormulaRC** properties, exclude the leading equal sign. These entities expect strings.

Numbers in formulas can be followed by a percent sign (%). Numbers with trailing percent signs are treated as percentages (e.g., 100% is evaluated as 1).

If text is encountered when a number is expected, the text is converted to a number. For example, the formula 1 + "3" returns 4, because "3" is converted to a number. If the text cannot be converted to a valid number (e.g., 1 + "Text"), #VALUE! is returned.

Likewise, if a number is encountered when text is expected, the number is converted to text. The formula "The number is "&3 converts to the text string "The number is 3".

The value TRUE always converts to 1; while FALSE converts to 0. If a number is encountered when a logical value is expected, a zero is converted to FALSE. All other numbers are converted to TRUE. If text is encountered when a logical value is expected, "TRUE" is converted to TRUE; "FALSE" is converted to FALSE. All other text returns #VALUE!.

Dates and times are recognized and converted to their serial values. For example, "10/10/94" - "10/1/94" equals 9.

Limiting Formula Entry

Displaying Formulas

Entering Constant Values

Formula Operators

Cell References

Worksheet Errors

Built-In Worksheet Functions

Using Names

Calculating Worksheets

Formula Operators

When creating formulas, Formula One provides a set of operators for specifying the type of calculation or evaluation to be performed on the formula data. The following table lists the formula operators.

Operator Type	Operator	Description
Arithmetic	+	Addition
	-	Subtraction
	/	Division
	*	Multiplication
	%	Percentage
	^	Exponentiation
Text	&	Concatenation
Comparison	=	Equal to
	>	Greater than
	<	Less than
	>=	Greater than or equal to
	<=	Less then or equal to
	<>	Not equal to
Reference	:, .., .	Range - produces a reference that includes all the cells between the two references (e.g., A1:A5 includes cells A1 and A5 and all cells in between).
	,	Union - produces one reference that includes the two references (e.g., A1:A10,C1:C10).

Entering Formulas

Operator Precedence

Operator Precedence

When combining operators in a formula, Formula One uses a specific order of precedence to calculate the formula. The following table lists the order of precedence for formula operators.

Operator	Description
( )	Parentheses
:, .., .	Range
,	Union
-	Negation (single operand)
%	Percentage
^	Exponentiation
* and /	Multiplication and Division
+ and -	Addition and Subtraction
&	Text concatenation
= < > <= >= <>	Comparison

Operators of like precedence are evaluated left to right. Parentheses should be used when it is necessary to change the order of evaluation. The following example illustrates how the result of a formula can be altered by adding parentheses to change the order of precedence.

Formula	Result
1+2*37	75
(1+2)*37	111

As illustrated in the previous table, the multiplication operator (*) has higher precedence than the addition operator (+). It is evaluated first unless parentheses are used to force the addition to take place first.

Entering Formulas

Formula Operators

Cell References

A reference identifies a cell by referring to the row and column coordinates of the cell. References are based on the row and column headings. For example, A1 refers to the cell at the intersection of row 1 and column A. References can be used in formulas to access data from a worksheet.

A range of cells is specified by placing a colon (:) between two cell references. For example, the reference A1:C3 refers to the range anchored by cells A1 and C3. The range includes all cells in columns A, B, and C of rows 1, 2, and 3.

Entering Formulas

Absolute and Relative References

References to Other Worksheets

References to Multiple Worksheets

External References

Automatically Entering Cell References

Using Names

Absolute and Relative References

There are two types of cell references - relative and absolute.

- Relative references point to a cell based on its relative position to the current cell. When the cell containing the reference is copied, the reference is adjusted to point to a new cell with the same relative offset as the originally referenced cell.
- Absolute references point to a cell at an exact location. When the cell containing the formula is copied, the reference does not change. Absolute references are designated by placing a dollar sign ($) in front of the row and column that is to be absolute.

References can be part absolute and part relative. These are called mixed references. The following table lists the reference types.

Reference	Type
A1	Relative reference pointing to cell A1.
$A$1	Absolute reference pointing to cell A1.
$A1	Absolute column reference, relative row reference pointing to cell A1.
A$1	Relative column reference, absolute row reference pointing to cell A1.

The reference operators can be used to specify multiple ranges in the same reference. For example, A1:C1,A10:C10 specifies the three cells A1, B1, and C1 and the three cells A10, B10, and C10. The formula =SUM(A1:C1,A10:C10) adds the values in all six cells.

Cell References

References to Other Worksheets

References to Multiple Worksheets

External References

Automatically Entering Cell References

References to Other Worksheets

You can reference cells in other worksheets.

To reference a worksheet in the same workbook, use the following syntax:

```
Sheet3!A1
```

To reference a worksheet in a different workbook, use the following syntax:

```
[F1Book1]Sheet1!A1:B2
```

Workbooks are referenced by the value of their Title property, and must be loaded in another control for the reference to work.

Cell References

Absolute and Relative References

References to Multiple Worksheets

External References

Automatically Entering Cell References

References to Multiple Worksheets

You can also refer to multiple worksheets, or ranges in multiple worksheets. The following example references cell A1 in Sheet1 and cell A1 in Sheet2.

```
Sheet1C:Sheet2!A1
```

Important Worksheets must be referenced in index order. For example, the reference to the sheet indexed 1 must come before the reference to the sheet indexed 2. Remember that the worksheet index is usually different that the sheet name that appears on the sheet tab. Worksheets are indexed from left to right, beginning with 1.

The following syntax references the range A1 to B2 in Sheet1 and the range A1 to B2 in Sheet2.

```
Sheet1:Sheet2!A1:B2
```

You can also reference multiple worksheet ranges in a different workbooks by referencing the workbook name at the beginning of the syntax.

```
[F1Book1]Sheet1:Sheet2!A1
```

```
[F1Book2]Sheet1:Sheet2!A1:B2
```

Cell References

Absolute and Relative References

References to Other Worksheets

External References

Automatically Entering Cell References

Solving Circular References

External References

References can point to cells in other workbooks. This type of reference is called an external reference. An external reference is created by placing a workbook name in brackets, followed by the worksheet name and an exclamation point, and finally a cell reference. The following table shows examples of external references.

Reference	Type
[Sales]Sheet1!A1	Relative reference pointing to cell A1 in the first worksheet of a workbook titled Sales.
[FY91]Sheet2!$A$1	Absolute reference pointing to cell A1 in the second worksheet of a workbook titled FY91.
[Q1]Sheet1:Sheet2!$A1	Absolute column reference, relative row reference pointing to cell A1 in the first and second worksheets in a workbook titled Q1.
[Store1]Sheet1:Sheet4!A1:F1	Relative row and column reference, pointing to the range A1 to F1 in a workbook titled Store1.

Cell References

Absolute and Relative References

References to Other Worksheets

References to Multiple Worksheets

Automatically Entering Cell References

Solving Circular References

Automatically Entering Cell References

Cell references can be automatically entered as you enter a formula.

To automatically enter a cell reference:

1. Enter the formula to the point of the range reference.
2. With the mouse, select the cell or range you want to reference.

The reference of the range you select is automatically placed in the formula.

When you enter a cell reference in this manner, Formula One assumes it is a relative reference.

Cell References

Absolute and Relative References

References to Other Worksheets

References to Multiple Worksheets

External References

Solving Circular References

Worksheet Errors

When a formula cannot be properly calculated, an error is returned in the cell. The following table lists the errors that can be generated.

Error	Cause
#ARRAY_FORMULA!	Formula One read in a file that contained an array formula. Since this feature is not supported in Formula One, this error value is placed in the cell which used to contain the array formula.
#DIV/0!	Divide by zero. May be caused by a reference to a blank cell or a cell containing zero.
#N/A	No value is available. May be caused by inappropriate values in the formula or a reference to a cell containing the #N/A value.
#NAME?	Name is not recognized. May be caused by a user defined name that is not defined.
#NULL!	Null intersection. An intersection of two ranges was defined that does not intersect.
#NUM!	Number problem. May be caused by inappropriate numbers in functions, an iteration that cannot solve for a value, or a formula that results in a number too large or too small to represent.
#REF!	Reference error. May be caused by referring to a cell that was deleted.
#VALUE!	Wrong argument type. May be caused by entering text where a number was expected, or supplying a range to an operator or function that was expecting a single value.

Entering Formulas

Built-In Worksheet Functions

Understanding Functions

Built-In Worksheet Functions

Formula One contains a set of 130 built-in worksheet functions that provide the ability to perform complex calculations with very little work.

Worksheet functions:

- calculate and evaluate data.
- can be used alone or in a formula.
- are entered directly in the worksheet.

Like formulas, worksheet functions return data to the cell in which they are entered.

Each function performs a specific calculation. The SQRT function is an example of a built in function. With this function, you can easily calculate the square root of a number. The following example calculates the square root of 118:

```
=SQRT(118)
```

Understanding Functions

Entering Functions

Nesting Functions

Entering Arguments

Syntax Errors

Calculating Worksheets

Understanding Functions

Most worksheet functions are composed of keywords and arguments. Every worksheet function contains a keyword, but not all functions require arguments.

- The keyword identifies the function and tells the worksheet what type of calculation or evaluation is performed. Each function keyword is unique.
- Arguments provide the data for the function to calculate or evaluate. The arguments for a function immediately follow the function keyword and are enclosed in parentheses.

Built-In Worksheet Functions

Entering Functions

Nesting Functions

Entering Arguments

Syntax Errors

Using Names

Calculating Worksheets

Entering Functions

When entering functions in a worksheet, all functions are preceded by an equal sign (=). The leading equal sign tells the worksheet that the following information is to be evaluated or calculated.

The function keyword follows the equal sign. It can be entered in lowercase or uppercase characters. After the function is entered, the worksheet records the function keyword in uppercase characters, regardless of how it was entered.

If a function requires multiple arguments, the arguments are separated by commas. Some functions contain optional arguments. If you omit an optional argument, a default value is assumed for the argument.

Functions that do not require arguments still require a set of parentheses following the function keyword.

Built-In Worksheet Functions

Understanding Functions

Nesting Functions

Entering Arguments

Syntax Errors

Nesting Functions

A function can be used as an argument for another function. When a function is used in this manner, you are nesting functions. The nested function must return the appropriate type of data for the function in which it is nested. You must also provide the necessary arguments for the nested function.

In the following example, the AVERAGE function is used as an argument for the SUM function. In this case, AVERAGE is nested in SUM.

```
=SUM(5.23, 6.82, AVERAGE(2.45, 5.62, 7.74), 8.95, 9.01)
```

Built-In Worksheet Functions

Understanding Functions

Entering Functions

Entering Arguments

Syntax Errors

Entering Arguments

The arguments for a function can be:

- Numerical values
- Logical values
- Text strings
- Error values
- References to cells or ranges

For most arguments, you can substitute a cell or range reference for the data required by an argument. For example, if an argument requires a number, you can substitute a reference to a cell that contains a number. The number in the referenced cell is used in the calculation of the function. The data in the referenced cell must be appropriate for the argument for which it is used.

Built-In Worksheet Functions

Understanding Functions

Entering Functions

Nesting Functions

Syntax Errors

Syntax Errors

If the worksheet function you enter contains syntax errors, Formula One does not allow the function to be entered. You must correct the errors before proceeding with other tasks.

Built-In Worksheet Functions

Understanding Functions

Entering Functions

Nesting Functions

Entering Arguments

Using Names

User defined names are an easy way to identify a cell, a group of cells, a value, or a formula. For example, the formula "= Sales - Expenses" is much clearer than "=A10 - A6".

You can also use names to identify constants and formula expressions. For example, you might define the name LightSpeed as 186000. You could then use the name LightSpeed in all your formulas. Or, you could define the name SqRtTwo as the formula SQRT(2).

To define names in the Workbook Designer:

1. If you are naming a range, select it.
2. Choose Defined Name from the Data'menu.
3. Enter the name you want to define.
4. Enter the formula that describes the name.

   If this is a range, enter a range reference. If this is a formula, enter the formula.
5. Click Add to add the new name to the list.

- You can delete any name from the list by selecting it in the list and clicking Delete.

To define names in code:

- Use the **DefinedName** property. The following code uses this property to define a name.

```
F1Book1.DefinedName ("Sales") = "$A$10"
```

This example defines the name "Sales" as $A$10. The name "Sales" can then be used in formulas instead of the reference.

Formula One has a set of built-in names. These names are used by the print functions. The built-in names are listed in the following table.

Built in Name	Purpose
Print_Area	Defines the print area used during printing. This name can contain one or more ranges (e.g., A1:C3,A11:C13).
Print_Titles	Defines the row and column titles that are printed on each new page. Entire rows and columns must be selected when you define this name.

Cell References

Calculating Worksheets

Formula One calculates cells in natural order. In natural order calculation, formulas are calculated in such a way that all dependencies are calculated before their dependents. This ensures that the formula results are always correct.

When a worksheet is edited, formula references are adjusted so they point to the correct cells. Then, Formula One determines the natural order of the formulas.

When a change is made to a cell, the formulas are recalculated to keep all worksheets in the workbook current, ensuring that data is always valid.

Setting Automatic Recalculation

Solving Circular References

Setting Automatic Recalculation

Normally, automatic recalculation is enabled. In this mode, the worksheet is recalculated each time a cell is changed and system processing is idle.

For moderate sized worksheets, recalculation operations happen in a fraction of a second. But for large worksheets or situations where many cells are changed by code, this reorganization and recalculation process can slow system processing.

In these situations, it is sometimes desirable to disable automatic recalculation while your code operates on the worksheet. Automatic recalculation can be disabled with the **AutoRecalc** property. After the completion of an operation, automatic recalculation can be enabled and the worksheet updated.

Calculating Worksheets

Solving Circular References

Solving Circular References

There are some circumstances where a formula refers to its own cell, either directly or indirectly. This is called a circular reference. To solve a formula that contains a circular reference, iteration must be used. Iteration is the process of repeatedly calculating a worksheet until a specific condition is met.

Formula One supports iteration using the **SetIteration** and **CalculationDlg** methods. These methods allow you to specify the maximum number of iterations and the maximum change between iterations. The iteration continues until one of those two conditions is met.

The following example includes a circular reference:

Suppose your small business has 10,000 shares of stock owned by four shareholders. You decide to let a fifth shareholder enter your partnership. In return for his investment, you give him 10 percent of the company. How many more shares will the company have to issue to give the new investor 10% of the company?

The following illustration shows the results of this example as it is entered in a worksheet.

	A	B	C	D
1	Old Shares	10,000		
2	Total Shares	= Old Shares + New Shares		
3	New Shares	= Total Shares * 10%		
4				

	A	B	C	D
1	Old Shares	10,000		
2	Total Shares	11,111		
3	New Shares	1,111		
4				

The formulas in B2 and B3 create a circular reference in this example worksheet.

The first worksheet shows the formula text, the second worksheet shows the results of the formulas.

Calculating Worksheets

Setting Automatic Recalculation

Cut, Copy, and Paste Methods

Ranges of data can be edited using one of several editing methods. Formula One automatically adjusts cell references when cells are moved. Thus, the integrity of worksheet formulas remains intact.

Formula One maintains its own internal clipboard and also supports text on the Windows clipboard. The internal clipboard is more flexible than the Windows clipboard. The internal clipboard retains formulas and allows cell references to be adjusted when cells are pasted. The Windows clipboard only holds text, and formatting; cell references are not maintained by the Windows clipboard.

The following table describes the methods that interact with the clipboards.

Method	Operation
ClearClipboard	Clears the internal clipboard.
CopyAll	Copies the contents of the active worksheet in the specified view to the active worksheet in the current view.
EditCopy	Copies the current selection to the internal clipboard and the Windows clipboard (in text format only). If there is more than one selection, only the first selection is copied.
EditCut	Cuts the current selection to the internal clipboard. If there is more than one selection, only the first selection is cut.
EditPaste	Pastes the contents of the internal clipboard to the current selection. If the internal clipboard is empty, text is pasted from the Windows clipboard. You can also paste tab-delimited blocks of data.
EditPasteValues	Pastes values from the clipboard to the current worksheet selection. Any formatting applied to the values is ignored. In addition, only formula results are pasted; formulas are ignored.

- If you cut a cell to which formulas refer, the formula references are maintained while the cell remains in the clipboard. If the cell is subsequently pasted, references in the original formulas are adjusted to point to the cell's new location.
- If a cell containing a formula is copied and subsequently pasted, its relative references are adjusted to point to a new location.

Copying Data Across Ranges

Copying Data Interactively

Interactively Copying Ranges

Moving Data

Inserting Cells, Rows, and Columns

Clearing and Deleting Cells, Rows, and Columns

Copying Data Across Ranges

Four methods copy data within and between worksheets. The following table describes these methods.

Method	Operation
EditCopyDown	Copies the top row of the selection down. Relative references are automatically adjusted.
EditCopyRight	Copies the left column of the selection right. Relative references are automatically adjusted.
CopyRange	Copies a range from one range to another, within the same workbook or between workbooks.
CopyRangeEx	Copies a range from one worksheet to another, within the same workbook or between workbooks.

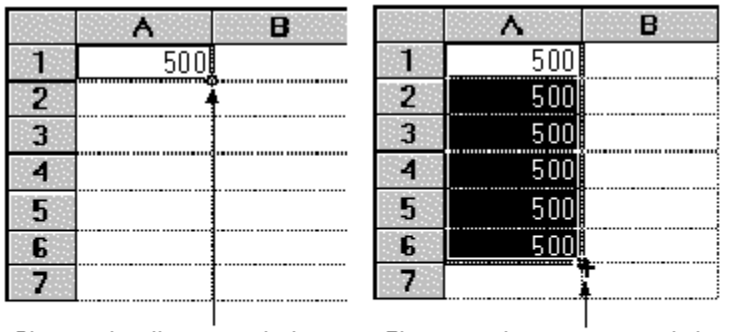
Cut, Copy, and Paste Methods

Copying Data Interactively

Interactively Copying Ranges

Copying Data Interactively

You can copy data interactively by dragging the copy handle of a selection. The copy handle is the small knob in the lower right corner of a selection. When you copy data using the copy handle, the pointer changes to a small crosshair.



In this illustration, the number in A1 is copied to the cells below when the copy handle is dragged downward.

You can disable the user's ability to copy data by setting the **AllowFillRange** property to False. This disables interactive data copying.

Cut, Copy, and Paste Methods
Copying Data Across Ranges
Interactively Copying Ranges

Interactively Copying Ranges

You can copy data interactively by dragging a selected range in a worksheet.

To interactively copy a range:

1. Position the pointer on the border of the selection you want to copy.

   When placed on the selection border, the pointer changes to an arrow.

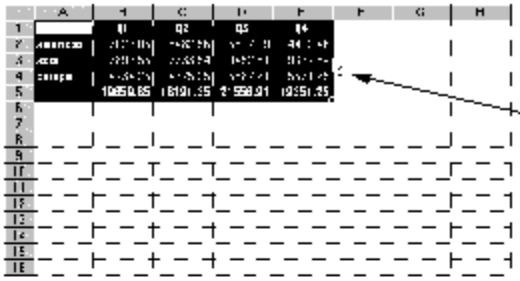
2. Press the CTRL key.

3. With the pointer on the selection border, click and drag the range.

   An outline of the selected range moves as you drag the pointer.

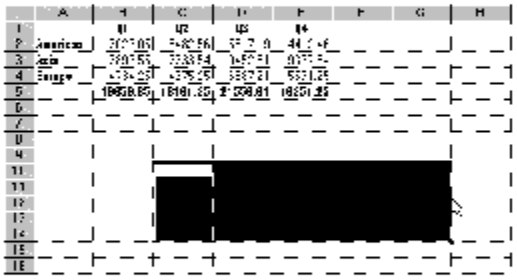
4. Release the mouse button at the location where you want to place the copied range.

The original range is not moved. The following illustration shows the steps required to interactively copy a range.

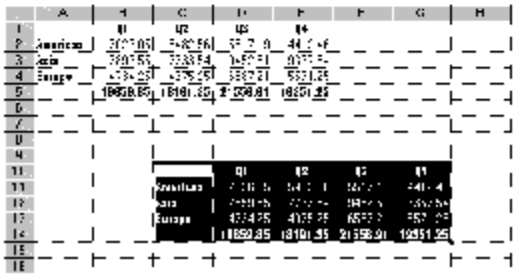


The data in the selected range, A1:E5, can be copied interactively.

The pointer appears as an arrow when positioned on a selection border.



When you click and drag a selection border while pressing CTRL, the highlighted area moves to indicate the placement of the copied range.



The data in the range is copied to its new location when you release the mouse button.

Cut, Copy, and Paste Methods
Copying Data Across Ranges
Copying Data Interactively

Moving Data

There are several ways you can move ranges of data. The easiest way is to use the **MoveRange** method. When you use this method, the integrity of formula cell references is maintained.

If there is special processing that must be performed when data is moved, you can use a loop in C or C++ code to move the data. However, cell references are not adjusted using this technique.

Copying Data Across Ranges

Interactively Moving Ranges

Interactively Moving Ranges

In addition to interactively copying ranges, you can move ranges interactively by clicking and dragging a selection.

To interactively move a range:

1. Position the pointer on the border of the selection you want to move.

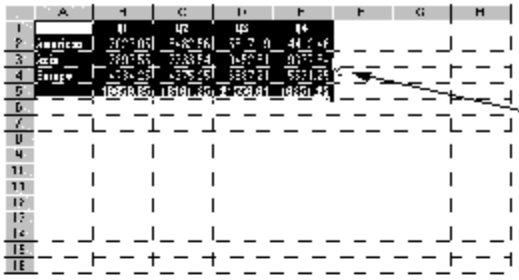
   When placed on the selection border, the pointer changes to an arrow.

2. With the pointer on the selection border, click and drag the range.

   An outline of the selected range moves as you drag the pointer.

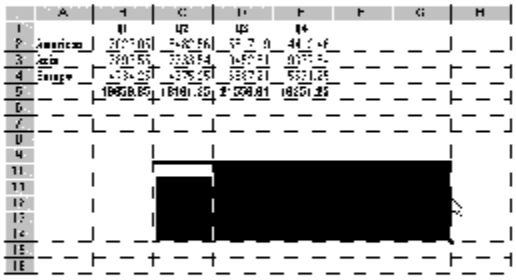
3. Release the mouse button at the location where you want to place the selected range.

The following illustration shows the steps required to interactively move a range.

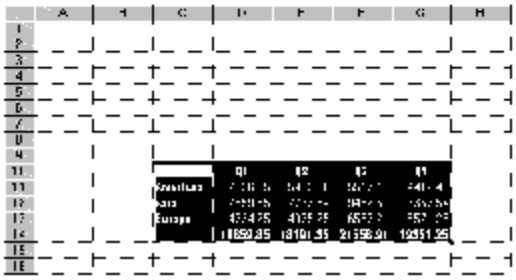


The data in the selected range, A1:E5, can be moved interactively.

The pointer appears as an arrow when positioned on a selection border.



When you click and drag a selection border, the highlighted area moves to indicate the placement of the moved range.



The data in the range is moved to its new location when you release the mouse button.

You can disable the user's ability to move data by setting **AllowMoveRange** property to False. This disables interactive data moving.

Moving Data

Inserting Cells, Rows, and Columns

You can insert cells, rows or columns in the Workbook Designer or through code.

To insert new cells in the Workbook Designer:

1. Select one of the following, depending on what you want to insert.

- **Range**. Select a range of cells in the size and position you want to insert new cells.
- **Row**. Select the row where you want the new row inserted by clicking on the row heading. Select as many rows as you want to insert. To select additional row headings, hold down Shift when you click on the heading. For example, to insert 3 rows, select three rows.
- **Column**. Select the column where you want the new column inserted by clicking on the column heading. Select as many columns as you want to insert. To select additional column headings, hold down Shift when you click on the heading. For example, to insert 3 columns, select three columns.

2. Select Insert from the Edit menu.

If you insert a range or columns, any existing data is shifted right. If you insert columns, any existing data is shifted down.

To insert new cells in code:

- Use the **InsertRange** method to insert new cells in a worksheet. You supply a range where new cells are inserted and specify how the current cells in that range should be shifted to make room for the new cells.

  The following example inserts a two by two block of cells starting at B2. The current cells in the range B2:C3 are shifted downward to make room for the new cells.

```
F1Book1.InsertRange 2, 2, 3, 3, F1ShiftVertical
```

- Use the **EditInsert** method to insert cells, rows, and columns. You specify whether rows, columns, or cells should be inserted. This method call uses the currently selected range to determine how many rows, columns, or cells to insert.

  When new cells are inserted, cell references in formulas are adjusted so the formulas remain correct.

  The next four examples assume the range A4:B5 is selected (a two by two range). In the following code, data in all columns and rows 4 and below is shifted down two rows to allow room for the inserted cells.

```
F1Book1.EditInsert F1ShiftRows
```

  The following code shifts all data in the worksheet right two columns to allow room for the inserted cells.

```
F1Book1.EditInsert F1ShiftCols
```

  In the following code, data in all columns of rows 4 and 5 is shifted right two columns to allow room for the inserted cells

```
F1Book1.EditInsert F1ShiftHorizontal
```

  In the following code, data in columns A and B in rows 4 and below is shifted down two rows to allow room for the inserted cells

```
F1Book1.EditInsert F1ShiftVertical
```

Clearing and Deleting Cells, Rows, and Columns

Clearing and Deleting Cells, Rows, and Columns

You can delete or clear cells, rows or columns in the Workbook Designer or through code. Deleting cells removes the cells and shifts the surrounding data to fill the space. Clearing cells, leaves the cells, but deletes the data.

To delete cells in the Workbook Designer:

1. Select the cells, rows, or columns you want to delete.
2. Select Delete from the Edit menu.

   If you delete a range or columns, any data to the right of the deleted cells is shifted left. If you delete rows, any existing data below the deleted rows is shifted up.

To clear cells in the Workbook Designer:

1. Select the cells, rows, or columns you want to clear.
2. Select Clear from the Edit menu to display the Clear dialog box.
3. Select All, Formats, or Values, depending on what you want to clear from the cell. All clears the value and its formatting. Formats leaves the value, but clears any formatting. Values clears only the value.
4. Click OK.

To delete or clear cells in code:

Several methods delete and clear data. The following table lists these methods.

Method	Operation
EditDelete	Deletes the current selection.
DeleteRange	Deletes the specified range.
EditClear	Clears the current selection.
ClearRange	Clears the specified range.

- **EditDelete** is similar to the **EditInsert** method. For **EditDelete**, you specify whether cells, rows, or columns should be deleted. The number of cells, rows, or columns deleted is determined from the current selection. For example, to delete rows (based on the current selection), you could use the following code:

```
F1Book1.EditDelete F1ShiftRows
```

If you delete cells (e.g., using **EditDelete** or **DeleteRange**) to which a formula refers, those formulas return a #REF! error because the referenced cells no longer exist.

- To delete a specific range instead of the current selection, use the DeleteRange method. This method allows you to explicitly specify the range to delete. The following code uses this method.

```
F1Book1.DeleteRange 1, 1, 3, 3, F1ShiftRows
```

Clearing a cell can clear the value or format in a cell, or both. You can also specify whether or not formulas are cleared. Clearing does not shift other cells in the worksheet. The cleared cell has a value of zero. Formulas that refer to cleared cells obtain a value of zero from those cells.

- You can use **EditClear** or **ClearRange** to clear a cell or range of cells. The following example clears the current selection.

```
F1Book1.EditClear F1ClearAll
```

Alternately, you can use the following example to clear specific rows or columns instead of the

current selection.

```
F1Book1.ClearRange 1, 1, 3, 3, F1ClearAll
```

Inserting Cells, Rows, and Columns

Sorting Data in Worksheets

You can sort the data in a worksheet and specify the keys by which the data is sorted. **SortDlg** displays a dialog box that allows the user to specify sort keys, sort rows or columns, and ascending or descending sort order. Before using the sort dialog box, a range in a worksheet must be selected. The data in the selected range is the data that is sorted.

You can also sort worksheet data using the **Sort** or **Sort3** methods.

Formatting Worksheets

Formula One supports a rich set of data formatting capabilities. When a worksheet is first created, all cells use the General format. As you enter data in the worksheet, Formula One determines the type of data and applies the appropriate format (e.g., if you enter a date, a date format is applied).

Built-in Number Formats

You can apply a number of built-in number formats in the Workbook Designer by highlighting a range of cells and selecting the format from the Format menu.

The following table lists the built-in number formats and the result after the format is applied to a positive, negative, and decimal number.

Category	Format	3	-3	.3
All	General	3	-3	.3
Currency	$#,##0_);($#,##0)	$3	($3)	$0
	$#,##0_);[RED]($#,##0)	$3	($3) in red	$0
	$#,##0.00_);($#,##0.00)	$3.00	($3.00)	$0.30
	$#,##0.00_);[RED]$(#,##0.00)	$3.00	($3.00) in red	$0.30
Fixed	0	3	-3	0
	0.00	3.00	-3.00	0.30
	#,##0	3	-3	0
	#,##0.00	3.00	-3.00	0.30
	#,##0_);(#,##0)	3	(3)	0
	#,##0_);[RED](#,##0)	3	(3) in red	0
	#,##0.00_);(#,##0.00)	3.00	(3.00)	0.30
	#,##0.00_);[RED](#,##0.00)	3.00	(3.00) in red	0.30
Percent	0%	300%	-300%	30%
	0.00%	300.00%	-300.00%	30.00%
Fraction	# ?/?	3	-3	2/7
	# ??/??	3	-3	3/10
Scientific	0.00E+00	3.00E+00	-3.00E+00	3.00E-01

Obtaining Formatted Text

Custom Formatting

Formatting Rows and Columns

If you format a row or column, that format is applied to all cells in the row or column. When you enter data in a cell in a formatted row or column, the data assumes the designated format.

Formula One allocates memory by rows. Formatting empty rows or columns does not use memory. A format is merely attached to a row or column. Formatting empty ranges is treated differently. If you format a range of empty cells, a group of formatted, empty cells is created. Each new formatted, empty cells consumes memory.

Built-in Number Formats

Obtaining Formatted Text

Custom Formatting

Obtaining Formatted Text

You can obtain the formatted text in a cell by retrieving the value of the **FormattedText** or **FormattedTextRC** properties. These properties return text exactly as it is displayed in the worksheet.

Built-in Number Formats

Formatting Rows and Columns

Custom Formatting

Custom Formatting

In addition to the built-in formats, you can define custom formats. Each custom format can have as many as four sections - one for positive numbers, one for negative numbers, one for zeros, and one for text. Each section is optional, The sections are separated by semicolons. The following example shows a custom format.

```
#,###;(#,###);0;"Error: Entry must be numeric"
```

To define a custom number format in the Workbook Designer:

1. Select a range of cells to be formatted.
2. Choose Custom Number from the Format'menu to display the Custom Format dialog box.
3. Enter a format built from the custom format characters described later in this chapter.

To define a custom format programmatically:

- Use the **NumberFormat** property. The following code sets **NumberFormat** to format numbers in the current selection with two decimal places and negative numbers with parentheses.

```
F1Book1.NumberFormat = ("#,##0.00_);(#,##0.00)
```

- Use **FormatNumberDlg** to display the Custom Format dialog box. This dialog box allows you to select existing formats as well as define custom formats. The selected format is applied to all selections. The following code displays the Custom Format dialog box.

```
F1Book1.FormatNumberDlg
```

The following table lists the format symbols that can be used in a custom format string.

Format Symbol	Description
General	Displays the number in General format.
0	Digit placeholder. If the number contains fewer digits than the format contains placeholders, the number is padded with 0's. If there are more digits to the right of the decimal than there are placeholders, the decimal portion is rounded to the number of places specified by the placeholders. If there are more digits to the left of the decimal than there are placeholders, the extra digits are retained.
#	Digit placeholder. This placeholder functions the same as the 0 placeholder except the number is not padded with 0's if the number contains fewer digits than the format contains placeholders.
?	Digit placeholder. This placeholder functions the same as the 0 placeholder except that spaces are used to pad the digits.
. (period)	Decimal point. Determines how many digits (0's or #'s) are displayed on either side of the decimal point. If the format contains only #'s left of the decimal point, numbers less than 1 begin with a decimal point. If the format contains 0's left of the decimal point, numbers less than 1 begin with a 0 left of the decimal point.
%	Displays the number as a percentage. The number is multiplied by 100 and the % character is appended.
, (comma)	Thousands separator. If the format contains commas separated by #'s or 0's, the number is displayed with commas separating thousands. A comma following a

	placeholder scales the number by a thousand. For example, the format 0, scales the number by 1000 (e.g., 10,000 would be displayed as 10).
E- E+ e- e+	Displays the number as scientific notation. If the format contains a scientific notation symbol to the left of a 0 or # placeholder, the number is displayed in scientific notation and an E or an e is added. The number of 0 and # placeholders to the right of the decimal determines the number of digits in the exponent. E- and e- place a minus sign by negative exponents. E+ and e+ place a minus sign by negative exponents and a plus sign by positive exponents.
$ - + / ( ) : space	Displays that character. To display a character other than those listed, precede the character with a back slash (\) or enclose the character in double quotation marks (" "). You can also use the slash (/) for fraction formats.
\	Displays the next character. The backslash is not displayed. You can also display a character or string of characters by surrounding the characters with double quotation marks (" ").
	The backslash is inserted automatically for the following characters: ! ^ & ` (left quote) ' (right quote) ~ { } = < >
* (asterisk)	Repeats the next character until the width of the column is filled. You cannot have more than one asterisk in each format section.
_ (underline)	Skips the width of the next character. For example, to make negative numbers surrounded by parentheses align with positive numbers, you can include the format _) for positive numbers to skip the width of a parenthesis.
"text"	Displays the text inside the quotation marks.
@	Text placeholder. If there is text in the cell, the text replaces the @ format character.
m	Month number. Displays the month as digits without leading zeros (e.g., 1-12). Can also represent minutes when used with h or hh formats.
mm	Month number. Displays the month as digits with leading zeros (e.g., 01-12). Can also represent minutes when used with the h or hh formats.
mmm	Month abbreviation. Displays the month as an abbreviation (e.g., Jan-Dec).
mmmm	Month name. Displays the month as a full name (e.g., January-December).
d	Day number. Displays the day as digits with no leading zero (e.g., 1-2).
dd	Day number. Displays the day as digits with leading zeros (e.g., 01-02).
ddd	Day abbreviation. Displays the day as an abbreviation (e.g., Sun-Sat).
dddd	Day name. Displays the day as a full name (e.g., Sunday-Saturday).
yy	Year number. Displays the year as a two-digit number

	(e.g., 00-99).
yyyy	Year number. Displays the year as a four-digit number (e.g., 1900-2078).
h	Hour number. Displays the hour as a number without leading zeros (1-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.
hh	Hour number. Displays the hour as a number with leading zeros (01-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.
m	Minute number. Displays the minute as a number without leading zeros (0-59). The m format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.
mm	Minute number. Displays the minute as a number with leading zeros (00-59). The mm format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.
s	Second number. Displays the second as a number without leading zeros (0-59).
ss	Second number. Displays the second as a number with leading zeros (00-59).
AM/PM am/pm A/P a/p	12-hour time. Displays time using a 12-hour clock. Displays AM, am, A, or a for times between midnight and noon; displays PM, pm, P, or p for times from noon until midnight.
[h]	Outputs total number of hours
[m]	Outputs total number of minutes
[s]	Outputs total number of seconds
s.0, s.00, s.000, ss.0, ss.00, ss.000	Outputs fractional part of second.
[BLACK]	Displays cell text in black.
[BLUE]	Displays cell text in blue.
[CYAN]	Displays cell text in cyan.
[GREEN]	Displays cell text in green.
[MAGENTA]	Displays cell text in magenta.
[RED]	Displays cell text in red.
[WHITE]	Displays cell text in white.
[YELLOW]	Displays cell text in yellow.
[COLOR n]	Displays cell text using the corresponding color in the color palette. n is a color in the color palette.
[conditional value]	Each format can have as many as four sections - one each for positive numbers, negative numbers, zeros, and text. Using the conditional value brackets ([ ]), you can designate a different condition for each section. For example, you might want positive numbers displayed in black, negative numbers in red, and zeros in blue. The following string formats a number for these conditions: [>0] [BLACK]General; [<0] [RED]General;

[BLUE]General

The following table shows some examples of custom number formats and numbers displayed using the custom formats.

Format	Cell Data	Display
#.##	123.456	123.46
	0.2	.2
#.0#	123.456	123.46
	123	123.0
#,##0"CR";#,##0"DR";0	1234.567	1,235CR
	0	0
	-123.45	123DR
#,	10000	10
"Sales="0.0	123.45	Sales=123.5
	-123.45	-Sales=123.5
"X="0.0;"x="-0.0	-12.34	x=-12.3
$* #,##0.00;$* -#,##0.00	1234.567	$ 1,234.57
	-12.34	$ -12.34
000-00-0000	123456789	123-45-6789
"Cust. No." 0000	1234	Cust. No. 1234
;;;	Anything	(Not Displayed)
"The End"	123.45	The End
	-123.45	-The End
	text	text
m-d-yy	2/3/94	2-3-94
mm dd yy	2/3/94	02 03 94
mmm d, yy	2/3/94	Feb 3, 94
mmmm d, yyyy	2/3/94	February 3, 1994
d mmmm yyyy	2/3/94	3 February 1994
hh"h" mm"m"	1:32 AM	01h 32m
h.mm AM/PM	14:56	2.56 PM
hhmm "hours"	3:15	0315 hours

Aligning Data

Formula One allows you to specify how data is aligned within a cell. The standard alignment places text along the left edge of the cell and numbers along the right edge of the cell. Logical and error values are centered.

To align text in the Workbook Designer:

1. Select the cells whose contents you want to align.
2. Choose Alignment from the Format menu to display the Alignment dialog box.

   In this dialog box, you can specify the horizontal and vertical alignment of data in the selected cells. In addition, you can specify whether long strings of data can wrap to multiple lines within the cell.

To align text in code:

- Use the **SetAlignment** method to set horizontal and vertical alignment and word wrapping for data in the selected cells.

  To set the alignment in the currently selected ranges, you could use the following code:

```
F1Book1.SetAlignment F1HAlignLeft, FALSE, F1VAlignBottom, 0
```

  In the preceding example, F1HAlignLeft specifies that the cell data is left aligned, FALSE specifies that word wrap is disabled, F1VAlignBottom indicates that text is positioned at the bottom of the cell. The 0 is a placeholder for the orientation argument (not implemented in this version).

- Use **FormatAlignmentDlg** to invoke the Alignment dialog box.

  The following code invokes the Alignment dialog box.

```
F1Book1.FormatAlignmentDlg
```

Changing Row Heights and Column Widths

The width of columns and the height of rows can be changed interactively or set with properties and methods. Interactive column and row sizing can be performed in the Workbook Designer at design time or in a workbook at run time.

Interactively Sizing Rows and Columns

Sizing Rows and Columns with Properties and Methods

Fixing Rows and Columns

Formatting Row and Column Headings

Interactively Sizing Rows and Columns

When you position the pointer on the right edge of a column heading or the bottom edge of a row heading, the pointer changes to a double arrow to indicate that the row or column can be resized. Simply click and drag to resize the column or row.

If multiple rows are selected when you resize a row, all selected rows are resized as you drag a row border. Multiple columns can be resized in the same manner.

You can also set the size of a selected group of columns or rows to match the size of an existing row or column. First, select the group of rows or columns you want to resize, including the row or column whose size you want to match. Then, click the right border of the column header or the bottom border of the row whose size you want to match. The selected rows are resized to match the size of the row or column you clicked.

You can disable interactive sizing of rows and columns by setting the **AllowResize** property to False.

Changing Row Heights and Column Widths

Sizing Rows and Columns with Properties and Methods

Sizing Rows and Columns with Properties and Methods

The following table lists the properties and methods that allow you to size rows and columns.

Property/Method	Operation
ColHidden	Sets or returns the display status of an individual column.
ColWidth	Sets or returns the width of a single column in units of 1/256 of a average character's width in the default font or twips (1/1440th of an inch) depending on the setting of ColWidthUnits.
ColWidthDlg	Displays the Column Width dialog box.
ColWidthTwips	Sets or returns the width of the specified columns in twips.
ColWidthUnits	Sets or returns whether column widths are stored and displayed in twips or character units.
SetColWidth	Sets the width of the specified columns in units of 1/256 of an average character's width in the default font or twips (1/1440th of an inch) depending on the setting of ColWidthUnits.
SetColWidthAuto	Automatically sets the width of the specified columns to accommodate the largest data in the column.
SetColWidthTwips	Changes the width of one or more columns to the specified number of twips.
RowHeight	Sets or returns the height of a single row in twips.
RowHidden	Sets or returns the display status of an individual row.
SetRowHeight	Sets the height of the specified rows in twips (one twip equals 1/1440 inch).
SetRowHeightAuto	Automatically sets the height of the specified rows to accommodate the tallest data in the row.
RowHeightDlg	Displays the Row Height dialog box.

SetRowHeight and **SetColWidth** set the size of one or more rows or columns. For example, the following code sets the height of rows 1 through 10 to 1/2 inch, and the width of columns 1 through 10 (A through J) to 10 characters wide.

```
F1Book1.SetRowHeight 1, 10, 720, FALSE
F1Book1.SetColWidth 1, 10, 2560, FALSE
```

SetColWidthAuto and **SetRowHeightAuto** automatically size rows and columns to accommodate the largest data in the row or column. For example, the following code automatically sets the row and column sizes of rows 1 through 10, and columns 1 through 10 (A through J).

```
F1Book1.SetRowHeightAuto 1, 1, 10, 10, TRUE
F1Book1.SetColWidthAuto 1, 1, 10, 10, TRUE
```

Changing Row Heights and Column Widths

Interactively Sizing Rows and Columns

Fixing Rows and Columns

Rows and columns in a worksheet can be fixed so that they do not scroll when you perform worksheet scrolling. Fixed rows remain stationary when you perform vertical scrolling; fixed columns remain stationary when you perform horizontal scrolling. Fixing rows and columns allows you to place titles in your worksheet that are always displayed.

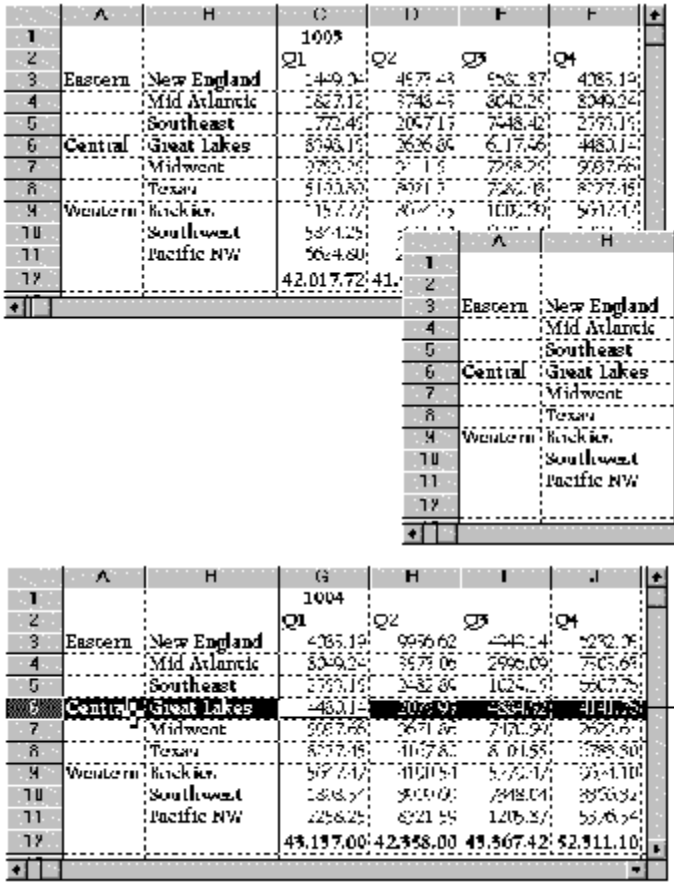
When fixing rows and columns, you specify the starting row or column to fix and the number of rows or columns to fix. Rows and columns preceding the fixed area are not displayed in the worksheet. For example, if you fix rows 4 through 6, rows 1, 2, and 3 are not displayed.

To fix rows or columns in the Workbook Designer:

1. Select the rows or columns you want to fix by clicking on the column or row header. To select multiple rows or columns, hold down Shift while you click.
2. Choose Fix Rows (or Columns) from the Sheet menu, depending on what you have selected.

To fix rows or columns in code:

- Setting the **FixedRow** and **FixedCol** properties specify the starting row and column to fix. **FixedRows** and **FixedCols** determines how many rows or columns are fixed from the first row or column. The following illustration shows a worksheet with fixed columns.



Setting Cell Borders and Colors

Cells and ranges can be formatted with borders, colors, and patterns. These attributes can be set using the Workbook Designer or methods.

Borders can be applied to the top, bottom, left, and right sides of a cell. You can select the type and color of line used for the border. When adding a border to a range, you can place a border around the outside of the range.

When applying colors and patterns to a cell or range, you specify the pattern and foreground and background colors used to fill the cells.

To format cells from the Workbook Designer:

1. Select the cells you want to format.
2. To change cell borders, choose Border from the Format menu to display the Borders dialog box. To change cell colors and patterns, choose Pattern from the Format menu to display the Pattern dialog box.

To format cells from code:

- Use **SetBorder** to format the border, outline, shading, and color for the selected cells. The following code uses this method:

```
F1Book1.SetBorder 2, 1, 1, 1, 1, 1, 3, 4, 4, 4, 4
```

- Use **SetPattern** to format the color and pattern for the selected cells. The following code uses this method:

```
F1Book1.SetPattern 2, 128, 0
```

- Use **FormatBorderDlg** to invoke the Borders dialog box. The following code displays this dialog box:

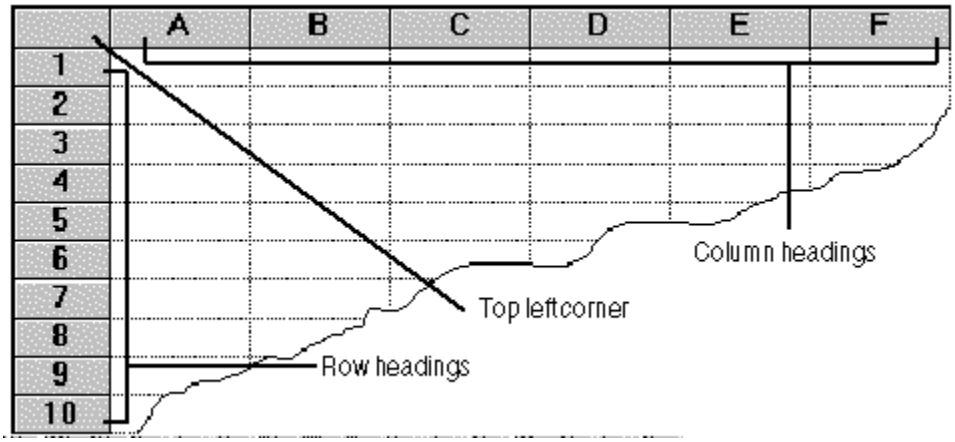
```
F1Book1.FormatBorderDlg
```

- Use **FormatPatternDlg** to invoke the Pattern dialog box. The following code displays this dialog box:

```
F1Book1.FormatPatternDlg
```

Formatting Row and Column Headings

In addition to formatting worksheet cells, many aspects of row and column headings can be formatted. Worksheet headings contain three areas: the row headings, column headings, and the box in the top left corner of the worksheet where the row and column headings intersect. The following illustration highlights these three areas.



Selecting Row and Column Heading Areas
Sizing Row and Column Headings
Setting Row and Column Text

Selecting Row and Column Heading Areas

Row and column headings can be selected interactively or programmatically.

To select headings interactively:

- Press CTRL+Shift and click the heading area.

To select a heading area by programmatically:

- Use **SetHdrSelection**. The following code selects the column heading area.

```
F1Book1.SetHdrSelection FALSE, FALSE, TRUE
```

After a heading area is selected, you can set:

- the alignment of the heading text.
- the font and color of the heading text.
- the pattern and fill color of the heading area.
- the border used to frame heading cells.

Formatting Row and Column Headings

Selecting Row and Column Heading Areas

Sizing Row and Column Headings

Sizing Row and Column Headings

The size of row, column, and top left headings can be set interactively and programmatically.

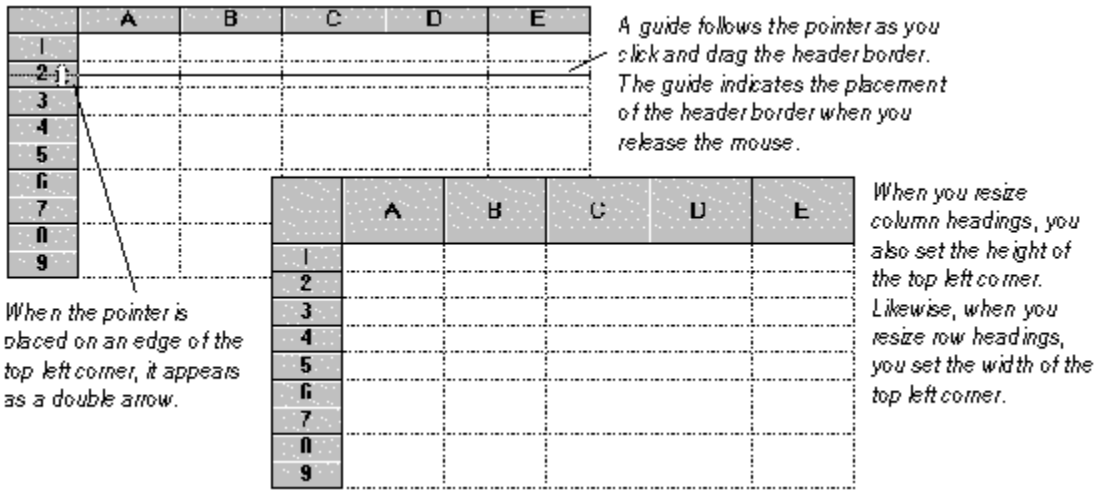
To interactively change the size of column headings:

- Click and drag the bottom edge of the top left corner.

To interactively change the size of row headings:

- Click and drag the right edge of the top left corner.

The following illustration shows interactive resizing of the column headings.



To change the size of column headings programmatically:

- Use **HdrHeight**.

To change the size of row headings programmatically:

- Use **HdrWidth**.

Formatting Row and Column Headings

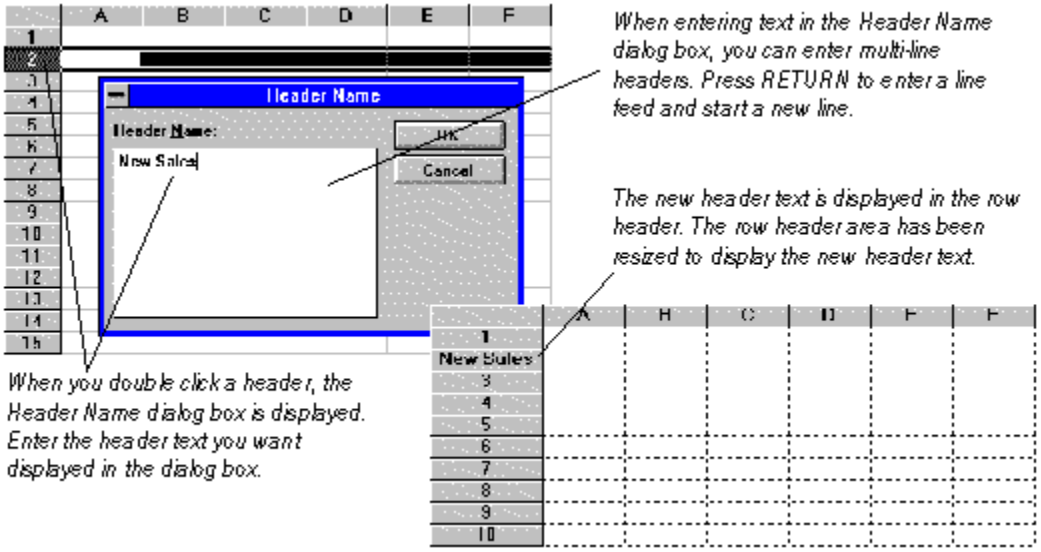
Selecting Row and Column Heading Areas

Setting Row and Column Text

Like other column and row header attributes, the text displayed in header cells can be changed interactively or programmatically.

To interactively change the text for a row or column header:

1. Double click on the header for which you want to enter text to display the Header Name dialog box.
2. Enter one or more lines of text to serve as the header name.



You can control interactive editing of headers by setting the **AllowEditHeaders** property. You can return the value of **AllowEditHeaders** to determine whether interactively editing headers is allowed.

To set or return the text displayed in column headers programmatically:

- Use **ColText**. Use **RowText** to set or return the text displayed in row headers. **TopLeftText** sets or returns the text displayed in the top left corner.

With **ColText** and **RowText**, you must specify the column or row for which you are setting header text. The following example sets the heading for column 4 to "Orders" instead of the default "D".

```
F1Book1.ColText (3) = "Orders"
```

The following illustration shows the result of the example code.

	A	B	Orders	D	E
1					
2					
3					
4					
5					
6					
7					

The header text for column C is replaced with "Orders" by the **ColText** *property.*

Note Rows, columns, and cells are still referred to by their default numbers and letters in functions, properties, and formulas even if the header text for rows and columns has been changed. For example, the cell at the intersection of column B and row 2 is still referred to as B2 even if the header text for row 2 has been set to "New Sales."

Formatting Row and Column Headings

Reading and Writing Files

Formula One can read and write a number of file formats. The following table lists the formats and the associated file name extensions.

Format	Extension	Description
Formula One 3.0	.VTS	Formula One native format.
Formula One 2.x	.VTS	Formula One native format. (New 3.0 features not supported.)
Excel 5.0	.XLS	Excel 5.0 format.
Excel 4.0	.XLS	Excel 4.0 format.
Tabbed-Text	.TXT	Tab-delimited text file including number formatting information.
Tabbed-Text (Values Only)	.TXT	Tab-delimited text without formatting information.

Since Formula One has some features not supported by Excel, files saved in the VTS file format cannot be read by Excel. The XLS format is based on records where each record represents a unique feature or property of the workbook.

If the file you save contains features not supported by Excel, they are removed when the workbook is saved as an XLS file. Likewise, Excel contains features not supported by Formula One. Unsupported features are ignored when Formula One loads an Excel worksheet or workbook.

Important If you load an Excel file that contains features not supported by Formula One, such as charts or function arrays, those features are ignored. If the imported file is then written from Formula One as an Excel file and subsequently read by Excel, those features are omitted and irretrievable.

Formula One cannot read password protected Excel files. If you intend to read files from Excel, they should not be password protected.

The following methods and properties are available for reading and writing files in Formula One applications:

Property/Method	Description
Read	Reads a worksheet from disk.
ReadFromBlob	Reads a worksheet that has been stored in memory in a blob variable.
SaveFileDlg	This dialog box allows you to save the current file in Formula One, Excel, or tabbed text format.
Write	Saves the worksheet to a file.
WriteToBlob	Writes a worksheet to a blob variable.

Using BLOB access

Using BLOB access

A Formula One workbook can also read data from or write data to a memory variable defined as a Binary Large Object (BLOB.) This allows you to store worksheets or workbooks in a database table and later retrieve them from the database table.

To retrieve a worksheet or workbook from a database table:

1. Write code outside Formula One to read a worksheet or workbook from a database table into a BLOB variable.
2. Call Formula One's **ReadFromBlob** method to display that worksheet or workbook in the workbook control.

To store a worksheet or workbook in a database table:

1. Call Formula One's **WriteToBlob** method to copy the worksheet or workbook from the Formula One control to a BLOB variable.
2. Write code outside Formula One to write the worksheet or workbook from the BLOB variable to a database table.

Reading and Writing Files

ABS

Description

Returns the absolute value of a number.

Syntax

ABS (*number*)

Parameter	Description
number	Any number.

Remarks

An absolute value does not display a positive or negative sign.

Examples

These functions both return 1:

```
ABS(-1)
ABS(1)
```

ACOS

Description

Returns the arc cosine of a number.

Syntax

ACOS (*number*)

Parameter	Description
number	The cosine of the angle. The cosine can range from 1 to –1.

Remarks

The resulting angle is returned in radians (from 0 to p). To convert the resulting radians to degrees, multiply the radians by 180/PI( ).

Examples

This function returns 1.05:

```
ACOS(.5)
```

This function returns 1.77:

```
ACOS(–.2)
```

See Also

COS

ACOSH

Description

Returns the inverse hyperbolic cosine of a number.

Syntax

ACOSH (*number*)

Parameter	Description
number	Any number equal to or greater than 1.

Examples

This function returns .62:

```
ACOSH(1.2)
```

This function returns 1.76:

```
ACOSH (3)
```

See Also

ASINH

ATANH

COSH

ADDRESS

Description

Creates a cell address as text.

Syntax

ADDRESS (*row*, *column*, *ref_type* [, *a1*] [, *sheet*])

Parameter	Description	
row	The row number for the cell address.	
column	The column number for the cell address.	
ref_type	The cell reference type. Following are the valid values for this argument.	
	1	Absolute
	2	Absolute row, relative column
	3	Relative row, absolute column
	4	Relative
a1	The reference format. This argument must be TRUE( ) to represent an A1 reference format; Formula One does not support the R1C1 reference format.	
sheet	The name of an external worksheet view control. Omitting this argument assumes that the reference exists in the current spreadsheet.	

Examples

This function returns $F$5:

```
ADDRESS(5, 6, 1)
```

This function returns SALES!F5:

```
ADDRESS(5, 6, 4, TRUE(), “SALES.”)
```

See Also

COLUMN

OFFSET

ROW

AND

Description

Returns True if all arguments are true; returns False if at least one argument is false.

Syntax

AND (*logical_list*)

Parameter	Description
logical_list	A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error #VALUE! is returned.

Examples

This function returns True because both arguments are true:

```
AND(1+1=2, 5+5=10)
```

This function returns False:

```
AND(TRUE(), FALSE())
```

See Also

ROW

NOT

OR

ASIN

Description

Returns the arcsine of a number.

Syntax

ASIN (*number*)

Parameter	Description
number	The sine of the resulting angle, ranging from –1 to 1.

Remarks

The resulting angle is returned in radians (ranging from –p/2 to p/2). To convert the resulting radians to degrees, multiply the radians by 180/PI().

Examples

This function returns –1.57:

```
ASIN(–1)
```

This function returns .41:

```
ASIN(.4)
```

See Also

ASINH

PI

SIN

ASINH

Description

Returns the inverse hyperbolic sine of a number.

Syntax

ASINH (*number*)

Parameter	Description
number	Any number.

Examples

This function returns 2.37:

```
ASINH(5.3)
```

This function returns –2.09:

```
ASINH(-4)
```

See Also

ACOSH

ASIN

ATANH

SINH

ATAN

Description

Returns the arctangent of a number.

Syntax

ATAN (*number*)

Parameter	Description
number	The tangent of the angle.

Remarks

The resulting angle is returned in radians, ranging from –p/2 to p/2. To convert the resulting radians to degrees, multiply the radians by 180/PI( ).

Examples

This function returns 1.29:

```
ATAN(3.5)
```

This function returns –1.33:

```
ATAN(–4)
```

See Also

ATAN2

ATANH

PI

TAN

ATAN2

Description

Returns the arctangent of the specified coordinates.

Syntax

ATAN2 (*x*, *y*)

Parameter	Description
x	The x coordinate.
y	The y coordinate.

Remarks

The arctangent is the angle from the x axis to a line with end points at the origin (0, 0) and a point with the given coordinates (x, y). The angle is returned in radians, ranging from $-\pi$ to π, excluding $-\pi$.

Examples

This function returns 1.11:

```
ATAN2(3, 6)
```

This function returns 3.04:

```
ATAN2(-1, .1)
```

See Also

ATAN

ATANH

PI

TAN

ATANH

Description

Returns the inverse hyperbolic tangent of a number.

Syntax

ATANH (*number*)

Parameter	Description
number	A number between –1 and 1, excluding –1 and 1.

Examples

This function returns .55:

```
ATANH(.5)
```

This function returns –.26:

```
ATANH(–.25)
```

See Also

ACOS

ASINH

TANH

AVERAGE

Description

Returns the average of the supplied numbers. The result of **AVERAGE** is also known as the arithmetic mean.

Syntax

AVERAGE (*number_list*)

Parameter	Description
number_list	A list of numbers separated by commas. As many as 30 numbers can be included in the list, and the list can contain numbers or a reference to a range that contains numbers. Text, logical expressions, or empty cells in a referenced range are ignored. All numeric values (including 0) are used.

Examples

This function returns 8.25:

```
AVERAGE(5, 6, 8, 14)
```

This function returns 134, the average of the values in the range C15:C17:

```
AVERAGE(C15:C17)
```

See Also

MIN

MAX

CEILING

Description

Rounds a number up to the nearest multiple of a specified significance.

Syntax

CEILING (*number*, *significance*)

Parameter	Description
number	The value to round.
significance	The multiple to which to round.

Remarks

Regardless of the sign of the number, the value is rounded up, away from zero. If number is an exact multiple of significance, no rounding occurs.

If number or significance is non-numeric, the error #VALUE! is returned. When the arguments have opposite signs, the error #NUM! is returned.

Examples

This function returns 1.25:

```
CEILING(1.23459, .05)
```

This function returns –150:

```
CEILING(–148.24, –2)
```

See Also

EVEN

FLOOR

INT

ODD

ROUND

TRUNC

CHAR

Description

Returns a character that corresponds to the supplied ASCII code.

Syntax

CHAR (*number*)

Parameter	Description
number	A value between 1 and 255 that specifies an ASCII character.

Remarks

The character and associated numeric code are defined by Windows in the ASCII character set.

Examples

This function returns F:

```
CHAR(70)
```

This function returns #:

```
CHAR(35)
```

See Also

CODE

CHOOSE

Description

Returns a value from a list of numbers based on the index number supplied.

Syntax

CHOOSE (*index*, *item_list*)

Parameter	Description
index	A number that refers to an item in item_list.
item_list	A list of numbers, formulas, or text separated by commas. This argument can also be a range reference. You can specify as many as 29 items in the list.

Remarks

Index can be a cell reference; index can also be a formula that returns any value from 1 to 29. If *index* is less than 1 or greater than the number of items in *item_list*, #VALUE! is returned. If *index* is a fractional number, it is truncated to an integer.

Examples

This function returns Q2:

```
CHOOSE(2,”Q1”, “Q2”, “Q3”, “Q4”)
```

This function returns the average of the contents of range A1:A10:

```
AVERAGE(CHOOSE(1, A1:A10, B1:B10, C1:C10))
```

See Also

INDEX

CLEAN

Description

Removes all nonprintable characters from the supplied text.

Syntax

CLEAN (*text*)

Parameter	Description
text	Any worksheet information.

Remarks

Text that is imported from another environment may require this function.

Examples

This function returns Payments Due because the character returned by CHAR (8) is nonprintable:

```
CLEAN(“Payments “ & CHAR(8) & “Due”)
```

See Also

CHAR

TRIM

CODE

Description

Returns a numeric code representing the first character of the supplied string.

Syntax

CODE (*text*)

Parameter	Description
text	Any string.

Remarks

The numeric code and associated string are defined in your computer's character set. The character set used by Windows is the ANSI character set.

Examples

This function returns 65:

```
CODE("A")
```

This function returns 98:

```
CODE("b")
```

See Also

CHAR

COLUMN

Description

Returns the column number of the supplied reference.

Syntax

COLUMN (reference)

Parameter	Description
reference	A reference to a cell or range. Omitting the argument returns the number of the column in which COLUMN is placed.

Examples

This function returns 2:

```
COLUMN(B3)
```

This function returns 4 if the function is entered in cell D2:

```
COLUMN()
```

See Also

COLUMNS

ROW

COLUMNS

Description

Returns the number of columns in a range reference.

Syntax

COLUMNS (*range*)

Parameter	Description
range	A reference to a range of cells.

Example

This function returns 4:

```
COLUMNS(A1:D5)
```

See Also

COLUMN

ROWS

CONCATENATE

Description

Joins several text strings into one string.

Syntax

CONCATENATE (*text1*, *text2*, ...)

Parameter	Description
text1, *text2*, ...	Up to 30 text items to be joined into a single text item. The text items can be strings, numbers, or single-cell references.

Remarks

The "&" operator can be used instead of CONCATENATE to join text items.

Examples

The following example returns "Sale Price" it is the same as typing "Sale"&" "&"Price":

```
CONCATENATE ("Sale ", "Price")
```

Suppose in an inventory worksheet, C2 contains "extruder1", C5 contains " gaskets", and C8 contains the number 15. The following example returns "Inventory currently holds 15 gaskets for extruder1.":

```
CONCATENATE ("Inventory currently holds ", C8, " ", C5," for ", C2)
```

See Also

COLUMN

ROWS

COS

Description

Returns the cosine of an angle.

Syntax

COS (*number*)

Parameter	Description
number	The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by PI()/180.

Examples

This function returns .126:

```
COS(1.444)
```

This function returns .28:

COS(5)

See Also

ACOS

ASINH

ATANH

COSH

PI

COSH

Description

Returns the hyperbolic cosine of a number.

Syntax

COSH (*number*)

Parameter	Description
number	Any number.

Examples

This function returns 4.14:

```
COSH(2.10)
```

This function returns 1.03:

```
COSH(.24)
```

See Also

ASINH

ATANH

COS

COUNT

Description

Returns the number of values in the supplied list.

Syntax

COUNT (*value_list*)

Parameter	Description
value_list	A list of values. The list can contain as many as 30 values.

Remarks

COUNT only numerates numbers or numerical values such as logical values, dates, or text representations of dates. If you supply a range, only numbers and numerical values in the range are counted. Empty cells, logical values, text, and error values in the range are ignored.

Examples

This function returns 2:

```
COUNT(5, 6, "Q2")
```

This function returns 3:

```
COUNT("03/06/94", "06/21/94", "10/19/94")
```

See Also

AVERAGE

COUNTA

SUM

COUNTIF

Description

Returns the number of cells within a range which meet the given criteria.

Syntax

COUNTIF (*range*, *criteria*)

Parameter	Description
range	Range of cells you want to count.
criteria	Number, expression, or text that defines which cells are counted.

See Also

AVERAGE

COUNTA

SUM

SUMIF

COUNTA

Description

Returns the number of nonblank values in the supplied list.

Syntax

COUNTA (*expression_list*)

Parameter	Description
expression_list	A list of expressions. As many as 30 expressions can be included in the list.

Remarks

COUNTA returns the number of cells that contain data in a range. Null values ("") are counted, but references to empty cells are ignored.

Examples

This function returns 4:

```
COUNTA(32, 45, “Earnings”, "")
```

This function returns 0 when the specified range contains empty cells:

```
COUNTA(C38:C40)
```

See Also

AVERAGE

COUNT

PRODUCT

SUM

DATE

Description

Returns the serial number of the supplied date.

Syntax

DATE (*year*, *month*, *day*)

Parameter	Description
year	A number from 1900 to 2078. If year is between 1920 to 2019, you can specify two digits to represent the year; otherwise specify all four digits.
month	A number representing the month (for example, 12 represents December). If a number greater than 12 is supplied, the number is added to the first month of the specified year.
day	A number representing the day of the month. If the number you specify for day exceeds the number of days in that month, the number is added to the first day of the specified month.

Examples

This function returns 34506:

```
DATE(94, 6, 21)
```

This function returns 36225:

```
DATE(99, 3, 6)
```

See Also

DATEVALUE

DAY

MONTH

NOW

TIMEVALUE

TODAY

YEAR

DATEVALUE

Description

Returns the serial number of a date supplied as a text string.

Syntax

DATEVALUE (*text*)

Parameter	Description
text	A date in text format between January 1, 1900, and December 31, 2078. If you omit the year, the current year is used.

Examples

This function returns 34399:

```
DATEVALUE("3/6/94")
```

This function returns 35058:

```
DATEVALUE("12/25/95")
```

See Also

NOW

TIMEVALUE

TODAY

DAY

Description

Returns the day of the month that corresponds to the date represented by the supplied number.

Syntax

DAY (*serial_number*)

Parameter	Description
serial_number	A date represented as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Examples

This function returns 6:

```
DAY(34399)
```

This function returns 21:

```
DAY("06-21-94")
```

See Also

NOW

HOUR

MINUTE

MONTH

SECOND

TODAY

WEEKDAY

YEAR

DB

Description

Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method.

Syntax

DB (*cost*, *salvage*, *life*, *period* [, *months*])

Parameter	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine period and life must match.
months	The number of months in the first year of the item's life. Omitting this argument assumes there are 12 months in the first year.

Example

This function returns 1451.52:

```
DB(10000, 1000, 7, 3)
```

See Also

DDB

SLN

SYD

VDB

DDB

Description

Returns the depreciation of an asset for a specific period of time using the double-declining balance method or a declining balance factor you supply.

Syntax

DDB (*cost*, *salvage*, *life*, *period* [, *factor*])

Parameter	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine period and life must match.
factor	The rate at which the balance declines. Omitting this argument assumes a default factor of 2, the double-declining balance factor.

Remarks

The double-declining balance method uses an accelerated rate where the highest depreciation occurs in the first period, decreasing in successive periods.

All arguments for this function must be positive numbers.

Example

This function returns 1457.73:

```
DDB(10000,1000, 7, 3)
```

See Also

DB

SLN

SYD

VDB

DOLLAR

Description

Returns the specified number as text, using currency format and the supplied precision.

Syntax

DOLLAR (*number* [, *precision*])

Parameter	Description
number	A number, a formula that evaluates to a number, or a reference to a cell that contains a number.
precision	A value representing the number of decimal places to the right of the decimal point. Omitting this argument assumes two decimal places.

Examples

This function returns $1023.79:

```
DOLLAR(1023.789)
```

This function returns $500:

```
DOLLAR(495.301, -2)
```

See Also

FIXED

TEXT

VALUE

ERROR.TYPE

Description

Returns a number corresponding to an *error*.

Syntax

ERROR.TYPE (*error_ref*)

Parameter	Description
error_ref	A cell reference.

Remarks

The following error text or numbers can be returned by this function.

Number	Description
1	#NULL!
2	#DIV/0!
3	#VALUE!
4	#*REF*!
5	#NAME?
6	#NUM!
7	#N/A
#N/A	Other

Example

This function returns 2 if the formula in cell A1 attempts to divide by zero:

```
ERROR.TYPE(A1)
```

See Also

ISERR

ISERROR

EVEN

Description

Rounds the specified number up to the nearest even integer.

Syntax

EVEN (*number*)

Parameter	Description
number	Any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

Examples

This function returns 4:

```
EVEN(2.5)
```

This function returns 2032:

```
EVEN(2030.45)
```

See Also

CEILING

FLOOR

INT

ODD

ROUND

TRUNC

EXACT

Description

Compares two expressions for identical, case-sensitive matches. True is returned if the expressions are identical; False is returned if they are not.

Syntax

EXACT (*expression1*, *expression2*)

Parameter	Description
expression1	Any text.
expression2	Any text.

Examples

This function returns True:

```
EXACT("Match", "Match")
```

This function returns False:

```
EXACT("Match", "match")
```

See Also

LEN

SEARCH

EXP

Description

Returns e raised to the specified power. The constant e is 2.71828182845904 (the base of the natural logarithm).

Syntax

EXP (*number*)

Parameter	Description
number	Any number as the exponent.

Examples

This function returns 12.18:

```
EXP(2.5)
```

This function returns 20.09:

```
EXP(3)
```

See Also

LN

LOG

FACT

Description

Returns the factorial of a specified number.

Syntax

FACT (*number*)

Parameter	Description
number	Any non-negative integer. If you supply a real number, **FACT** truncates the number to an integer before calculation.

Examples

This function returns 2:

```
FACT(2.5)
```

This function returns 720:

```
FACT(6)
```

See Also

PRODUCT

FALSE

Description

Returns the logical value False. This function always requires the trailing parentheses.

Syntax

FALSE ( )

See Also

TRUE

FIND

Description

Searches for a string of text within another text string and returns the character position at which the search string first occurs.

Syntax

FIND (search_text, text [, start_position])

Parameter	Description
search_text	The text to find. If you specify an empty string (""), **FIND** matches the first character in text.
text	The text to be searched.
start_position	The character position in text where the search begins. The first character in text is character number 1. When you omit this argument, the default starting position is character number 1.

Remarks

FIND is case-sensitive. You cannot use wildcard characters in the search_text.

Examples

This function returns 12:

```
FIND(“time”, “There’s no time like the present”)
```

This function returns 19:

```
FIND(“4”, “Aisle 4, Part 123-4-11”, 9)
```

See Also

EXACT

LEN

MID

SEARCH

FIXED

Description

Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.

Syntax

FIXED (*number* [, *precision*][, *no_commas*])

Parameter	Description
number	Any number.
precision	The number of digits that appear to the right of the decimal place. When this argument is omitted, a default precision of 2 is used. If you specify negative precision, number is rounded to the left of the decimal point. You can specify a precision as great as 127 digits.
no_commas	Determines if thousands separators (commas) are used in the result. Use 1 to exclude commas in the result. If no_commas is 0 or the argument is omitted, thousands separators are included (for example, 1,000.00).

Examples

This function returns 2,000.500:

```
FIXED(2000.5, 3)
```

This function returns 2010:

```
FIXED(2009.5, -1, 1)
```

See Also

DOLLAR

ROUND

TEXT

VALUE

FLOOR

Description

Rounds a number down to the nearest multiple of a specified significance.

Syntax

FLOOR (*number*, *significance*)

Parameter	Description
number	The value to round.
significance	The multiple to which to round.

Remarks

Regardless of the sign of the number, the value is rounded down, toward zero. If number is an exact multiple of significance, no rounding occurs.

If number or significance is non-numeric, #NAME? is returned. When the arguments have opposite signs, #NUM! is returned.

Examples

This function returns 1.2:

```
FLOOR(1.23459, .05)
```

This function returns –148:

```
FLOOR(–148.24, –2)
```

See Also

CEILING

EVEN

INT

ODD

ROUND

TRUNC

FV

Description

Returns the future value of an annuity based on regular payments and a fixed interest rate.

Syntax

FV (*interest*, *nper*, *payment* [, *pv*] [, *type*])

Parameter	Description
interest	The fixed interest rate.
nper	The number of payments in an annuity.
payment	The fixed payment made each period.
pv	The present value, or the lump sum amount, the annuity is currently worth. When you omit this argument, a present value of 0 is assumed.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Remarks

The units used for interest must match those used for nper. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for nper.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns 4,774.55:

```
FV(5%, 8, –500)
```

This function returns 531,550.86:

```
FV(10%/12, 240, –700, 1)
```

See Also

IPMT

NPER

PMT

PPMT

PV

RATE

HLOOKUP

Description

Searches the top row of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Syntax

HLOOKUP (*search_item*, *search_range*, *row_index*)

Parameter	Description
search_item	A value, text string, or reference to a cell containing a value that is matched against data in the top row of *search_range*.
search_range	A reference to the range (table) to be searched. The cells in the first row of *search_range* can contain numbers, text, or logical values. The contents of the first row must be in ascending order (for example, –2, –1, 0, 2...A through Z, False, True). Text searches are not case-sensitive.
row_index	The row in search_range from which the matching value is returned. row_index can be a number from 1 to the number of rows in *search_range*. If *row_index* is less than 1, the error #VALUE! is returned. When *row_index* is greater than the number of rows in the table, the error #REF! is returned.

Remarks

HLOOKUP compares the information in the top row of *search_range* to the supplied *search_item*. When a match is found, information located in the same column and supplied row (*row_index*) is returned.

If *search_item* cannot be found in the top row of *search_range*, the largest value that is less than search_item is used. When *search_item* is less than the smallest value in the first row of the *search_range*, the error #REF! is returned.

Examples

The following examples use this worksheet.

	A	B	C	D	E
1		Midwest	Northeast	Pacific	South
2	Q1	48.23	278.21	61.97	164.80
3	Q2	163.83	22.63	161.73	183.96
4	Q3	43.96	233.56	278.16	171.98
5	Q4	245.69	167.09	245.23	163.00

This function returns 22.63:

```
HLOOKUP("Northeast", B1:E5, 3)
```

This function returns #REF!:

```
HLOOKUP("Pacific", B1:E5, 7)
```

See Also

INDEX

LOOKUP

MATCH

VLOOKUP

HOUR

Description

Returns the hour component of the specified time in 24-hour format.

Syntax

HOUR (*serial_number*)

Parameter	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Remarks

The result is an integer ranging from 0 (12:00 AM) to 23 (11:00 PM).

Examples

This function returns 9:

```
HOUR(34259.4)
```

This function returns 23:

```
HOUR(34619.976)
```

See Also

DAY

MINUTE

MONTH

NOW

SECOND

WEEKDAY

YEAR

IF

Description

Tests the condition and returns the specified value.

Syntax

IF (*condition*, *true_value*, *false_value*)

Parameter	Description
condition	Any logical expression.
true_value	The value to be returned if condition evaluates to True.
false_value	The value to be returned if condition evaluates to False.

Example

This function returns Greater if the contents of A1 is greater than 10 and Less if the contents of A1 is less than 10:

```
IF(A1>10, “Greater”, “Less”)
```

See Also

AND

FALSE

NOT

OR

TRUE

INDEX

Description

Returns the contents of a cell from a specified range.

Syntax

INDEX (*reference* [, *row*] [, *column*] [, *range_number*])

Parameter	Description
reference	A reference to one or more ranges. If reference specifies more than one range, separate each reference with a comma and enclose reference in parentheses. For example, (A1:C6, B7:E14, F4). If each range in reference contains only one row or column, you can omit the row or column argument. For example, if reference is A1:A15, you can omit the column argument INDEX(A1:A15, 3,, 1).
row	The row number in reference from which to return data.
column	Column number in reference from which to return data.
range_number	Specifies the range from which data is returned if reference contains more than one range. For example, if reference is (A1:A10, B1:B5, D14:E23), A1:A10 is range_number 1, B1:B5 is range_number 2, and D14:E23 is range_number 3.

Remarks

If row, column, and range_number do not point to a cell within reference, #REF! is returned. If row and column are omitted, INDEX returns the range in reference specified by range_number.

Examples

The following examples use this worksheet.

	A	B	C	D	E
1	Sales Group 1			Sales Group 2	
2	Adams	$1,225.14		Cash	$1,819.47
3	Baker	$1,415.35		Johnson	$1,733.67
4	Martinez	$1,573.57		Nelson	$1,138.23
5	Smith	$1,469.78		Randall	$1,634.58
6	White	$1,390.89		Schultz	$1,093.82

This function returns $1415.35:

```
INDEX(A2:B6, 2, 2)
```

This function returns $1634.58:

```
INDEX((A2:B6, D2:E6), 4, 2, 2)
```

See Also

CHOOSE

HLOOKUP

LOOKUP

MATCH

VLOOKUP

INDIRECT

Description

Returns the contents of the cell referenced by the specified cell.

Syntax

INDIRECT (*ref_text* [, *a1*])

Parameter	Description
ref_text	A reference to a cell that references a third cell. If ref_text is not a valid reference, the error #REF! is returned.
a1	The reference format. This argument must be TRUE() to represent an A1 reference format; Formula One does not support the R1C1 reference format.

Example

This function returns the contents of the cell that C1 references. If C1 contains "D1," then the contents of D1is returned:

```
INDIRECT(C1)
```

See Also

OFFSET

INT

Description

Rounds the supplied number down to the nearest integer.

Syntax

INT (*number*)

Parameter	Description
number	Any real number.

Examples

This function returns 10:

```
INT(10.99)
```

This function returns –11:

```
INT(–10.99)
```

See Also

CEILING

FLOOR

MOD

ROUND

TRUNC

IPMT

Description

Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.

Syntax

IPMT (*interest*, *per*, *nper*, *pv*, [*fv*], [*type*])

Parameter	Description
interest	The fixed periodic interest rate.
per	The period for which to return the interest payment. This number must be between 1 and nper.
nper	The number of payments.
pv	The present value, or the lump sum amount the annuity is currently worth.
fv	The future value, or the value after all payments are made. If this argument is omitted, the future value is assumed to be 0.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Remarks

The units used for interest must match those used for nper. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for nper.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –117.87:

```
IPMT(8%/12, 2, 48, 18000)
```

This function returns –117.09:

```
IPMT(8%/12, 2, 48, 18000, 0, 1)
```

See Also

FV

PMT

PPMT

RATE

IRR

Description

Returns internal rate of return for a series of periodic cash flows.

Syntax

IRR (*cash_flow* [, *guess*])

Parameter	Description
cash_flow	A reference to a range that contains values for which to calculate the internal rate of return. The values must contain at least one positive and one negative value. During calculation, **IRR** uses the order in which the values appear to determine the order of the cash flow. Text, logical values, and empty cells in the range are ignored.
guess	The estimate of the internal rate of return. If no argument is supplied, a rate of return of 10 percent is assumed.

Remarks

The internal rate of return is the interest rate received for an investment consisting of payments (specified by negative numbers) and investments (specified by positive numbers).

IRR is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for guess.

Examples

The following examples use this worksheet.

	A	B
1	Investment	($60,000.00)
2	1989 income	$9,590.00
3	1990 income	$10,580.00
4	1991 income	$12,790.00
5	1992 income	$15,830.00
6	1993 income	$18,930.00

This function returns 3.72 percent:

```
IRR(B1:B6)
```

This function returns –49.26 percent:

```
IRR(B1:B3, –20%)
```

See Also

MIRR

NPV

RATE

ISBLANK

Description

Determines if the specified cell is blank.

Syntax

ISBLANK (*reference*)

Parameter	Description
reference	A reference to any cell.

Remarks

If the referenced cell is blank, True is returned. False is returned if the cell is not blank.

Example

This function returns True if A1 is a blank cell:

```
ISBLANK(A1)
```

See Also

ISERR

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

ISERR

Description

Determines if the specified expression returns an error value.

Syntax

ISERR (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns any error except #N/A!, True is returned. Otherwise, False is returned.

Example

This function returns True if A1 contains a formula that returns an error such as #NUM!:

```
ISERR(A1)
```

See Also

ISBLANK

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

ISERROR

Description

Determines if the specified expression returns an error value.

Syntax

ISERROR (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns any error value, such as #N/A!, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!, True is returned. Otherwise, False is returned.

Examples

This function returns True:

```
ISERROR(4/0)
```

This function returns False if A1 contains a formula that does not return an error.

```
ISERROR(A1)
```

See Also

ISBLANK

ISERR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

ISLOGICAL

Description

Determines if the specified expression returns a logical value.

Syntax

ISLOGICAL (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns a logical value, True is returned. Otherwise, False is returned.

Example

This function returns True because ISBLANK returns a logical value:

```
ISLOGICAL(ISBLANK(A1))
```

See Also

ISBLANK

ISERR

ISERROR

ISNA

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

ISNA

Description

Determines if the specified expression returns the value not available error.

Syntax

ISNA (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns the #N/A! error, True is returned. Otherwise, False is returned.

Example

This function returns True if cell A1 contains the NA ( ) function or returns the error value #N/A!:

```
ISNA(A1)
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

ISNONTEXT

Description

Determines if the specified expression is not text.

Syntax

ISNONTEXT (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns any value that is not text, True is returned. Otherwise, False is returned.

Examples

This function returns True if cell F3 contains a number or is a blank cell:

```
ISNONTEXT(F3)
```

This function returns False:

```
ISNONTEXT("text")
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNA

ISNUMBER

ISREF

ISTEXT

ISNUMBER

Description

Determines if the specified expression is a number.

Syntax

ISNUMBER (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns a number, True is returned. Otherwise, False is returned. If expression returns a number represented as text (for example, “12”), False is returned.

Examples

This function returns True:

```
ISNUMBER(123.45)
```

This function returns False:

```
ISNUMBER(“123”)
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISREF

ISTEXT

ISREF

Description

Determines if the specified expression is a range reference.

Syntax

ISREF (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns a range reference, True is returned. Otherwise, False is returned.

Example

This function returns True:

```
ISREF(A3)
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISTEXT

ISTEXT

Description

Determines if the specified expression is text.

Syntax

ISTEXT (*expression*)

Parameter	Description
expression	Any expression.

Remarks

If the expression returns text, True is returned. Otherwise, False is returned.

Example

This function returns True:

```
ISTEXT("2nd Quarter")
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISREF

LEFT

Description

Returns the leftmost characters from the specified text string.

Syntax

LEFT (*text* [, *num_chars*])

Parameter	Description
text	Any text string.
num_chars	The number of characters to return. This value must be greater than or equal to zero. If num_chars is greater than the number of characters in text, the entire string is returned. Omitting this argument assumes a value of 1.

Examples

This function returns 2:

```
LEFT(“2nd Quarter”)
```

This function returns 2nd:

```
LEFT(“2nd Quarter”, 3)
```

See Also

MID

RIGHT

LEN

Description

Returns the number of characters in the supplied text string.

Syntax

LEN (*text*)

Parameter	Description
text	Any text string. Spaces in the string are counted as characters.

Examples

This function returns 11:

```
LEN("3rd Quarter")
```

This function returns 3:

```
LEN("1-3")
```

See Also

EXACT

SEARCH

LN

Description

Returns the natural logarithm (based on the constant e) of a number.

Syntax

LN (*number*)

Parameter	Description
number	Any positive real number.

Remarks

LN is the inverse of the **EXP** function.

Examples

This function returns 2.50:

```
LN(12.18)
```

This function returns 3.00:

```
LN(20.09)
```

See Also

EXP

LOG

LOG10

LOG

Description

Returns the logarithm of a number to the specified base.

Syntax

LOG (*number* [, *base*])

Parameter	Description
number	Any positive real number.
base	The base of the logarithm. Omitting this argument assumes a base of 10.

Examples

This function returns 0:

```
LOG(1)
```

This function returns 1:

```
LOG(10)
```

See Also

EXP

LN

LOG10

LOG10

Description

Returns the base-10 logarithm of a number.

Syntax

LOG10 (*number*)

Parameter	Description
number	Any positive real number.

Examples

This function returns 2.41:

```
LOG10(260)
```

This function returns 2:

```
LOG10(100)
```

See Also

EXP

LN

LOG

LOOKUP

Description

Searches for a value in one range and returns the contents of the corresponding position in a second range.

Syntax

LOOKUP (*lookup_value*, *lookup_range*, *result_range*)

Parameter	Description
lookup_value	The value for which to search in the first range.
lookup_range	The first range to search and contains only one row or one column. The range can contain numbers, text, or logical values. To search lookup_range correctly, the expressions in the range must be placed in ascending order (for example, –2, –1, 0, 1, 2...A through Z, False, True). The search is not case-sensitive.
result_range	A range of one row or one column that is the same size as lookup_range.

Remarks

If *lookup_value* does not have an exact match in *lookup_range*, the largest value that is less than or equal to *lookup_value* is found and the corresponding position in *result_range* is returned. When *lookup_value* is smaller than the data in *lookup_range*, #N/A is returned.

Examples

The following examples use this worksheet.

	A	B
1	**Region**	**Headquarters**
2	Midwest	Kansas City
3	North	Detroit
4	Northeast	Philadelphia
5	Pacific	Portland
6	South	Atlanta
7	Southwest	Phoenix

This function returns Detroit:

```
LOOKUP("North", A2:A7, B2:B7)
```

This function returns #N/A:

```
LOOKUP("Alabama", A2:A7, B2:B7)
```

See Also

HLOOKUP

INDEX

VLOOKUP

LOWER

Description

Changes the characters in the specified string to lowercase characters. Numeric characters in the string are not changed.

Syntax

LOWER (*text*)

Parameter	Description
text	Any string.

Examples

This function returns 3rd quarter:

```
LOWER(“3rd Quarter”)
```

This function returns john doe:

```
LOWER(“JOHN DOE”)
```

See Also

PROPER

UPPER

MATCH

Description

A specified value is compared against values in a range. The position of the matching value in the search range is returned.

Syntax

MATCH (*lookup_value*, *lookup_range*, *comparison*)

Parameter	Description
lookup_value	The value against which to compare. It can be a number, text, or logical value or a reference to a cell that contains one of those values.
lookup_range	The range to search and contains only one row or one column. The range can contain numbers, text, or logical values.
comparison	A number that represents the type of comparison to be made between lookup_value and the values in lookup_range. When you omit this argument, comparison method 1 is assumed. When comparison is 0, the first value that is equal to lookup_value is matched. When using this comparison method, the values in lookup_range can be in any order. When comparison is 1, the largest value that is less than or equal to lookup_value is matched. When using this comparison method, the values in lookup_range must be in ascending order (for example, ...–2, –1, 0, 1, 2..., A through Z, False, True). When comparison is –1, the smallest value that is greater than or equal to lookup_value is matched. When using this comparison method, the values in lookup_range must be in descending order (for example, True, False, Z through A, ...2, 1, 0, –1, –2...).

Remarks

When using comparison method 0 and *lookup_value* is text, *lookup_value* can contain wildcard characters. The wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character.

When no match is found for *lookup_value*, #N/A is returned.

Examples

The following examples use this worksheet.

	A	B
1	Mfr. Code	Stock No.
2	BAJ	0677
3	DOD	0753
4	FMH	0816
5	JMP	0913
6	PLY	7534
7	TJL	7763

This function returns 5:

```
MATCH(7600, B2:B7,1)
```

This function returns 2:

```
MATCH("D*", A2:A7,0)
```

See Also

HLOOKUP

INDEX

LOOKUP

VLOOKUP

MAX

Description

Returns the largest value in the specified list of numbers.

Syntax

MAX (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas.
	The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values.
	Error values or text that cannot be translated into numbers return errors.
.	If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored
	If there are no numbers in the list, 0 is returned.

Examples

This function returns 500:

```
MAX(50, 100, 150, 500, 200)
```

This function returns the largest value in the range:

```
MAX(A1:F12)
```

See Also

AVERAGE

MIN

MID

Description

Returns the specified number of characters from a text string, beginning with the specified starting position.

Syntax

MID (*text*, *start_position*, *num_chars*)

Parameter	Description
text	The string from which to return characters.
start_position	The position of the first character to return from text.
	If *start_position* is 1, the first character in text is returned.
	If *start_position* is greater than the number of characters in text, an empty string ("") is returned.
	If *start_position* is less than 1, #VALUE! is returned.
num_chars	The number of characters to return. If *num_chars* is negative, #VALUE! is returned.

Remarks

If *start_position* plus the number of characters in *num_chars* exceeds the length of text, the characters from *start_position* to the end of text are returned.

Examples

This function returns Expenses:

```
MID(“Travel Expenses”, 8, 8)
```

This function returns 45:

```
MID(“Part #45-7234”, 7, 2)
```

See Also

CODE

FIND

LEFT

RIGHT

SEARCH

MIN

Description

Returns the smallest value in the specified list of numbers.

Syntax

MIN (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored. If there are no numbers in the list, 0 is returned

Examples

This function returns 50:

```
MIN(50, 100, 150, 500, 200)
```

This function returns the smallest value in the range:

```
MIN(A1:F12)
```

See Also

AVERAGE

MAX

MINUTE

Description

Returns the minute that corresponds to the supplied date.

Syntax

MINUTE (*serial_number*)

Parameter	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Remarks

The result is an integer ranging from 0 to 59.

Examples

This function returns 36:

```
MINUTE(34506.4)
```

This function returns 48:

```
MINUTE(34399.825)
```

See Also

DAY

HOUR

MONTH

NOW

SECOND

WEEKDAY

YEAR

MIRR

Description

Returns the modified internal rate of return for a series of periodic cash flows.

Syntax

MIRR (*cash_flows*, *finance_rate*, *reinvest_rate*)

Parameter	Description
cash_flow	A reference to a range that contains values for which to calculate the modified internal rate of return. The values must contain at least one positive and one negative value. Values that represent cash received should be positive; negative values represent cash paid. During calculation, **MIRR** uses the order in which the values appear to determine the order of cash flow. Text, logical values, and empty cells in the range are ignored.
finance_rate	The interest rate paid on money used in the cash flow.
reinvest_rate	The interest rate received on money reinvested from the cash flow.

Remarks

The modified internal rate of return considers the cost of the investment and the interest received on the reinvestment of cash.

Examples

The following examples use this worksheet.

This function returns 5.20 percent:

```
MIRR(B1:B6, 12%, 8%)
```

This function returns –40.93 percent:

```
MIRR(B1:B3, 12%, 8%)
```

See Also

IRR

NPV

RATE

MOD

Description

Returns the remainder after dividing a number by a specified divisor.

Syntax

MOD (*number*, *divisor*)

Parameter	Description
number	Any number.
divisor	Any nonzero number. If divisor is 0, #DIV/0! is returned.

Examples

This function returns 1:

```
MOD(-23, 3)
```

This function returns -2:

```
MOD(-23, -3)
```

See Also

INT

ROUND

TRUNC

MONTH

Description

Returns the month that corresponds to the supplied date.

Syntax

MONTH (*serial_number*)

Parameter	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Remarks

MONTH returns a number ranging from 1 (January) to 12 (December).

Examples

This function returns 6:

```
MONTH("06-21-94")
```

This function returns 10:

```
MONTH(34626)
```

See Also

DAY

NOW

HOUR

MINUTE

SECOND

TODAY

WEEKDAY

YEAR

N

Description

Tests the supplied value and returns the value if it is a number.

Syntax

N (*value*)

Parameter	Description
value	A value or a reference to a cell containing a value to test.

Remarks

Numbers are returned as numbers, serial numbers formatted as dates are returned as serial numbers, and the logical function TRUE() is returned as 1. All other expressions return 0.

Examples

This function returns 32467:

```
N(32467)
```

This function returns 1 if A4 contains the logical function TRUE:

```
N(A4)
```

See Also

T

VALUE

NA

Description

Returns the error value #N/A, which represents ‘‘not available.’’

Syntax

NA ( )

Remarks

Use **NA** to mark cells that lack data without leaving them empty. Empty cells may not be correctly represented in some calculations.

Although **NA** does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also

ISNA

NOT

Description

Returns a logical value that is the opposite of its value.

Syntax

NOT (*logical*)

Parameter	Description
logical	An expression that returns a logical value such as True or False.

Remarks

If logical is false, **NOT** returns True. Conversely, if logical is true, **NOT** returns False.

Examples

This function returns False:

```
NOT(TRUE())
```

This function returns False:

```
NOT(MONTH("12/25/94") = 12)
```

See Also

AND

IF

OR

NOW

Description

Returns the current date and time as a serial number.

Syntax

NOW ( )

Remarks

In a serial number, numbers to the left of the decimal point represent the date; numbers to the right of the decimal point represent the time. The result of this function changes only when a recalculation of the worksheet occurs.

See Also

DATE

DAY

HOUR

MINUTE

MONTH

SECOND

TODAY

WEEKDAY

YEAR

NPER

Description

Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate.

Syntax

NPER (*interest*, *pmt*, *pf* [, *fv*] [, *type*])

Parameter	Description
interest	The fixed interest rate.
pmt	The fixed payment made each period. Generally, pmt includes the principle and interest, not taxes or other fees.
pf	The present value, the lump-sum amount that a series of future payments is currently worth.
fv	The future value, the balance to attain after the final payment. Omitting this argument assumes a future balance of 0.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Examples

This function returns 36.67:

```
NPER(12%/12, –350, –300, 16000, 1)
```

This function returns 36.98:

```
NPER(1%, –350, –300, 16000)
```

See Also

FV

IPMT

PMT

PPMT

PV

RATE

NPV

Description

Returns the net present value of an investment based on a series of periodic payments and a discount rate.

Syntax

NPV (*discount_rate*, *value_list*)

Parameter	Description
discount_rate	The rate of discount for one period.
value_list	A list of as many as 29 arguments or a reference to a range that contains values that represent payments and income. During calculation, NPV uses the order in which the values appear to determine the order of cash flow. Numbers, empty cells, and text representations of numbers are included in the calculation. Errors and text that cannot be translated into numbers are ignored. If value_list is a range reference, only numeric data in the range is included in the calculation. Other types of data in the range, such as empty cells, logical values, text, and error values, are ignored.

Remarks

The time span **NPV** uses for calculation begins one period before the first cash flow date and ends when the last cash flow payment is made. This function is based on future cash flows. When your first cash flow occurs at the beginning of the first period, the first value must be added to the **NPV** result, not supplied as a value in *value_list*.

Example

This function returns 811.57:

```
NPV(8%, –12000, 3000, 3000, 3000, 7000)
```

See Also

FV

IRR

PV

ODD

Description

Rounds the specified number up to the nearest odd integer.

Syntax

ODD (*number*)

Parameter	Description
number	Any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

Examples

This function returns 5:

```
ODD(3.5)
```

This function returns 7:

```
ODD(6)
```

See Also

CEILING

EVEN

FLOOR

INT

ROUND

TRUNC

OFFSET

Description

Returns the contents of a range that is offset from a starting point in the spreadsheet.

Syntax

OFFSET (*reference*, *rows*, *columns* [, *height*] [, *width*])

Parameter	Description
reference	A reference to a cell from which the offset reference is based. If you specify a range reference, #VALUE! is returned.
rows	The number of rows from reference that represents the upper-left cell of the offset range. A positive number represents rows below the starting cell; a negative number represents rows above the starting cell. If rows places the upper-left cell of the offset range outside the spreadsheet boundary, #REF! is returned.
columns	The number of columns from reference that represents the upper-left cell of the offset range. A positive number represents columns right of the starting cell; a negative number represents columns left of the starting cell. If columns places the upper-left cell of the offset range outside the spreadsheet boundary, #REF! is returned.
height	A positive number representing the number of rows to include in the offset range. Omitting this argument assumes a single row.
width	A positive number representing the number of columns to include in the offset range. Omitting this argument assumes a single column.

Remarks

OFFSET does not change the current selection in the worksheet. Because it returns a reference, **OFFSET** can be used in any function that requires or uses a cell or range reference as an argument.

Examples

This function returns the contents of cell D4:

```
OFFSET(B1, 3, 2, 1, 1)
```

This function returns the sum of the values in the range E3:F5:

```
SUM(OFFSET(A1, 2, 4, 3, 2))
```

OR

Description

Returns True if at least one of a series of logical arguments is true.

Syntax

OR (*logical_list*)

Parameter	Description
logical_list	A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error value #VALUE! is returned.

Example

This function returns True because one of the arguments is true:

```
OR(1 + 1 = 1, 5 + 5 = 10)
```

See Also

AND

IF

NOT

PI

Description

Returns the value of pi (p), which is approximately 3.14159265358979 when calculated to 15 significant digits.

Syntax

PI ( )

Remarks

Although **PI** does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also

COS

SIN

TAN

PMT

Description

Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.

Syntax

PMT (*interest*, *nper*, *pv* [, *fv*] [, *type*])

Parameter	Description
interest	The fixed periodic interest rate.
nper	The number of periods in the annuity.
pv	The present value, or the amount the annuity is currently worth.
fv	The future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Remarks

PMT returns only the principal and interest payment, it does not include taxes or other fees.

The units used for interest must match those used for nper. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for nper.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –439.43:

```
PMT(8%/12, 48, 18000)
```

This function returns –436.52:

```
PMT(8%/12, 48, 18000, 0, 1)
```

See Also

IPMT

FV

NPER

PPMT

PV

RATE

PPMT

Description

Returns the principle paid on an annuity for a given period.

Syntax

PPMT (*interest*, *per*, *nper*, *pv*, [*fv*], [*type*])

Parameter	Description
interest	The fixed periodic interest rate.
per	The period for which to return the principle.
nper	The number of periods in the annuity.
pv	The present value, or the amount the annuity is currently worth.
fv	The future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Remarks

The units used for interest must match those used for nper. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for nper.

Examples

This function returns –321.56:

```
PPMT(8%/12, 2, 48, 18000)
```

This function returns –319.43:

```
PPMT(8%/12, 2, 48, 18000, 0, 1)
```

See Also

FV

IPMT

NPER

PMT

PV

RATE

PRODUCT

Description

Multiplies a list of numbers and returns the result.

Syntax

PRODUCT (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored. All numeric values, including 0, are used in the calculation.

Example

This function returns 24:

```
PRODUCT(1, 2, 3, 4)
```

See Also

FACT

SUM

PROPER

Description

Returns the specified string in proper-case format.

Syntax

PROPER (*text*)

Parameter	Description
text	Any string.

Remarks

In proper-case format, the first alphabetic character in a word is capitalized. If an alphabetic character follows a number, punctuation mark, or space, it is capitalized. All other alphabetic characters are lowercase. Numbers are not changed by **PROPER**.

Examples

This function returns 3rd Quarter:

```
PROPER(“3rd Quarter”)
```

This function returns John Doe:

```
PROPER(“JOHN DOE”)
```

See Also

LOWER

UPPER

PV

Description

Returns the present value of an annuity, considering a series of constant payments made over a regular payment period.

Syntax

PV (*interest*, *nper*, *pmt* [, *fv*] [, *type*])

Parameter	Description
interest	The fixed periodic interest rate.
nper	The number of payment periods in the investment.
pmt	The fixed payment made each period.
fv	The future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.

Remarks

The units used for *interest* must match those used for *nper*. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for *nper*.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns –17999.89:

```
PV(8%/12, 48, 439.43)
```

This function returns 17999.89:

```
PV(8%/12, 48, –439.43)
```

See Also

FV

IPMT

NPER

PMT

PPMT

RATE

RAND

Description

Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1.

Syntax

RAND ( )

Remarks

Although RAND does not use arguments, you must supply the empty parentheses to correctly reference the function.

Example

This function returns a random number greater than or equal to 0 and less than 10:.

```
RAND()*10
```

RATE

Description

Returns the interest rate per period of an annuity, given a series of constant cash payments made over a regular payment period.

Syntax

RATE (*nper*, *pmt*, *pv* [, *fv*] [, *type*] [, *guess*])

Parameter	Description
nper	The number of periods in the annuity.
pmt	The fixed payment made each period. Generally, pmt includes only principle and interest, not taxes or other fees.
pv	The present value of the annuity.
fv	The future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed.
type	Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. When you omit this argument, 0 is assumed.
guess	Your estimate of the interest rate. If no argument is supplied, a value of 0.1 (10 percent) is assumed.

Remarks

RATE is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for guess.

Example

The following example returns the monthly interest rate of .0067; the annual interest rate (.0067 multiplied by 12) is 8 percent:

```
RATE(48, –439.43, 18000)
```

See Also

FV

IPMT

NPER

PMT

PPMT

PV

REPLACE

Description

Replaces part of a text string with another text string.

Syntax

REPLACE (*orig_text*, *start_position*, *num_chars*, *repl_text*)

Parameter	Description
orig_text	The original text string.
start_position	The character position where the replacement begins. If start_position is greater than the number of characters in orig_text, repl_text is appended to the end of orig_text. If start_position is less than 1, #VALUE! is returned.
num_chars	The number of characters to replace. If this argument is negative, #VALUE! is returned.
repl_text	The replacement text string.

Examples

This function returns "For the year: 1994":

```
REPLACE(“For the year: 1993”, 18, 1, “4”)
```

See Also

MID

SEARCH

TRIM

REPT

Description

Repeats a text string the specified number of times.

Syntax

REPT (*text*, *number*)

Parameter	Description
text	Any text string.
number	The number of times you want text to repeat. If number is 0, empty text ("") is returned.

Remarks

The result of **REPT** cannot exceed 255 characters.

Example

This function returns error-error-error-:

```
REPT(“error-”, 3)
```

RIGHT

Description

Returns the rightmost characters from the given text string.

Syntax

RIGHT (*text* [, *num_chars*])

Parameter	Description
text	Any text string.
num_chars	The number of characters to return. The value must be greater than or equal to zero. If num_chars is greater than the number of characters in text, the entire string is returned. Omitting this argument assumes a value of 1.

Examples

This function returns r:

```
RIGHT("2nd Quarter")
```

This function returns Quarter:

```
RIGHT("2nd Quarter", 7)
```

See Also

LEFT

MID

ROUND

Description

Rounds the given number to the supplied number of decimal places.

Syntax

ROUND (*number*, *precision*)

Parameter	Description
number	Any value.
precision	The number of decimal places to which number is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by precision are replaced with zeros. If precision is 0, number is rounded to the nearest integer.

Example

This function returns 123.46:

```
ROUND(123.456, 2)
```

This function returns 9900:

```
ROUND(9899.435, -2)
```

See Also

- **CEILING**
- **FLOOR**
- **INT**
- **MOD**
- **ROUNDDOWN**
- **ROUNDUP**
- **TRUNC**

ROUNDDOWN

Description

Rounds a number down.

Syntax

ROUNDDOWN (*number*, *numberOfDigits*)

Parameter	Description
number	Any real number you want to round.
numberOfDigits	The number of decimal places to which number is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by precision are replaced with zeros. If precision is 0, number is rounded down to the nearest integer.

Example

This function returns 31.141:

```
ROUNDDOWN(3.14159, 3)
```

This function returns 31.400:

```
ROUNDDOWN(31415.92654, -2)
```

See Also

CEILING

FLOOR

INT

MOD

ROUND

ROUNDUP

TRUNC

ROUNDUP

Description

Rounds the given number up to the supplied number of decimal places.

Syntax

ROUNDUP (*number*, *numberOfDigits*)

Parameter	Description
number	Any value you want to round up.
numberOfDigits	The number of decimal places to which number is rounded. When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by precision are replaced with zeros. If precision is 0, number is rounded up to the nearest integer.

Example

This function returns 77:

```
ROUNDUP(76.9,0)
```

This function returns 3150:

```
ROUNDUP(31415.92654, -2)
```

See Also

CEILING

FLOOR

INT

MOD

ROUND

ROUNDDOWN

TRUNC

ROW

Description

Returns the row number of the supplied reference.

Syntax

ROW (*reference*)

Parameter	Description
reference	A cell or range reference. Omitting this argument returns the row number of the cell in which ROW is entered.

Examples

This function returns 3:

```
ROW(B3)
```

See Also

COLUMN

ROWS

ROWS

Description

Returns the number of rows in a range reference.

Syntax

ROWS (*range*)

Parameter	Description
range	A reference to a range of cells.

Examples

This function returns 5:

```
ROWS(A1:D5)
```

This function returns 6:

```
ROWS(C30:F35)
```

See Also

COLUMNS

ROW

SEARCH

Description

Locates the position of the first character of a specified text string within another text string.

Syntax

SEARCH (*search_text*, *text* [, *start_position*])

Parameter	Description
search_text	The text to find. To search for an asterisk or question mark, include a tilde (~) before the character. The search string can contain wildcard characters. The available wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character.
text	The text to be searched.
start_position	The character position where the search begins. If the number you specify is less than 0 or greater than the number of characters in text, #VALUE! is returned. Omitting this argument assumes a starting position of 1.

Remarks

Text is searched from left to right, starting at the position specified. The search is not case-sensitive. If text does not contain the search string, #VALUE! is returned.

Examples

This function returns 6:

```
SEARCH("?5", "Bin b45")
```

This function returns 5:

```
SEARCH("b", "Bin b45", 4)
```

See Also

FIND

MID

REPLACE

SUBSTITUTE

SECOND

Description

Returns the second that corresponds to the supplied date.

Syntax

SECOND (*serial_number*)

Parameter	Description
serial_number	The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Examples

This function returns 58:

```
SECOND(.259)
```

This function returns 46:

```
SECOND(34657.904)
```

See Also

DAY

HOUR

MINUTE

MONTH

NOW

WEEKDAY

SIGN

Description

Determines the sign of the specified number.

Syntax

SIGN (*number*)

Parameter	Description
number	Any number.

Remarks

SIGN returns 1 if the specified number is positive, –1 if it is negative, and 0 if it is 0.

Examples

This function returns –1:

```
SIGN(–123)
```

This function returns 1:

```
SIGN(123)
```

See Also

ABS

SIN

Description

Returns the sine of the supplied angle.

Syntax

SIN (*number*)

Parameter	Description
number	The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by PI( )/180.

Examples

This function returns .85:

```
SIN(45)
```

This function returns .89:

```
SIN(90)
```

See Also

ASIN

PI

SINH

Description

Returns the hyperbolic sine of the specified number.

Syntax

SINH (*number*)

Parameter	Description
n*u*mber	Any number.

Examples

This function returns 1.18:

```
SINH(1)
```

This function returns 10.02:

```
SINH(3)
```

See Also

ASINH

PI

SLN

Description

Returns the depreciation of an asset for a specific period of time using the straight-line balance method.

Syntax

SLN (*cost*, *salvage*, *life*)

Parameter	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods of the useful life of the asset.

Example

This function returns 1285.71:

```
SLN(10000, 1000, 7)
```

See Also

DDB

SYD

VDB

SQRT

Description

Returns the square root of the specified number.

Syntax

SQRT (*number*)

Parameter	Description
number	Any positive number. If you specify a negative number, the error #NUM! is returned.

Examples

This function returns 3:

```
SQRT(9)
```

This function returns 1.58:

```
SQRT(2.5)
```

See Also

SUMSQ

STDEV

Description

Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax

STDEV (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example

This function returns .56:

```
STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEVP

VAR

VARP

STDEVP

Description

Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax

STDEVP (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example

This function returns .52:

```
STDEVP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV

VAR

VARP

SUBSTITUTE

Description

Replaces a specified part of a text string with another text string.

Syntax

SUBSTITUTE (*text*, *old_text*, *new_text* [, *instance*])

Parameter	Description
text	A text string that contains the text to replace. You can also specify a reference to a cell that contains text.
old_text	The text string to be replaced.
new_text	The replacement text.
instance	Specifies the occurrence of old_text to replace. If this argument is omitted, every instance of old_text is replaced.

Examples

This function returns "Second Quarter Results":

```
SUBSTITUTE(“First Quarter Results”, “First”, “Second”)
```

This function returns "Shipment 45, Bin 52":

```
SUBSTITUTE(“Shipment 45, Bin 45”, “45”, “52”, 2)
```

See Also

REPLACE

TRIM

SUM

Description

Returns the sum of the supplied numbers.

Syntax

SUM (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.

Examples

This function returns 6000:

```
SUM(1000, 2000, 3000)
```

This function returns 4000 when each cell in the range contains 1000:

```
SUM(A10:D10)
```

See Also

AVERAGE

COUNT

COUNTA

PRODUCT

SUMSQ

SUMIF

Description

Returns the sum of the specified cells based on the given criteria.

Syntax

SUMIF (*range*, *criteria*, *sum_range*)

Parameter	Description
range	The range of cells you want evaluated.
criteria	A number, expression, or text that defines which cells are added. For example, criteria can be expressed as 15, "15", ">15", "cars".
sum_range	The actual cells to sum. These cells are only summed if their corresponding cells in range match the criteria. If this argument is omitted, the cells in range are summed.

See Also

AVERAGE

COUNT

COUNTA

COUNTIF

PRODUCT

SUM

SUMSQ

Description

Squares each of the supplied numbers and returns the sum of the squares.

Syntax

SUMSQ (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.

Example

This function returns 302:

```
SUMSQ(9, 10, 11)
```

See Also

SUM

SYD

Description

Returns the depreciation of an asset for a specified period using the sum-of-years method. This depreciation method uses an accelerated rate, where the greatest depreciation occurs early in the useful life of the asset.

Syntax

SYD (*cost*, *salvage*, *life*, *per*)

Parameter	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
period	The period for which to calculate the depreciation. The time units used to determine period and life must match.

Example

This function returns 1607.14:

```
SYD(10000, 1000, 7, 3)
```

See Also

DDB

SLN

VDB

T

Description

Tests the supplied value and returns the value if it is text.

Syntax

T (*value*)

Parameter	Description
value	The value to test.

Remarks

Empty text ("") is returned for any value that is not text.

Examples

This function returns Report:

```
T(“Report”)
```

This function returns empty text (" ") if A4 contains a number:

```
T(A4)
```

See Also

N

VALUE

TAN

Description

Returns the tangent of the specified angle.

Syntax

TAN (*number*)

Parameter	Description
number	The angle in radians. To convert a number expressed as degrees to radians, multiply the degrees by PI( )/180.

Examples

This function returns 0.752:

```
TAN(0.645)
```

This function returns 1:

```
TAN(45*PI()/180)
```

See Also

ATAN

ATAN2

PI

TANH

TANH

Description

Returns the hyperbolic tangent of a number.

Syntax

TANH (*number*)

Parameter	Description
number	Any number.

Examples

This function returns –.96:

```
TANH(–2)
```

This function returns .83:

```
TANH(1.2)
```

See Also

ATANH

COSH

SINH

TAN

TEXT

Description

Returns the given number as text, using the specified formatting.

Syntax

TEXT (*number*, *format*)

Parameter	Description
number	Any value, a formula that evaluates to a number, or a reference to a cell that contains a value.
format	A string representing a number format. The string can be any valid format string including “General,” “M/DD/YY,” or “H:MM AM/PM.” The format must be surrounded by a set of double quotation marks. Asterisks cannot be included in format.

Examples

This function returns 123.620:

```
TEXT(123.62, “0.000”)
```

This function returns 10/19/94:

```
TEXT(34626.2, “MM/DD/YY”)
```

See Also

DOLLAR

FIXED

T

VALUE

TIME

Description

Returns a serial number for the supplied time.

Syntax

TIME (*hour*, *minute*, *second*)

Parameter	Description
hour	A number from 0 to 23.
minute	A number from 0 to 59.
second	A number from 0 to 59.

Examples

This function returns .52:

```
TIME(12, 26, 24)
```

This function returns .07:

```
TIME(1, 43, 34)
```

See Also

HOUR

MINUTE

NOW

SECOND

TIMEVALUE

TIMEVALUE

Description

Returns a serial number for the supplied text representation of time.

Syntax

TIMEVALUE (*text*)

Parameter	Description
text	A time in text format.

Examples

This function returns .07:

```
TIMEVALUE("1:43:43 am")
```

This function returns .59:

```
TIMEVALUE("14:10:07")
```

See Also

HOUR

MINUTE

NOW

SECOND

TIME

TODAY

Description

Returns the current date as a serial number.

Syntax

TODAY ( )

Remarks

This function is updated only when the worksheet is recalculated.

See Also

DATE

DAY

NOW

TRIM

Description

Removes all spaces from text except single spaces between words.

Syntax

TRIM (*text*)

Parameter	Description
text	Any text string or a reference to a cell that contains a text string.

Remarks

Text that is imported from another environment may require this function.

Example

This function returns Level 3, Gate 45:

```
TRIM(“ Level 3, Gate 45 ”)
```

See Also

CLEAN

MID

REPLACE

SUBSTITUTE

TRUE

Description

Returns the logical value True. This function always requires the trailing parentheses.

Syntax

TRUE ( )

See Also

FALSE

TRUNC

Description

Truncates the given number to an integer.

Syntax

TRUNC (*number* [, *precision*])

Parameter	Description
number	Any value.
precision	The number of decimal places allowed in the truncated number. Omitting this argument assumes a precision of 0.

Remarks

TRUNC removes the fractional part of a number to the specified precision without rounding the number.

Examples

This function returns 123.45:

```
TRUNC(123.456, 2)
```

This function returns 9800:

```
TRUNC(9899.435, –2)
```

See Also

CEILING

FLOOR

INT

MOD

ROUND

TYPE

Description

Returns the argument type of the given expression.

Syntax

TYPE (*expression*)

Parameter	Description
expression	Any expression.

Remarks

The valid values returned by this argument are:

Number	Description
1	Number
2	Text string
4	Logical value
16	Error value

Examples

This function returns 1 if cell A1 contains a number:

```
TYPE(A1)
```

This function returns 2:

```
TYPE(“Customer”)
```

See Also

ISBLANK

ISERR

ISERROR

ISLOGICAL

ISNA

ISNONTEXT

ISNUMBER

ISREF

ISTEXT

UPPER

Description

Changes the characters in the specified string to uppercase characters.

Syntax

UPPER (*text*)

Parameter	Description
text	Any string.

Remarks

Numeric characters in the string are not changed.

Examples

This function returns 3RD QUARTER:

```
UPPER(“3rd Quarter”)
```

This function returns JOHN DOE:

```
UPPER(“JOHN DOE”)
```

See Also

LOWER

PROPER

VALUE

Description

Returns the specified text as a number.

Syntax

VALUE (*text*)

Parameter	Description
text	Any text string, a formula that evaluates to a text string, or a cell reference that contains a text string. You can also specify a date or time in a recognizable format (for example, M/DD/YY for dates or H:MM AM/PM for time). If the format is not recognized, #VALUE! is returned.

Examples

This function returns 9800:

```
VALUE(9800)
```

This function returns 123:

```
VALUE(“123”)
```

See Also

DOLLAR

FIXED

TEXT

VAR

Description

Returns the variance of a population based on a sample of values.

Syntax

VAR (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example

This function returns .31:

```
VAR(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV

STDEVP

VARP

VARP

Description

Returns the variance of a population based on an entire population of values.

Syntax

VARP (*number_list*)

Parameter	Description
number_list	A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example

This function returns .27:

```
VARP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also

STDEV

STDEVP

VAR

VDB

Description

Returns the depreciation of an asset for a specified period using a variable method of depreciation.

Syntax

VDB (*cost*, *salvage*, *life*, *start_period*, *end_period* [, *factor*] [, *method*])

Parameter	Description
cost	The initial cost of the asset.
salvage	The salvage value of the asset.
life	The number of periods in the useful life of the asset.
start_period	The beginning period for which to calculate the depreciation. The time units used to determine *start_period* and *life* must match.
end_period	The ending period for which to calculate the depreciation. The time units used to determine *end_period* and *life* must match.
factor	The rate at which the balance declines. Omitting this argument assumes a default of 2, which is the double-declining balance factor.
method	A logical value that determines if you want to switch to straight-line depreciation when depreciation is greater than the declining balance calculation. Use True to maintain declining balance calculation; use False or omit the argument to switch to straight-line depreciation calculation.

Example

This function returns 1041.23:

```
VDB(10000, 1000, 7, 3, 4)
```

See Also

DDB

SLN

SYD

VLOOKUP

Description

Searches the first column of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Syntax

VLOOKUP (*search_item*, *search_range*, *column_index*)

Parameter	Description
search_item	A value, text string, or reference to a cell containing a value that is matched against data in the top row of *search_range*.
search_range	The reference of the range (table) to be searched. The cells in the first column of *search_range* can contain numbers, text, or logical values. The contents of the first column must be in ascending order (for example, –2, –1, 0, 2...A through Z, False, True). Text searches are not case-sensitive.
column_index	The column in the search range from which the matching value is returned. *column_index* can be a number from 1 to the number of rows in the search range. If *column_index* is less than 1, #VALUE! is returned. When *column_index* is greater than the number of rows in the table, #REF! is returned.

Remarks

VLOOKUP compares the information in the first column of *search_range* to the supplied *search_item*. When a match is found, information located in the same row and supplied column (*column _index*) is returned.

If *search_item* cannot be found in the first column of *search_range*, the largest value that is less than search_item is used. When *search_item* is less than the smallest value in the first column of the search_range, #REF! is returned.

Examples

The following examples use this worksheet.

	A	B	C	D	E
1	Employee	Start Date	Emp. No.	Salary	Exempt
2	Anderson	10/15/84	2348	$37,800	Y
3	Clark	2/6/90	4891	$28,700	N
4	Davis	6/21/80	2480	$46,950	Y
5	Franklin	4/20/88	3793	$30,275	Y
6	Lee	8/30/89	3961	$25,000	N
7	Olson	11/1/81	2578	$45,780	Y
8	Turner	2/15/93	5129	$26,100	N
9	Wilson	9/1/89	3965	$31,650	Y

This function returns $28,700:

```
VLOOKUP("Clark", A2:E9, 4)
```

This function returns 3961:

```
VLOOKUP("Lee", A2:E9, 3)
```

See Also

HLOOKUP

INDEX

LOOKUP

MATCH

WEEKDAY

Description

Returns the day of the week that corresponds to the supplied date.

Syntax

WEEKDAY (*serial_number*)

Parameter	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Remarks

WEEKDAY returns a number ranging from 1 (Sunday) to 7 (Saturday).

Examples

This function returns 1, indicating Sunday:

```
WEEKDAY(34399.92)
```

This function returns 3, indicating Tuesday:

```
WEEKDAY("06/21/94")
```

See Also

DAY

NOW

TEXT

TODAY

YEAR

Description

Returns the year that corresponds to the supplied date.

Syntax

YEAR (*serial_number*)

Parameter	Description
serial_number	The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Examples

This function returns 1993:

```
YEAR(34328)
```

This function returns 1994:

```
YEAR("06/21/94")
```

See Also

DAY

HOUR

MINUTE

MONTH

NOW

SECOND

TODAY

WEEKDAY

Working with Objects

Formula One provides the ability to create objects in a worksheet. Among the objects you can create are charts, lines, rectangles, ovals, arcs, polygons, push buttons, check boxes, and drop down list boxes. As with other worksheet elements, Formula One provides a wide range of options for formatting and manipulating the appearance of objects you create.

Creating Objects

Formula One provides several methods for creating objects in worksheets:

- Objects can created by calling methods in your code.
- The Workbook Designer allows objects to be created interactively.
- You can set the mode of the mouse to allow object drawing at any time.

Creating Objects with Methods

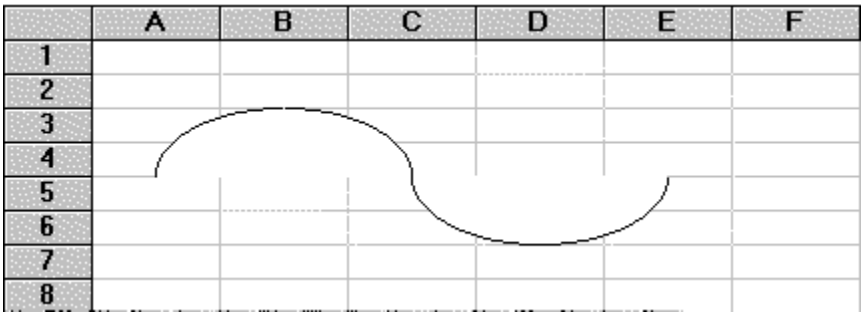
Formula One provides several methods that allow you to create objects from your application code.

- **ObjNew** allows you to create lines, rectangles, ovals, arcs, buttons, check boxes, drop down list boxes, and charts.
- **OjbNewPicture** creates a new metafile picture object on a worksheet.

The following example creates four adjoining arcs using the **ObjNew** method:

```
Dim objID1 As Long
Dim objID2 As Long
Dim objID3 As Long
Dim objID4 As Long
F1Book1.ObjNew F1ObjArc, .5, 2, 1.5, 4, objID1
F1Book1.ObjNew F1ObjArc, 2.5, 2, 1.5, 4, objID2
F1Book1.ObjNew F1ObjArc, 2.5, 6, 3.5, 4, objID3
F1Book1.ObjNew F1ObjArc, 4.5, 6, 3.5, 4, objID4
```

The following illustration shows the result of the example code.



Creating Objects
Interactively Drawing Objects
Picture Objects
Setting Mouse Mode

Interactively Drawing Objects

The Workbook Designer toolbar contains buttons that allow you to interactively create and edit objects. The following table describes the buttons on the toolbar.

Button	Name	Description
	Worksheet tool	Performs standard worksheet editing. Also selects objects for editing. Press CTRL when selecting buttons, check boxes, and drop down list boxes.
	Polygon editing mode	Toggles between normal polygon editing and polygon point editing. When polygon point editing is enabled, the points of a polygon can be moved.
	Line tool	Draws lines.
	Rectangle tool	Draws rectangles and squares.
	Oval tool	Draws ovals and circles.
	Arc tool	Draws arcs.
	Polygon tool	Draws polygons.
	Button tool	Draws push buttons.
	Check box tool	Draws check box objects.
	List box tool	Draws drop down list box objects.
	Chart tool	Charts the selected range of data.

Interactively drawing objects in the Workbook Designer is as simple as point, click, and drag.

To draw an object in the Workbook Designer:

1. Select the tool for the object you want to draw.

   The pointer appears as a small cross when positioned in the worksheet.

2. Position the pointer at the point where you want to begin drawing.

3. Click and drag to draw the object.

   An outline of the object you are drawing appears and moves as you drag the mouse.

4. Release the mouse button to set the object.

   Note When drawing an object, press ALT to align the object to the cell grid.

Creating Objects

Creating Objects with Methods

Picture Objects

Setting Mouse Mode

Picture Objects

You can place a picture object on a worksheet and fill it with a metafile. Formula One provides two methods for doing this, one for drawing the picture object, and one for filling it with a metafile.

- Use **ObjNewPicture** to create a picture object on the current worksheet. You must specify the position for the new picture object.
When specifying the location of the picture object, integers place the edge of the object on a row or column border; fractional numbers place the edge of the object between borders.

- Use **ObjSetPicture** to place a metafile in an existing object. You must provide a handle to the metafile and the ID number of the picture object into which you want the metafile placed. Any metafile previously contained by the picture object is freed from memory.
These methods also pass information about the dimensions of the picture and whether the picture can be stretched. Formula One manages the memory associated with a metafile once a picture object has been created, including freeing memory when the object is deleted.

You should be familiar with Windows metafiles and their structure before using these methods.

Creating Objects

Creating Objects with Methods

Interactively Drawing Objects

Setting Mouse Mode

Setting Mouse Mode

By setting the mode for mouse actions, you can allow objects to be drawn interactively in a worksheet.

- Set the **Mode** property to change the mouse mode. You can specify that the mouse draw charts, lines, rectangles, ovals, arcs, buttons, polygons, check boxes, and drop down list boxes. You can also specify that the mouse assume normal worksheet editing mode.
- Return the value of the **Mode** property to return the current mouse mode.

The following example uses **Mode** to set the mouse mode to rectangle drawing when you double click a worksheet.

```
Private Sub F1Book1_Click(ByVal nRow As Long, ByVal nCol As Long)
    F1Book1.Mode = F1ModeRectangle
End Sub
```

Interactively Drawing Objects

Identifying Objects

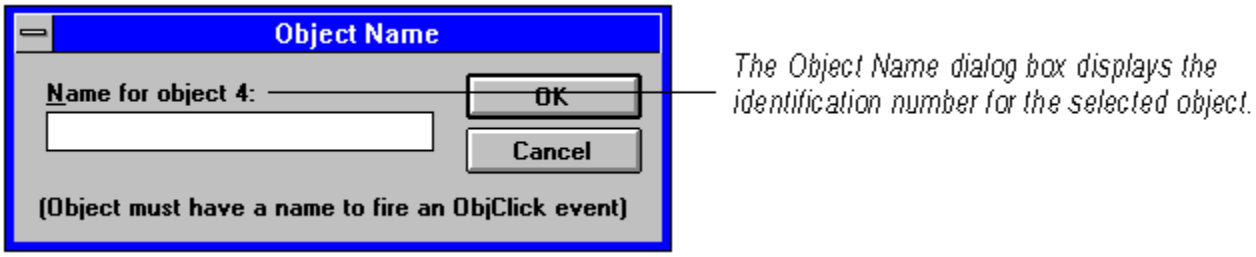
When you create an object – whether by method, in the Workbook Designer, or by setting the mouse mode – the object is assigned an identification number. Many methods or properties require an object identification number to tell Formula One on which object to operate.

If you are uncertain of an object's identification number, there are several ways to determine it.

To determine an objects number in the Workbook Designer:

1. Select the object for which you want to determine the identification number.
2. Choose Name from the Object menu to display the Object Name dialog box.

The dialog box displays the identification number of the selected object.



The Object Name dialog box displays the identification number for the selected object.

To determine an objects number programmatically, use one of the following methods:

- If an object is selected, the **ObjGetSelection** method returns the identification number of the selected object.
- Formula One maintains a list of objects in each worksheet within a workbook. The order of the objects in that list is determined by the order in which objects are drawn in the worksheet. The farther to the back an object is placed or drawn, the higher the object is placed in the list; the closer to the front an object is placed or drawn, the lower the object is placed in the list. Therefore, when you call **ObjFirstID**, this method returns the identification number of the first object in the list, which is the object drawn farthest back in the layers of worksheet objects. Then, **ObjNextID** returns the identification number of the next closest object, and so on.
When you use the **ObjBringToFront** and **ObjSendToBack** methods or the Bring To Front and Send To Back commands from the Object menu in the Workbook Designer, you alter the order of the object list maintained by the worksheet.

- If an object has been named, you can return the identification number of the named object with the **ObjNameToID** method.

Creating Objects

Naming Objects

Selecting Objects

Moving, Sizing, and Arranging Objects

Formatting Objects

Naming Objects

As mentioned in the previous section, objects can be named after they are created. Object names do not take the place of object identification numbers. Rather, object names are used as a supplement to identification numbers, making it easier to track and manipulate objects.

Another purpose for naming objects is to enable the **ObjClick**, **ObjDblClick**, **ObjGotFocus**, **ObjLostFocus**, and **ObjValueChanged** events. Before these events can fire, you must name the object receiving the action.

Objects can be named in the Workbook Designer or programmatically.

To name an object in the Worksheet Designer:

1. Select the object you want to name.
2. Choose Name from the Object menu to display the Object Name dialog box.
3. In the entry field, enter the name to assign to the object.

To name an object in code:

- Set the **ObjName** property to name an object. Once an object is named, return the value of **ObjName** to obtain the name assigned to an object. You can also call **ObjNameDlg** to display the Object Name dialog box.

Note After an object is named, you must press CTRL to select it at run time. In addition, if the **ObjClick** or **ObjDblClick** events are enabled, you must press CTRL when clicking the object.

Creating Objects

Identifying Objects

Selecting Objects

Moving, Sizing, and Arranging Objects

Formatting Objects

Selecting Objects

When working in the Workbook Designer or in a workbook at run time, selecting objects is an important first step when you are formatting, moving, or resizing objects. In addition, many methods – such as **SetPattern**, **SetLineStyle**, **ObjBringToFront**, and **ObjSendToBack** – require that an object be selected for the method to work.

Interactively Selecting Objects

Selecting objects interactively in the Workbook Designer or in a worksheet at run time is as simple as pointing and clicking.

To interactively select an object:

1. Position the pointer on the object you want to select.

   The pointer appears as an arrow when positioned on an object.

2. Click the object.

When the object is selected, selection handles appear at the edges of the bounding box that surrounds the object.

Note For check boxes, list boxes, and buttons that have been named or if the **ObjClick** or **ObjDblClick** events are enabled, you must press CTRL when selecting the object. In addition, in a worksheet at run time, you must press CTRL to select arcs, lines, ovals, polygons, and rectangles that have been named.

To select multiple objects, press SHIFT as you select the objects. For objects that have been named, you must press CTRL+SHIFT to select multiple objects.

In the Workbook Designer, you can choose Select All from the Objects menu to select all the objects in a worksheet.

Selecting Objects

Limiting Interactive Object Selection

Selecting Objects Programmatically

Selecting Check Box and List Box Items

Limiting Interactive Object Selection

Setting the **AllowObjSelections** property allows you to specify whether users can interactively select objects in the Workbook Designer or in a worksheet at run time. Return the value of **AllowObjSelections** to determine if object selections are allowed.

Selecting Objects

Interactively Selecting Objects

Selecting Objects Programmatically

Selecting Objects Programmatically

To select objects programatically, use one or more of the following:

- **ObjSetSelection** selects the object you specify by object identification number. This method unselects any previously selected objects or worksheet ranges.
- **ObjAddSelection** adds an object to the current selection. Any previously selected objects remain selected.
- **ObjGetSelection** returns the identification number of the selected object. If more than one object is selected, you must indicate for which object to return an identification number.
- **ObjGetSelectionCount** returns the number of objects currently selected.

Selecting Objects

Interactively Selecting Objects

Limiting Interactive Object Selection

Moving, Sizing, and Arranging Objects

After an object is created, you can change both the position and size of the object. Formula One allows these changes to be made interactively or programmatically.

Interactively Positioning Objects

Interactively, objects can be resized and repositioned at run time or in the Workbook Designer.

To interactively resize and reposition an object:

1. Select the object to be moved or sized.

   Note To select a named button, check box, or list box, you must press CTRL when selecting the object. Or, if the **ObjClick** or **ObjDblClick** events are enabled, you must press CTRL when clicking the object.

   When you select an object, selection handles appear along the bounding box that surrounds the object.

2. If moving the object, position the pointer anywhere in the area occupied by the object. If resizing the object, position the pointer on one of the selection handles.

   When positioned in the object area, the pointer appears as an arrow. When positioned on a selection handle, the pointer appears as a two-headed arrow, indicating the direction in which the object can be resized.

3. Click and drag to move or resize the object.

   An outline of the object moves with the pointer as you drag the mouse.

   Note When moving or resizing an object, you can align the object to the worksheet grid by pressing ALT as you click and drag.

4. Release the mouse to set the object at its new position or at its new size.

Selecting Objects

Moving, Sizing, and Arranging Objects

Positioning Objects Programmatically

Determining Object Position and Size

Arranging Overlapping Objects

Arranging Objects and Object Order

Positioning Objects Programmatically

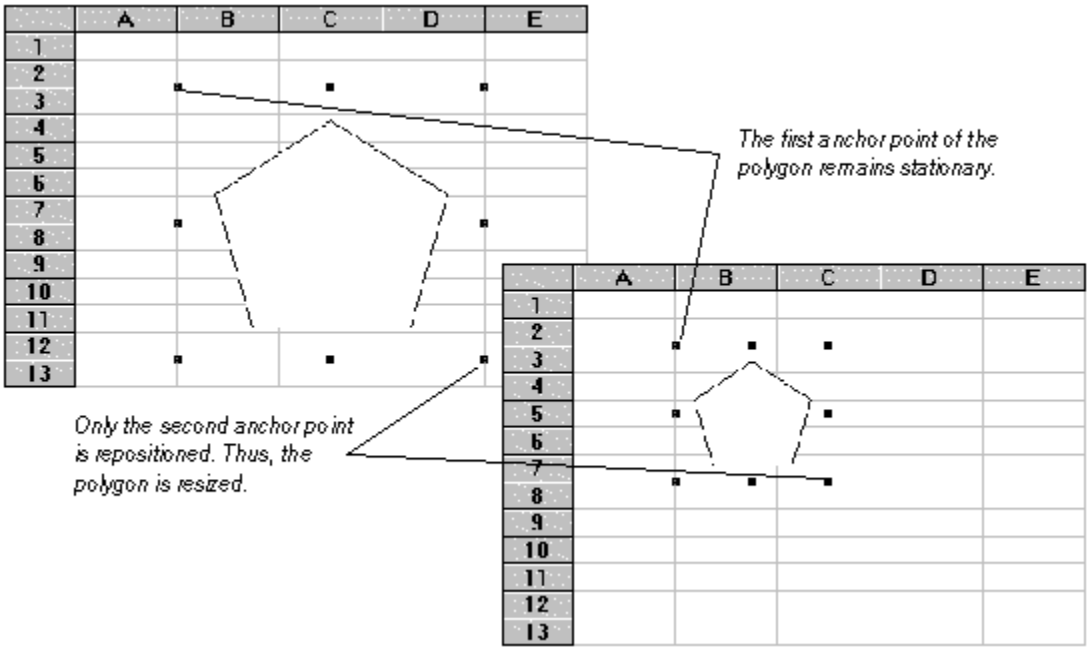
Use the **ObjSetPos** method to set the position of an object. Because this method sets the placement of both anchor points of the bounding box that surrounds an object, this method also resizes objects.

If you are uncertain of an object's location, use the **ObjGetPos** to return the object's position. The following example uses **ObjGetPos** to obtain the position of the object. Then, **ObjSetPos** resizes the object by moving the second anchor point left one and a half columns and up five rows.

```
Dim x1 As Single
Dim y1 As Single
Dim x2 As Single
Dim y2 As Single

F1Book1.ObjGetPos 1, x1, y1, x2, y2
F1Book1.ObjSetPos 1, x1, y1, (x2 - 1.5), (y2 - 5)
```

The following illustration shows the results of the example.



Selecting Objects Programmatically
Moving, Sizing, and Arranging Objects
Interactively Positioning Objects
Determining Object Position and Size
Arranging Overlapping Objects
Arranging Objects and Object Order

Determining Object Position and Size

The **ObjPosToTwips** method returns the placement and size of an object in twips. For this method, you provide the position of an object's anchor points in relation to the rows and columns of a worksheet. The method also determines whether the object is shown, partially shown, or hidden given the current dimensions of the view.

ObjPosToTwips does not reference a specific object on a worksheet and has no effect on any objects.

Moving, Sizing, and Arranging Objects

Interactively Positioning Objects

Positioning Objects Programmatically

Arranging Overlapping Objects

Arranging Objects and Object Order

Arranging Overlapping Objects

When you have multiple objects on a worksheet, they appear to be drawn on the same plane. However, when two objects overlap, the previously drawn object is covered by the latter-drawn object. You can change the order of object layering in a worksheet by sending an object behind other objects or bringing an object to the front of other objects.

To arrange objects in the Workbook Designer:

1. Select the object you want to move.
2. Choose Bring to Front or Send to Back from the Object menu. The object is moved in the direction you specify.

To arrange objects in code:

- Call the methods **ObjBringToFront** or **ObjSendToBack** to arrange objects. These methods move only the selected objects.

If multiple objects are selected when using these commands and methods, the order of objects within the selection remains unchanged. The selected objects are placed in front of or behind only the unselected objects.

Moving, Sizing, and Arranging Objects

Interactively Positioning Objects

Positioning Objects Programmatically

Determining Object Position and Size

Arranging Objects and Object Order

Arranging Objects and Object Order

When you use the **ObjBringToFront** and **ObjSendToBack** methods or the Bring To Front and Send To Back commands from the Object menu in the Workbook Designer, you alter the order of the object list maintained by the worksheet.

Moving, Sizing, and Arranging Objects

Interactively Positioning Objects

Positioning Objects Programmatically

Determining Object Position and Size

Arranging Overlapping Objects

Formatting Objects

Many elements of objects can be formatted including:

- fill patterns and colors, and line colors, widths and styles for arcs, lines, ovals, polygons, and rectangles.
- the lists of items contained by drop down list boxes.
- the text displayed by check boxes or push buttons.

Formatting Colors and Patterns

To set the fill colors and patterns for graphical objects – arcs, ovals, polygons, and rectangles – use the following methods:

To set fill colors and patterns for selected objects in the Workbook Designer:

- Choose Pattern from the Object menu. The Pattern dialog box is displayed from which you can make your pattern and color choices.

To set fill colors and patterns for selected object in code:

- Use **SetPattern** to set the fill colors and pattern for the selected objects. You can also call **FormatPatternDlg** to display the Pattern dialog box.

To set the line style, color, and weight for line objects and the lines that form the borders of arcs, ovals, polygons, and rectangles, use the following approaches:

To format lines for selected objects in the Workbook Designer:

- Choose Line Style from the Object menu. The Line Style dialog box is displayed from which you can choose line attributes for the selected objects.

To format lines for selected objects in code:

- Use **SetLineStyle** to set the line style, color, and weight for the selected objects. You can also call **LineStyleDlg** to display the Line Style dialog box.

The following table shows the line styles that can be applied to line objects and the borders of arcs, ovals, polygons, and rectangles:

Line styles	
Solid	
Dashed	
Ditted	
Dash-ditted	
Dash-dit-ditted	
None	

Solid lines can be 1/2 point, 1 point, 2 points, or 3 points in weight. Styled lines can only be 1/2 point in weight.

Formatting Objects

Showing and Hiding Objects

Formatting List Boxes

Formatting Check Boxes

Formatting Buttons

Editing Polygons

Showing and Hiding Objects

An object on a worksheet can be hidden by setting the **ObjVisible** property to False. To display a hidden object, set the property to True. If you are uncertain whether an object is shown or hidden, return the value of **ObjVisible**.

Formatting Objects

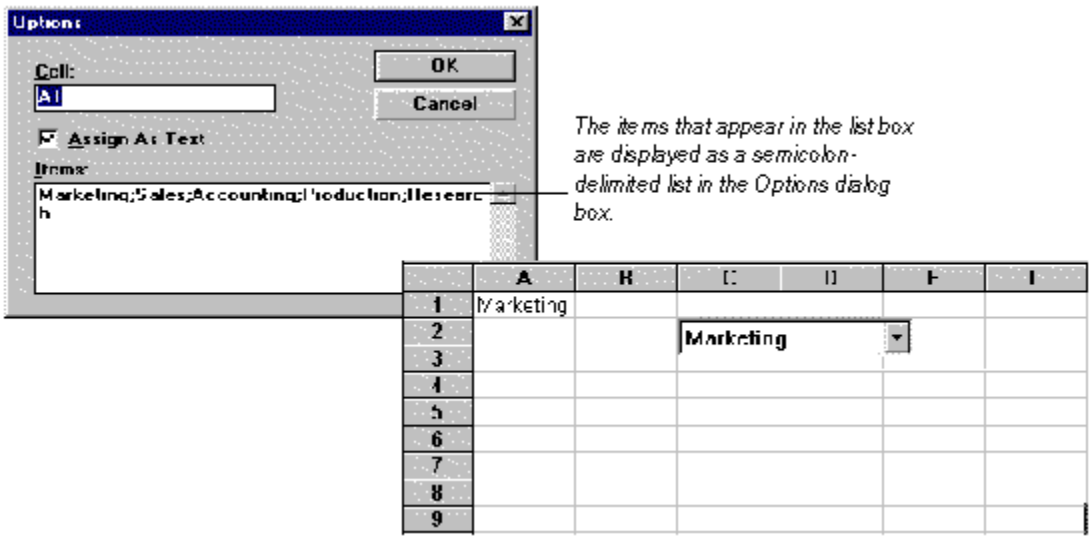
Formatting List Boxes

List boxes can be formatted by setting or changing the cell they reference, whether a selection appears in the referenced cell as a value or text, and specifying the list of selections in the list box.

To set or edit list box items in the Workbook Designer:

1. Select a list box
2. Choose Options from the Object menu to display the Options dialog box.
3. Enter or edit the list of items contained by the list box. The items must be entered as a semicolon-delimited list.

The following illustration shows the Options dialog box and a list box in a worksheet.



To set and manipulate the items contained in a list box programmatically, use the following properties:

- Set **ObjItems** to specify a list of items for a list box. For this property, you provide a semicolon-delimited list of items. The list you provide replaces any previously-specified lists. Return the value of **ObjItems** to get a semicolon-delimited list of items from a list box.
- To change an item in a list of items, set **ObjItem**. For this property, you provide the number of the item you want to change and the new value for the item. Return the value of **ObjItem** to get a specific item from a list box.
- To add an item to a list of items, use **ObjAddItem**. This method adds an item to the end of the current list. Use **ObjInsertItem** to add an item at a specific location within a list.
- Use **ObjDeleteItem** to delete an item from a list box.
- **ObjGetItemCount** returns the number of items contained by a list box.
- Use **ObjOptionsDlg** to display the Options dialog box.

Selecting Objects

Limiting Interactive Object Selection

Formatting Objects

Showing and Hiding Objects

Formatting Check Boxes

Formatting Buttons

Formatting Check Boxes

The text displayed by a check box can be set either through the Workbook Designer or programmatically.

To edit check box text in the Workbook Designer:

1. Select a check box.
2. Choose Options from the Object menu to display the Options dialog box. In this dialog box, you can enter and edit the text displayed by the check box.

To set the text displayed in a check box programmatically:

- Set the **ObjText** property. Return the value of **ObjText** to get the text displayed by a check box.

Selecting Objects

Limiting Interactive Object Selection

Selecting Objects Programmatically

Formatting List Boxes

Formatting Buttons

Selecting Check Box and List Box Items

Setting Values Interactively

Setting Values Programmatically

Setting Values by Cell Reference

Formatting Buttons

The text displayed on a button can be set either through the Workbook Designer or programmatically.

To edit button text in the Workbook Designer:

1. Select a button.
2. Choose Options from the Object menu to display the Options dialog box. In this dialog box, you can enter and edit the text displayed on the button.

To set the text displayed on a button programmatically:

- Set the **ObjText** property. Return the value of **ObjText** to get the text displayed on a button.

Selecting Objects

Limiting Interactive Object Selection

Selecting Objects Programmatically

Showing and Hiding Objects

Formatting List Boxes

Formatting Check Boxes

Selecting Check Box and List Box Items

Formula One provides a variety of methods for checking or unchecking check box objects and selecting items from drop down list box objects.

- At run time, items can be checked or selected interactively using the mouse.
- Properties and methods can set the value of a check box or list box object.
- By assigning a cell to an object, you can set the value in the cell by making a selection from the object. Likewise, if you enter a value that is in the list associated with a list box, you can change the value displayed in the list box. The cell reflects the value to which the object is set, regardless of the method used to set the value.

When you check or uncheck a check box, or select an item from list box, you set the value in the assigned cell.

- For a check box, the value of the assigned cell is True if the object is checked or False if unchecked.
- In a list box, you can choose to have the value of the assigned cell set to the number or the text of the selected item. Items in a list box are numbered consecutively, starting with 0 (e.g., the first item is item 0, the second item is item 1, and so on). -1 means that no item is selected in the list box.

Creating Objects

Selecting Objects

Limiting Interactive Object Selection

Selecting Objects Programmatically

Showing and Hiding Objects

Formatting List Boxes

Formatting Check Boxes

Setting Values Interactively

Setting Values Programmatically

Setting Values by Cell Reference

Setting Values Interactively

To set the value of a check box or list box interactively, the object itself cannot be selected and the pointer must be in normal worksheet editing mode.

- To set the value of a check box, position the pointer anywhere in the check box area and click. The check box toggles between checked and unchecked states.
- To set the value of a list box, position the pointer anywhere in the list box area and click. The list of items contained by the list box is displayed. If the list area is not large enough to display all the items, you may have to click the scroll areas to view the entire list. Then, click the item you want to select.

Formatting List Boxes

Formatting Check Boxes

Setting Values Programmatically

Set the **ObjValue** property to set the value of a check box or list box specified by object ID number.

- For check boxes, provide 1 to check a check box; provide 0 to uncheck the object.
- For list boxes, provide the number of the item you want to select. List box items are numbered consecutively, starting with 0. If you specify -1, no item is selected in the list box.

Return the value of **ObjValue** to get the value of the current selection.

- You can also use **ObjText** to set or return the text displayed on a button or next to a check box.

Formatting List Boxes

Formatting Check Boxes

Setting Values by Cell Reference

For both check boxes and list boxes, you can specify a cell that references the object. The referenced cell works in two ways:

- If you select an item from the object, the value or text of that selection is displayed in the referenced cell. For example, if you uncheck a check box, FALSE is placed in the cell the check box references.
- If you enter a valid value in the referenced cell, the current selection of the object reflects the cell value. For example, if you enter 0 in the cell, a list box that references the cell displays its first item as the current selection.

If multiple objects reference the same cell, making a selection in one object makes the same selection in all objects that reference the cell.

The cell reference for an object can be set in the Options dialog box, displayed by choosing Options from the Object menu when the object is selected. You can also specify a cell reference by calling the **ObjSetCell** method. **ObjGetCell** returns an object cell reference.

For list boxes only, you can specify whether the value or text of the selection is displayed in the cell.

Formatting List Boxes

Formatting Check Boxes

Editing Polygons

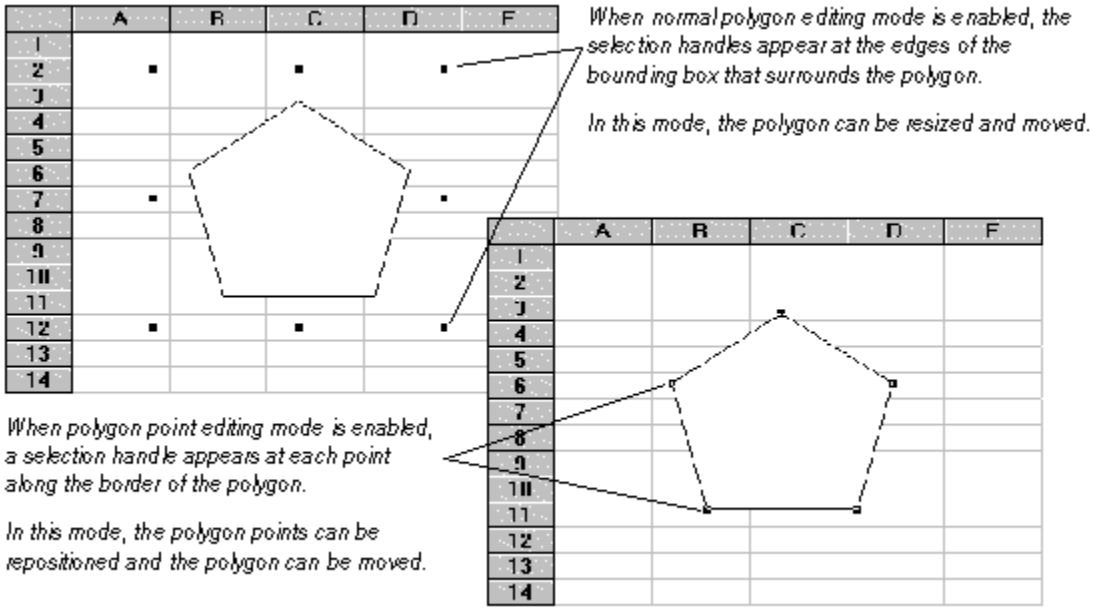
When editing polygons, there are two editing modes:

Normal Polygon Editing. This mode allows you to resize and move polygons. Editing of polygon points is not allowed in this mode.

Polygon Point Editing. This mode allows you to reposition polygon points, thus changing the shape of the polygon.

You can set the polygon editing mode by setting the **PolyEditMode** property. To determine the current polygon editing mode, return the value of **PolyEditMode**.

In the Workbook Designer, use the polygon editing mode button in the toolbar to toggle between the polygon editing modes. The following illustration shows a selected polygon when normal polygon editing and polygon point editing modes are enabled.



To interactively reshape a polygon:

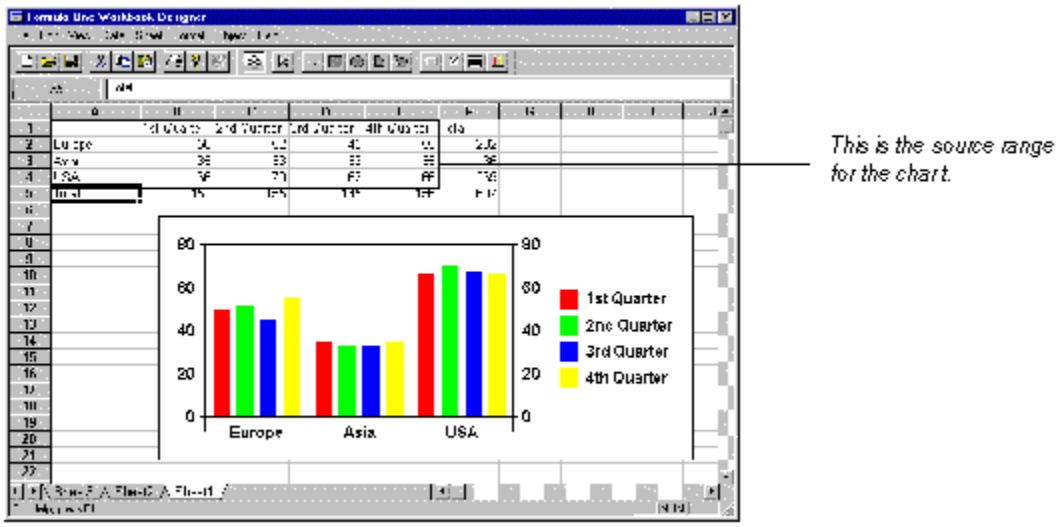
1. Select the polygon to be reshaped.
2. Make certain that polygon editing mode is enabled.

   When polygon editing mode is enabled, selection handles appear at each point along the border of the selected polygon.
3. Position the pointer on the polygon point that you want to move.
4. Click the point and drag the mouse.

   An outline of the lines adjoining the point move as you drag the polygon point.
5. Release the mouse button to place the point at its new location.

Working with Chart Objects

If you have also purchased Visual Component's First Impression OCX, you can automatically chart worksheet data. In order to draw a chart, First Impression must be properly installed on your system. The following illustration shows an example of a chart on a worksheet.



To create a chart programmatically;

- Use the **ObjNew** method with the F1ModeChart constant. This draws a chart object on the worksheet in the position you specify and charts the currently selected range.
- To change a chart's data range, you must select it, open the Object Options dialog box and provide a range reference or defined name for the chart.

To interactively chart data:

1. Select a range of data to be charted.
2. Click the chart button in the Tool bar.
3. Draw a rectangle where you want to place the chart.

   The Chart Wizard appears to assist you in designing the chart appearance.
4. Make any necessary selections from the Chart Wizard and click Finished.

Using the Chart Wizard

Chart Options

Referencing Data on Another Worksheet

Using the Chart Wizard

The tabs within the Chart Wizard guide you through most common design decisions required when you create or modify a chart. The Chart Wizard is displayed automatically when you draw a new chart on a worksheet. You can also display the Chart Wizard at any time to assist you in formatting an existing chart.

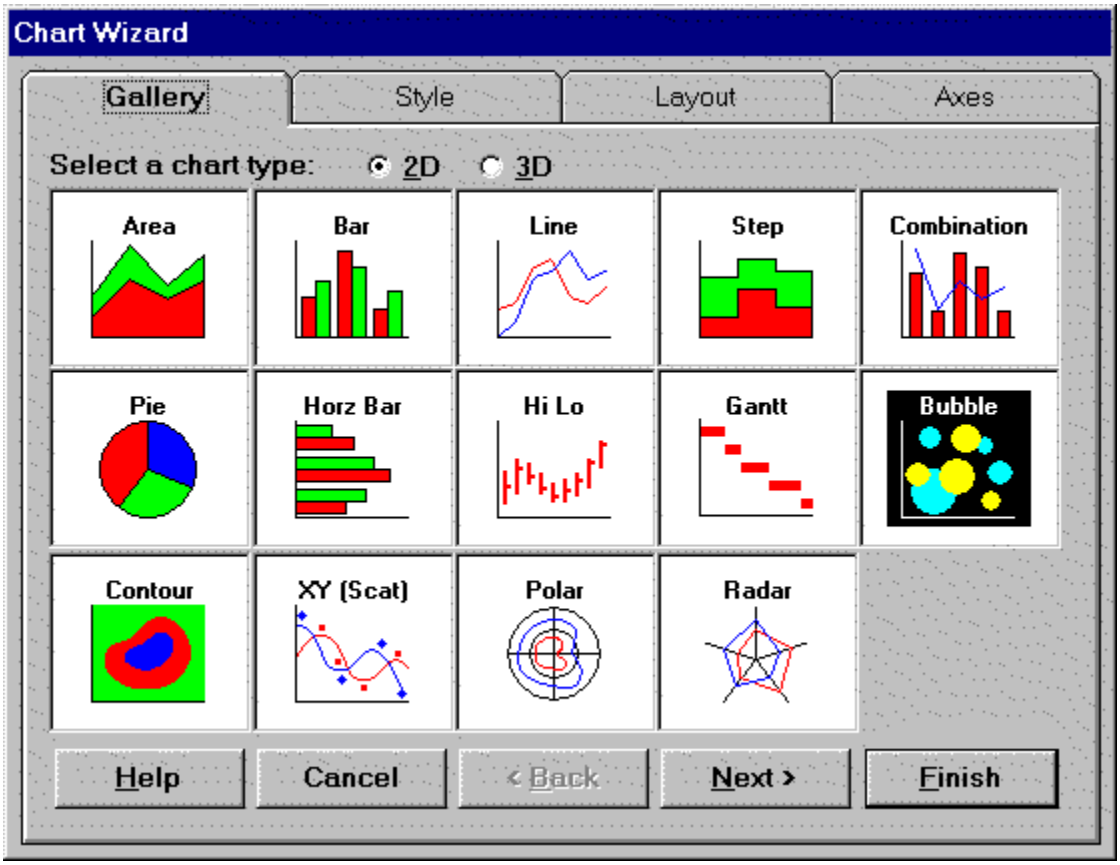
Important Each chart type has individual requirements as to how the data must be laid out. Separate formatting options are also available for different chart types. You should read your First Impression User's Guide to familiarize yourself with the requirements of various chart types.

To access the Chart Wizard:

1. Double-click on the chart to activate it.
2. Right-click the chart to display the shortcut menu.
3. Choose Wizard from the menu.

The Chart Wizard provides four tabs that control various design aspects such as choosing a chart type, setting chart options, controlling chart layout, and specifying chart and axis titles. You can navigate through these tabs by clicking on the tab at the top of the dialog box, or by using the navigation buttons at the bottom of the dialog box.

Click on the following illustration to display additional information about the Chart Wizard.



Working with Chart Objects

Using the Chart Wizard

The tabs within the Chart Wizard guide you through most common design decisions required when you create or modify a chart. The Chart Wizard is displayed automatically when you draw a new chart on a worksheet. You can also display the Chart Wizard at any time to assist you in formatting an existing chart.

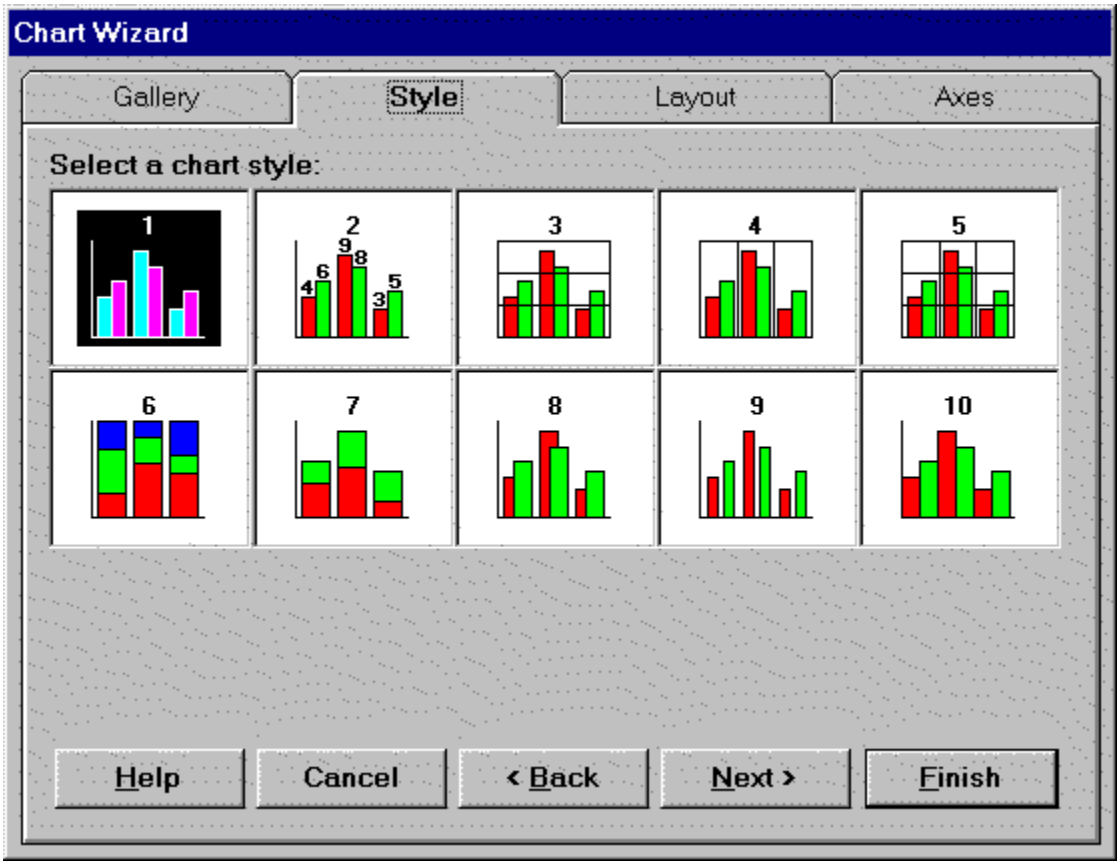
Important Each chart type has individual requirements as to how the data must be laid out. Separate formatting options are also available for different chart types. You should read your First Impression User's Guide to familiarize yourself with the requirements of various chart types.

To access the Chart Wizard:

1. Double-click on the chart to activate it.
2. Right-click the chart to display the shortcut menu.
3. Choose Wizard from the menu.

The Chart Wizard provides four tabs that control various design aspects such as choosing a chart type, setting chart options, controlling chart layout, and specifying chart and axis titles. You can navigate through these tabs by clicking on the tab at the top of the dialog box, or by using the navigation buttons at the bottom of the dialog box.

Click on the following illustration to display additional information about the Chart Wizard.



Working with Chart Objects

Using the Chart Wizard

The tabs within the Chart Wizard guide you through most common design decisions required when you create or modify a chart. The Chart Wizard is displayed automatically when you draw a new chart on a worksheet. You can also display the Chart Wizard at any time to assist you in formatting an existing chart.

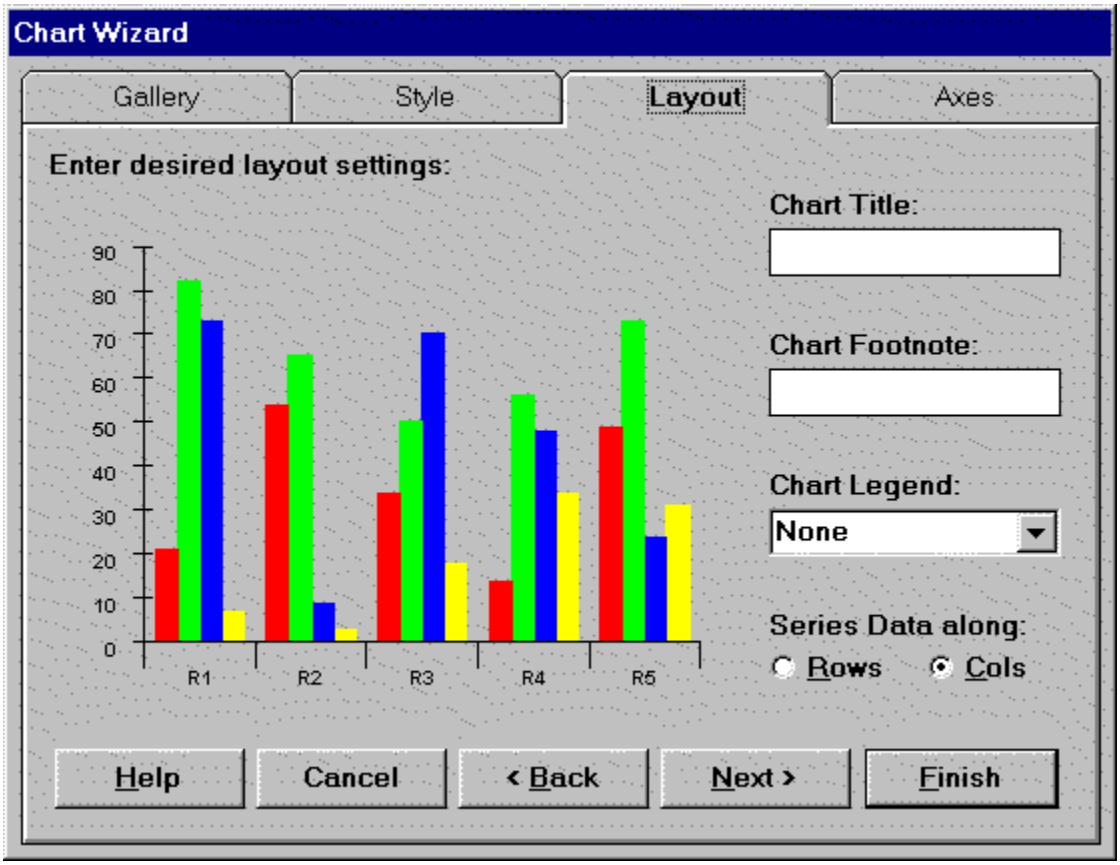
Important Each chart type has individual requirements as to how the data must be laid out. Separate formatting options are also available for different chart types. You should read your First Impression User's Guide to familiarize yourself with the requirements of various chart types.

To access the Chart Wizard:

1. Double-click on the chart to activate it.
2. Right-click the chart to display the shortcut menu.
3. Choose Wizard from the menu.

The Chart Wizard provides four tabs that control various design aspects such as choosing a chart type, setting chart options, controlling chart layout, and specifying chart and axis titles. You can navigate through these tabs by clicking on the tab at the top of the dialog box, or by using the navigation buttons at the bottom of the dialog box.

Click on the following illustration to display additional information about the Chart Wizard.



Working with Chart Objects

Using the Chart Wizard

The tabs within the Chart Wizard guide you through most common design decisions required when you create or modify a chart. The Chart Wizard is displayed automatically when you draw a new chart on a worksheet. You can also display the Chart Wizard at any time to assist you in formatting an existing chart.

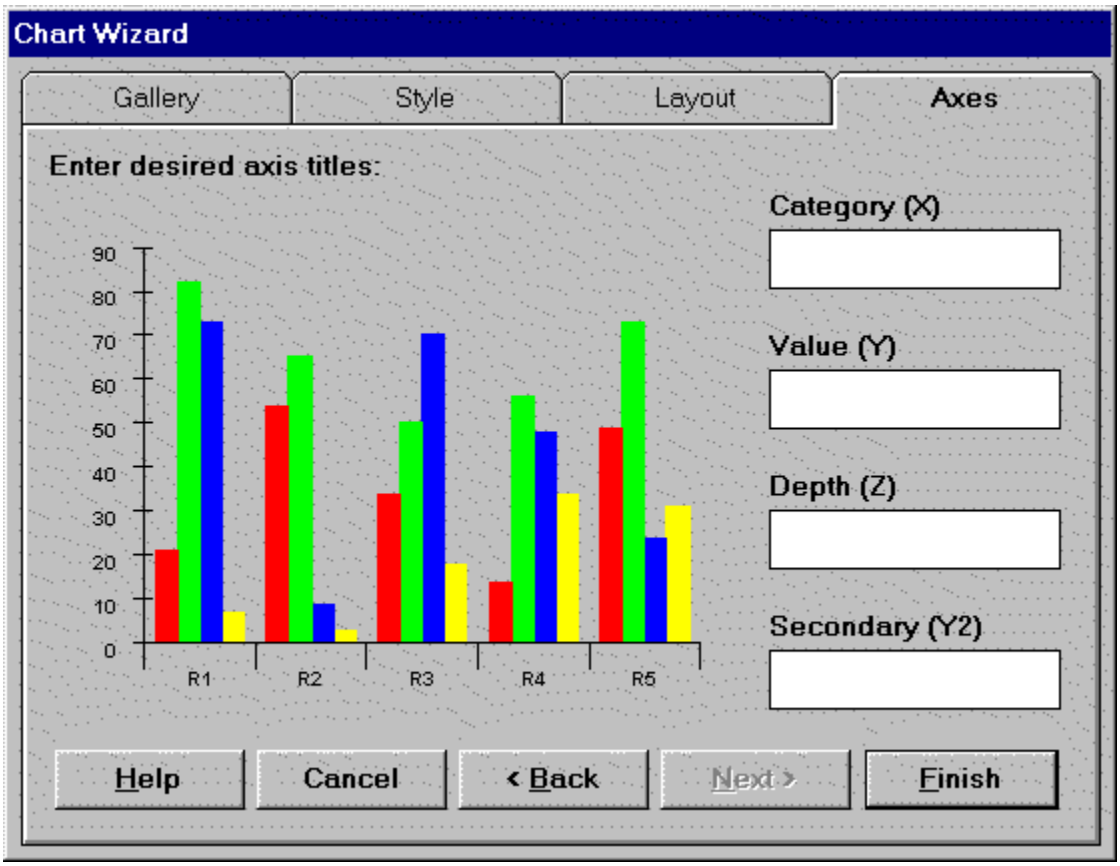
Important Each chart type has individual requirements as to how the data must be laid out. Separate formatting options are also available for different chart types. You should read your First Impression User's Guide to familiarize yourself with the requirements of various chart types.

To access the Chart Wizard:

1. Double-click on the chart to activate it.
2. Right-click the chart to display the shortcut menu.
3. Choose Wizard from the menu.

The Chart Wizard provides four tabs that control various design aspects such as choosing a chart type, setting chart options, controlling chart layout, and specifying chart and axis titles. You can navigate through these tabs by clicking on the tab at the top of the dialog box, or by using the navigation buttons at the bottom of the dialog box.

Click on the following illustration to display additional information about the Chart Wizard.



Working with Chart Objects

In the Gallery Tab, select the type of chart you wish to design. Two radio buttons control whether 2D chart or 3D chart types are displayed.

In the Style Tab, select a style for the chart type you have selected. Each style is a predefined set of display options such as series labels, stacking and bar gap.

In the Layout Tab, specify an optional chart title, footnote, chart legend position, and whether the series data in the charted range should be charted along rows or columns. The chart preview image on this tab shows you how the chart will look with these modifications.

In the Axes Tab, enter optional axis titles for any or all axes on your chart. The chart preview image on this tab shows you how the chart will look with the addition of the axis title.

Displays Wizard Help

Aborts changes.

Displays previous tab.

Displays next tab.

Applies modifications.

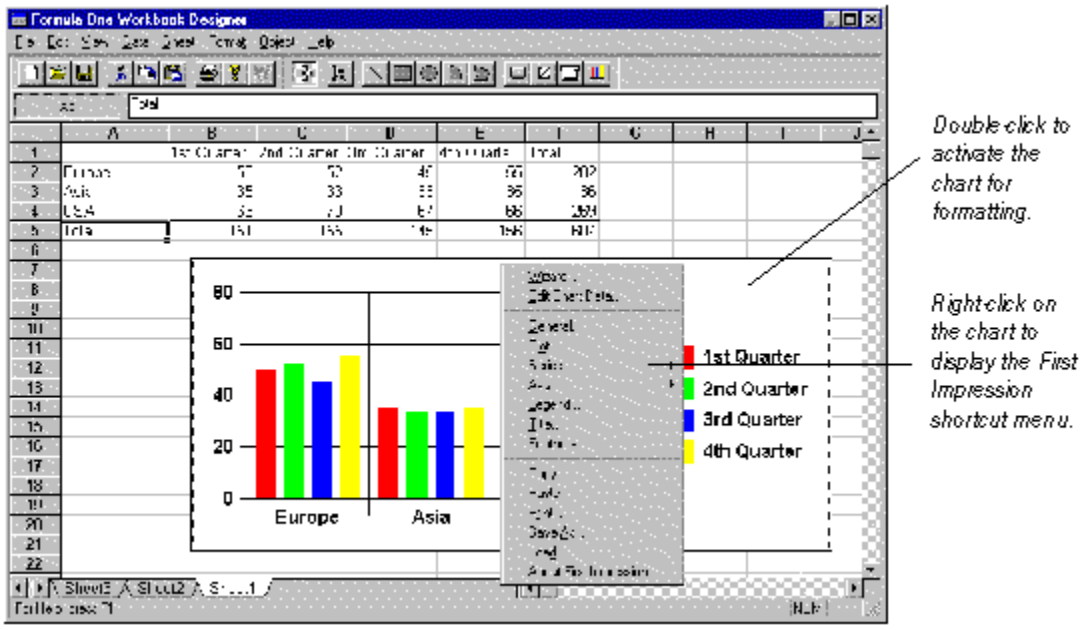
Additional Chart Formatting

The Chart Wizard provides a quick and easy method for applying some of the most common formatting options to your chart. Additional formatting options are available by using the First Impression dialog boxes. These dialog boxes are displayed by double-clicking on chart elements in an activated chart object, or selecting items from the First Impression shortcut menu. Consult your First Impression documentation for more information about these.

To access First Impression dialog boxes:

1. Double-click on the chart object to activate it.
2. Right-click on the chart to display the shortcut menu.
3. Make a selection from the menu.

Important One of the options on the First Impression shortcut menu is Edit Chart Data. This displays First Impression's Data Grid View tool. In this tool you can modify the size and content of the chart's data source. Any changes you make in the Data Grid View tool are reflected in the chart, but are NOT reflected in the worksheet. Also, if you use the Data Grid View tool to edit the chart data and then recalculate the workbook in Formula One while the chart still references a range in the worksheet, the worksheet data overwrites your changes. To prevent this, set the charts formula to an empty string (" ") before using the Data Grid View tool to edit the data.



Using the Chart Wizard

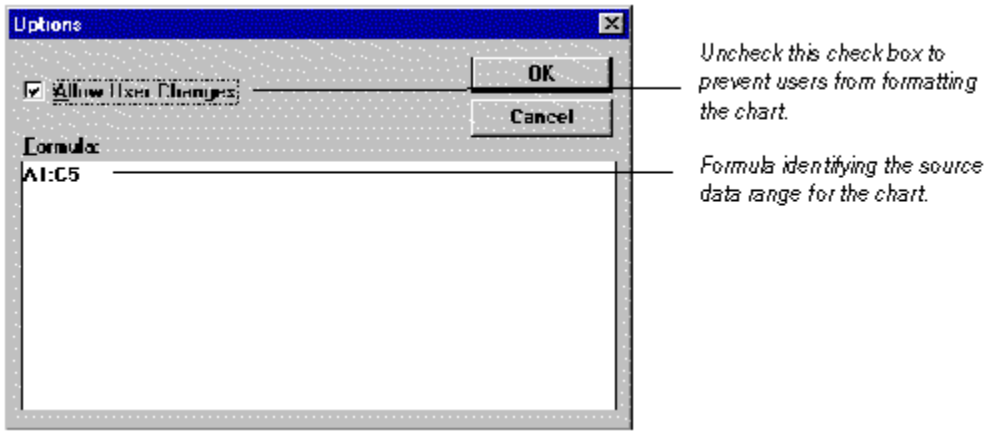
Chart Options

Within Formula One, you can control whether or not users can edit the chart, and change the source data range for a chart using the Object Options dialog box.

To edit chart options:

1. Select the chart.
2. Choose Options from the Object menu to display the object options dialog box.

   Developer's Note If you make the Workbook designer available to your end-users, they will have access to the Allow User Change check box. To prevent them from modifying a chart regardless of this setting, set the **AllowObjSelection** property to False.



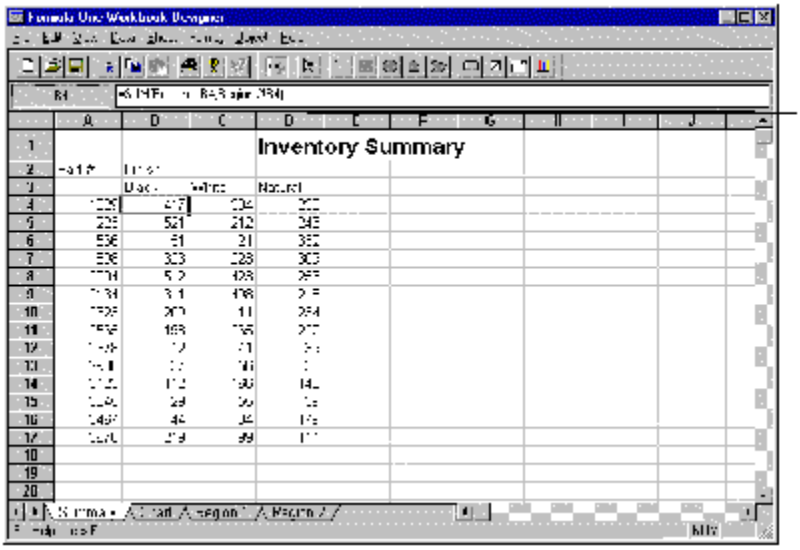
Referencing Data on Another Worksheet

Referencing Data on Another Worksheet

You can also display a chart on one worksheet that references data on a separate worksheet.

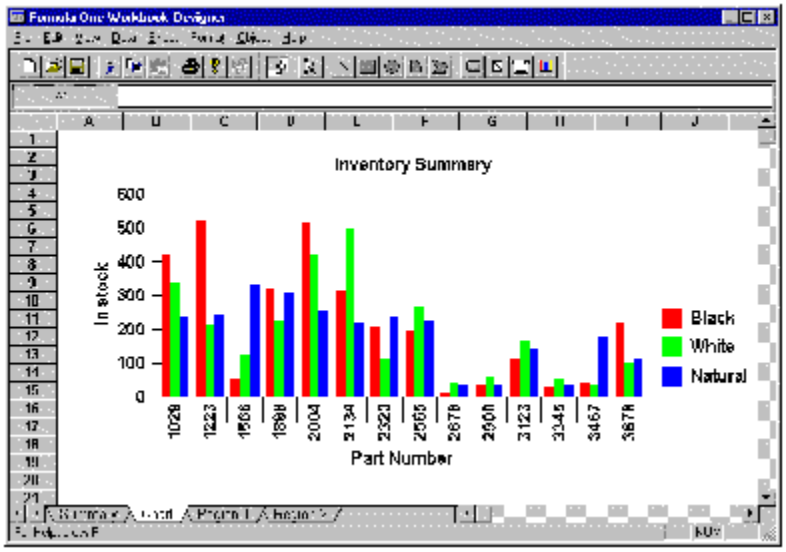
To create a chart on a separate worksheet:

1. Select the chart tool.
2. Draw the chart on a worksheet.
3. Choose Options from the Object menu to display the Object Options dialog box.
4. Enter the formula referencing the data source on the other spreadsheet.



This worksheet provides summary information about the inventory data stored on the worksheets named Region 1 and Region 2.

These cells contain formulas that sum information from the worksheets individual reagins.



The data source formula for this chart object is Summary!A3:D17

Chart Options

Working with Databases

Database connectivity is one of Formula One's most powerful features. You can use Formula One, along with ODBC drivers, to retrieve data from a database and use it to populate a Formula One worksheet at the starting row and column position you specify. This ODBC connection offers incredible speed and flexibility in populating your worksheet.

Installing the ODBC Drivers

In order to connect a Formula One worksheet to a database via ODBC, you must have the 32 bit version of the ODBC drivers installed on your system. These drivers come with most 32 bit development environments such as Visual Basic 4.0, Office95, and Visual C++ 2.x. When you install these environments be sure to select the ODBC option. If you have already installed your development environment, you can re-install and check only the ODBC option. Be aware that different environments offer different drivers. Please contact your development environment vender for information about the drivers available.

Working with Databases

Setting up a Data Source

Overview of Formula One Connections

Setting up a Data Source

You can create a new data source from within Formula One, or you may find it more convenient to set up your data sources outside Formula One.

To set up a data source outside Formula One:

1. Double-click the ODBC icon in the Control Panel to bring up the Data Sources dialog box.
2. Press the Add button to add a connection to the database you want to reach.
3. In the Add Data Source dialog box, choose the appropriate 32 bit driver for the database you are connecting to and press OK. This brings up an ODBC dialog box based on the driver you chose.
4. Enter a Data Source Name and Database Name that describe the database you want to connect to and press OK.

Overview of Formula One Connections

Database connections are made on a per worksheet basis. This means that you can populate each worksheet in a workbook with data from a different query, or even a different database. This is a powerful feature because Formula One worksheet functions work across multiple worksheets.

As an example, you can query district sales information, and place information from each district in a separate worksheet within the same workbook. On each worksheet, you can do summary calculations to show statistics for that district. You can also add additional worksheets to the workbook that do summaries or analysis on any combination of the districts.

If you also purchase Visual Component's First Impression charting tool, you can select a range of data in a worksheet and draw a chart illustrating that data right on the Formula One worksheet.

Installing the ODBC Drivers

Connecting to the Data Source

Querying the Data Source

Disconnecting from the Data Source

Connecting to the Data Source

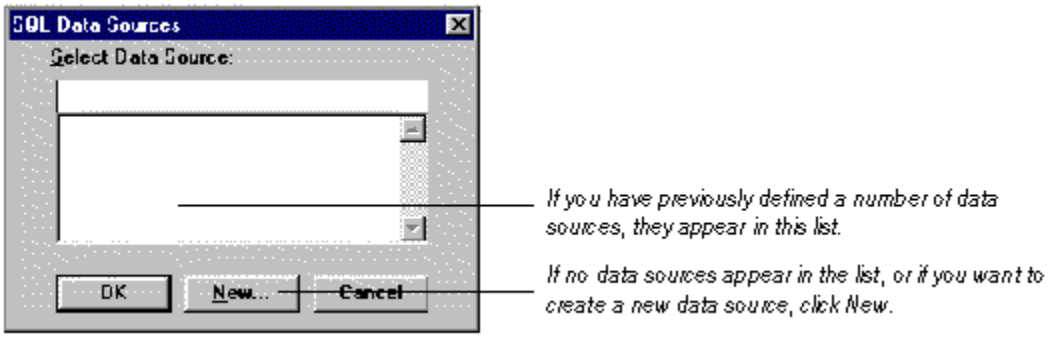
Formula One provides the **ODBCConnect** method to connect the active worksheet to a database. You provide ODBCConnect with:

- a variable containing a connect string,
- a boolean that controls whether SQL errors are displayed,
- and a variable that receives the returned SQL status code.

The following code example shows the use of this method:

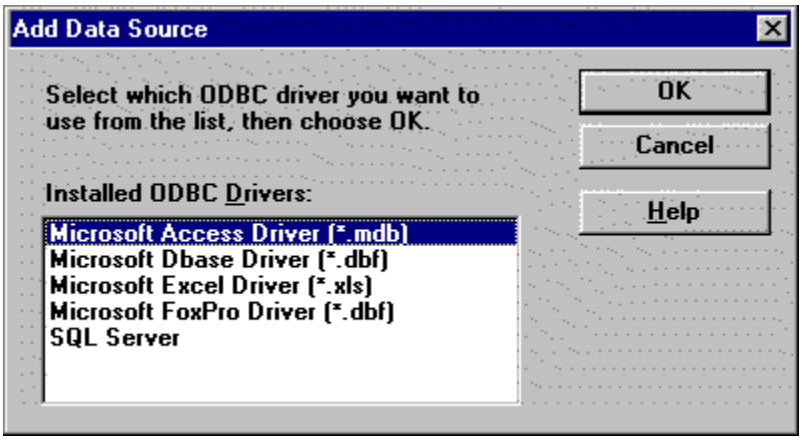
```
On Error GoTo ConnectError
    Dim returnCode As Integer, pConnect As String
    F1Book1.ODBCConnect pConnect, True, returnCode
Exit Sub
ConnectError:
    MsgBox Error
```

The connect string must be a variable. This allows you to let the user select the database at run time and returns the data source in the connect string. When you pass in a null string for the connect string, the following dialog box is displayed, allowing the user to select the data source.



To create a new data source within Formula One:

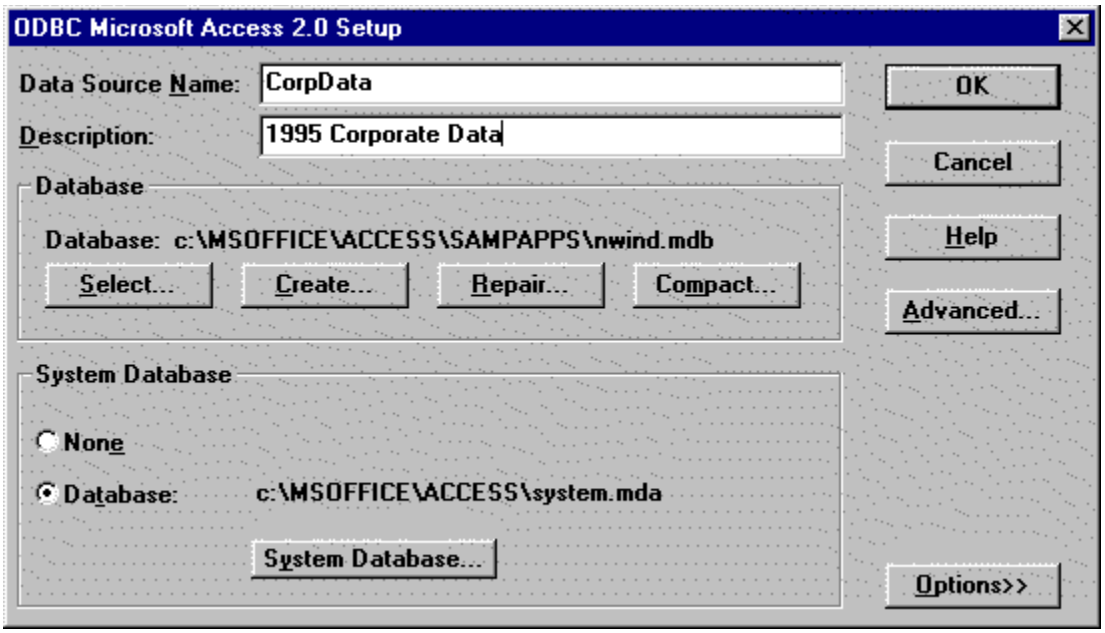
1. Click New in the SQL Data Sources dialog box to display the following dialog box.



2. Select an ODBC driver to use with the data source you are defining and click OK to display the Setup dialog for that driver.

If the appropriate driver does not appear in the list see the instructions under Installing ODBC drivers earlier in this chapter.

3. In the Setup dialog, enter a data source name and optional description. Click Select to identify the database associated with this data source.



4. Click OK to redisplay the SQL Data Sources dialog box.

5. Select the data source to which you want to connect and click OK.

Querying the Data Source

Disconnecting from the Data Source

Querying the Data Source

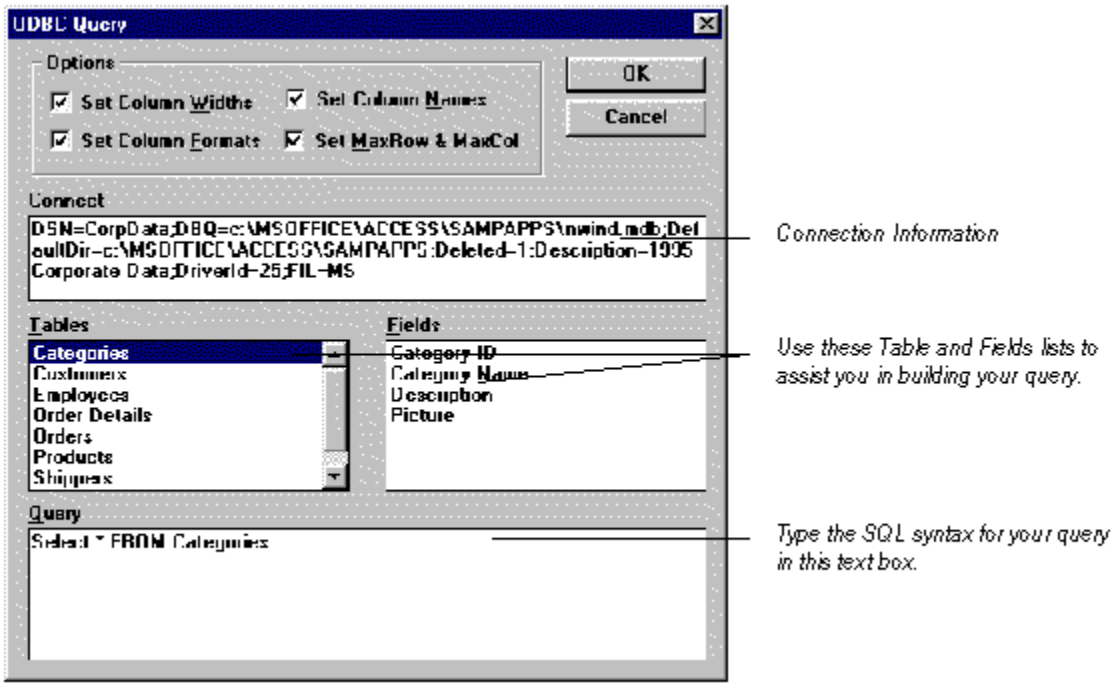
Once you have connected to a data source, you can build and execute a query. Formula One's **ODBCQuery** method accomplishes both. You must provide **ODBCQuery** the following information:

- a variable that specifies the query syntax. If you pass a null string for this argument, the **ODBCQuery** dialog box is displayed to allow the user to build the query at run time. The query string must be passed as a variable. This allows you to let the user build the query at run time and returns the final query string.
- row and column coordinates that identify where in the active worksheet the returned data is to be placed.
- a boolean that determines whether the **ODBCQuery** dialog box is displayed.
- variables that control whether data and data type information is used to format the worksheet size, column headings, column width, and cell formatting.
- a variable that received the returned status code.

Below is the code that implements this in the demo.

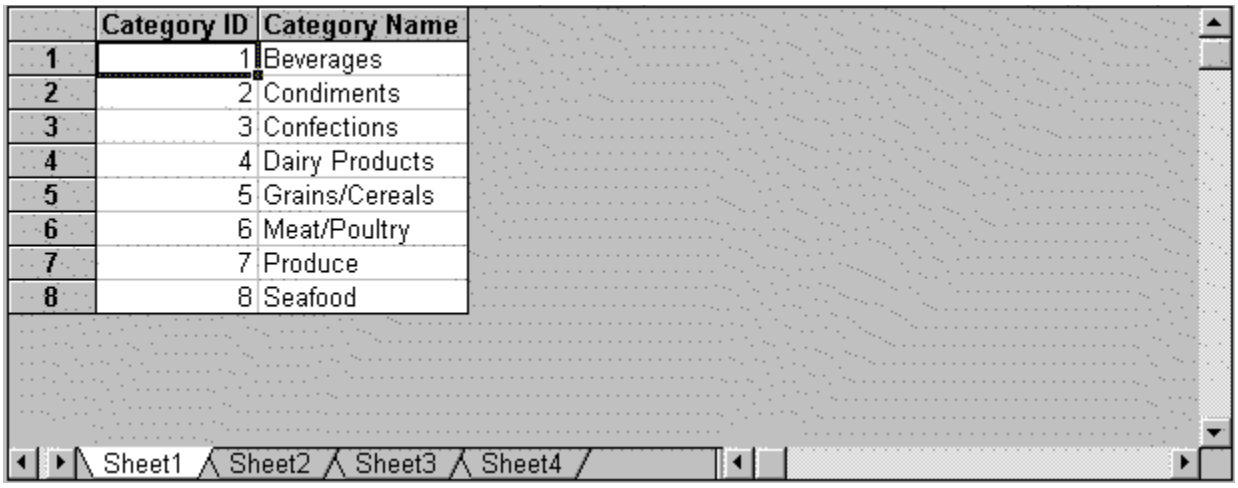
```
On Error GoTo FetchError
Dim returnCode As Integer, query As String
Dim setColNames As Boolean, setColFormats As Boolean
Dim setColWidths As Boolean, setMaxRC As Boolean
Let query = cboQueries.TEXT
setColNames = chkSetColNames.Value
setColFormats = chkSetColFormats.Value
setColWidths = chkSetColWidths.Value
setMaxRC = chkSetMaxRC.Value
F1Book1.ODBCQuery query, Val(txtStartRow.TEXT), Val(txtStartCol.TEXT), optShowDialog.Value,
        setColNames, setColFormats, setColWidths, setMaxRC, returnCode
Exit Sub
FetchError:
    MsgBox Error
```

If the query string is null or you set the optShowDialog value to true, the query dialog is displayed.



- **Set Column Widths**. Check this box to automatically set the width of each column to be wide enough to display the widest data in the column.
- **Set Column Names.** Check this box to display field names instead of the standard alphabetic column headings. Even though field names are displayed as the column headings, formulas must still use the standard cell referencing conventions (e.g., A1).
- **Set Column Formats.** Check this box to have formats for date, time, and currency fields set automatically when data is placed in the worksheet. If you do not check this box, you must set the formats for these columns manually.
- **Set MaxRow & MaxCol.** Check this box to have the maximum number of worksheet rows and columns set to the number of records and fields returned by the query.

When you click OK in the ODBC Query dialog box, the query is executed. The following example shows the results of a returned query.



Notice that Set Column Names and Set MaxRow and Max Column have been set to True. Column labels have been replaced with database field names and the size of the worksheet has been adjusted to the number of returned columns and rows.

Note To maintain Excel compatibility, each cell is limited to 256 characters. If the returned data exceeds

this limit, the text is truncated to fit in the cell
Notice that Set Column Names and Set MaxRow and Max Column have been set to True. Column labels have been replaced with database field names and the size of the worksheet has been adjusted to the number of returned columns and rows.

Connecting to the Data Source

Disconnecting from the Data Source

Disconnecting from the Data Source

After your query you should disconnect from the database. This is done with the **ODBCDisconnect** method. Following is a code example showing the use of this method.

```
F1Book1.ODBCDisconnect
```

Connecting to the Data Source

Querying the Data Source

Printing Worksheets

You can print the active worksheet from the Workbook Designer, or from code.

To print the active sheet from the Workbook Designer:

1. Choose Print from the File menu to display the Print dialog box.
2. Make any necessary adjustments to the settings and click OK.

To print the active sheet from code:

- Use the **FilePrint** method. The following code uses this method to print a worksheet.

```
F1Book1.FilePrint TRUE
```

- When you call **FilePrint**, the Print dialog box can be displayed, allowing you to specify the pages to print, the number of copies to print, and other related items.
- Use **FilePageSetupDlg** to display the Page Setup dialog box which gives easy access to setting margins, headers, footers, headings, grid printing, page ordering, and output alignment. The following code displays the Page Setup dialog box.

```
F1Book1.FilePageSetupDlg
```

- Use **FilePrintSetupDlg** to invoke the Print Setup dialog box, the standard Windows printer setup dialog box is displayed. It allows you to select a printer, select the paper source, and select the page orientation (portrait or landscape). The following code displays the Print Setup dialog box.

```
F1Book1.FilePrintSetupDlg
```

Specifying Print Areas

Specifying Row and Column Print Titles

Specifying Headers and Footers

Specifying Page Breaks

Page Break Methods

Setting Printing Orientation

Using the Windows API DEVMODE Structure

Specifying Print Areas

Formula One prints the entire active worksheet unless you specify the ranges you want to print. To specify the areas you want to print, you must set the Print_Area name to reflect the worksheet area to be printed.

To set the print area in the Workbook Designer:

1. Select the ranges to print.
2. Choose Set Print Area from Sheet.

To set the print area in code:

- Use the **PrintArea** property.
  The following example uses the **PrintArea** property to set A1:D25 as the area to be printed.

```
F1Book1.PrintArea "A1:D25"
```

  You can select multiple ranges to print. If you specify multiple ranges, the ranges do not have to be adjacent. For example, a print area could be comprised of two ranges, A1:D4 and F5:I8.

  In the following example, the print area is set to the ranges A1:D4 and F5:I8.

```
F1Book1.PrintArea "A1:D4,F5:I8"
```

Specifying Row and Column Print Titles

Specifying Headers and Footers

Specifying Page Breaks

Specifying Row and Column Print Titles

You can specify row or column titles that you want printed on each page of your worksheet. If you select a row, it is printed at the top of each page. If you select a column, it is printed at the left edge of each page. You can select multiple rows or columns, but they must be adjacent.

Important When setting print titles, you must select entire rows and columns.

The Print_Titles name holds the row and column titles specification.

To set print titles in the Workbook Designer:

1. Select the rows or columns to use as print titles.
2. Choose Print Titles on the Sheet menu.

To set print titles in code:

- Set the **PrintTitles** property.
  The following example uses the **PrintTitles** property to set rows 1 and 2 and column A as print titles.

```
F1Book1.PrintTitles "A1:IV2, A1:A16384"
```

Specifying Print Areas

Specifying Headers and Footers

Specifying Page Breaks

Specifying Headers and Footers

Headers and footers are printed at the top and bottom of each page. The header and footer definition is accessible in the Page Setup dialog box. You can also define headers and footers through the **PrintFooter** and **PrintHeader** properties.

Headers and footers can contain text and special formatting codes. The following table lists the special formatting codes. Header and footer codes can be entered in upper or lower case.

Format Code	Description
&L	Left-aligns the characters that follow
&C	Centers the characters that follow
&R	Right-aligns the characters that follow
&D	Prints the current date
&T	Prints the current time
&F	Prints the workbook name
&A	Prints the worksheet name
&P	Prints the page number
&P+number	Prints the page number plus number
&P-number	Prints the page number minus number
&&	Prints an ampersand
&N	Prints the total number of pages in the document

Codes and text are, by default, centered unless &L or &R is specified.

The following font codes must appear before other codes and text or they are ignored. The alignment codes (e.g., &L, &C, and &R) restart each section; new font codes can be specified after an alignment code.

Format Code	Description
&B	Use a bold font
&I	Use an italic font
&U	Underline the header
&S	Strikeout the header
&O	Ignored
&H	Ignored
&"fontname"	Use the specified font
&nn	Use the specified font size - must be a two digit number

Specifying Print Areas

Specifying Row and Column Print Titles

Specifying Page Breaks

Specifying Page Breaks

Both horizontal and vertical page breaks can be specified on a worksheet. Page breaks can be specified interactively using the Workbook Designer, or you can use methods and properties.

In the Workbook Designer, page breaks are always placed adjacent to the active cell. When using methods, page breaks can be placed adjacent to the active cell or a cell that you specify.

- Horizontal (row) page breaks are placed adjacent to the top edge of the active or specified cell.
- Vertical (column) page breaks are placed adjacent to the left edge of the active or specified cell.

To set page breaks in the Workbook Designer:

1. Select the cell adjacent to which you want to place page breaks.
2. Choose Set Page Breaks from the Sheet menu.

Page Break Methods

Page Break Methods

There are several categories of page break methods. The **AddPageBreak** and **RemovePageBreak** methods add page breaks adjacent to the active cell. The following example uses these methods:

```
F1Book1.AddPageBreak

F1Book1.RemovePageBreak
```

AddRowPageBreak, **AddColPageBreak**, **RemoveRowPageBreak**, and **RemoveColPageBreak** add and remove page breaks adjacent to the row or column that you specify in the method. The following example uses the **AddRowPageBreak** method:

```
Dim theRow As Long

theRow = 8

F1Book1.AddRowPageBreak theRow
```

NextRowPageBreak returns the next page break below the row that you specify in the method. **NextColPageBreak** returns the next page break to the right of the column that you specify in the method. The following example uses the **NextRowPageBreak** method:

```
Dim nextBreak As Long

Dim theRow As Long

theRow = 20

nextBreak = F1Book1.NextRowPageBreak (therow )
```

Specifying Page Breaks

Setting Printing Orientation

Worksheet pages can be printed with a portrait or landscape orientation. The **PrintLandscape** property controls the printing orientation.

To set printing orientation in the Worksheet Designer:

- Choose Print Setup from the File menu and set the printing orientation in the Print Setup dialog box.

Using the Windows API DEVMODE Structure

You can also customize your printing session by using the **PrintDevMode** property to read and write the Windows API DEVMODE structure. This structure sets a number of common printing attributes such as device name, orientation, paper size, print scale, and print resolution.

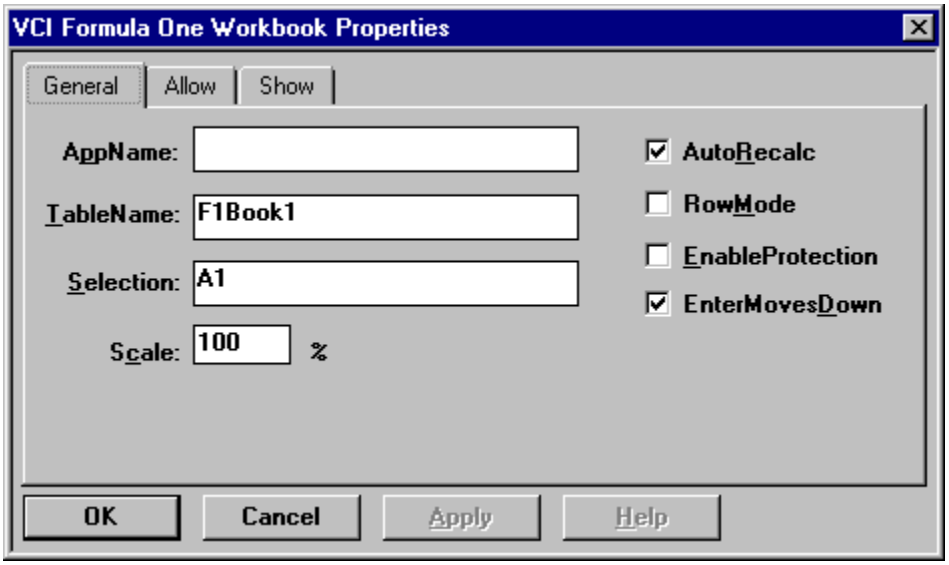
Formula One Workbook Properties

A number of the most commonly used properties can be set in the Formula One Workbook Properties dialog box.

To display the Formula One Workbook Properties dialog box:

1. Right-click on the workbook control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a control in the following illustration for more information about that particular control.



Enter an application name to display in the title bar of all Formula One error dialog boxes.

AppName property

Enter a name to change the default name assigned to the workbook associated with the current view. A number of methods, such as Attach, reference a workbook by its name instead of its handle.

TableName property

This control displays a set of cell references identifying the current selection on the current worksheet. Enter a cell reference here to select a new range and move the active cell to the top left cell in the range.

Selection Property

This control shows the current display scale for the workbook. Enter a number between 10 and 400 to change the display scale.

ViewScale Property

Check this box to immediately recalculate the worksheet if recalculation is necessary. Thereafter, any change to the workbook causes all formulas to be recalculated. Uncheck this box to recalculate the worksheet only if you specifically request a recalculation.

AutoRecalc Property

Check this box to select an entire row when you click on a cell. Uncheck this box to select only a cell when you click on a cell.

RowMode Property

Check this box to enable protection for all selected sheets. Uncheck this box to disable protection. Enabling protection means that any cells marked as hidden or locked, are actually hidden and locked.

EnableProtection Property

Check this box to cause pressing the enter key to move the active cell down to the next cell. Uncheck this box to prevent the enter key from moving to the next cell.

EnterMovesDown Property

Formula One Workbook Properties

A number of the most commonly used properties can be set in the Formula One Workbook Properties dialog box.

To display the Formula One Workbook Properties dialog box:

1. Right-click on the workbook control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a control in the following illustration for more information about that particular control.

The controls in this section specify whether you can use certain mouse actions to perform certain tasks. Check the Resize Rows/Cols box to allow rows and columns to be resized by dragging row or column heading borders. Check the Fill Range box to fill a worksheet range by dragging a selection's copy handle. Check the Move Range box to move ranges by dragging a cell.

AllowResize Property

AllowFillRange Property

AllowMoveRange Property

The controls in this section specify whether you can perform certain editing activities. Check the Formulas box to allow formulas to be entered. Check In Cell Editing to allow data to be entered directly into a cell, bypassing the edit bar. Check the Selections box to allow ranges and objects to be selected with the keyboard or by clicking and dragging the mouse.

AllowFormulas Property

AllowInCellEditing Property

AllowSelections Property

The controls in this section specify the behavior of certain keyboard actions. Check Use Arrows to allow use of the arrow keys to reposition the active cell. Check Use Tabs to allow use the tab key to reposition the active cell in a selected range. Check Use Delete to allow use of the delete key deletes records and clears selections. The delete key deletes a record if an entire row is selected. The current selection is cleared if less than a row is selected or if data browsing mode is disabled.

AllowTabs Property

AllowArrows Property

AllowDelete Property

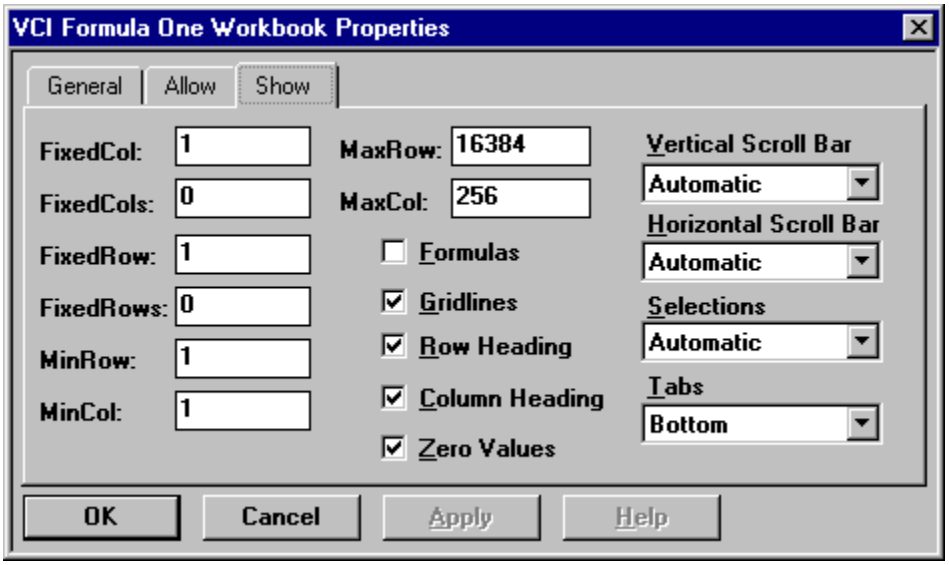
Formula One Workbook Properties

A number of the most commonly used properties can be set in the Formula One Workbook Properties dialog box.

To display the Formula One Workbook Properties dialog box:

1. Right-click on the workbook control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a control in the following illustration for more information about that particular control.



Enter the column number of the first column you want to fix in the active worksheet. Fixed columns do not scroll when the worksheet is scrolled horizontally. Data in fixed columns cannot be edited.

FixedCol Property

Enter the number of columns (from FixedCol) you want to fix. Fixed columns do not scroll when the worksheet is scrolled horizontally. Data in fixed columns cannot be edited.

FixedCols Property

Enter the row number of the first row you want to fix in the active worksheet. Fixed rows do not scroll when the worksheet is scrolled vertically. Data in fixed rows cannot be edited.

FixedRow Property

Enter the number of rows (from FixedRow) you want to fix. Fixed rows do not scroll when the worksheet is scrolled vertically. Data in fixed rows cannot be edited.

FixedRows Property

Enter the number of the first row that can be displayed in the active worksheet. Rows before the first row are not displayed but can be used to hold data and formulas.

MinRow Property

Enter the number of the first column that can be displayed in the active worksheet. Columns before the first row are not displayed but can be used to hold data and formulas.

MinCol

Enter the number of the last displayable row in the active worksheet. Rows beyond the last row are not displayed but can be used to hold data and formulas.

MaxRow Property

Enter the number of the last displayable column in the active worksheet. Columns beyond the last column are not displayed but can be used to hold data and formulas.

MaxCol Property

Check this box to display formula text in cells instead of the values formulas produce. Uncheck the box to display formula results.

ShowFormulas Property

Check this box to display grid lines on the active worksheet. Uncheck this box to hide the gridlines.

ShowGridLines Property

Check this box to display row headings on the active worksheet. Uncheck this box to hide the row headings.

ShowRowHeading Property

Check this box to display column headings on the active worksheet. Uncheck this box to hide the column headings.

ShowColHeading Property

Check this box to display zeros in cells with zero values. Uncheck this box to display zero value cells as blanks.

ShowZerosValues Property

Controls the appearance of the vertical scroll bar. Select On to turn the scroll bar on, Off to turn scroll bar off, and Automatic to display scroll bar when the workbook is too tall to display in the control.

ShowVScrollBar

Controls the appearance of the horizontal scroll bar. Select On to turn the scroll bar on, Off to turn scroll bar off, and Automatic to display scroll bar when the workbook is too tall to display in the control.

ShowHScrolBar

Controls how selections are displayed on the active worksheet. Select On to display selections, Off to hide selections, or Automatic to display selections only when the control has focus.

ShowSelections Property

Controls the appearance and position of sheet name tabs. Select Top to display the tabs at the top of the workbook, Bottom to display the tabs at the bottom of the workbook, or Off to hide the sheet tabs.

ShowTabs Property

AboutBox Method

Description

Displays the About Formula One dialog box. This dialog box contains information about your version of Formula One, including your serial number. You must have your serial number to receive technical support or upgrade pricing on future product releases.

Syntax

*F1Book1.***AboutBox**

AddColPageBreak Method

Description

Adds a vertical page break adjacent to the left edge of the specified column.

Syntax

*F1Book1.***AddColPageBreak** *nCol*

Part	Type	Description
nCol	Long	Indicates the column where the page break is added.

See Also

AddPageBreak method

AddRowPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemoveColPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

Example

The following example adds a vertical page break at the left edge of the column containing the active cell (identified by the **Col** property):

```
F1Book1.AddColPageBreak F1Book1.Col
```

AddPageBreak Method

Description

Adds a horizontal and vertical page break adjacent to the active cell.

Syntax

*F1Book1.***AddPageBreak**

Remarks

When page breaks are added adjacent to the active cell, the horizontal page break is added at the cell's top edge; the vertical page break is added at the cell's left edge.

See Also

AddColPageBreak method

AddRowPageBreak method

RemoveColPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

Example

The following example puts a page break at the top left corner of a range of cells with the defined name of SalesReport2:

```
F1Book1.Selection = "SalesReport2"
F1Book1.AddPageBreak
```

AddRowPageBreak Method

Description

Adds a horizontal page break adjacent to the top edge of the specified row.

Syntax

*F1Book1.***AddRowPageBreak** *nRow*

Part	Type	Description
nRow	Long	Indicates the row where the page break is added.

See Also

AddPageBreak method

AddColPageBreak method

RemoveColPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

Example

The following example implements a menu item that adds row page breaks at the active cell:

```
Private Sub mnuFileAddRowPageBreak
    F1Book1.AddRowPageBreak F1Book1.Row
End Sub
```

AddSelection Method

Description

Adds a new selection to the current selection list.

Syntax

F1Book1.**AddSelection** *nRow1*, *nCol1*, *nRow2*, *nCol2*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	*Long*	Indicate the row and column coordinates of the selection to add to the selection list. To include all rows in the selection, set *nRow1* to -1. To include all columns in the selection, set *nCol1* to -1. For example, to add all of row 4 to the selection list, use 4, -1, 4, -1.

Remarks

Multiple selections allow operations such as formatting or clearing to be performed on non-contiguous areas.

See Also

Selection property

Example

The following example sets a medium black border on the bottom of every 5th row for the first 100 rows:

```
Dim i As Integer, savedRange As String
savedRange = F1Book1.Selection
F1Book1.SetSelection 5, -1, 5, -1
For i = 10 To 100 Step 5
    F1Book1.AddSelection i, -1, i, -1
Next i
F1Book1.SetBorder 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0
F1Book1.Selection = savedRange
```

AllowArrows Property

Description

Sets or returns whether the arrow keys can reposition the active cell.

Syntax

*F1Book1.***AllowArrows** [= *boolean*]

If this property is True, the arrow keys on your keyboard move the active cell in the spreadsheet. By default, this property is True.

AllowAutoFill Property

Description

Sets or returns whether the AutoFill feature is enabled.

Syntax

F1Book1.**AllowAutoFill** [= *boolean*]

If this property is True, you can use the mouse or the keyboard to autofill a range.

See Also

AutoFillItems property

AutoFillItemsCount property

DeletAutoFillItems method

AllowDelete Property

Description

Sets or returns whether the delete key clears selections.

Syntax

*F1Book1.***AllowDelete** [= *boolean*]

If this property is True, the delete key deletes the current selection. If this property is False, the delete key does not clear selections. By default, the allow delete flag is True.

AllowDesigner Property

Description

Sets or returns whether the Workbook Designer can be launched at run-time.

Syntax

F1Book1.**AllowDesigner** [= *boolean*]

Remarks

When this property is True, the default state, you can display the Workbook Designer by double-clicking the right mouse on the workbook control. When it is False, the Workbook Designer is only available at design time.

If you make the Workbook Designer available at run-time, it will also be available to your end users.

See Also

LaunchDesigner method

AllowEditHeaders Property

Description

Sets or returns whether row, column, and top left headers can be edited.

Syntax

*F1Book1.***AllowEditHeaders** [= *boolean*]

If this property is True, you can edit the names displayed in row, column, and top left headers by double-clicking on a header. This displays the Header Name dialog box, allowing you to enter a new header name. If this property is False, editing of headers is not allowed and a **DblClick** event is passed when a header is double clicked, if **DoDblClick** is True.

AllowFillRange Property

Description

Sets or returns whether you can fill a worksheet range by dragging.

Syntax

*F1Book1.***AllowFillRange** [= *boolean*]

If this property is True, you can drag a selection's copy handle to fill a range.

AllowFormulas Property

Description

Sets or returns whether the user is allowed to enter or edit formulas.

Syntax

*F1Book1.***AllowFormulas** [= *boolean*]

If this property is False, formulas cannot be added or edited by the user at run time.

AllowInCellEditing Property

Description

Sets or returns whether in-cell editing is allowed.

Syntax

*F1Book1.***AllowInCellEditing** [= *boolean*]

If this property is False, in-cell editing is inactive and you cannot enter or edit data in a cell without using the edit bar.

AllowMoveRange Property

Description

Sets or returns whether you can move ranges by dragging them.

Syntax

*F1Book1.***AllowMoveRange** [= *boolean*]

If this property is True, you can move ranges by dragging a selection.

AllowObjSelections Property

Description

Sets or returns whether you can select objects at runtime.

Syntax

*F1Book1.***AllowObjSelections** [= *boolean*]

If this property is True, objects can be selected. If an object is already selected when you set this property to False, it remains selected.

AllowResize Property

Description

Sets or returns whether you can resize rows and columns by dragging them.

Syntax

*F1Book1.***AllowResize** [= *boolean*]

If this property is True, rows and columns can be sized by dragging row or column heading borders.

AllowSelections Property

Description

Sets or returns whether you can select ranges and objects.

Syntax

*F1Book1.***AllowSelections** [= *boolean*]

If this property is True, you can select ranges and objects with the keyboard or by clicking and dragging with the mouse.

AllowTabs Property

Description

Sets or returns whether you can use the tab key to reposition the active cell in a selected range.

Syntax

*F1Book1.***AllowTabs** [= *boolean*]

If this property is True, you can use the tab key to move the active cell through a selected range. When tabbing through a range, the active cell moves from left to right through each row in the range. By default, the allow tabs flag is True.

AppName Property

Description

Sets or returns the application name that is displayed in the title bar of Visual Components error dialog boxes.

Syntax

*F1Book1.***AppName** [= *name*]

Part	Type	Description
name	String	Application name.

Remarks

When the application name is set, only the name displayed in the title bar of error dialog boxes is affected. Other dialog boxes display functional names (e.g., Alignment, Custom Format, Font).

Attach Method

Description

Searches for a workbook with the given title and attaches it to a view.

Syntax

*F1Book1.***Attach** *pTitle*

Part	Type	Description
pTitle	String	Indicates the title of the workbook for which to search.

Remarks

This method searches for a workbook with the given title. If the workbook is found, it is attached to the specified view. After a successful attachment, both the workbook's original view and the new view display the same workbook.

This method is similar to **AttachToSS**, except that the workbook is referenced by title, rather than by handle.

Use the **Title** property to return the title associated with a particular workbook.

AttachToSS Method

Description

Searches for a workbook by handle, and attaches it to a view.

Syntax

*F1Book1.***AttachToSS** *hSrcSS*

Part	Type	Description
hSrcSS	Long	Handle to the workbook.

Remarks

After a successful attachment, both the workbook's original view and the new view display the same workbook. This method is similar to **Attach**, except that the workbook is referenced by handle, rather than by title.

Use the **SS Method** method to return the handle associated with a particular workbook.

See Also

Attach method

SwapTables method

Example

The following example shows how to present different worksheets in response to user selection. F1BookMain is the only visible Formula One control. Each time the user clicks the button, he is shown a different view of the same worksheet. He can view regional, district, or national sales information at the click of a button:

```
Sub SwapViews (toWhat As Integer)
    Select Case toWhat
        Case kRegionSales
            F1BookMain.AttachToSS F1BookHidden1.SS
        Case kDistrictSales
            F1BookMain.AttachToSS F1BookHidden2.SS
        Case kNationalSales
            F1BookMain.AttachToSS F1BookHidden3.SS
        Case Else
            MsgBox "debug: invalid toWhat in SwapViews # " & toWhat
    End Select
End Sub
```

AutoFillItems Property

Description

Sets or returns an autofill list.

Syntax

*F1Book1.***AutoFillItems** (*nIndex*) [= *list*]

Part	Type	Description
nIndex	Integer	Identifies the list's position within all defined autofill lists. The first autofill list has an index of one.
list	String	The semi-colon delimited list of autofill items.

Remarks

Autofill lists are frequently used series of text such as months and days of the week. When you enter a text value from an autofill list and move one cell down or to the right, the next text value in the list is proposed for that cell. You can also use the fill handle to autofill a single row or column.

Lists are case sensitive. To delete any existing autofill list, use **DeleteAutoFillItems** method.

Following are the default lists provided with Formula One:

Index	List
1	"Sun; Mon; Tue; Wed; Thu; Fri; Sat"
2	"Sunday; Monday; Tuesday; Wednesday; Thursday; Friday"
3	"Jan; Feb; Mar; Apr; May; Jun; Jul; Aug; Sep; Oct; Nov; Dec"
4	"January; February; March; April; May; June; July; August; September; October; November; December"

See Also

AutoFillItemsCount property

Example

The following example creates a new autofill list.

```
F1Book1.AutoFillItems (F1Book1.AutoFillItemsCount+1) = "North Region; South Region; East
        Region; West Region"
```

AutoFillItemsCount Property

Description

Returns the number of existing autofill lists.

Syntax

[*count* =] *F1Book1.***AutoFillItemsCount**

Part	Type	Description
count	Integer	A variable that receives the number of lists.

Remarks

Autofill lists are frequently used series of text such as months and days of the week. When you enter a value from an autofill list and move one cell down or to the right, the next value in the list is proposed for that cell. You can also use the fill handle to autofill a single row or column.

See Also

AutoFillItems property

DeleteAutoFillItems method

Example

The following example uses creates a new autofill list.

```
F1Book1.AutoFillItems (F1Book1.AutoFillItemsCount+1) = "First Quarter;Second Quarter;Third
        Quarter;Fourth Quarter"
```

AutoRecalc Property

Description

Sets or returns whether automatic recalculation is enabled

Syntax

F1Book1.**AutoRecalc** [= *boolean*]

Remarks

If this property is True, the workbook is recalculated if needed. Thereafter, any change to the workbook causes all formulas to be recalculated.

You may notice that the workbook is not recalculated immediately after each change you make from your program. This is a speed optimization. To force the workbook to be recalculated immediately, call **Recalc**.

Setting **Text**, **TextRC**, **Number**, **NumberRC**, **FormattedText**, **FormattedTextRC**, **TypeRC** and **LogicalRC** also causes the workbook to be recalculated immediately, if needed.

See Also

SSUpdate method

BackColor Property

Description

Sets or returns the background color of the view.

Syntax

F1Book1.**BackColor** [= *color*]

Part	Type	Description
color	OLE_COLOR	This value can be one of the normal RGB Colors. These colors are specified using the color palette, or by using the RGB or QBColor functions. The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

Important Formula One does not support the system default colors.

Remarks

All cells within the view are set to the background color except those with patterns.

See Also

ExtraColor property

BorderStyle Property

Description

Sets or returns the border style for the Formula One window.

Syntax

F1Book1.**BorderStyle** [= *setting*]

Part	Type	Description
setting	Integer	Describes the type of border on the window.

Remarks

The **BorderStyle** property settings are:

Setting	Description
0	None
1	Fixed Single

CalculationDlg Method

Description

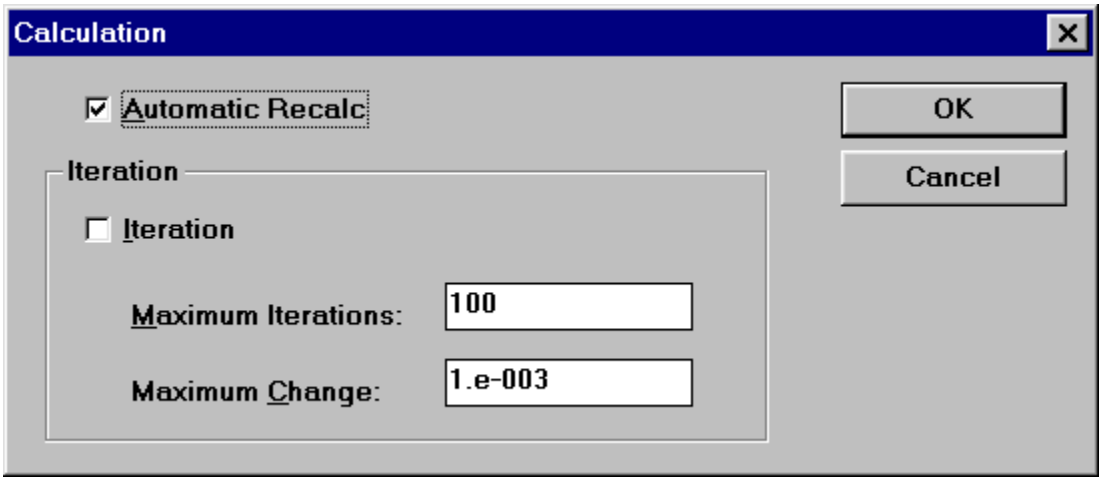
Displays the Calculation dialog box.

Syntax

F1Book1.**CalculationDlg**

Remarks

The Calculation dialog box allows you to enable and disable automatic recalculation and specify iteration values for calculating circular references.



See Also

AutoRecalc property

GetIteration method

CancelEdit Method

Description

Cancels edit mode and leaves the contents of the active cell unchanged.

Syntax

*F1Book1.***CancelEdit**

Remarks

CancelEdit aborts cell editing and exits edit mode without altering the contents of the active cell.

See Also

EndEdit method

Example

The following example uses CancelEdit in an **EndEdit** event to prevent the user from entering more than 15 characters in a cell:

```
Private Sub F1Book1_EndEdit (EditString As String, Cancel As Integer)
    Dim result As Integer
    If Len(EditString) >15 then
        result = MsgBox("Entry exceeds maximum length. Abort entry?", vbYesNo)
    EndIf
    Select Case result
        Case vbYes
            F1Book1.CancelEdit
        Case vbNo
            Cancel = True
    End Select
End Sub
```

CheckRecalc Method

Description

Recalculates the workbook if needed.

Syntax

F1Book1.**CheckRecalc**

Remarks

CheckRecalc determines if the workbook needs to be recalculated as a result of a change. If so, the workbook is recalculated. A workbook is usually recalculated when the result of a formula cell is required by some operation, a worksheet is printed or saved, or when the system becomes idle.

See Also

AutoRecalc property

ClearClipboard Method

Description

Clears the internal clipboard.

Syntax

F1Book1.**ClearClipboard**

Remarks

ClearClipboard clears the contents of the internal clipboard and releases all resources associated with it. This does not affect the contents of the Windows clipboard.

See Also

EditPaste method

ClearRange Method

Description

Clears the specified range in all selected sheets.

Syntax

F1Book1.**ClearRange** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *nClearType*

Part	Type	Description
nRow1, nCol1, nRow2, nCol2	Long	Coordinates that specify the range to clear. If nRow1 is -1, all rows are included in the range; if nCol1 is -1, all columns are included.
nClearType	Integer	F1ClearTypeConstants that determine what is cleared from the specified range:

Constants	Description
F1ClearDlg	Displays the Clear dialog box.
F1ClearAll	All (values and formats)
F1ClearFormats	Formats only
F1ClearValues	Values only (including formulas)

See Also

DeleteRange method

Clip Property

Description

Imports and exports tab-delimited text strings to and from workbooks. This is a run time only property.

Syntax

*F1Book1.***Clip** [= text]

Part	Type	Description
text	String	Identifies a tab-delimited text block.

Remarks

The **Clip** property can set or obtain a tab-delimited text block in a workbook.

- When placing a block of text in a worksheet, text placement begins with the active cell. Tab characters (ANSI character 9) in the text block move text placement to the next column; carriage returns (ANSI character 13), line feeds (ANSI character 10), and carriage return/line feed pairs move text placement to the following row.
- When obtaining a block of text from a worksheet, text is obtained from the currently selected range. If multiple ranges are selected, only text from the first range is used.

See Also

ClipValues property

Formula property

Number property

Text property

GetTabbedText method

SetTabbedText method

Example

The following example uses Formula One to sort text in a text box control by paragraphs:

```
F1Book1.Clip = RichTextBox1.Text
F1Book1.Sort -1,-1,-1,-1, True, 1
RichTextBox1.Text = F1Book1.Clip
```

ClipValues Property

Description

Imports and exports tab-delimited text strings to and from workbooks, ignoring any formatting applied to the text. This is a run-time only property.

Syntax

F1Book1.**ClipValues** [= *text*]

Part	Type	Description
text	String	Identifies a tab-delimited text block.

Remarks

Like the **Clip** property, **ClipValues** can set or obtain a tab-delimited text block in a workbook. However, **ClipValues** ignores formatting applied to the text block.

- When placing a block of text in a worksheet, text placement begins with the active cell. Tab characters (ANSI character 9) in the *text* block move text placement to the next column; carriage returns (ANSI character 13), line feeds (ANSI character 10), and carriage return/line feed pairs move text placement to the following row.
- When obtaining a block of text from a worksheet, text is obtained from the currently selected range. If multiple ranges are selected, only text from the first range is used.

See Also

Formula property

Number property

Text property

GetTabbedText method

SetTabbedText method

Col Property

Description

Determines the active column in the active worksheet. This is a run-time only property.

Syntax

F1Book1.**Col** [= *column*]

Part	Type	Description
column	Long	Identifies a *column* number. -1 indicates all columns.

Remarks

The **Col** property is used with the **Row** property to set the active cell in the worksheet. The **Col** property is automatically changed if a range is selected using the **SelStartCol** property, **SelStartRow** property, **SelEndCol** property and **SelEndRow** property properties.

You can specify -1 as the row and *column* number to indicate all rows or all columns. For example, setting **Row** to 1 and **Col** to -1 causes all columns in row 1 to be selected. Setting both **Row** and **Col** to -1 selects the entire worksheet.

See Also

GetActiveCell method

SetActiveCell method

ColHidden Property

Description

Sets or returns the display status of an individual column.

Syntax

F1Book1.**ColHidden** (*nCol*) [= *boolean*]

Part	Type	Description
nCol	Long	Identifies a column by number.
boolean	*Boolean*	If this property is True, the column is hidden. If it is False, the column is displayed.

Remarks

Use **SetColHidden** to change the display status of one or more columns.

See Also

RowHidden property

Example

The following example assumes a Formula One button named HideSales:

```
Sub F1Book1_ObjClick (ObjName As String, ByVal ObjID As Long)
    Dim i as Integer
    With F1Book1
        .ObjText = IIf(.ColHidden (kSalesStartCol), "Hide Sales", "Show Sales")
        For i = kSalesStartCol to kSalesEndCol
            .ColHidden (i) = Not , ColHidden (i)
        Next i
    End With
End Sub
```

ColorPaletteDlg Method

Description

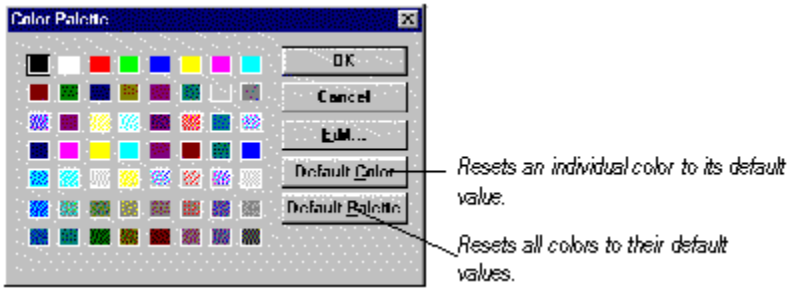
Displays the Color Palette dialog box.

Syntax

*F1Book1.***ColorPaletteDlg**

Remarks

The Color Palette dialog box allows you to edit colors in the color palette, specify a default color, and use the default color palette. Color palettes are workbook specific.



See Also

BackColor property

GetBorder method

ExtraColor property

GetFont method

GetPattern method

SetBorder method

SetFont method

SetPattern method

ColText Property

Description

Sets or returns the label for a column. Setting this property affects all selected sheets.

Syntax

F1Book1.**ColText** (*nCol*) [= *colText*]

Part	Type	Description
nCol	Long	Identifies a column by number.
colText	String	The column label text.

Remarks

Naming a column is useful for labeling columns so they reflect the data in the column (e.g., column G might be named Total Sales). The column name is displayed in the column heading and is used for display purposes only. The column is still referred to by letter reference in formulas.

The column name can be up to 9 lines and 254 characters. A CR (carriage return and LF (line feed) combination are counted as two characters.

Set **ColText** to change a column label and get the value of **ColText** to return a string that contains the current column label.

See Also

RowText property

TopLeftText property

Example

The following example restores the original column heading text after modifications have been made:

```
Dim i As Integer
For i = 1 to 26
    F1Book1.ColText (i) = Chr(i+64)
Next i
    For i = 27 to 256
F1Book1.ColText ( i ) = Left$( F1Book1.FormatRCNr ( 1, i, False), 2)
Next i
```

ColWidth Property

Description

Sets or returns the width of a single column. Setting this property affects all selected sheets.

Syntax

*F1Book1.***ColWidth** (*nCol*) [= *width*]

Part	Type	Description
nCol	Long	Identifies a column by number.
width	Integer	Indicates the column width value.

Remarks

Column width can be specified in units equal to 1/256th of an average character's width in the default font, or twips, depending on the setting of the **ColWidthUnits** property.

Use the **SetColWidth** method to change the width of multiple columns. Use **ColWidthTwips** to set or return the width of a single column in twips.

See Also

ColWidthDlg method

RowHeight property

SetColWidthAuto method

ColWidthDlg Method

Description

Displays the Column Width dialog box.

Syntax

F1Book1.**ColWidthDlg**

Remarks

The Column Width dialog box allows you to set the width of the selected columns, specify default column widths, and automatically set column widths to the display the widest column text. In addition, you can specify whether the selected columns are shown or hidden. Settings made in this dialog box affect all selected sheets.

See Also

ColWidth property

SetColWidth method

SetColWidthAuto method

ColWidthTwips Property

Description

Sets or returns the width of a specified column in twips.

Syntax

*F1Book1.***ColWidthTwips** (*nCol*) [= *width*]

Part	Type	Description
nCol	Long	Identifies a column by number for which the current width is returned.
width	Integer	Indicates the column width value in twips.

See Also

ColWidth property

ColWidthDlg method

RowHeight property

SetColWidthAuto method

Example

The following example creates a resizable form that is filled with a one column workbook. It also assumes that **ColWidthUnits** is set to twips:

```
F1Book1.Width = Form1.ScaleWidth
F1Book1.Height = Form1.ScaleHeight
F1Book1.ColWidthTwips (1) = F1Book1.Width - F1Book1.HdrWidth
```

ColWidthUnits Property

Description

Sets or returns whether column widths are stored and displayed in twips or character units.

Syntax

*F1Book1.***ColWidthUnits** [= *nColWidthUnits*]

Part	Type	Description
nColWidthUnits	Integer	F1ColWidthUnitsConstants that determine the type of units used for column widths:

Constant	Description
F1ColWidthUnitsCharacters	All column widths are converted to units equal to 1/256th of an average characters width in the default font. This is the default column width unit.
F1ColWidthUnitsTwips	All widths in the workbook are converted to twips. This can allow more consistent sizing between printer output and screen display.

Remarks

When you change the column width unit you are changing how widths are stored. Use care when switching back and forth. If you switch from character units to twips and back again, the character unit values may be slightly different due to mathematical rounding used in the conversion process.

See Also

ColWidth property

ColWidthDlg method

SetColWidth method

SetColWidthAuto method

CopyAll Method

Description

Copies the contents of one workbook to another workbook.

Syntax

F1Book1.**CopyAll** *hSrcSS*

Part	Type	Description
hSrcSS	Long	A handle to the source view.

Remarks

CopyAll copies an entire worksheet from the source view control to the current view control.

See Also

CopyRange method

MoveRange method

SS method

Example

The following example copies the contents of workbook F1Book1 to the workbook F1Book2:

```
F1Book1.CopyAll F1Book2.SS
```

CopyRange Method

Description

Copies a range of data to the specified location. This method can be used to copy ranges within a worksheet, or from the active worksheet in the specified workbook to the active worksheet in the current workbook.

Syntax

F1Book1.**CopyRange** *nDstR1*, *nDstC1*, *nDstR2*, *nDstC2*, *hSrcSS*, *nSrcR1*, *nSrcC1*, *nSrcR2*, *nSrcC2*

Part	Type	Description
nDstR1, *nDstC1*, *nDstR2*, *nDstC2*	Long	Coordinates that define the range to hold the copied data in the current view.
hSrcSS	Long	Handle to the source view.
nSrcR1, *nSrcC1*, *nSrcR2*, *nSrcC2*	Long	Coordinates defining the range holding the data to be copied in the specified view.

Remarks

The source and the destination ranges can be in different workbooks, allowing ranges to be copied between workbooks. The copy operation is the same as if a copy and paste operation had occurred. Cell references are adjusted appropriately in the destination range.

See Also

CopyAll method

EditCopy method

EditPaste method

CopyRangeEx Method

Description

Copies a range of data from one worksheet to another.

Syntax

F1Book1.**CopyRangeEx** *nDstSheet*, *nDstR1*, *nDstC1*, *nDstR2*, *nDstC2*, *hSrcSS*, *nSrcSheet*, *nSrcR1*, *nSrcC1*, *nSrcR2*, *nSrcC2*

Part	Type	Description
nDstSheet	Long	Index that identifies a specific worksheet in the current workbook. Worksheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.
nDstR1, nDstC1, nDstR2, nDstC2	Long	Coordinates that define the range to hold the copied data in the current workbook.
hSrcSS	Long	Handle to the source workbook.
nSrcSheet	Long	Index that identifies a specific worksheet in the source workbook. Worksheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.
nSrcR1, nSrcC1, nSrcR2, nSrcC2	Long	Coordinates defining the range holding the data to be copied in the specified workbook.

Remarks

The source and the destination worksheets can be in the same workbook, or a different workbook. The copy operation is the same as if a copy and paste operation had occurred. Cell references are adjusted appropriately in the destination range.

DefinedName Property

Description

Sets or returns the formula associated with a defined name.

Syntax

F1Book1.**DefinedName** (*name*) [= *formula*]

Part	Type	Description
name	String	A defined name. Enter an existing name if you are returning the formula associated with the name or changing the value associated with the name. Enter a unique name if you are creating a new value.
formula	String	Describes the range represented by name. A name can refer to a cell, a group of cells, a value, or a formula. When setting a name, do not include a leading equal sign (=) in the formula.

Remarks

DefinedName returns a string describing the formula for the specified defined name. For example, if the range $B$10:$F$10 is named TotalSales, a string is returned containing the range reference $B$10:$F$10. Relative references in defined name formulas are relative to the active cell. Changing the active cell will offset relative references in defined names.

Set **DefinedName** to define or change a user-defined name.

See Also

DefinedNameDlg method

DeleteDefinedName method

DefinedNameByIndex Property

Description

Changes or returns a defined name.

Syntax

F1Book1.**DefinedNameByIndex** (*index*) [= *name*]

Part	Type	Description
index	Long	Identifies a name by number. Defined names are numbered in the order they are created, beginning with 1.
name	String	Defined name.

See Also

DefinedName property

DefinedNameCount method

DefinedNameDlg method

DeleteDefinedName method

Example

The following example clears all defined names:

```
Dim i As Integer
For i = 1 to F1Book1.DefinedNameCount
    F1Book1.DeleteDefinedName x = F1Book1.DefinedNameByIndex( i )
Next i
```

DefinedNameCount Method

Description

Returns the number of defined names in a view.

Syntax

count = *F1Book1.***DefineNameCount**

Part	Type	Description
count	Long	Variable that receives the returned number of defined ranges.

See Also

DefinedName property

DefinedNameByIndex property

DefinedNameDlg method

DeleteDefinedName method

DefinedNameDlg Method

Description

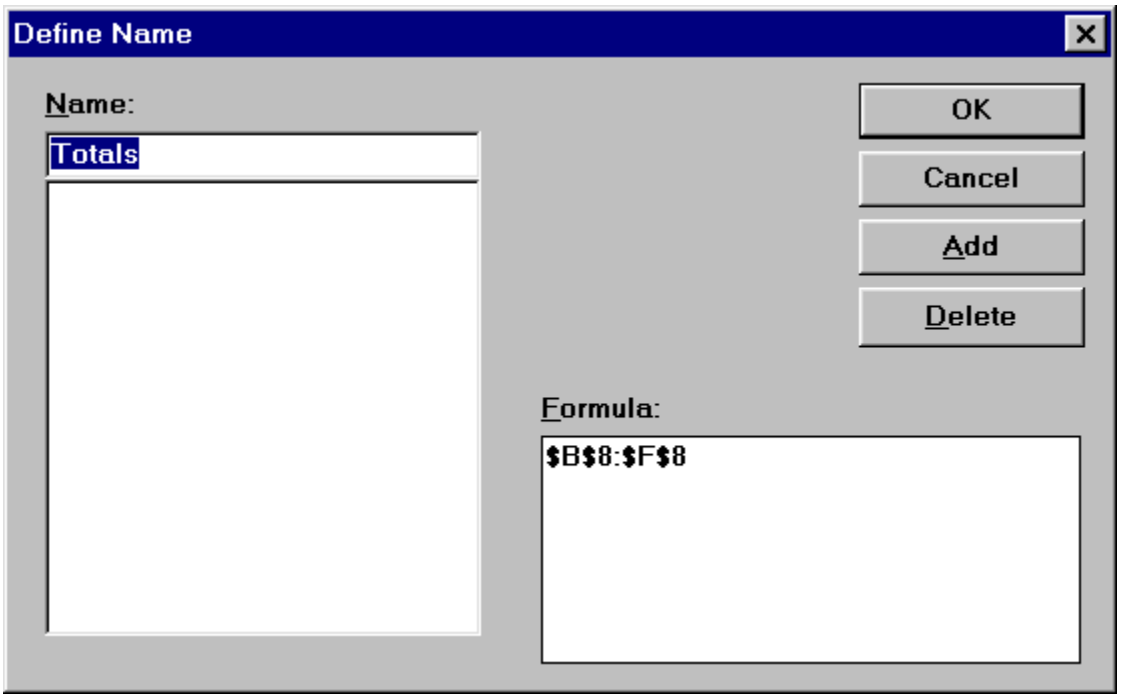
Displays the Define Name dialog box.

Syntax

F1Book1.**DefinedNameDlg**

Remarks

The Define Name dialog box allows you to add and delete user defined names.



Important Do not type a leading equal sign (=) at the beginning of the defined name formula.

See Also

DefinedName property

DeleteDefinedName method

DeleteAutoFillItems Method

Description

Deletes the specified autofill list.

Syntax

*F1Book1.***DeleteAutoFillItems** *nIndex*

Part	Type	Description
nIndex	Integer	Identifies the list's position within all defined autofill lists. The first autofill list has an index of one.

Remarks

Autofill lists are frequently used series of data such as months and days of the week. When you enter a value from an autofill list and move one cell down or to the right, the next value in the list is proposed for that cell. You can also use the fill handle to autofill a single row or column.

See Also

AutoFillItems property

AutoFillItemsCount property

Example

The following example deletes the standard autofill item lists.

```
Dim i Integer
For i = 1 to F1Book1.AutoFillItemsCount
    F1Book1.DeleteAutoFillItems ( i )
Next i
```

DeleteDefinedName Method

Description

Deletes the specified user-defined name.

Syntax

F1Book1.**DeleteDefinedName** *pName*

Part	Type	Description
pName	String	Identifies the user-defined name to delete.

Remarks

You cannot delete a defined name that is currently referenced by a formula in a cell, other defined name, or a chart object.

See Also

DefinedName property

DefinedNameDlg method

DeleteRange Method

Description

Deletes cells, rows, or columns from the specified range in all selected sheets.

Syntax

F1Book1.**DeleteRange** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *nShiftType*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Coordinates that specify the range to delete. If nRow1 is -1, all rows are included in the range; if *nCol1* is -1, all columns are included.
nShiftType	Integer	F1ShiftTypeConstants that determine how the delete should occur:

Constants	Description
F1ShiftHorizontal	Cells to the right of the range are shifted left to fill the vacated space
F1ShiftVertical	Cells below the range are shifted up to fill the vacated space
F1ShiftRows	Rows in which the range resides are deleted and lower rows are shifted up to fill the vacated space
F1ShiftCols	Columns in which the range resides are deleted and the rightmost columns are shifted left to fill the vacated space

See Also

ClearRange method

EditDelete method

DeleteSheets Method

Description

Deletes one or more worksheets from a workbook.

Syntax

F1Book1.**DeleteSheets** *nSheet*, *nSheets*

Part	Type	Description
nSheet	Long	Specifies the index to the first worksheet you want to delete. Worksheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.
nSheets	Long	Specifies how many worksheets to delete. For example, if you specify 3 for this argument, Formula One deletes the worksheet referenced in *nSheet* and the two immediate to its right.

Remarks

The names displayed on sheet tabs are not affected by deletions. However the worksheet index is adjusted to match its new position.

See Also

InsertSheets method

DoCancelEdit Property

Description

Determines if the **CancelEdit** event can be fired.

Syntax

*F1Book1.***DoCancelEdit** [= *boolean*]

Remarks

If True, this property allows the **CancelEdit** event to be fired when the user aborts editing a cell.

See Also

DoEndEdit property

DoStartEdit property

EndEdit event

StartEdit event

DoClick Property

Description

Determines if the **Click** event can be fired.

Syntax

F1Book1.**DoClick** [= *boolean*]

Remarks

If True, this property allows the **Click** event to be fired when the user clicks the Formula One control with the left mouse button. If False, the event is not fired.

DoDblClick Property

Description

Determines if the DblClick event can be fired.

Syntax

F1Book1.**DoDblClick** [= *boolean*]

Remarks

If True, this property allows the **DblClick** event to be fired when the user double clicks the Formula One control with the left mouse button. If False, the event is not fired and in-cell editing is activated when the user double clicks the control.

The default for this property is True.

Note Operations in **Click** event may inhibit the **DblClick**.

DoEndEdit Property

Description

Determines if the **EndEdit** event can be fired.

Syntax

F1Book1.**DoEndEdit** [= *boolean*]

Remarks

If True, this property allows the **EndEdit** event to be fired when the user finishes editing a cell.

See Also

CancelEdit event

StartEdit event

DoCancelEdit property

DoStartEdit property

DoEndRecalc Property

Description

Determines if the **EndRecalc** event can be fired.

Syntax

F1Book1.**DoEndRecalc** [= *boolean*]

Remarks

If True, this property allows the **EndRecalc** event to be fired when the workbook finishes recalculation. If you disable this event, processing is accelerated when you perform large operations on a workbook with Visual Basic code.

See Also

AutoRecalc property

DoStartRecalc property

DoObjClick Property

Description

Determines if the **ObjClick** event can be fired.

Syntax

F1Book1.**DoObjClick** [= *boolean*]

Remarks

If True, this property allows the **ObjClick** event to be fired when the user clicks a named Formula One object with the left mouse button. The default for this property is True. For an object to receive an **ObjClick** event, it must be named.

When an object is named and the **ObjClick** event is enabled, you must press CTRL when interactively selecting the object.

DoObjDblClick Property

Description

Determines if the **ObjDblClick** event can be fired.

Syntax

F1Book1.**DoObjDblClick** [= *boolean*]

Remarks

If True, this property allows the **ObjDblClick** event to be fired when the user double clicks a Formula One object with the left mouse button. The default for this property is True. For an object to receive an **ObjDblClick** event, it must be named.

When an object is named and the **ObjDblClick** event is enabled, you must press CTRL when interactively selecting the object.

DoObjGotFocus Property

Description

Determines whether an event is triggered when an object gets focus.

Syntax

F1Book1.**DoObjGotFocus** [= *boolean*]

Remarks

If this property is true a **ObjGotFocus** event is triggered when the user clicks on an object.

See Also

DoObjLostFocus property

DoObjLostFocus Property

Description

Determines whether an event is triggered when an object loses focus.

Syntax

F1Book1.**DoObjLostFocus** [= *boolean*]

Remarks

If this property is true, an **ObjLostFocus** event is triggered when a user clicks off of an object.

See Also

DoObjGotFocus property

DoObjValueChanged Property

Description

Determines whether an event is triggered when the value of a list box or check box is changed.

Syntax

F1Book1.**DoObjValueChanged** [= *boolean*]

Remarks

If this property is true, an **ObjValueChanged** event is triggered when a user changes the selection for a list box or check box.

DoRClick Property

Description

Determines if the **RClick** event can be fired.

Syntax

F1Book1.**DoRClick** [= *boolean*]

Remarks

If True, this property allows the **RClick** event to be fired when the user clicks the Formula One control with the right mouse button. If False, the event is not fired.

DoRDblClick Property

Description

Determines if the **RDblClick** event can be fired.

Syntax

F1Book1.**DoRDblClick** [= *boolean*]

Remarks

If True, this property allows the **RDblClick** event to be fired when the user double clicks the Formula One control with the right mouse button. The default for this property is True.

DoSelChange Property

Description

Determines if the **SelChange** event can be fired.

Syntax

F1Book1.**DoSelChange** [= *boolean*]

Remarks

If True, this property allows the **SelChange** event to be fired when the current selection changes. If you disable this event, processing is accelerated when you perform large operations on a workbook with Visual Basic code.

DoStartEdit Property

Description

Determines if the **StartEdit** event can be fired.

Syntax

F1Book1.**DoStartEdit** [= *boolean*]

Remarks

If True, this property allows the **StartEdit** event to be fired when the current cell enters edit mode.

See Also

CancelEdit event

EndEdit event

DoCancelEdit property

DoEndEdit property

DoStartRecalc Property

Description

Determines if the **StartRecalc** event can be fired.

Syntax

F1Book1.**DoStartRecalc** [= *boolean*]

Remarks

If True, this property allows the **StartRecalc** event to be fired when the workbook begins recalculation. If you disable this event, processing is accelerated when you perform large operations on a workbook with Visual Basic code.

See Also

DoEndRecalc property

DoTopLeftChanged Property

Description

Determines if the **TopLeftChanged** event can be fired.

Syntax

F1Book1.**DoTopLeftChanged**[= *boolean*]

Remarks

If True, this property allows the **TopLeftChanged** event to be fired when the top left edge of any cell changes position. This event is fired every time any column or row is resized, a column or row is hidden or displayed, or a worksheet is scrolled in any direction. This event is also fired when a workbook is first created. The execution of this event is deferred until the system is idle.

DragIcon Property

Description

Determines the icon displayed in a drag-and-drop operation.

Syntax

F1Book1.**DragIcon** = *icon*

Part	Type	Description
icon	Picture	Identifies the icon displayed during drag-and-drop operations. Valid settings are:

Setting	Description
(None)	Default Windows Icon.
Icon	A custom mouse pointer. See Microsoft documentation.

Remarks

For additional information, refer to the description of the DragIcon property in the Microsoft Visual Basic documentation.

See Also

DragMode property

DragMode Property

Description

Determines the dragging mode for drag-and-drop operations.

Syntax

F1Book1.**DragMode** [= *mode*]

Part	Type	Description
mode	Integer	Identifies the dragging mode. Valid settings are:

Setting	Description
0 (Default)	**Manual**: Requires the drag method to initiate dragging.
1	**Automatic**: Clicking the source control initiates dragging.

Remarks

For additional information, refer to the description of the DragMode property in the Microsoft Visual Basic documentation.

See Also

DragIcon property

Draw Method

Description

This method draws a workbook to the specified device context, such as a printer or window.

Syntax

F1Book1.**Draw** *hDC*, *X*, *Y*, *cX*, *cY*, *nRow*, *nCol*, *pRows*, *pCols*, *nFixedRow*, *nFixedCol*, *nFixedRows*, *nFixedCols*

Part	Type	Description
hDC	OLE_HANDLE	Handle to a device context and specifies where the workbook is drawn. This handle must be established outside of Formula One.
X, *Y*	Long	Coordinates of the upper left corner of the drawn workbook.
cX	Long	Width of the drawn workbook.
cY	Long	Height of the drawn workbook.
nRow, *nCol*	Long	Beginning row and column in the workbook to be drawn.
pRows, *pCols*	Long	Variables, passed by reference, that specify the number of rows and columns to be drawn. Formula One attempts to draw the number of rows and columns specified by these arguments at the current print scale. If necessary, Formula One reduces the print scale until the rows and columns fit in the specified drawing area or until the print scale reaches 10 percent. If *pRows* and pCols are set to 0, Formula One draws as many rows and columns as will fit using the current print scale; and, *pRows* and pCols are set to the number of rows and columns that are drawn.
nFixedRow, *nFixedCol*	Long	Starting fixed row and column of the drawn worksheet.
nFixedRows, *nFixedCols*	Long	Number of rows and columns to fix in the drawn worksheet.

Remarks

The settings for the following properties and methods affect how the workbook is drawn on the specified device:

Property/Method
PrintColHeading property
PrintGridLines property
PrintHCenter property
PrintNoColor property
PrintRowHeading property
PrintVCenter property
SetPrintScale method

EditClear Method

Description

Clears all cells in the selected ranges and all selected objects in all selected sheets.

Syntax

F1Book1.**EditClear** *nClearType*

Part	Type	Description
nClearType	Integer	F1ClearTypeConstants that determine what is cleared from the selected range:

Constants	Description
F1ClearDlg	Displays the Clear dialog box.
F1ClearAll	All (values, formats, and objects)
F1ClearFormats	Formats only
F1ClearValues	Values only (including formulas)

Remarks

EditClear clears cells in all selected ranges. Non-contiguous ranges can be cleared simultaneously if **AddSelection** or the CTRL key were used to select multiple ranges. In addition, all selected objects are cleared.

See Also

ClearRange method

EditDelete method

EditCopy Method

Description

Copies the selected range or objects to the clipboard.

Syntax

F1Book1.**EditCopy**

Remarks

EditCopy copies the selected range or objects to the clipboard. Only one range can be selected. If more than one range is selected, the error F1ErrorOnlyOneRange is returned.

Cells cut to the internal clipboard can be pasted to other areas of Formula One and retain their formulas, formatting, and data. When data is copied to other programs, the windows clipboard is used and the cells become formatted text representations of the data they contain. If you want to copy data from the Windows clipboard to Formula One and the internal clipboard contains data, you must first clear the internal clipboard.

See Also

EditCut method

EditPaste method

EditCopyDown Method

Description

Copies cells in the top row of a selection to the other rows in the selected range.

Syntax

F1Book1.**EditCopyDown**

Remarks

EditCopyDown copies data in the top row of a selection to the other rows in the selection and adjusts relative cell references appropriately.

See Also

EditCopyRight method

EditCopyRight Method

Description

Copies cells in the left column of a selection to the other columns in the selected range.

Syntax

F1Book1.**EditCopyRight**

Remarks

EditCopyRight copies data in the left column of a selection to the other columns in the selection and adjusts relative cell references appropriately.

See Also

EditCopyDown method

EditCut Method

Description

Cuts the selected range or objects to the clipboard.

Syntax

F1Book1.**EditCut**

Remarks

EditCut cuts the selected range or objects to the clipboard and clears the range or objects from the active worksheet. Only one range can be selected. If more than one range is selected, the error F1ErrorOnlyOneRange is returned.

Cells cut to the internal clipboard can be pasted to other areas of Formula One and retain their formulas, formatting, and data. When data is copied to other programs, the windows clipboard is used and the cells become formatted text representations of the data they contain. If you want to copy data from the Windows clipboard to Formula One and the internal clipboard contains data, you must first clear the internal clipboard.

See Also

EditCopy method

EditPaste method

EditDelete Method

Description

Deletes the selected range in all selected sheets.

Syntax

F1Book1.**EditDelete** *nShiftType*

Part	Type	Description
nShiftType	Integer	F1ShiftTypeConstants that determine how the delete should occur:

Constants	Description
F1ShiftHorizontal	Cells to the right of the range are shifted left to fill the vacated space
F1ShiftVertical	Cells below the range are shifted up to fill the vacated space
F1ShiftRows	Rows in which the range resides are deleted and lower rows are shifted up to fill the vacated space
F1ShiftCols	Columns in which the range resides are deleted and the rightmost columns are shifted left to fill the vacated space

See Also

DeleteRange method

EditDeleteSheets method

EditInsert method

EditDeleteSheets Method

Description

Deletes the selected sheets.

Syntax

F1Book1.**EditDeleteSheets**

Remarks

Sheet names displayed on the tabs are not affected by the deletion. However, the remaining worksheets have their indexes adjusted to reflect their new position.

You cannot delete the last sheet out of a workbook.

See Also

DeleteSheets method

EditDelete method

EditInsertSheets method

EditInsert Method

Description

Moves the selected range in all selected worksheets to insert new cells, rows, or columns.

Syntax

F1Book1.**EditInsert** *nShiftType*

Part	Type	Description
nShiftType	Integer	F1ShiftTypeConstants that determine how the insert should occur:

Constants	Description
F1ShiftHorizontal	Cells of the selected range are shifted right to make room for the inserted cells.
F1ShiftVertical	Cells of the selected range are shifted down to make room for the inserted cells.
F1ShiftRows	Rows in which the range resides are shifted down to make room for the inserted cells.
F1ShiftCols	Columns in which the range resides are shifted right to make room for the inserted cells.

Remarks

EditInsert inserts cells, rows, or columns by moving the selected range. It uses the shape of the selected range to determine what to insert. For example, if you select a two-by-two range of cells and call **EditInsert** with the F1ShiftRows option, a new two-by-two range is inserted by shifting the rows in the selected range down.

Formats are copied from the cells above when inserting and shifting down and from the cells to the left when inserting and shifting right. All formatting is copied except that only borders that are the same above and below or left and right of the new cells are copied.

See Also

EditDelete method

EditInsertSheets method

InsertRange method

EditInsertSheets Method

Description

Inserts one or more new worksheets, depending on the number and position of the currently selected worksheets.

Syntax

F1Book1.**EditInsertSheets**

Remarks

EditInsertSheets inserts one or more worksheets in the workbook. It uses the number and position of selected worksheets in the workbook to determine how many worksheets to add and where to add them. For example, if you select the second and third worksheet in the workbook and call **EditInsertSheets**, two new worksheets are added and the second worksheet becomes the fourth.

The newly inserted worksheets are given the next available sheet name, regardless of their position. For example, if there are 5 worksheets in the workbook and you insert two worksheets before the second worksheet, they are given the names Sheet6 and Sheet7. However, all worksheets in the workbook are still indexed from left to right, beginning with 1.

See Also

EditDelete method

EditDeleteSheets method

EditInsert method

InsertSheets method

EditPaste Method

Description

Pastes the contents of the clipboard to the selected range.

Syntax

F1Book1.**EditPaste**

Remarks

EditPaste pastes information from the clipboard to the selected range in the active worksheet. How the information is pasted in the worksheet depends on the size of the selected range.

- If the selected range consists of a single cell, all information in the clipboard is pasted to the worksheet.
- If the selected range is smaller than the clipboard information, only as much information as will fit in the range is pasted.
- If the selected range is larger than the clipboard information, the clipboard information is replicated to fill the range.

Formula One can also paste tab-delimited blocks of data from the clipboard. Objects are also pasted by this command.

See Also

EditCopy method

EditCut method

EditPasteValues method

EditPasteValues Method

Description

Pastes values from the clipboard to the selected range.

Syntax

F1Book1.**EditPasteValues**

Remarks

EditPasteValues only pastes values from the clipboard to the selected range in the active worksheet. If the data on the clipboard contains formulas, only formula results are pasted, not the formula itself. In addition, formatting applied to the data on the clipboard is ignored by this method.

See Also

EditCopy method

EditCut method

EditPaste method

Enabled Property

Description

Determines if the Formula One object is enabled.

Syntax

F1Book1.**Enabled** [= *boolean*]

Remarks

When True, this property enables the Formula One object; when False, the Formula One object is disabled.

EnableProtection Property

Description

Sets or returns whether protection is enabled for all selected sheets.

Syntax

F1Book1.**EnableProtection** [= *boolean*]

Remarks

If this property is True, protection is enabled. Enabling protection means that any cells marked as hidden or locked, are actually hidden and locked. Cells can be marked as locked and hidden using the **SetProtection** and **ProtectionDlg** methods.

See Also

GetProtection method

EndEdit Method

Description

Ends edit mode and applies changes to the active cell.

Syntax

F1Book1.**EndEdit**

Remarks

EndEdit ends cell editing and applies any changes made during edit mode to the active cell. If an invalid entry has been made (e.g., an incorrect formula), edit mode cannot end. In this case, F1ErrorGeneral is returned.

See Also

CancelEdit method

StartEdit method

EnterMovesDown Property

Description

Sets or returns whether pressing the enter key moves the active cell down to the next cell.

Syntax

F1Book1.**EnterMovesDown** [= *boolean*]

If this property is True, pressing the enter key moves the active cell down to the next row, even if no range is selected. If False and a single cell is selected, pressing the enter key does not advance the active cell.

When you create a new workbook, this property is True by default.

Entry Property

Description

Enters text in the active cell of all selected worksheets, or returns the contents of the active cell in the first selected worksheet in the same format as it would be displayed while in edit mode.

Syntax

F1Book1.**Entry** [= *data*]

Part	Type	Description
data	String	The cell value.

Remarks

The text returned by this property is in the same format as if you were entering or editing the cell's value. If the cell contains a formula, the text of the formula is returned. Formulas are returned preceded by an equal sign (=).

Setting this property allows you to enter information in a cell just as a user would enter information. The program automatically determines the kind of data entered (e.g., number, text, formula). It also recognizes dates, times, percentages, currency, fractions, and scientific notation and applies an appropriate number format. When setting formulas, precede the formula with an equal sign (=).

See Also

EntryRC property

Text property

EntryRC Property

Description

Sets the value of a specified cell of all selected worksheets, or returns the text value of the specified cell in the first selected worksheet in the same format as it would be displayed while in edit mode.

Syntax

F1Book1.**EntryRC** (*nRow*, *nCol*) [= *entry*]

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers that identify a cell.
entry	String	Cell value

Remarks

The text returned by this property is in the same format as if you were entering or editing the cell's value. If the cell contains a formula, the text of the formula is returned.

Setting this property allows you to enter information in a cell just as a user would enter information. The program automatically determines the kind of data entered (e.g., number, text, formula). It also recognizes dates, times, percentages, currency, fractions, and scientific notation and applies an appropriate number format.

See Also

Entry property

Text property

ErrorNumberToText Method

Description

Returns the error text corresponding to the specified error number.

Syntax

errorText = *F1Book1*.**ErrorNumberToText** (*nError*)

Part	Type	Description
nError	Integer	F1ErrorConstants that describe the error for which text is returned.
errorText	String	Variable which receives the returned error text. The string must be of sufficient length to hold the returned text.

ExtraColor Property

Description

Sets or returns the color of the view area outside the workbook.

Syntax

F1Book1.**ExtraColor** [= *color*]

Part	Type	Description
color	OLE_COLOR	This value can be one of the normal RGB Colors. These colors are specified using the color palette, or by using the RGB or QBColor functions. The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

Important Formula One does not support the system default colors.

Remarks

The color of the workbook itself is controlled with the BackColor method.

See Also

BackColor property

FileName Property

Description

Attaches a worksheet or workbook file to a view control or returns the name of the file currently attached to the control.

Syntax

F1Book1.**FileName** [= *filename*]

Remarks

If two controls are in existence at the same time with the same **FileName** setting, they are attached to the same table. This provides the same result as using the **Attach** method to attach two or more views to one workbook.

If **FileName** is set to an existing file at design time, a dialog box asks whether the file should be read immediately, not read immediately, or if the read request should be canceled. If the file is read immediately, the worksheet or workbook is loaded. The file can be a Formula One file, an Excel 4.0 or 5.0 file, or a tab-delimited text file. When the form is saved, the worksheet or workbook is saved in the file specified in the **FileName** property as a Formula One file.

If the **FileName** property is blank, the worksheet or workbook is saved with the form instead of in a separate file.

You can also use the Read command on the File menu in the Workbook Designer to read in a file that you want to save in a form.

Caution If you set **FileName** to the name of an Excel file, this file is overwritten with a Formula One file when the form is saved. Excel features not supported in Formula One are lost.

Use **TableName** and not **FileName** to change the workbook name used by external references in formulas.

See Also

TableName property

FilePageSetupDlg Method

Description

Displays the Page Setup dialog box.

Syntax

F1Book1.**FilePageSetupDlg**

Remarks

The Page Setup dialog box allows you to define header and footer text, page margins, page print order, page centering, worksheet-related print options.

See Also

FilePrint method

FilePrintSetupDlg method

FilePrint Method

Description

Prints the active worksheet.

Syntax

F1Book1.**FilePrint** *bShowPrintDlg*

Part	Type	Description
bShowPrintDlg	Boolean	Sets the show print dialog flag. If this flag is True, the Print dialog box is displayed before printing. The Print dialog box allows the user to set printing parameters such as the page range and number of copies to print.

Remarks

FilePrint prints the worksheet or selections as directed by the user.

If the user defined name Print_Area is defined, only those ranges specified in Print_Area are printed. If Print_Area is not defined, the entire formatted section of the worksheet is printed.

See Also

FilePageSetupDlg method

FilePrintSetupDlg method

FilePrintSetupDlg Method

Description

Displays the standard Windows Print Setup dialog box.

Syntax

F1Book1.**FilePrintSetupDlg**

Remarks

The Print Setup dialog box allows you to select the printer to which the workbook is sent, the page orientation, and paper size.

See Also

FilePageSetupDlg method

FilePrint method

FixedCol Property

Description

Return or set the starting fixed column for the active worksheet.

Syntax

F1Book1.**FixedCol** [= *column*]

Part	Type	Description
column	Long	Number of the first fixed column.

Remarks

The **FixedCol** property is used with the **FixedCols** property to fix one or more columns at the left edge of the active worksheet. The fixed columns do not scroll when the worksheet is scrolled horizontally. Individual cells in fixed columns cannot be selected with the mouse or keyboard.

See Also

FixedRow Property

FixedRows Property

FixedCols Property

Description

Sets or returns how many columns to fix at the left edge of the active worksheet.

Syntax

*F1Book1.***FixedCols** [= *columns*]

Part	Type	Description
columns	Long	The number of columns to fix.

Remarks

The **FixedCols** property is used with the **FixedCol** property to fix one or more columns at the left edge of the worksheet. The fixed columns do not scroll when the worksheet is scrolled horizontally. Individual cells in fixed columns cannot be selected with the mouse or keyboard.

See Also

FixedRow property

FixedRows property

FixedRow Property

Description

Sets or returns the starting fixed row in the active worksheet.

Syntax

*F1Book1.***FixedRow** [= *row*]

Part	Type	Description
row	Long	Number of first fixed row.

Remarks

The **FixedRow** property is used with the **FixedRows** property to fix one or more rows at the top of the worksheet. The fixed rows do not scroll when the worksheet is scrolled vertically. Individual cells in fixed rows cannot be selected with the mouse or keyboard.

See Also

FixedCol property

FixedCols property

FixedRows Property

Description

Sets or returns how many rows to fix at the top of the active worksheet.

Syntax

*F1Book1.***FixedRows** [= *rows*]

Part	Type	Description
rows	Long	Number of rows to fix.

Remarks

The **FixedRows** property is used with the **FixedRow** property to fix one or more rows at the top of the worksheet. The fixed rows do not scroll when the worksheet is scrolled vertically. Individual cells in fixed columns cannot be selected with the mouse or keyboard.

See Also

FixedCol property

FixedCols property

FormatAlignmentDlg Method

Description

Displays the Alignment dialog box.

Syntax

F1Book1.**FormatAlignmentDlg**

Remarks

The Alignment dialog box allows you to specify the horizontal and vertical alignment of data in the selected range. In addition, you can enable and disable word wrapping. Settings in this dialog box affect all selected sheets.

See Also

GetAlignment method

SetAlignment method

FormatBorderDlg Method

Description

Displays the Border dialog box.

Syntax

F1Book1.**FormatBorderDlg**

Remarks

The Border dialog box allows you to specify the placement of borders in the selected range. In addition, you can specify the border line style and color. Settings in this dialog box affect all selected sheets.

See Also

GetBorder method

SetBorder method

FormatCurrency0 Method

Description

Formats the selected ranges on all selected worksheets with currency format and no decimal places.

Syntax

F1Book1.**FormatCurrency0**

Remarks

Currency (0) format displays numbers with a currency symbol and no decimal places. Both the currency symbol and its position are taken from the Windows international setting.

See Also

FormatCurrency2 method

NumberFormat property

FormatCurrency2 Method

Description

Formats the selected ranges on all selected worksheets with currency format and two decimal places.

Syntax

F1Book1.**FormatCurrency2**

Remarks

Currency (2) format displays numbers with a currency symbol and two decimal places. Both the currency symbol and its position are taken from the Windows international setting.

See Also

NumberFormat property

FormatCurrency0 method

FormatDefaultFontDlg Method

Description

Displays the Default Font dialog box.

Syntax

F1Book1.**FormatDefaultFontDlg**

Remarks

This dialog box allows you to set the default font for the current workbook. In addition to setting the font and font size used to display data in a workbook, the default font can affect the widths of worksheet columns if the **ColWidthUnits** property is set to character units instead of twips. If column widths are stored as character units, each unit is equal to 1/256th of the character 0 (zero) in the default font.

Because the basic unit for measuring columns changes when you change the default font, you may need to adjust the widths of columns – including the row header column – to achieve the desired appearance for your workbook.

Note By default, Formula One uses Arial as the default font. You must use a TrueType font as your default font in order for **ViewScale** and **SetPrintScale** to work correctly.

See Also

GetDefaultFont method

ColWidth property

SetColWidth method

SetDefaultFont method

FormatFixed Method

Description

Formats the selected ranges on all selected worksheets with fixed format and no decimal places.

Syntax

F1Book1.**FormatFixed**

Remarks

Fixed format includes thousands separators (normally commas).

See Also

NumberFormat property

FormatFixed2 Method

Description

Formats the selected ranges on all selected worksheets with fixed format and two decimal places.

Syntax

F1Book1.**FormatFixed2**

Remarks

Fixed format includes thousands separators (normally commas).

See Also

NumberFormat property

FormatFontDlg Method

Description

Displays the Font dialog box.

Syntax

F1Book1.**FormatFontDlg**

Remarks

The Font dialog box allows you to specify the font, point size, font style, and color of data in the selected range. Settings in this dialog box affect all selected sheets.

See Also

GetFont method

SetFont method

FormatFraction Method

Description

Formats the selected ranges on all selected worksheets with the fraction format.

Syntax

F1Book1.**FormatFraction**

Remarks

The fraction format displays numbers in a fractional format - with a numerator and denominator separated by a slash (e.g. .5 is displayed as 1/2).

See Also

NumberFormat property

FormatGeneral Method

Description

Formats the selected ranges on all selected worksheets with the general format.

Syntax

F1Book1.**FormatGeneral**

Remarks

The general format displays numbers with as many decimal places as necessary; thousands separators (normally commas) are not used.

See Also

NumberFormat property

FormatHmmampm Method

Description

Formats the selected ranges on all selected worksheets with the 12-hour time format.

Syntax

F1Book1.**FormatHmmampm**

Remarks

All selected ranges are formatted with the h:mm AM/PM format (e.g., 1:00 AM).

See Also

NumberFormat property

FormatMdyy Method

Description

Formats the selected ranges on all selected worksheets with the date format.

Syntax

F1Book1.**FormatMdyy**

Remarks

All selected ranges are formatted with the m/d/yy format (e.g., 12/31/93).

See Also

NumberFormat property

FormatNumberDlg Method

Description

Displays the Custom Number dialog box.

Syntax

F1Book1.**FormatNumberDlg**

Remarks

The Custom Format dialog box allows you to define custom number formats for data in the selected range. Settings in this dialog box affect all selected sheets.

See Also

NumberFormat property

FormatPatternDlg Method

Description

Displays the Pattern dialog box.

Syntax

F1Book1.**FormatPatternDlg**

Remarks

The Pattern dialog box allows you to specify the fill pattern and foreground and background colors for the selected range or objects. Settings in this dialog box affect all selected sheets.

See Also

GetPattern method

SetPattern method

FormatPercent Method

Description

Formats the selected ranges on all selected worksheets in percent format.

Syntax

F1Book1.**FormatPercent**

Remarks

Percent format displays numbers with a trailing percent sign and no decimal places.

See Also

NumberFormat property

FormatRCNr Method

Description

Returns a string containing a formatted row and column reference.

Syntax

reference = *F1Book1*.**FormatRCNr** (*nRow*, *nCol*, *bDoAbsolute*)

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers of the reference to format.
bDoAbsolute	Boolean	Specifies whether absolute or relative cell references are used. Use True for absolute references, False for relative references.
reference	String	Variable which received the returned format string. The string must be of sufficient length to hold the returned reference.

See Also

Selection property

Example

The following example displays the string "B4-$B$4" when you click on cell B4.

```
Private Sub F1Book1_Click(ByVal nrow As Long, ByVal nCol As Lon)
    MsgBox F1Book1.FormatRCNr (nRow, nCol, False) & "-" & F1Book1.FormatRCNr(nRow, nCol,
                True)
End Sub
```

FormatScientific Method

Description

Formats the selected ranges on all selected worksheets in scientific format.

Syntax

F1Book1.**FormatScientific**

See Also

NumberFormat property

FormattedText Property

Description

Returns the formatted text value of the active cell.

Syntax

[*text* =] *F1Book1.***FormattedText**

Part	Type	Description
text	String	Variable that receives the returned text.

Remarks

This property returns the text as it is seen in the spreadsheet, including all formatting. To return unformatted text, use **Text** or **Entry**.

See Also

Entry property

FormattedTextRC property

Text property

FormattedTextRC Property

Description

Returns the formatted text value of the specified cell.

Syntax

text = *F1Book1*.**FormattedTextRC** (*nRow*, *nCol*)

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers of the cell from which the text is returned.
text	String	Variable which receives the returned text.

Remarks

This property returns the text as it is seen in the spreadsheet, including all formatting.

See Also

EntryRC property

FormattedText property

NumberRC property

TextRC property

Formula Property

Description

Enters a formula in the active cell of all selected sheets, or returns the text version of the formula already in the active cell of the active worksheet.

Syntax

F1Book1.**Formula** [= *formula*]

Part	Type	Description
formula	String	The formula string.

Remarks

Set this property to enter a formula in the active cell. The formula should not have a leading equal sign (=). This property returns a string containing the text of a formula already in the active cell.

See Also

Entry property

FormulaRC property

Number property

Text property

FormulaRC Property

Description

Enters a formula in the specified cell of all selected sheets, or returns the text version of the formula already in the specified cell of the active worksheet.

Syntax

F1Book1.**FormulaRC** (*nRow*, *nCol*) [= *formula*]

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers that identify the cell.
formula	String	The formula string.

Remarks

Set this property to enter a formula in the specified cell. The formula should not have a leading equal sign (=). This method returns a string containing the text of a formula already in the specified cell.

See Also

Entry property

Formula property

Number property

Text property

GetActiveCell Method

Description

Returns the row and column coordinates of the active cell.

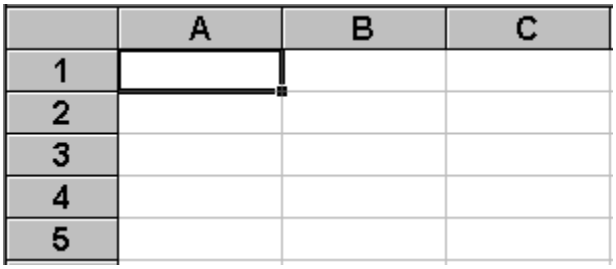
Syntax

F1Book1.**GetActiveCell** *pRow*, *pCol*

Part	Type	Description
pRow	Long	A variable, passed by reference, that receives the returned row number of the active cell.
pCol	Long	A variable, passed by reference, that receives the returned column number of the active cell.

Remarks

The active cell is the cell on which the cursor is currently located. It is the cell in which data is entered or edited if the user starts typing.



Cell A1 is the active cell in this worksheet. The active cell is highlighted by a heavy border.

See Also

Col property

Row property

SetActiveCell method

Example

The following example moves the active cell down one row:

```
Dim theRow As Long
Dim theCol As Long
F1Book1.GetActiveCell theRow, theCol
F1Book1.SetActiveCell theRow+1, theCol
```

GetAlignment Method

Description

Returns data alignment information for the active cell.

Syntax

F1Book1.**GetAlignment** *pHorizontal*, *pWordWrap*, *pVertical*, *pOrientation*

Part	Type	Description
pHorizontal	Integer	A variable, passed by reference that receives the returned horizontal alignment setting. Following are the F1HAlignConstants that can be returned to this variable:

Constant	Description
F1HAlignGeneral	General
F1HAlignLeft	Left
F1HAlignCenter	Center
F1HAlignRight	Right
F1HAlignFill	Fill
F1HAlignJustify	Justify
F1HAlignCenterAcrossCells	Center across cells

Part	Type	Description
pWordWrap	Boolean	A variable, passed by reference, that receives the returned value of the word wrap flag. If the flag is true, text wraps when it exceeds the cell width.
pVertical	Integer	A variable, passed by reference, that receives the returned vertical alignment value. Following are the F1VAlignConstants that can be returned to this variable:

Constant	Description
F1VAlignTop	Top
F1VAlignCenter	Center
F1VAlignBottom	Bottom

Part	Type	Description
pOrientation	Integer	A variable, passed by reference, that receives the returned orientation value. Following are the values that can be returned to this variable. Because this feature is not implemented in this version, this argument always returns 0.

Value	Description
0	Horizontal
1	Vertical
2	Upward
3	Downward

Remarks

If a range is selected, only settings for the active cell are returned.

See Also

FormatAlignmentDlg method

SetAlignment method

Example

The following example returns the current alignment settings for the active cell, changes the horizontal alignment and writes the new settings for the current selection:

```
Dim hAlign as Integer
Dim wordWrap as Boolean
Dim vAlign as Integer
Dim orient as Integer

F1Book1.GetAlignment hAlign, wordWrap, vAlign, orient
F1Book1.SetAlignment kHAlignCenter, wordWrap, vAlign, orient
```

GetBorder Method

Description

Returns the border styles used to display cells.

Syntax

F1Book1.**GetBorder** *pLeft*, *pRight*, *pTop*, *pBottom*, *pShade*, *pcrLeft*, *pcrRight*, *pcrTop*, *pcrBottom*

Part	Type	Description
pLeft, *pRight*, *pTop*, *pBottom*	Integer	Variables, passed by reference that receive the border type for a side of the active cell. Following are the values that can be returned to these variables:

Setting	Description
0	No Border
1	Thin Line
2	Medium Line
3	Dashed Line
4	Dotted Line
5	Thick Line
6	Double Line
7	Hairline

pShade	Integer	A variable, passed by reference, that receives the border *shade* setting; the value corresponds to the built-in shades (not implemented in this version).
pcrLeft, *pcrRight*, *pcrTop*, *pcrBottom*	OLE_COLOR	Variables, passed by reference, that receive the values that specify the colors of the cell border sides. Each is an RGB color that has been translated into one of the 56 colors in the color palette.

See Also

FormatBorderDlg method

BorderStyle property

SetBorder method

Example

The following example retrieves the border settings for the current selection and changes the color of the border lines. Notice the -1 option is used to prevent changes to the outline of the selection.

```
Dim lBorder As Integer
Dim rBorder As Integer
Dim tBorder As Integer
Dim bBorder As Integer
Dim shade As Integer
Dim lColor As Long
Dim rColor As Long
Dim tColor As Long
```

```
Dim bColor As Long
Dim newColor As Long
F1Book1.GetBorder lBorder, rBorder, tBorder, bBorder, shade, lColor, rColor, tColor, bColor
newcolor= RGB(255,0,255)
F1Book1.SetBorder -1, lBorder, rBorder, tBorder, bBorder, shade, -1, newColor, newColor,
        newColor, newColor
```

GetDefaultFont Method

Description

Returns the default font and font size for the specified workbook.

Syntax

F1Book1.**GetDefaultFont** *pFont*, *pSize*

Part	Type	Description
pFont	String	Variable, passed by reference, that receives the default font name.
pSize	Integer	Variable, passed by reference, that receives the default font size in twips. The size is returned in twips, even if the font size was originally specified in points. To convert twips to points, divide the number of twips by 20.

Remarks

In addition to setting the font and font size used to display data in a workbook, the default font can affect the widths of worksheet columns if the **ColWidthUnits** property is set to character units instead of twips. If column widths are stored as character units, each unit is equal to 1/256th of the character 0 (zero) in the default font.

Because the basic unit for measuring columns changes when you change the default font, you may need to adjust the widths of columns – including the row header column – to achieve the desired appearance for your workbook.

Note By default, Formula One uses Arial as the default font. You must use a TrueType font as your default font in order for **ViewScale** and **SetPrintScale** to work correctly.

See Also

FormatDefaultFontDlg method

SetColWidth method

SetDefaultFont method

GetFont Method

Description

Returns font information for the active cell.

Syntax

F1Book1.**GetFont** *pFont*, *pSize*, *pBold*, *pItalic*, *pUnderline*, *pStrikeout*, *pcrColor*, *pOutline*, *pShadow*

Part	Type	Description
pFont	String	Variable, passed by reference, that receives the font name.
pSize	Integer	Variable, passed by reference that receives the font size. The font size is always returned in twips. To convert twips to points, divide the number of twips by 20.
pBold, *pItalic*, *pUnderline*, *pStrikeout*	Boolean	Variables, passed by reference, that receive whether these attributes are turned on for the font. True means the font has the attribute; False means the font does not have the attribute.
pcrColor	OLE_COLOR	Variable, passed by reference that receives the color used to display the font.
pOutline, *pShadow*	Boolean	Variables, passed by reference, that receive whether these attributes are turned on for the font. These attributes are not supported in this version of Formula One.

See Also

FormatFontDlg method

SetFont method

GetHdrSelection Method

Description

Returns whether the row and column headings are selected.

Syntax

F1Book1.**GetHdrSelection** *pTopLeftHdr*, *pRowHdr*, *pColHdr*

Part	Type	Description
pTopLeftHdr	Boolean	Variable, passed by reference, that receives the returned value of the top left header flag. If True, the heading area at the intersection of the row and column headings is selected. If False, the area is not selected.
pRowHdr	Boolean	Variable, passed by reference, that receives the returned value of the row header selection flag. If True, the row headings are selected. If False, the row headings are not selected.
pColHdr	Boolean	Variable, passed by reference, that receives the returned value of the column header selection flag. If True, the column headings are selected. If False, the column headings are not selected.

Remarks

To interactively select this header intersection area, use CTRL+ SHIFT click.

See Also

SetHdrSelection method

GetIteration Method

Description

Returns iteration information.

Syntax

F1Book1.**GetIteration** *pIteration*, *pMaxIterations*, *pMaxChange*

Part	Type	Description
pIteration	Boolean	Variable, passed by reference, that receives the returned state of the iteration flag. If True, iteration is enabled. If False, iteration is disabled.
pMaxIterations	Integer	Variable, passed by reference, that receives the returned maximum number of iterations.
pMaxChange	Double	Variable, passed by reference, that receives the returned maximum change value.

Remarks

Iteration can be used to solve circular references. Formula One calculates until it iterates the number of times specified by *nMaxIterations* or until all cells change by less than the amount specified in *nMaxChange*.

See Also

SetIteration method

GetLineStyle Method

Description

Returns the weight, color, and style for the selected line object or the line forming the border around the selected object.

Syntax

F1Book1.**GetLineStyle** *pStyle*, *pcrColor*, *pWeight*

Part	Type	Description
pStyle	Integer	Variable, passed by reference that receives the returned line style. Following are valid values returned by this variable:

Values	Description
0	Solid
1	Dashed
2	Ditted
3	Dash-ditted
4	Dash-dit-ditted
5	None

Part	Type	Description
pcrColor	OLE_COLOR	Variable, passed by reference, that receives the returned line color.
pWeight	Integer	Variable, passed by reference, that receives the returned line weight. Following are the valid values returned to this variable:

Values	Line weight
0	1/2 point (displayed as 1 point rule on low resolution monitors)
1	1 point
2	2 points
3	3 points

Remarks

Solid lines can assume any of the line weights; styled lines appear solid if set wider than 1/2 point.

See Also

LineStyleDlg method

SetLineStyle method

GetPattern Method

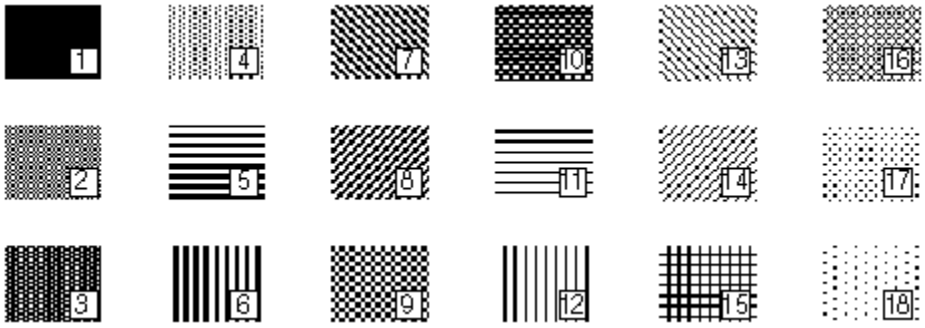
Description

Returns the pattern for the active cell or the first selected object.

Syntax

F1Book1.**GetPattern** *pPattern*, *pcrFG*, *pcrBG*

Part	Type	Description
pPattern	Integer	Variable, passed by reference, that receives the returned pattern number used to display the cells or objects. The pattern number can range from 0 (no pattern) to 18 and represents one of the 18 patterns.
crFG, *crBG*	OLE_COLOR	Variables, passed by reference, that receive the returned foreground and background colors for the pattern.



See Also

FormatPatternDlg method

SetPattern method

GetPrintScale Method

Description

Returns the current print scale settings for the active worksheet.

Syntax

F1Book1.**GetPrintScale** *pScale*, *pFitToPage*, *pVPages*, *pHPages*

Part	Type	Description
pScale	Integer	Variable, passed by reference, that receives the returned scale factor. Scale factor can range from 10 to 400.
pFitToPage	Boolean	Variable, passed by reference, that receives the returned state of the fit to page flag. If the flag is False, the scale percentage returned in *scale* is used to print the workbook. If the flag is True, the worksheet is printed on the number of vertical and horizontal pages returned by *vPages* and *hPages*.
pVPages	Long	Variable, passed by reference, that receives the returned number of vertical pages to which each range in the print job is fit.
pHPages	Long	Variable, passed by reference, that receives the returned number of horizontal pages to which each range in the print job is fit.

Remarks

Each separate print range is printed starting on a new page. When fitting a print job to a specified number of pages, Formula One begins with the page scale specified by *pScale* and reduces the scaling until the job fits in the specified horizontal and vertical pages.

To print a worksheet at the largest scale possible, set *pScale* to 400 when *pFitToPage* is True. Then, specify the number of pages on which you want the worksheet printed.

See Also

FilePageSetup method

SetPrintScale method

GetProtection Method

Description

Returns the protection status of the active cell.

Syntax

F1Book1.**GetProtection** *pLocked*, *pHidden*

Part	Type	Description
pLocked	Boolean	Variable, passed by reference, that receives the returned status of the locked cell flag. If the locked cell flag is True, the active cell is locked.
pHidden	Boolean	Variable, passed by reference, that receives the returned status of the hide formulas flag. If the hide formulas flag is True, the formulas are hidden (formula results are not hidden).

Remarks

After locking cells and hiding formulas, you must enable protection for the workbook before cell locking and formula hiding is enabled. Protection for a workbook is enabled using the **EnableProtection** property.

See Also

SetProtection method

ProtectionDlg method

GetSelection Method

Description

Returns the row and column coordinates of the currently selected range.

Syntax

F1Book1.**GetSelection** *nSelection pRow1*, *pCol1*, *pRow2*, *pCol2*

Part	Type	Description
nSelection	Integer	Identifies the index of the selection. An index of 0 returns the row and column coordinates of the first selection.
pRow1, *pCol1*, *pRow2*, *pCol2*	Long	Variables, passed by reference, that receive the row and column coordinates of the current selection.

See Also

Selection property

SelectionCount property

SetSelection method

GetTabbedText Method

Description

Takes the specified range of data and converts it to a tab-delimited block of text.

Syntax

F1Book1.**GetTabbedText** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *bValuesOnly*, *phText*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Specify the range from which the tab-delimited block of text is obtained.
bValuesOnly	Boolean	Determines if the text is obtained as unformatted text (True) or as formatted text (False).
phText	OLE_HANDLE	Variable, passed by reference, that receives the returned handle to the tab-delimited block of text. You must free this handle by calling GlobalFree. To get a pointer to access the text, you must use GlobalLock and GlobalUnlock. The text is null terminated, and its size is limited only by available memory.

See Also

Clip property

ClipValues property

SetTabbedText method

Text property

TextRC property

GetValidationRule Method

Description

Returns the validation rule for the currently selected range of cells.

Syntax

F1Book1.**GetValidationRule** *pRule*, *pText*

Part	Type	Description
pRule	String	Variable, passed by reference, that receives the returned formula used to test the entered value.
pText	String	Variable, passed by reference, that receives the returned text to display in cell if validation fails.

Remarks

Validation rules can be used to test data entered in a cell or a range of cells. A validation rule consists of a formula to test, and text to display if the validation fails. If the formula returns True, the value is entered. If the formula returns a text string, the string is displayed and the value is not entered. If the formula returns False, the value is not entered and the validation text is displayed.

You can use relative references in validation rules. These references are considered to be relative to the active cell. This allows a validation rule to be properly applied to an entire range.

See Also

ValidationRuleDlg method

SetValidationRule method

GotoDlg Method

Description

Displays the Goto dialog box.

Syntax

*F1Book1.***GotoDlg**

Remarks

GotoDlg displays the Goto dialog box. When you enter a location in this dialog box, Formula One moves the visible area of the workbook to display this selection. To make a selection in another worksheet, type the sheet name, followed by an exclamation mark before the range reference (Sheet3!A1:H10).

To enter multiple ranges, separate ranges with a comma. (A1:C5,D4). You can also enter a defined name if it specifies a range.

See Also

GetActiveCell method

Selection property

SetActiveCell method

HdrHeight Property

Description

Sets or returns the height of the column headers. Setting this property affects all selected sheets.

Syntax

F1Book1.**HdrHeight** [= *height*]

Part	Type	Description
height	Integer	Indicates the height of the column headers.

Remarks

Header height is specified in twips. A twip is 1/1440 of an inch.

See Also

HdrWidth property

RowHeight property

HdrWidth Property

Description

Sets or returns the width of the row headers. Setting this property affects all selected sheets.

Syntax

F1Book1.**HdrWidth** [= *width*]

Part	Type	Description
width	Integer	Indicates the width of the row headers.

Remarks

The units used to store or display width units depends on the setting of **ColWidthUnits**. If **ColWidthUnits** is set to characters, width is specified in units equal to 1/256th of an average character's width in the default font. If **ColWidthUnits** is set to twips, columns are set in twips which are 1/440th of an inch.

If you do any addition or subtraction of widths in your code, make sure that the units of the items you are using are the same.

See Also

ColWidth property

HdrHeight property

HeapMin Method

Description

Calls the Visual C++ _heapmin( ) function and releases unused memory that was allocated for Formula One workbooks.

Syntax

F1Book1.**HeapMin**

Remarks

Formula One uses the memory allocation routines that are provided by Microsoft Visual C++. These routines allocate large blocks of memory from Windows and divide them into smaller memory blocks for use by Formula One. When Formula One releases these memory blocks the memory is not returned to Windows unless the _heapmin( ) function is called. The only time Formula One automatically calls this method is after an application deletes it's last workbook.

You can manually call _heapmin( ) by calling HeapMin. By doing this, you can be certain that Formula One has released as much memory back to Windows as possible without deleting all of your workbooks.

hWnd Property

Description

Returns a handle to the specified view window. This is a read-only property.

Syntax

handle = *F1Book1.***hWnd**

Part	Type	Description
handle	OLE_HANDLE	The handle you want to assign to the view window

Remarks

The Windows environment identifies each window in an application by assigning it a handle, or hWnd. This hWnd is used with Windows API calls. Many Windows environment functions require the hWnd of the current window as an argument. Refer to your development environment documentation for more information about window handles.

InitTable Method

Description

Initializes the workbook associated with the current view control.

Syntax

F1Book1.**InitTable**

Remarks

InitTable initializes the workbook attached to a view. If there is no workbook attached to the view, a new workbook is created.

Initializing the workbook deletes all information in the workbook, resetting it to its default state.

InsertRange Method

Description

Moves the specified range in order to insert new cells, rows, or columns.

Syntax

F1Book1.**InsertRange** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *nShiftType*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Coordinates that specify the range that identifies how many cells rows or columns are inserted and where they are inserted. If *nRow1* is -1, all rows are included in the selection; if *nCol1* is -1, all columns are included.
nShiftType	Integer	F1ShiftTypeConstants that determine how the insert should occur:

Constants	Description
F1ShiftHorizontal	Cells of the specified range are shifted right to make room for the inserted cells.
F1ShiftVertical	Cells of the specified range are shifted down to make room for the inserted cells.
F1ShiftRows	Rows in which the specified range resides are shifted down to make room for the inserted cells.
F1ShiftCols	Columns in which the specified range resides are shifted right to make room for the inserted cells.

Remarks

InsertRange uses the shape of the specified range to determine what to insert. For example, if you specify a two-by-two range of cells and call **InsertRange** with the F1ShiftRows option, a new two-by-two range is inserted by shifting the rows in the specified range down.

Formats are copied from the cells above when inserting and shifting down and from the cells to the left when inserting and shifting right. All formatting is copied except that only borders that are the same above and below or left and right of the new cells are copied.

InsertRange acts on all selected sheets.

See Also

DeleteRange method

EditInsert method

EditDelete method

InsertSheets Method

Description

Inserts one or more sheets at the specified location.

Syntax

F1Book1.**InsertSheets** *nSheet*, *nSheets*

Part	Type	Description
nSheet	Long	Identifies the index of the worksheet in front of which you want to insert the new sheets. Sheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.
nSheets	Long	Determines how many sheets are inserted before *nSheet*.

Remarks

Inserted sheets are given the next sequential names in the worksheet list. To rename a worksheet use the **SheetName** method. The index of all sheets is adjusted to reflect their new position after the insertion.

See Also

DeleteSheets method

LastCol Property

Description

Returns the number of the last occupied column.

Syntax

[*column* =] *F1Book1.***LastCol**

Part	Type	Description
column	Long	Variable that receives the returned column number.

Remarks

This property returns the last column that is not empty, including cells that contain only formatting.

See Also

LastColForRow property

LastRow property

LastColForRow Property

Description

Returns the last occupied column in the specified row.

Syntax

[*pLastColForRow* =] *F1Book1*.**LastColForRow** (*nRow*)

Part	Type	Description
nRow	Long	Identifies a row.
pLastColForRow	Long	Variable that receives the last column value for *nRow*.

Remarks

This property returns the last column in the specified row that is not empty, including cells that contain only formatting.

See Also

LastCol property

LastRow property

LastRow Property

Description

Returns the number of the last occupied row.

Syntax

[*row* =] *F1Book1.***LastRow**

Part	Type	Description
row	Long	Variable that receives the number of the last row.

Remarks

This property returns the last row that is not empty, including rows with cells that contain only formatting.

See Also

LastCol property

LastColForRow property

Example

The following example searches for a string in the active worksheet, moves it to the top left corner and selects it:

```
Sub FindString ( target As String )
    Dim i As Long, j As Long
    For i = 1 to F1Book1.LastRow
        For j = 1 to F1Book1.LastColForRow
            If F1Book1.TypeRC (i, j) = 2 then 'Text only search
               If StrComp (F1Book1.TextRC (i, j), target) Then
                   F1Book1.TopRow = i
                   F1Book1.LeftCol = j
                   F1Book1.SetSelection (i,j,i,j)
                   Exit Sub
               End If
            End If
        Next j
    Next i
End Sub
```

LaunchDesigner Method

Description

Displays the Workbook Designer.

Syntax

F1Book1.**LaunchDesigner**

Remarks

Calling this method displays the Workbook Designer, regardless of the setting of the **AllowDesigner** property.

LeftCol Property

Description

Sets or returns the leftmost column currently displayed in the view.

Syntax

F1Book1.**LeftCol** [= *col*]

Part	Type	Description
col	Long	Indicates the number of the leftmost column.

LineStyleDlg Method

Description

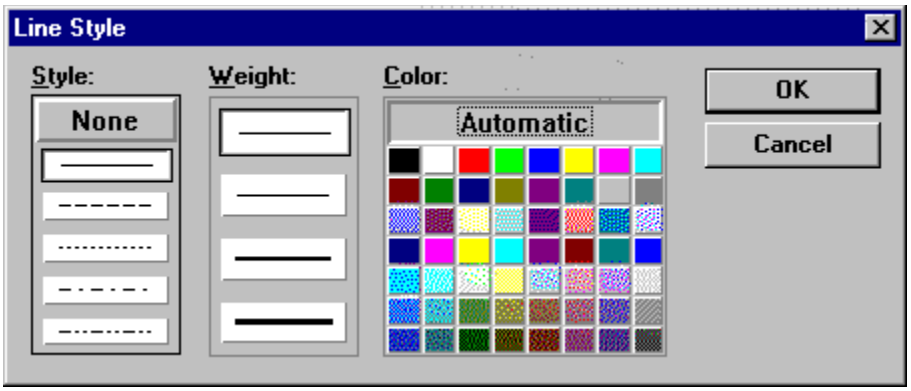
Displays the Line Style dialog box.

Syntax

F1Book1.**LineStyleDlg**

Remarks

The Line Style dialog box allows you to set the weight, color, and style for the selected line object or the line forming the border around the selected object.



Note Styled lines (e.g., lines with dashes and dots) that are wider than one pixel are not supported by this version of Formula One. If you select a heavier weight, the line appears solid.

See Also

GetLineStyle method

SetLineStyle method

Logical Property

Description

Sets or returns the logical (True or False) value of the active cell. Setting this property affects all selected worksheets.

Syntax

F1Book1.**Logical** [= *boolean*]

Remarks

Set this property to change the logical value of the active cell. This property returns the cell's current logical value.

If the cell contains a number, it's logical value is True for nonzero values, and False for zero values. If the cell has text that can be converted to a number, the text is converted and treated as a numeric cell. If the cell contains a formula, the above rules apply depending on the formula's result. All other cells, including empty cells, have a False logical value.

The text "True" and "False" are returned as True and False, respectively.

If the cell contains a formula, the formula is deleted when the logical value is assigned.

See Also

LogicalRC property

LogicalRC Property

Description

Sets or returns the logical (True or False) value of the specified cell. Setting this property affects all selected sheets.

Syntax

F1Book1.**LogicalRC** (*nRow*, *nCol*) [= *value*]

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers that identify a cell
value	Boolean	Indicates the specified cell's logical value.

Remarks

If the cell contains a number, it's logical value is True for nonzero values, and False for zero values. If the cell has text that can be converted to a number, the text is converted and treated as a numeric cell. If the cell contains a formula, the above rules apply depending on the formula's result. All other cells, including empty cells, have a False logical value.

The text "True" and "False" are returned as True and False, respectively.

If the cell contains a formula, the formula is deleted when the logical value is assigned.

See Also

Logical property

MaxCol Property

Description

Sets or returns the last displayable column in a the active worksheet.

Syntax

F1Book1.**MaxCol** [= *col*]

Part	Type	Description
col	Long	Indicates the number of the last displayable column.

Remarks

Columns beyond the last column are not displayed but can be used to hold data and formulas.

Set this property to change the last displayable column. This property returns the current last displayable column.

See Also

MaxRow property

MinCol property

MinRow property

MaxRow Property

Description

Sets or returns the last displayable row in the active worksheet.

Syntax

F1Book1.**MaxRow** [= *row*]

Part	Type	Description
row	Long	Indicate the number of the last displayable row.

Remarks

Rows beyond the last row are not displayed but can be used to hold data and formulas.

See Also

MaxCol property

MinCol property

MinRow property

MinCol Property

Description

Sets or returns the first column that can be displayed in the active worksheet.

Syntax

F1Book1.**MinCol** [= *col*]

Part	Type	Description
col	Long	Indicates the column number of the first displayable column in the view.

Remarks

Columns before the first column are not displayed but can be used to hold data and formulas.

See Also

MaxCol property

MaxRow property

MinRow property

MinRow Property

Description

Sets or returns the first row that can be displayed in the active worksheet.

Syntax

F1Book1.**MinRow** [= *row*]

Part	Type	Description
row	Long	Indicates the first displayable row in a view.

Remarks

Rows before the first row are not displayed but can be used to hold data and formulas.

See Also

MaxCol property

MaxRow property

MinCol property

Mode Property

Description

Sets or returns the current mode for mouse actions in a view.

Syntax

F1Book1.**Mode** [= *mode*]

Part	Type	Description
mode	Integer	F1ModeConstants that indicate the mouse mode:

Constants	Description
F1ModeNormal	Normal worksheet editing
F1ModeLine	Line drawing
F1ModeRectangle	Rectangle drawing
F1ModeOval	Oval drawing
F1ModeArc	Arc drawing
F1ModeChart	Chart drawing
F1ModeField	Field drawing (not implemented in this version)
F1ModeButton	Button drawing
F1ModePolygon	Polygon drawing
F1ModeCheckBox	Check box drawing
F1ModeDropDown	Drop down list box drawing

See Also

PolyEditMode property

Modified Property

Description

Sets or returns the state of the modified flag which indicates whether modifications have been made to a view.

Syntax

F1Book1.**Modified** [= *boolean*]

Remarks

The value of this property indicates whether modifications have been made. When a new view is created the **Modified** property is set to False. When something in the view is changed, **Modified** is set to True and the **Modified** event is fired. If further changes are made, the **Modified** event is not fired again. It will not be fired again until you set this property to False.

Setting **Modified** to True only sets it for the specified view, setting it to False sets it to False for all views attached to a table.

To cause the **Modified** event to be fired every time a change is made to the workbook, set the modified flag to False from within the **Modified** event.

MouseIcon Property

Description

Sets a custom mouse icon.

Syntax

F1Book1.**MouseIcon** [= *picture*]

Part	Type	Description
picture	IPictureDisp	Identifies the picture to be used as a mouse icon.

Remarks

The **MouseIcon** property provides a custom icon that is used when the **MousePointer** property is set to Custom.

MousePointer Property

Description

Returns or sets a value indicating the type of mouse pointer displayed when the mouse is over a particular part of an object at run time.

Syntax

F1Book1.**MousePointer** [= *value*]

Part	Type	Description
value	Integer	F1MousePointerConstants that specify the type of mouse pointer displayed:

Constant	Description
F1Default	Shape determined by the object.
F1Arrow	Arrow head shape.
F1Cross	Cross-hair pointer.
F1IBeam	I-Beam shape.
F1Icon	Small square within a square.
F1Size	Four-pointed arrow pointing north, south, east, and west. WindowsNT only
F1SizeNESW	Double arrow pointing northeast and southwest.
F1SizeNS	Double arrow pointing north and south.
F1SizeNWSE	Double arrow pointing northwest and southeast.
F1SizeWE	Double arrow pointing west and east.
F1UpArrow	Arrow pointing up.
F1Hourglass	Hourglass icon usually associated with waiting.
F1NoDrop	No Drop. Not Implemented.
F1Custom	Custom icon specified by the MouseIcon property.

Remarks

Use this property when you want to indicate changes in functionality as the mouse pointer passes over the Formula One control. The Hourglass setting (11) is useful for indicating that the user should wait for a process or operation to finish.

See Also

MouseIcon property

Example

The following example displays the mouse pointer as an hour glass while the workbook is recalculating:

```
Sub RecalcAll ( )
    F1Book1.MousePointer = F1HourGlass
        . . . 'Do the recalc
    F1Book1.MousePointer = F1Default
End Sub
```

MoveRange Method

Description

Moves a range in all selected sheets.

Syntax

F1Book1.**MoveRange** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *nRowOffset*, *nColOffset*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Coordinates that specify the source range. If *nRow1* is -1, all rows are included in the selection; if *nCol1* is -1, all columns are included.
nRowOffset, *nColOffset*	Long	Specify the offset of the destination range from the source range.

Remarks

When **MoveRange** moves a range, the source range becomes blank. If the cells in the destination range contain data, the data in those cells is lost. References to the moved cells are adjusted to refer to their new location. References to any cells that are overwritten by the moved cells are converted to errors.

See Also

CopyRange method

CopyRangeEx method

EditCut method

EditPaste method

NextColPageBreak Method

Description

Returns the next column where there is a page break.

Syntax

nextCol = *F1Book1*.**NextColPageBreak** (*nCol*)

Part	Type	Description
nCol	Long	Identifies the starting column.
nextCol	Long	Variable that receives the returned column number of the next vertical page break, or zero if there is no page break after *nCol*.

See Also

AddColPageBreak method

AddPageBreak method

AddRowPageBreak method

NextRowPageBreak method

RemoveColPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

NextRowPageBreak Method

Description

Returns the next row where there is a page break.

Syntax

nextRow = *F1Book1*.**NextRowPageBreak** (*nRow*)

Part	Type	Description
nRow	Long	Identifies the starting row.
nextRow	Long	Variable that receives the returned row number of the next horizontal page break, or zero if there is no page break after *nRow*.

See Also

AddColPageBreak method

AddPageBreak method

AddRowPageBreak method

NextColPageBreak method

RemoveColPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

Number Property

Description

Sets or returns the numeric value of the active cell. Setting this property affects all selected sheets.

Syntax

F1Book1.**Number** [= *number*]

Part	Type	Description
number	Double	Indicates the cell value.

Remarks

Set this property to enter a number in the active cell and get the value of this property to return the numeric value currently in the active cell.

When returning the value, cells containing a formula return the numeric result of the formula. If the cell contains text, an attempt is made to convert the text to a number. If the text cannot be converted, 0 (No Error) is returned.

If the active cell contains a formula, the formula is deleted when the numeric value is assigned.

See Also

Entry property

Formula property

NumberRC property

Text property

NumberFormat Property

Description

Sets or returns the number format for the active cell in all selected sheets.

Syntax

F1Book1.**NumberFormat** [= *format*]

Part	Type	Description
format	String	Indicates the number format string.

Remarks

Set this property to change the format used to display numbers in the active cell. This property returns the format string currently used to display numbers in the active cell.

The following table lists the symbols that can be used in the format string.

Format Symbol	Description
General	Displays the number with up to 14 significant digits..
0	Digit placeholder. If the number contains fewer digits than the *format* contains placeholders, the number is padded with 0's. If there are more digits to the right of the decimal than there are placeholders, the decimal portion is rounded to the number of places specified by the placeholders. If there are more digits to the left of the decimal than there are placeholders, the extra digits are retained.
#	Digit placeholder. This placeholder functions the same as the 0 placeholder except the number is not padded with 0's if the number contains fewer digits than the *format* contains placeholders.
?	Digit placeholder. This placeholder functions the same as the 0 placeholder except that spaces are used to pad the digits.
. (period)	Decimal point. Determines how many digits (0's or #'s) are displayed on either side of the decimal point. If the format contains only #'s left of the decimal point, numbers less than 1 begin with a decimal point. If the format contains 0's left of the decimal point, numbers less than 1 begin with a 0 left of the decimal point.
%	Displays the number as a percentage. The number is multiplied by 100 and the % character is appended.
, (comma)	Thousands separator. If the format contains commas separated by #'s or 0's, the number is displayed with commas separating thousands. A comma following a placeholder scales the number by a thousand. For example, the format 0, scales the number by 1000 (e.g., 10,000 would be displayed as 10).
E- E+ e- e+	Displays the number as scientific notation. If the format contains a scientific notation symbol to the left of a 0 or # placeholder, the number is displayed in scientific notation and an E or an e is added. The number of 0 and # placeholders to the right of the decimal determines the number of digits in the exponent. E- and e- place a minus sign by negative exponents. E+ and e+ place a minus sign by negative exponents and a plus sign by positive exponents.
$ - + / ( ) : space	Displays that character. To display a character other than those listed, precede the character with a back slash (\) or enclose the character in double quotation marks (" "). You can also use the slash (/) for fraction formats.
\	Displays the next character. The backslash is not displayed. You can also display a character or string of characters by surrounding the characters with double quotation marks (" ").
	The backslash is inserted automatically for the following characters: ! ^ & ` (left quote) ' (right quote) ~ { } = < >
* (asterisk)	Repeats the next character until the width of the column is filled. You cannot have more than

	one asterisk in each *format* section.
_ (underline)	Skips the width of the next character. For example, to make negative numbers surrounded by parentheses align with positive numbers, you can include the format _) for positive numbers to skip the width of a parenthesis.
"text"	Displays the text inside the quotation marks.
@	Text placeholder. If there is text in the cell, the text replaces the @ format character.
m	Month number. Displays the month as digits without leading zeros (e.g., 1-12). Can also represent minutes when used with h or hh formats.
mm	Month number. Displays the month as digits with leading zeros (e.g., 01-12). Can also represent minutes when used with the h or hh formats.
mmm	Month abbreviation. Displays the month as an abbreviation (e.g., Jan-Dec).
mmmm	Month name. Displays the month as a full name (e.g., January-December).
d	Day number. Displays the day as digits with no leading zero (e.g., 1-2).
dd	Day number. Displays the day as digits with leading zeros (e.g., 01-02).
ddd	Day abbreviation. Displays the day as an abbreviation (e.g., Sun-Sat).
dddd	Day name. Displays the day as a full name (e.g., Sunday-Saturday).
yy	Year number. Displays the year as a two-digit number (e.g., 00-99).
yyyy	Year number. Displays the year as a four-digit number (e.g., 1900-2078).
h	Hour number. Displays the hour as a number without leading zeros (1-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.
hh	Hour number. Displays the hour as a number with leading zeros (01-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock.
m	Minute number. Displays the minute as a number without leading zeros (0-59). The m format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.
mm	Minute number. Displays the minute as a number with leading zeros (00-59). The mm format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number.
s	Second number. Displays the second as a number without leading zeros (0-59).
ss	Second number. Displays the second as a number with leading zeros (00-59).
AM/PM, am/pm A/P, a/p	12-hour time. Displays time using a 12-hour clock. Displays AM, am, A, or a for times between midnight and noon; displays PM, pm, P, or p for times from noon until midnight.
[h]	Outputs total number of hours
[m]	Outputs total number of minutes
[s]	Outputs total number of seconds
s.0, s.00, s.000, ss.0, ss.00, ss.000	Outputs fractional part of second.
[BLACK]	Displays cell text in black.
[BLUE]	Displays cell text in blue.
[CYAN]	Displays cell text in cyan.
[GREEN]	Displays cell text in green.
[MAGENTA]	Displays cell text in magenta.
[RED]	Displays cell text in red.

[WHITE]	Displays cell text in white.
[YELLOW]	Displays cell text in yellow.
[COLOR n]	Displays cell text using the corresponding color in the color palette. n is a number 1 to 56 that represents a color in the color palette.
[conditional value]	Each format can have as many as four sections - one each for positive numbers, negative numbers, zeros, and text. Using the conditional value brackets ([ ]), you can designate a different condition for each section. For example, you might want positive numbers displayed in black, negative numbers in red, and zeros in blue. The following string formats a number for these conditions: [>=0] [BLACK]General; [<0] [RED]General; [BLUE]General

The following table shows examples of custom number formatting.

Format	**Cell Data**	**Display**
#.##	123.456	123.46
	0.2	.2
#.0#	123.456	123.46
	123	123.0
#,##0"CR";#,##0"DR";0	1234.567	1,235CR
	0	0
	-123.45	123DR
#,	10000	10
"Sales="0.0	123.45	Sales=123.5
	-123.45	-Sales=123.5
"X="0.0;"x="-0.0	-12.34	x=-12.3
$* #,##0.00;$* -#,##0.00	1234.567	$ 1,234.57
	-12.34$	12.34$
000-00-0000	123456789	123-45-6789
"Cust. No." 0000	1234	Cust. No. 1234
;;;	Anything	(Not Displayed)
"The End"	123.45	The End
	-123.45	-The End
	text	text
m-d-yy	2/3/94	2-3-94
mm dd yy	2/3/94	02 03 94
mmm d, yy	2/3/94	Feb 3, 94
mmmm d, yyyy	2/3/94	February 3, 1994
d mmmm yyyy	2/3/94	3 February 1994
hh"h" mm"m"	1:32 AM	01h 32m
h.mm AM/PM	14:56	2.56 PM
hhmm "hours"	3:15	0315 hours

See Also

FormatNumberDlg method

NumberRC Property

Description

Sets or returns the numeric value of the specified cell. Setting this property affects all selected sheets.

Syntax

F1Book1.**NumberRC** (*nRow*, *nCol*) [= *number*]

Part	Type	Description
nRow, *nCol*	Long	Row and column coordinates that identify a cell.
number	Double	Indicates the cell value.

Remarks

When returning the value, cells containing a formula return the numeric result of the formula. If the cell contains text, an attempt is made to convert the text to a number. If the text cannot be converted, 0 (No Error) is returned.

If the specified cell contains a formula, the formula is deleted when the numeric value is assigned.

You can use **TypeRC** to determine the type of value in a cell.

See Also

EntryRC property

FormulaRC property

Number property

TextRC property

NumSheets Property

Description

Sets or returns the number of worksheets in the current workbook.

Syntax

F1Book1.**NumSheets** [= *sheets*]

Part	Type	Description
sheets	Long	Indicates the number of worksheets that make up the workbook.

Remarks

Set **NumSheets** to change the number of worksheets in the workbook. Worksheets are added to or deleted from the end of the current list of worksheets. This property returns the number of existing worksheets.

To add worksheets between existing worksheets, use **InsertSheets**. To delete worksheets between existing worksheets, use **DeleteSheets**.

See Also

Sheet property

SheetName property

EditInsertSheets method

EditDeleteSheets method

ObjAddItem Method

Description

Adds an item to a list box object.

Syntax

F1Book1.**ObjAddItem** *ID*, *pItem*

Part	Type	Description
ID	Long	Identification number of the list box object to which the item is added.
pItem	String	Reference to a string to be added to the list box.

Remarks

The new item is added to the end of the list of items in the object.

See Also

ObjDeleteItem method

ObjGetItemCount method

ObjInsertItem method

ObjItem property

ObjItems property

ObjAddSelection Method

Description

Selects an additional object in a view.

Syntax

F1Book1.**ObjAddSelection** *ID*

Part	Type	Description
ID	Long	Identification number of the object to select.

Remarks

All previously selected objects remain selected as the additional object is selected.

See Also

ObjGetSelectionCount method

ObjGetSelection method

ObjBringToFront Method

Description

Places the selected objects in front of all unselected objects in the active worksheet.

Syntax

F1Book1.**ObjBringToFront**

Remarks

If the selected objects overlap unselected objects in the active worksheet, the selected objects are displayed on top of the unselected objects. If a group of objects are selected, the order of the objects within the selection is unchanged.

See Also

ObjSendToBack method

ObjDeleteItem Method

Description

Deletes an item from a list box object.

Syntax

F1Book1.**ObjDeleteItem** *ID*, *nItem*

Part	Type	Description
ID	Long	Identification number of the list box object from which and item is deleted.
nItem	Integer	Number of the item to delete. Items in a list box are numbered sequentially starting with 0.

See Also

ObjAddItem method

ObjItem property

ObjItems property

ObjGetItemCount method

ObjInsertItem method

ObjFirstID Method

Description

Returns the object identification number for the first object in the active worksheet.

Syntax

ID = *F1Book1*.**ObjFirstID**

Part	Type	Description
ID	Long	Variable which receives the returned identification number.

Remarks

This property can be used in conjunction with **ObjNextID** to loop through all the objects in a worksheet.

See Also

ObjNameToID method

ObjGetCell Method

Description

Returns the worksheet cell assigned to hold the value displayed by the specified check box or list box object.

Syntax

F1Book1.**ObjGetCell** *ID*, *pHasCell*, *pRow*, *pCol*

Part	Type	Description
ID	Long	Identification number of the object.
pHasCell	Integer	Variable, passed by reference, that receives the returned state of the object cell flag. This flag controls what is placed in the cell identified by *pRow* and *pCol* when a selection is made from the check box or dropdown listbox. Following are the F1ControlCellConstants for this argument:

Constant	Description
F1ControlNoCell	Cell value is not changed when selection is made from dropdown listbox.
F1ControlCellValue	Cell value is set to index of selection in dropdown listbox.
F1ControlCellText	Cell value is set to text of selection in dropdown listbox.

Part	Type	Description
pRow, *pCol*	Long	Variables, passed by reference, that receive the returned row and column numbers that identify the cell in which the object value is placed.

See Also

ObjValue property

ObjSetCell method

ObjGetItemCount Method

Description

Returns the number of items in a list box object.

Syntax

pItems = *F1Book1.***ObjGetItemCount** (*ID*)

Part	Type	Description
ID	Long	Identification number of the object for which the item count is returned.
pItems	Integer	Variable, passed by reference, that receives the returned item count.

See Also

ObjItem property

ObjItems property

ObjGetPos Method

Description

Returns the position of an object.

Syntax

F1Book1.**ObjGetPos** *ID*, *pX1*, *pY1*, *pX2*, *pY2*

Part	Type	Description
ID	Long	Identification number of the object for which position values is returned.
pX1	Single	Variable, passed by reference, that receives the left edge position of the object.
pY1	Single	Variable, passed by reference, that receives the top edge position of the object.
pX2	Single	Variable, passed by reference, that receives the right edge position of the object.
pY2	Single	Variable, passed by reference, that receives the bottom edge position of the object.

Remarks

The values returned by *pX1* and *pX2* are measured in columns from the left edge of the worksheet. The values returned by *pY1* and *pY2* are measured in rows from the top edge of the worksheet.

In the measurements returned by *pX1*, *pY1*, *pX2*, and *pY2*, integers indicate the edge of the object is placed on a row or column border; fractional numbers indicate the edge of the object is placed between borders. For example, if *pX1* is 1.25, the left edge of the object is positioned one and a quarter columns from the left edge of the worksheet; if *nY1* is 3, the top edge of the object is positioned exactly three rows from the top of the worksheet.

See Also

ObjSetPos method

ObjGetSelection Method

Description

Returns the identification number of a selected object.

Syntax

*F1Book1.***ObjGetSelection** *nSelection, pID*

Part	Type	Description
nSelection	Integer	Number of the object in the group of selected objects for which the identification number is to be returned. This is a 0 based index, that is the first object is given the number 0, second object 1 etc.
pID	Integer	Variable, passed by reference, that receives the returned object identification number.

ObjGetSelectionCount Method

Description

Returns the number of selected objects.

Syntax

count = *F1Book1.***ObjGetSelectionCount**

Part	Type	Description
count	Integer	Variable that receives the returned number of selected objects.

See Also

ObjAddSelection method

ObjGetSelection method

Selection property

ObjGetType Method

Description

Returns the type of the specified object.

Syntax

pType = *F1Book1*.**ObjGetType** (*ID*)

Part	Type	Description
ID	Long	Identification number of the object.
pType	Integer	Variable, passed by reference, that receives the returned object type. Following are the valid F1ObjTypeConstants:

Constants	Description
F1ObjLine	Line
F1ObjRectangle	Rectangle
F1ObjChart	Chart
F1ObjOval	Oval
F1ObjArc	Arc
F1ObjButton	Button
F1ObjPolygon	Polygon
F1ObjCheckBox	Check box
F1ObjDropDown	Drop down list box
F1ObjPicture	Picture

Example

The following example hides all shape objects in the current selection:

```
Dim i As Long, id As Long
For i = 1 to F1Book1.ObjSelectionCount
    F1Book1.ObjGetSelection i, id
    Select Case F1Book1.ObjGetType ( id )
        Case F1ObjLine, F1ObjRectangle, F1ObjOval, F1ObjArc:
        F1Book1.ObjVisible ( id ) = False
    End Select
Next i
```

ObjInsertItem Method

Description

Inserts an item in a list box object.

Syntax

F1Book1.**ObjInsertItem** *ID*, *nItem*, *pItem*

Part	Type	Description
ID	Long	Identification number of the list box object in which an item is inserted.
nItem	Integer	Number of the item before which the new item is inserted. Items in a list box are numbered sequentially starting with 0.
pItem	String	Reference to the item to be inserted.

Remarks

If you specify the number of items in the list box for *nItem*, the new item is added to the end of the list. For example, if a list box has four items and you specify 4 for *nItem*, the new item is added after the fourth item in the list box.

See Also

ObjAddItem method

ObjDeleteItem method

ObjItem property

ObjItems property

ObjGetItemCount method

Example

The following example fills a list box with all defined names in the current worksheet:

```
Dim i As Long, id As Long
With F1Book1
    id = .ObjNameToId ( LBDefinedNames )
    For i = 1 to .DefinedNameCount
        .ObjInsertItem id, 0, .DefinedNameByIndex ( i )
    Next i
End With
```

ObjItem Property

Description

Sets or returns an item from a list box object.

Syntax

*F1Book1.***ObjItem** *(ID, nItem)* [= *item*]

Part	Type	Description
ID	Long	Identification number of the object for which the item is returned.
nItem	Integer	Identifies the item in the list box. Items in a list box are numbered sequentially starting with 0.
item	String	Identifies the list box item.

See Also

ObjGetItemCount method

ObjItems property

Example

The following example puts list box entries in proper case and trims them:

```
Dim i As Long, str As Long
With F1Book1.
    For i = 0 to .ObjGetItemCount ( 1 ) -1
        .ObjItem (1,i) = Trim$ (StrConv (.ObjItem (1, i ), vbProperCase))
    Next i
End With
```

ObjItems Property

Description

Assigns a list of values to, or returns a semicolon-delimited list of items from a list box object.

Syntax

*F1Book1.***ObjItems** (*ID*) [= *items*]

Part	Type	Description
ID	Long	Identification number of the object for which the list is returned.
items	*String*	Semicolon-delimited list of values.

See Also

ObjItem property

ObjGetItemCount method

ObjName Property

Description

Sets or returns the name of the specified object.

Syntax

*F1Book1.***ObjName** (*ID*) [= *name*]

Part	Type	Description
ID	Long	Object identification number.
name	String	Object name. A name can be of any length.

See Also

ObjNameToID method

ObjNameDlg Method

Description

Displays the Object Name dialog box.

Syntax

*F1Book1.***ObjNameDlg**

Remarks

The Object Name dialog box allows you to specify a name for the currently selected object. An object must be selected for this dialog box to display.

ObjNameToID Method

Description

Returns the object identification number for an object specified by name.

Syntax

ID = *F1Book1*.**ObjNameToID** (*pName*)

Part	Type	Description
pName	String	Object name.
ID	Long	Variable that receives the object's identification number.

Remarks

Use this property to obtain the identification number for an object that was created and named for which you do not know the identification number.

See Also

ObjFirstID method

ObjNextID method

ObjNew Method

Description

Creates and adds an object to the active worksheet.

Syntax

F1Book1.**ObjNew** *nType*, *nX1*, *nY1*, *nX2*, *nY2*, *pID*

Part	Type	Description
nType	Integer	Identifies the type of object to create. Following are the F1ObjTypeConstants:

Constants	Description
F1ObjLine	Line
F1ObjChart	Chart
F1ObjRectangle	Rectangle
F1ObjOval	Oval
F1ObjArc	Arc
F1ObjButton	Button
F1ObjCheckBox	Check box
F1ObjDropDown	Drop down list box

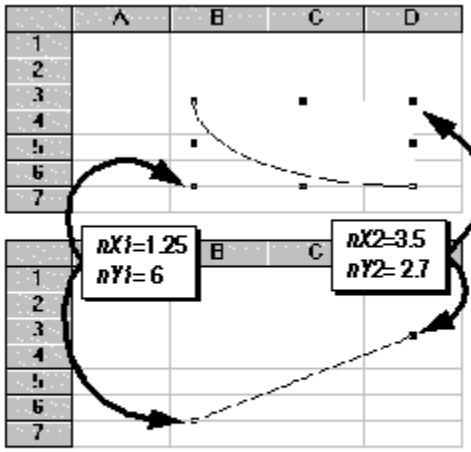
nX1, *nY1*	Single	Coordinates that represent the first anchor point of the object. *nX1* is measured in columns from the left edge of the worksheet; *nY1* is measured in rows from the top edge of the worksheet.
nX2, *nY2*	Single	Coordinates that represent the second anchor point of the object. *nX2* is measured in columns from the left edge of the worksheet; *nY2* is measured in rows from the top edge of the worksheet.
pID	Long	Variable, passed by reference, that receives the identification number of the new object.

Remarks

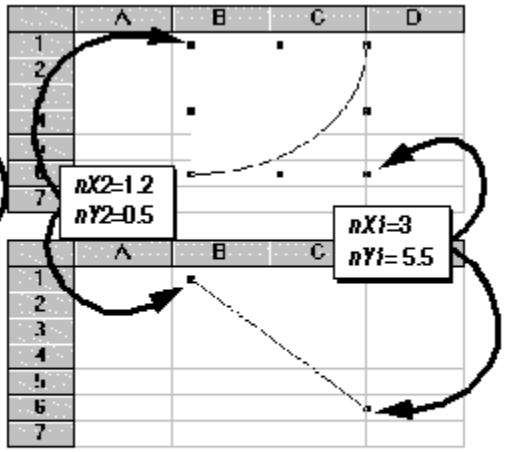
When specifying the location of the object, integers place the edge of the object on a row or column border; fractional numbers place the edge of the object between borders. For example, if *nX1* is 1.25, the first anchor point is placed one and a quarter columns from the left edge of the worksheet; if *nY1* is 3, the first anchor point is placed exactly three rows from the top of the worksheet.

The first anchor point (identified by *nX1* and *nY1*) can be to the right and below the second anchor point. When drawing rectangles, ovals, buttons, list boxes, and check boxes, the order of the anchor points is of no consequence. However, when drawing lines and arcs, the order and placement of the anchor points affects the angle and direction in which the object is drawn.

The following illustration demonstrates how the positioning of anchor points affects the drawing of arcs and lines.



The first anchor points for the arc and line in the above examples is below and to the left of the second anchor points.



The first anchor points for the arc and line in the above examples is below and to the right of the second anchor points.

ObjNewPicture Method

Description

Creates a new metafile picture object on the active worksheet.

Syntax

F1Book1.**ObjNewPicture** *nX1*, *nY1*, *nX2*, *nY2*, *pID*, *hMF*, *nMapMode*, *nWndExtentX*, *nWndExtentY*

Part	Type	Description
nX1, *nY1*	Single	Coordinates that represent the first anchor point of the object. nX1 is measured in columns from the left edge of the worksheet; nY1 is measured in rows from the top edge of the worksheet.
nX2, *nY2*	Single	Coordinates that represent the second anchor point of the object. nX2 is measured in columns from the left edge of the worksheet; nY2 is measured in rows from the top edge of the worksheet.
pID	Long	Variable, passed by reference, that receives the returned identification number of the new picture object.
hMF	OLE_HANDLE	Handle to the metafile to place in the picture object.
nMapMode	Long	Indicates how the metafile is mapped in the object. 7 indicates isotropic mode (keep aspect ratio); 8 is anisotropic mode (stretchable).
nWndExtentX, *nWndExtentY*	Long	Dimensions of the picture.

Remarks

hMF, *nMapMode*, and *nWndExtentX* and *nWndExtentY* usually accompany a metafile and are merely passed to this method. You should be familiar with Windows metafiles and their structure before using this method. Isotropic and anisotropic are Windows constants listed as MM_ISOTROPIC and MM_ANISOTROPIC.

Formula One manages the memory associated with a metafile once a picture object has been created, including freeing memory when the object is deleted.

When specifying the location of the picture object, integers place the edge of the object on a row or column border; fractional numbers place the edge of the object between borders. For example, if nX1 is 1.25, the first anchor point is placed one and a quarter columns from the left edge of the worksheet; if nY1 is 3, the first anchor point is placed exactly three rows from the top of the worksheet.

See Also

ObjSetPicture method

ObjNextID Method

Description

Given an object identification number, this method returns the object identification number for the next object in the active worksheet.

Syntax

[*pNextID* =] *F1Book1*.**ObjNextID** (*ID*)

Part	Type	Description
ID	Long	Identification number of the current object.
pNextID	Long	Variable that receives the returned identification number of the next object.

Remarks

Use this property if objects have been deleted from a worksheet and you are uncertain of the order or identification numbers of the remaining objects. In addition, this method can be used in conjunction with **ObjFirstID** to loop through all the objects in a worksheet.

See Also

ObjNameToID method

Example

The following example select all objects and deletes them:

```
Sub DeleteAllObjects ( )
    Dim id As Long
    On Error Goto DeleteObjects
    With F1Book1
        id = .ObjFirstID
        .ObjSetSelection id
        While True
            id = .ObjNextID (id)
            .ObjAddSelection id
        Wend
    End With
    DeleteObjects:
        F1Book1.EditClear F1ClearAll
End Sub
```

ObjOptionsDlg Method

Description

Displays the Object Options dialog box.

Syntax

*F1Book1.***ObjOptionsDlg**

Remarks

The Object Options dialog box allows you to set options for the currently selected object. An object must be selected for this dialog box to display. The contents of the dialog box change according to the selected object.

ObjPosToTwips Method

Description

Given an object position in relation to rows and columns, this method returns the object position in twips.

Syntax

F1Book1.**ObjPosToTwips** *nX1*, *nY1*, *nX2*, *nY2*, *pX*, *pY*, *pCX*, *pCY*, *pShown*

Part	Type	Description
nX1	Single	Number of columns from the left edge of the worksheet where the left edge of the object is placed.
nY1	Single	Number of rows from the top edge of the worksheet where the top of the object is placed.
nX2	Single	Number of columns from the left edge of the worksheet where the right edge of the object is placed.
nY2	Single	Number of rows from the top edge of the worksheet where the bottom of the object is placed.
pX	Long	Variable, passed by reference, that received the returned left edge position of the object in twips.
pY	Long	Variable, passed by reference, that received the returned top edge position of the object in twips.
pCX	Long	Variable, passed by reference, that received the returned width of the object in twips.
pCY	Long	Variable, passed by reference, that received the returned height of the object in twips.
pShown	Integer	Variable, passed by reference, that received the returned display status of the object at the given position. Following are valid values returned to this variable

Values	Description
0	Not shown
1	Shown
2	Partially shown

Remarks

This method does not reference a specific object on a worksheet and has no effect on any objects.

See Also

ObjGetPos method

ObjSetPos method

ObjSendToBack Method

Description

Places the selected objects behind all unselected objects in a view.

Syntax

F1Book1.**ObjSendToBack**

Remarks

If the selected objects overlap unselected objects in the active worksheet, the selected objects are displayed behind the unselected objects. If a group of objects are selected, the order of the objects within the selection is unchanged.

See Also

ObjBringToFront method

ObjSetCell Method

Description

Assigns a worksheet cell to hold the value displayed by the specified check box or list box object.

Syntax

F1Book1.**ObjSetCell** *ID*, *nHasCell*, *nRow*, *nCol*

Part	Type	Description
ID	Long	Identification number of the object from which a value is placed.
nHasCell	Integer	Sets the object cell flag. This flag controls what is placed in the cell identified by *nRow* and *nCol* when a selection is made from the check box or dropdown list box. Following are the valid F1ControlCellConstants:

Constant	Description
F1ControlNoCell	Cell value is not changed when selection is made from dropdown listbox.
F1ControlCellValue	Cell value is set to index of selection in dropdown listbox.
F1ControlCellText	Cell value is set to text of selection in dropdown listbox.

nRow	Long	Number of the row in which the object value is placed.
nCol	Long	Number of the column in which the object value is placed.

Remarks

If *pHasCell* is set to F1ControlCellValue or F1ControlCellText, the value displayed by the specified check box or list box object is placed in the cell specified by *nRow* and *nCol*. If it is set to F1ControlCellNone, no value is placed and the row and column values are ignored.

See Also

ObjGetCell method

ObjValue property

ObjSetPicture Method

Description

Places a metafile in an existing picture object.

Syntax

F1Book1.**ObjSetPicture** (*ID*, *hMF*, *nMapMode*, *nWndExtentX*, *nWndExtentY*)

Part	Type	Description
ID	Long	Identification number of the picture object in which the metafile specified by *hMF* is placed.
hMF	OLE_HANDLE	Handle to a metafile.
nMapMode	Integer	Indicates how the metafile is mapped in the object. 7 indicates isotropic mode (keep aspect ratio); 8 is anisotropic mode (stretchable).
nWndExtentX, *nWndExtentY*	Long	Dimensions of the picture.

Remarks

This method places a metafile picture in a previously created picture object. The previous metafile contained by the picture object is freed from memory. Picture objects are created with the **ObjNewPicture** method.

Formula One manages the memory associated with a metafile once a picture object has been created, including freeing memory when the object is deleted.

hMF, *nMapMode*, and *nWndExtentX* and *nWndExtentY* usually accompany a metafile and are merely passed to this method. You should be familiar with Windows metafiles and their structure before using this method. Isotropic and anisotropic are Windows constants listed as MM_ISOTROPIC and MM_ANISOTROPIC.

See Also

ObjNewPicture method

ObjSetPos Method

Description

Sets the position and size of an object.

Syntax

F1Book1.**ObjSetPos** *ID*, *nX1*, *nY1*, *nX2*, *nY2*

Part	Type	Description
ID	Long	Identification number of the object to position.
nX1, *nY1*	Single	Coordinates that represent the first anchor point of the object. *nX1* is measured in columns from the left edge of the worksheet; *nY1* is measured in rows from the top edge of the worksheet.
nX2, *nY2*	Single	Coordinates that represent the second anchor point of the object. *nX2* is measured in columns from the left edge of the worksheet; *nY2* is measured in rows from the top edge of the worksheet.

Remarks

When specifying the location of the object, integers place the edge of the object on a row or column border; fractional numbers place the edge of the object between borders. For example, if *nX1* is 1.25, the left edge of the object is placed one and a quarter columns from the left edge of the worksheet; if *nY1* is 3, the top edge of the object is placed exactly three rows from the top of the worksheet.

The first anchor point (identified by *nX1* and *nY1*) can be to the right and below the second anchor point (identified by *nX2* and *nY2*). When positioning rectangles, ovals, buttons, list boxes, and check boxes, the order of the anchor points is of no consequence. However, when positioning lines and arcs, the order and placement of the anchor points affects the angle and direction in which the object is placed. Refer to the illustration in the description of **ObjNew** for an example of anchor point placement.

See Also

ObjGetPos method

ObjSetSelection Method

Description

Selects an object in a view.

Syntax

F1Book1.**ObjSetSelection** *ID*

Part	Type	Description
ID	Long	Identification number of the object to select.

Remarks

All other items on the active worksheet are unselected before the specified object is selected.

See Also

ObjAddSelection method

ObjGetSelection method

ObjGetSelectionCount method

ObjText Property

Description

Sets or returns the text displayed by a specific button or check box object.

Syntax

F1Book1.**ObjText** (*ID*) [= *text*]

Part	Type	Description
ID	Long	Identification number of a button or check box object.
text	String	Text displayed by the object.

Remarks

Button text is displayed on the face of the button. Check box text is displayed adjacent to the check box.

ObjValue Property

Description

Sets or returns the value of a check box or list box object.

Syntax

F1Book1.**ObjValue** (*ID*) [= *pValue*]

Part	Type	Description
ID	Long	Object identification number.
pValue	Integer	Object value.

Remarks

For check box objects 0 is unchecked and 1 is checked. For list box objects -1 means no item is displayed, 0 means the first item is displayed, 1 means the second item is displayed, and so on.

ObjVisible Property

Description

Sets or returns whether an object is visible.

Syntax

*F1Book1.***ObjVisible** (*ID*) [= *boolean*]

Part	Type	Description
ID	Long	Object identification number.

Remarks

If this property is True, the specified object is displayed.

ODBCConnect Method

Description

Connects the current workbook to an ODBC database. To retrieve data from the database, you must call **ODBCConnect**, followed by as many calls to **ODBCQuery** as needed and then **ODBCDisconnect**.

Syntax

F1Book1.**ODBCConnect** *pConnect*, *bShowErrors*, *pRetCode*

Part	Type	Description
pConnect	*String*	*A variable, passed by reference, that identifies the ODBC database to which you want to connect. When you call* **ODBCConnect**, this string is passed to the ODBC call SQLDriverConnect. If *pConnect* is a null string (" ") SQLDriverConnect displays a dialog box prompting you to identify the database. After you call **ODBCConnect**, the value of *pConnect* is set to the connect string that was selected from the dialog box.
bShowErrors	Boolean	Determines whether ODBC errors are displayed in a dialog box. If this is true, all error statuses returned by SQLError ( ) are displayed in an error dialog box. If False, errors are not displayed.
pRetCode	Integer	Variable, passed by reference, that receives the ODBCRETCODE of the most recent ODBC call. This will be SQL_SUCCESS if **ODBCConnect** is successful.

Remarks

For more information about SQLDriverConnect, SQLError, and other ODBC information, refer to the ODBC 2.0 Programmers Reference.

ODBCDisconnect Method

Description
Disconnects the workbook from an ODBC database previously connect with the **ODBCConnect** method.

Syntax
F1Book1.**ODBCDisconnect**

ODBCQuery Method

Description

Builds and then runs a query on a ODBC database connected to the current workbook with the **ODBCConnect** method.

Syntax

F1Book1.**ODBCQuery** *pQuery*, *nRow*, *nCol*, *bForceShowDlg*, *pSetColNames*, *pSetColFormats*, *pSetColWidths*, *pSetMaxRC*, *pRetCode*

Part	Type	Description
pQuery	*Sting*	*Variable, passed by reference, that specifies the query to execute. If pQuery* is a null string (" "), or if *bForceShowDlg* is True, a dialog box is displayed to help you build your query and set other query options. The query string is passed to SQLPrepare and the resulting HSTMT is passed to SQLExecute. Rows are fetched by calls to SQLFetch. After executing **ODBCQuery**, this variable is set to the query string that was sent to SQLPrepare. You can test this value to determine the string built in the Query dialog box.
nRow, nCol	Long	Identifies the row and column coordinates of the cell in which to start placing query results.
bForceShowDlg	Boolean	Determines whether the Query dialog box is displayed. If True, the dialog box is displayed.
pSetColNames	Boolean	Indicates whether worksheet column headings are replaced by database field names. If True, field names are displayed instead of the standard alphabetic column headings. Even though field names are displayed as the column headings, formulas must still use the standard cell referencing conventions (e.g., A1). After this method is executed, you can test this variable to determine the setting for this flag made in the Query dialog box.
bSetColFormats	Boolean	Indicates if formats for date, time, and currency fields are set automatically when data is placed in the worksheet. If True, formats for columns containing these fields are set automatically. If False, you must set the formats for these columns manually in the Workbook Designer or in code. After this method is executed, you can test this variable to determine the setting for this flag made in the Query dialog box.
pSetColWidths	Boolean	Indicates if column widths are automatically set to accommodate the widest data in the column. If True, this property automatically sets the width of each column to be wide enough to display the widest data in the column. After this method is executed, you can test this variable to determine the setting for this flag made in the Query dialog box.
bSetMaxRC	Boolean	Indicates whether the maximum number of worksheet rows and columns are set to the number of records and fields returned by the query. After this method is executed, you can test this variable to determine the setting for this flag made in the Query dialog box.
pRetCode	Integer	Variable, passed by reference, that receives the ODBCRETCODE of the most recent ODBC call. This will be

SQL_SUCCESS if **ODBCQuery** is successful.

Remarks

Formula One does not clear the worksheet each time you run a query.

For more information about SQLPrepare, SQLFetch, and other ODBC information, refer to the ODBC 2.0 Programmers Reference.

OpenFileDlg Method

Description

Displays the Open File dialog box.

Syntax

F1Book1.**OpenFileDlg** *pTitle*, *hWndParent*, *pBuf*

Part	Type	Description
pTitle	String	Title of the dialog box. Use an empty string (" ") for default title.
hWndParent	OLE_HANDLE	Handle to a parent window.
pBuf	String	Name of the file to open.

Remarks

The Open File dialog box allows you to select a file to open.

See Also

SaveFileDlg method

PaletteEntry Property

Description

Sets or returns a color in the Formula One color palette.

Syntax

*F1Book1.***PaletteEntry** (*nEntry*) [= *colors*]

Part	Type	Description
nEntry	Long	One-based index of the entry to change. Formula One presents a color pallete with 56 color entries.
colors	OLE_COLOR	Points to an array of COLORREF values. A COLORREF value is a 32-bit value used to specify an RGB color.

Remarks

When specifying an explicit RGB color, the COLORREF value has the following hexadecimal form: 0x00bbggrr

The low-order byte contains a value for the relative intensity of red; the second byte contains a value for green; and the third byte contains a value for blue. The high-order byte must be zero. The maximum value for a single byte is 255.

Example

The following example shows how to use this property to set the first three entries to red, white and blue:

```
Private Sub RedWhitBlue_Click()
    F1Book1.PaletteEntry(1) = RGB(255, 0, 0) ' Red
    F1Book1.PaletteEntry(2) = RGB(255, 255, 255) ' White
    F1Book1.PaletteEntry(3) = RGB(0, 0, 255) ' Blue
    F1Book1.FormatPatternDlg ' Show palette in color pickers
End Sub
```

PolyEditMode Property

Description

Sets or returns the mode for interactive polygon editing.

Syntax

F1Book1.**PolyEditMode** [= *mode*]

Part	Type	Description
mode	Integer	Indicates the polygon editing mode. Following are the valid F1PolyEditModeConstants:

Constants	Description
F1PolyEditModeNormal	Normal polygon editing
F1PolyEditModePoints	Polygon point editing

Remarks

In normal polygon editing mode (0), only the width and height of polygons can be interactively edited. To reshape and edit the points on a polygon, polygon point editing mode (1) must be enabled.

See Also

Mode property

PrintArea Property

Description

Sets or returns the current print area.

Syntax

*F1Book1.***PrintArea** [= *formula*]

Part	Type	Description
formula	String	Indicates the print area formula. Each range is printed starting on a new page.

Remarks

Set this property to change the "Print_Area" user-defined name to the formula pointed to by formula. This name defines the worksheet ranges to be printed. This property returns a string containing a formula for the Print_Area user defined name.

The print area formula can contain one or more ranges (e.g. A1:C3,A11:C13). If "Print_Area" is Null (""), the selected portion of the active worksheet is printed.

See Also

PrintTitles property

SetPrintAreaFromSelection method

PrintBottomMargin Property

Description

Sets or returns the bottom page margin used during printing.

Syntax

*F1Book1.***PrintBottomMargin** [= *margin*]

Part	Type	Description
margin	Double	Indicates the bottom margin value in inches. This value can range from 0 to 48 inches.

PrintColHeading Property

Description

Sets or returns whether column headings are printed.

Syntax

*F1Book1.***PrintColHeading** [= *boolean*]

If this property is True, column headings are enabled and printed at the top of the worksheet.

PrintDevMode Property

Description

Reads and writes the Windows API DEVMODE printer structure.

Syntax

F1Book1.**PrintDevMode** [= *devmode*]

Part	Type	Description
devmode	OLE_HANDLE	Handle to a DEVMODE structure.

Remarks

The DEVMODE data structure contains information about the device initialization and environment of a printer. This property returns the current values of the DEVMODE structure, make any necessary changes, and set **PrintDevMode** to write out those changes. This property may return NULL if the DevMode structure in Formula One has not been initalized.

You can also allocate your own DEVMODE structure and then set **PrintDevMode**. If you allocate your own structure and pass it to **PrintDevMode**, Formula One frees it's own DEVMODE structure and keeps the one you passed in. You should not refer to this DEVMODE after passing it to **PrintDevMode**.

Important Never GlobalFree the DEVMODE returned via **PrintDevMode**.

Following is the DEVMODE structure. For more information about each field in the structure, refer to your Windows API documentation.

```
typedef struct _devicemode {    // dvmd
        BCHAR  dmDeviceName[32];
        WORD   dmSpecVersion;
        WORD   dmDriverVersion;
        WORD   dmSize;
        WORD   dmDriverExtra;
        DWORD  dmFields;
        short  dmOrientation;
        short  dmPaperSize;
        short  dmPaperLength;
        short  dmPaperWidth;
        short  dmScale;
        short  dmCopies;
        short  dmDefaultSource;
        short  dmPrintQuality;
        short  dmColor;
        short  dmDuplex;
        short  dmYResolution;
        short  dmTTOption;
        short  dmCollate;
        BCHAR  dmFormName[32];
        DWORD  dmLogPixels;
```

```
        DWORD  dmBitsPerPel;
        DWORD  dmPelsWidth;
        DWORD  dmPelsHeight;
        DWORD  dmDisplayFlags;
        DWORD  dmDisplayFrequency;
} DEVMODE;
```

PrintFooter Property

Description

Sets or returns the current page footer.

Syntax

F1Book1.**PrintFooter** [= *footer*]

Part	Type	Description
footer	String	List of special codes that describe the footer printed at the bottom of each page.

Remarks

The following tables list the special codes the footer text string can contain. By default, footer text is centered unless &L or &R is specified.

Format Code	Description
&L	Left-aligns the characters that follow
&C	Centers the characters that follow
&R	Right-aligns the characters that follow
&A	Prints the sheet name.
&D	Prints the current date
&T	Prints the current time
&F	Prints the workbook name
&P	Prints the page number
&P+number	Prints the page number plus number
&P-number	Prints the page number minus number
&&	Prints an ampersand
&N	Prints the total number of pages in the document

The following font codes must appear before other codes and text or they are ignored. The alignment codes (e.g., &L, &C, and &R) restart each section; new font codes can be specified after an alignment code.

Format Code	Description
&B	Use a bold font
&I	Use an italic font
&U	Underline the header
&S	Strikeout the header
&O	Ignored (Mac specific)
&H	Ignored (Mac specific)
&"fontname"	Use the specified font
&nn	Use the specified font size - must be a two digit number

PrintGridLines Property

Description

Sets or returns whether grid lines are printed.

Syntax

F1Book1.**PrintGridLines** [= *boolean*]

If this property is True, grid lines are printed.

PrintHCenter Property

Description

Sets or returns whether a worksheet is horizontally centered when printed.

Syntax

F1Book1.**PrintHCenter** [= *boolean*]

If this property is True, the worksheet is centered horizontally on the page when printed.

PrintHeader Property

Description

Sets or returns the page header printed at the top of each page.

Syntax

F1Book1.**PrintHeader** [= *header*]

Part	Type	Description
header	String	List of special codes that describe the header printed at the top of each page.

Remarks

The following tables list the special codes the header text string can contain. By default, header text is centered unless &L or &R is specified.

Format Code	Description
&L	Left-aligns the characters that follow
&C	Centers the characters that follow
&R	Right-aligns the characters that follow
&A	Prints the current sheet name.
&D	Prints the current date
&T	Prints the current time
&F	Prints the workbook name
&P	Prints the page number
&P+number	Prints the page number plus number
&P-number	Prints the page number minus number
&&	Prints an ampersand
&N	Prints the total number of pages in the document

The following font codes must appear before other codes and text or they are ignored. The alignment codes (e.g., &L, &C, and &R) restart each section; new font codes can be specified after an alignment code.

Format Code	Description
&B	Use a bold font
&I	Use an italic font
&U	Underline the header
&S	Strikeout the header
&O	Ignored (Mac specific)
&H	Ignored (Mac specific)
&"fontname"	Use the specified font
&nn	Use the specified font size - must be a two digit number

PrintLandscape Property

Description

Sets or returns whether a workbook is printed with a portrait or landscape orientation.

Syntax

F1Book1.**PrintLandscape** [= *boolean*]

Remarks

If this property is True, the workbook is printed with a landscape orientation. If this property is false, the workbook is printed with a portrait orientation.

PrintLeftMargin Property

Description

Sets or returns the left page margin used during printing.

Syntax

F1Book1.**PrintLeftMargin** [= *margin*]

Part	Type	Description
margin	Double	Indicates the left margin value in inches. This value can range from 0 to 48 inches.

PrintLeftToRight Property

Description

Sets or returns the order in which worksheet data is printed.

Syntax

F1Book1.**PrintLeftToRight** [= *boolean*]

Remarks

If this property is True, pages in a worksheet are printed left to right before printing top to bottom.

PrintNoColor Property

Description

Sets or returns whether the workbook is printed in color.

Syntax

F1Book1.**PrintNoColor** [= *boolean*]

Remarks

Color formats are translated by the printer driver and printed as patterns. This translation sometimes makes text unreadable. If this property is True, all workbook colors are converted to black and white, and all patterns are removed. A cleaner output is produced.

PrintRightMargin Property

Description

Sets or returns the right page margin used during printing.

Syntax

F1Book1.**PrintRightMargin** [= *margin*]

Part	Type	Description
margin	Double	Indicates the right margin value in inches. Margins can range from 0 to 48 inches.

PrintRowHeading Property

Description

Sets or returns whether row headings are printed.

Syntax

F1Book1.**PrintRowHeading** [= *boolean*]

If this property is True, row headings are enabled and printed at the left edge of the worksheet.

PrintTitles Property

Description

Sets or returns print titles to be printed at the top or left of each page.

Syntax

F1Book1.**PrintTitles** [= *formula*]

Part	Type	Description
formula	String	Formula for the Print_Titles user defined name. This formula can contain a single range or multiple ranges (e.g., A1:IV2, A1:A16384 prints the first two rows and the first column on every page)

Remarks

Print titles are row or column titles that are printed on each page. Row titles are printed at the top of each new page; column titles are printed on the left of each new page. If the property is set to null (""), no titles are printed.

Important When setting print titles, you must specify entire rows and columns.

See Also

SetPrintTitlesFromSelection method

PrintTopMargin Property

Description

Sets or returns the top page margin used during printing.

Syntax

*F1Book1.***PrintTopMargin** [= *margin*]

Part	Type	Description
margin	Double	Indicates the margin in inches. Page margins can range from 0 to 48 inches.

PrintVCenter Property

Description

Sets or returns whether the worksheet is vertically centered when printed.

Syntax

*F1Book1.***PrintVCenter** [= *boolean*]

Remarks

If this property is True, the worksheet is centered vertically on the page when printed.

ProtectionDlg Method

Description

Displays the Cell Protection dialog box.

Syntax

F1Book1.**ProtectionDlg**

Remarks

The Cell Protection dialog box allows you to set the locked attributes of a cell and hidden attributes of a formula. When a cell is locked, its contents cannot be altered. When a formula is hidden, formula text is hidden but formula results are still displayed.

After locking cells and hiding formulas, you must enable protection for the workbook before cell locking and formula hiding is enabled. Protection for a workbook is enabled using the **EnableProtection** property. Settings in this dialog box affect all selected sheets.

See Also

EnableProtection property

GetProtection method

SetProtection method

RangeToTwips Method

Description

Determines the offset, width, and height of the specified range in twips.

Syntax

F1Book1.**RangeToTwips** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *pX*, *pY*, *pCX*, *pCY*, *pShown*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Specify the range for which to find the offset, width, and height.
pX	Long	Variable, passed by reference, that receives the returned horizontal offset of the range
pY	Long	Variable, passed by reference, that receives the returned vertical offset of the range.
pCX	Long	Variable, passed by reference, that receives the returned width of the range.
pCY	Long	Variable, passed by reference, that receives the returned height of the range.
pShown	Integer	Variable, passed by reference, that receives the returned display status of the range. Following are the list of valid values returned to this variable:

Value	Description
0	Not shown
1	Shown
2	Partially shown

Remarks

The coordinates returned by this method are measured in twips from the upper left corner of the workbook control, excluding any borders on the workbook control, to the upper left corner of the specified range. The height and width of the range are also returned in twips.

Use **RangeToTwips** if you want to place a control or object in a worksheet at a specific range location.

See Also

TwipsToRC method

TopLeftChanged event

Read Method

Description

Reads a worksheet or workbook from disk.

Syntax

F1Book1.**Read** *pPathName*, *pFileType*

Part	Type	Description
pPathName	String	Name of the file to read. The name can include drive, path, and file name.
pFileType	Integer	Variable, passed by reference, that receives the returned type of the file that is read. This parameter is undefined if Read returns an error. Following is a list of valid F1FileTypeConstants returned to this variable:

Constants	Description
F1FileFormulaOne	Formula One format
F1FileExcel4	Excel 4.0 format
F1FileTabbedText	Tab-delimited text file
F1FileExcel5	Excel 5.0 format
F1FileFormulaOne3	Formula One v.3 format
F1FileTabbedTextValuesOnly	Tab-delimited text file values only

Remarks

Read initializes a workbook structure and reads a worksheet from the specified file. If there is not a workbook attached to the view, a new workbook is created.

See Also

ReadFromBlob method

Write method

ReadFromBlob Method

Description

Reads a worksheet or workbook that has been stored in memory in a BLOB variable.

Syntax

F1Book1.**ReadFromBlob** *hBlob*, *nReservedBytes*

Part	Type	Description
hBlob	OLE_HANDLE	Reference to a BLOB variable in memory.
nReservedBytes	Integer	Size of the BLOB variable. Not implemented in this version and must be 0.

Remarks

If a worksheet or workbook was previously saved as a blob in a database, it must be assigned to a blob type variable before it is passed to **ReadFromBlob**.

See Also

WriteToBlob method

Windows API

- GlobalAlloc
- GlobalFree
- GlobalLock
- GlobalUnlock

Recalc Method

Description

Recalculates the workbook attached to a view.

Syntax

F1Book1.**Recalc**

Remarks

Recalc recalculates all formulas in the workbook attached to the specified view.

See Also

AutoRecalc property

RemoveColPageBreak Method

Description

Removes a vertical page break adjacent to the left edge of the specified column.

Syntax

F1Book1.**RemoveColPageBreak** *nCol*

Part	Type	Description
nCol	Long	Column where the page break is removed.

See Also

AddColPageBreak method

AddPageBreak method

AddRowPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemovePageBreak method

RemoveRowPageBreak method

RemovePageBreak Method

Description

Removes page breaks adjacent to the active cell.

Syntax

F1Book1.**RemovePageBreak**

Remarks

If a horizontal page break is adjacent to the top edge of the active cell, it is removed. In addition, if a vertical page break is adjacent to the left edge of the active cell, it is also removed.

See Also

AddColPageBreak method

AddPageBreak method

AddRowPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemoveColPageBreak method

RemoveRowPageBreak method

RemoveRowPageBreak Method

Description

Removes a horizontal page break adjacent to the top edge of the specified row.

Syntax

F1Book1.**RemoveRowPageBreak** *nRow*

Part	Type	Description
nRow	Long	Row where the page break is removed.

See Also

AddColPageBreak method

AddPageBreak method

AddRowPageBreak method

NextColPageBreak method

NextRowPageBreak method

RemoveColPageBreak method

RemovePageBreak method

Repaint Property

Description

Sets or returns the repaint status for a view.

Syntax

*F1Book1.***Repaint** [= *boolean*]

Remarks

If this property is True, repainting occurs in the entire window when Windows sends a WM_PAINT message. No repainting occurs when the flag is False.

The repaint flag is not saved to disk. The default repaint setting for a new view is always True.

Row Property

Description

Determines the active row in the active worksheet. This is a run time only property.

Syntax

F1Book1.**Row** [= *row*]

Part	Type	Description
row	Long	Row number.

Remarks

The **Row** property is used along with the **Col** property to set the active cell in the worksheet. The **Row** property is automatically changed if a range is selected using the **SelStart**... and **SelEnd**... properties.

You can specify -1 as the row and column number to indicate all rows or all columns. For example, setting **Row** to -1 and **Col** to 1 causes all rows in column 1 to be selected. Setting both **Row** and **Col** to -1 selects the entire worksheet.

See Also

Col property

GetActiveCell method

SelEndCol property

SelEndRow property

SelStartCol property

SelStartRow property

SetActiveCell method

RowHeight Property

Description

Sets or returns the height of a single specified row. Setting this property affects all selected sheets

Syntax

F1Book1.**RowHeight** (*nRow*) [= *height*]

Part	Type	Description
nRow	Long	Row number.
height	Integer	Row height value in twips. A twip is 1/1440th of an inch.

See Also

ColWidth property

SetRowHeight method

RowHeightDlg Method

Description

Displays the Row Height dialog box.

Syntax

F1Book1.**RowHeightDlg**

Remarks

The Row Height dialog box allows you to set the height of the selected rows, specify default row heights, and specify automatic row height. In addition, you can specify whether the selected rows are shown or hidden. Settings in this dialog box affect all selected sheets.

See Also

RowHeight property

SetRowHeight method

SetRowHeightAuto method

RowHidden Property

Description

Sets or returns the display status of an individual row.

Syntax

F1Book1.**RowHidden** (*nRow*) [= *boolean*]

Part	Type	Description
nRow	Long	Identifies a row by number.

Remarks

If this property is true, the specified row is not displayed. If this property is False, the row is displayed.

See Also

ColHidden property

RowMode Property

Description

Sets or returns the row mode status for a view.

Syntax

F1Book1.**RowMode** [= *boolean*]

Remarks

If this property is True, an entire row is selected when you select a cell. Normal cell selection occurs when the flag is False.

RowText Property

Description

Sets or returns the name for the specified row. Setting this property affects all selected sheets.

Syntax

F1Book1.**RowText** (*nRow*) [= *rowText*]

Part	Type	Description
nRow	Long	Identifies a row by number.
rowText	String	Row name.

Remarks

Naming a row is useful for labeling rows so they reflect the data in the row (e.g., row 2 might be named Central Region). The row name is displayed in the row heading and is used for display purposes only. The row is still referred to by normal cell references in formulas.

Row names can be up to 254 characters long.

See Also

ColText property

TopLeftText property

SaveFileDlg Method

Description

Displays the Save As dialog box.

Syntax

F1Book1.**SaveFileDlg** *pTitle*, *pBuf*, *pFileType*

Part	Type	Description
pTitle	String	Title of the dialog box. Use a null string for the default title.
pBuf	String	Variable, passed by reference, that receives the returned name by which the workbook is saved.
pFileType	Integer	Variable, passed by reference, that receives the returned file type selected when the file is saved. Following are the F1FileTypeConstants:

Constants	Description
F1FileFormulaOne	Formula One format
F1FileExcel4	Excel 4.0 format
F1FileExcel5	Excel 5.0 format
F1FileFormulaOne3	Formula One v.3 format

Remarks

The Save As dialog box allows you to save and name a file.

See Also

OpenFileDlg method

SaveWindowInfo Method

Description

Saves the window specific information from a view to its workbook.

Syntax

F1Book1.**SaveWindowInfo**

Remarks

Window specific information from the view must be saved to its workbook if the information is to be saved the next time the workbook is written to disk.

The following table lists the window information that is saved.

Saved information		
AllowArrows	RowMode	Selection
AllowDelete	EnterMovesDown	ShowFormulas
AllowFillRange	ExtraColor	ShowGridLines
AllowInCellEditing	FixedCol	ShowColHeading
AllowMoveRange	FixedCols	ShowHScrollBar
AllowSelections	FixedRow	ShowRowHeading
AllowResize	FixedRows	ShowSelections
AllowTabs	LeftCol	ShowVScrollBar
AllowFormulas	MaxCol	ShowZeroValues
BackColor	MaxRow	TopRow

See Also

Write method

WriteToBlob method

ScrollToLastRC Property

Description

Sets or returns how scrollbars work.

Syntax

F1Book1.**ScrollToLastRC** [= *boolean*]

Remarks

Normally scroll bars can scroll the last filled cell to the top-left edge of the window. If **ScrollToLastRC** is set to False it causes scrolling to stop when the last filled row and column reach the bottom-right edge of the screen. The last row and column can still be set to the top-left edge of the window by setting **TopRow** or **LeftCol** properties to the appropriate value.

Selection Property

Description

Selects the specified range and moves the active cell to the top left cell in the range or returns a string that represents the row and column coordinates of the current selection.

Syntax

F1Book1.**Selection** [= *range*]

Part	Type	Description
range	String	Identifies the starting and ending rows and columns of the selection or a defined name that represents the selection.

See Also

AddSelection method

GetSelection method

ObjSetSelection method

SelectionCount property

SetSelection method

SelectionCount Property

Description

Returns the number of selected ranges.

Syntax

F1Book1.**SelectionCount** [= *count*]

Part	Type	Description
count	Integer	Number of selected ranges in the active worksheet.

Remarks

You can use this property to determine if only objects are selected. If you know you have a selection, and **SelectionCount** equals 0, you know the remaining selections must be objects.

See Also

AddSelection method

Selection property

GetSelection method

ObjGetSelectionCount method

SetSelection method

SelEndCol, SelEndRow, SelStartCol, and SelStartRow Properties

Description

These properties determine the starting column, starting row, ending column, and ending row of a selected range.

Syntax

F1Book1.**Sel**...**Col** [= *column*]

Part	Type	Description
column	Long	Identifies a column by number. Use -1 to specify all columns.

F1Book1.**Sel...Row** [= *row*]

Part	Type	Description
row	Long	Identifies a row by number. Use -1 to specify all rows

Remarks

SelEndCol, **SelEndRow**, **SelStartCol**, and **SelStartRow** define the starting and ending rows and columns when selecting a range. Use these properties to select a range before performing operations such as copying or deleting data.

If you need to select multiple ranges, you use the **AddSelection** method or **Selection** property.

See Also

Col property

GetSelection method

Row property

SelectionCount property

SetSelection method

SetActiveCell Method

Description

Makes the specified cell the active cell.

Syntax

F1Book1.**SetActiveCell** *nRow*, *nCol*

Part	Type	Description
nRow	Long	Indicates the number of the row that contains the cell you want to make active.
nCol	Long	Indicates the number of the column that contains the cell you want to make active.

Remarks

The active cell is the cell on which the cursor is currently located. It is the cell in which data is entered or edited if the user starts typing.

Cell A1 is the active cell in this worksheet. The active cell is highlighted by a heavy border.

If you use **SetActiveCell** to change the active cell to another cell in the same selection, only the active cell changes, the selection is not lost.

See Also

Col property

Row property

Example

The following example moves the active cell down one *row*:

```
Dim theRow As Long
Dim theCol As Long
F1Book1.GetActiveCell theRrow, theCol
F1Book1.SetActiveCell theRow+1, theCol
```

SetAlignment Method

Description

Changes the data alignment information for the currently selected cells in all selected sheets.

Syntax

F1Book1.**SetAlignment** *HAlign*, *bWordWrap*, *VAlign*, *nOrientation*

Part	Type	Description
HAlign	Integer	Indicates the horizontal alignment. Following are the valid F1HAlighConstants:
bWordWrap	Boolean	Indicates the value of the word wrap flag. If the flag is true, text wraps when it exceeds the cell width.
VAlign	Integer	Indicates the vertical alignment. Following are the valid F1VAlignConstants:
nOrientation	Integer	Indicates the orientation. Following are the valid settings. Because this feature is not implemented in this version, this argument must always be 0.

HAlign constants:

Constant	Description
F1HAlignGeneral	General
F1HAlignLeft	Left
F1HAlignCenter	Center
F1HAlignRight	Right
F1HAlignFill	Fill
F1HAlignJustify	Justify
F1HAlignCenterAcrossCells	Center across cells

VAlign constants:

Constant	Description
F1VAlignTop	Top
F1VAlignCenter	Center
F1VAlignBottom	Bottom

nOrientation values:

Value	Description
0	Horizontal
1	Vertical
2	Upward
3	Downward

See Also

FormatAlignmentDlg method

GetAlignment method

Example

The following example returns the current alignment settings for the active cell, changes the horizontal alignment and writes the new settings for the current selection:

```
Dim hAlign as Integer
Dim wordWrap as Boolean
Dim vAlign as Integer
Dim orient as Integer
F1Book1.GetAlignment hAlign, wordWrap, vAlign, orient
F1Book1.SetAlignment F1HAlignCenter, wordWrap, vAlign, orient
```

SetBorder Method

Description

Sets the border styles used to display cells in all selected sheets.

Syntax

F1Book1.**SetBorder** *nOutline*, *nLeft*, *nRight*, *nTop*, *nBottom*, *nShade*, *crOutline*, *crLeft*, *crRight*, *crTop*, *crBottom*

Part	Type	Description
nOutline	Integer	Indicates the outline border for the selected range. This border type is assigned to the top edge of cells in the top row, the left edge of cells in the left column, the right edge of cells in the right column, and the bottom edge of cells in the bottom row.
nLeft, nRight, nTop, nBottom	Integer	Indicates the border type for a side of the active cell. Following are the valid values for these arguments:

Setting	Description
-1	No Change
0	No Border
1	Thin Line
2	Medium Line
3	Dashed Line
4	Dotted Line
5	Thick Line
6	Double Line
7	Hairline

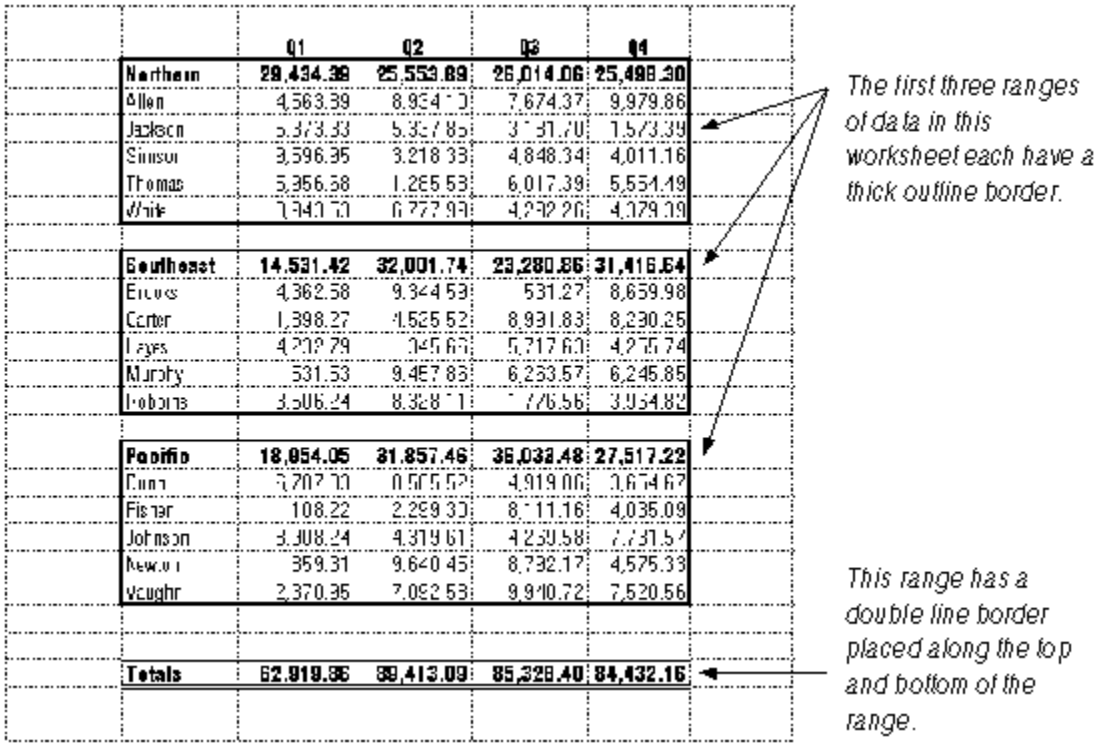
nShade	Integer	Indicates the border shade setting; the value corresponds to the built-in shades (not implemented in this version).
crOutline, *crLeft*, *crRight*, *crTop*, *crBottom*	OLE_COLOR	Indicates the values that specify the colors of the cell border sides or outline. Each is an RGB color that has been translated into one of the 56 colors in the color palette.

Remarks

Use the -1 option to indicate that a border or the outline should be unchanged. This option is most commonly used for the outline setting when specifying a top, left, right, or bottom border where there is not an outline border. The -1 option can also be used to format a range when you only want to change 1 attribute and leave the rest unchanged.

For example, if you want to apply a top border to a selected range, you would specify the border type, such as 5 for a thick border. Then for the outline setting, you would use -1 to indicate no change in the outline border. If you use 0 (no line) for the outline setting, the outline setting overrides the top border

setting you specified, and the top border is not displayed.



See Also

FormatBorderDlg method

GetBorder method

Example

The following example returns the border settings for the current selection and changes the color of the border lines. Notice the -1 option is used to prevent changes to the outline of the selection.

```
Dim lBorder As Integer
Dim rBorder As Integer
Dim tBorder As Integer
Dim bBorder As Integer
Dim shade As Integer
Dim lColor As Long
Dim rColor As Long
Dim tColor As Long
Dim bColor As Long
Dim newColor As Long

F1Book1.GetBorder lBorder, rBorder, tBorder, bBorder, shade, lColor, rColor, tColor, bColor
newcolor= RGB(255,0,255)
F1Book1.SetBorder -1, lBorder, rBorder, tBorder, bBorder, shade, -1, newColor, newColor,
        newColor, newColor
```

SetColHidden Method

Description

Sets or the display status of a range of columns.

Syntax

F1Book1.**SetColHidden** *nCol1*, *nCol2*, *bColHidden*

Part	Type	Description
nCol1	Long	Identifies the first column in a range.
nCol2	Long	Identifies the last column in a range.
bColHidden	Boolean	Identifies the status of the column hidden flag for all columns in the range identified by *nCol1* an *nCol2*. If the column hidden flag is true, the specified columns are not displayed. If the flag is False, the columns are displayed.

Remarks

Use the **ColHidden** property to return or change the display status of a single column.

See Also

SetRowHidden method

SetColWidth Method

Description

Changes the width of one or more columns in all selected sheets.

Syntax

F1Book1.**SetColWidth** *nCol1*, *nCol2*, *nWidth*, *bDefColWidth*

Part	Type	Description
nCol1	Long	Identifies the first column in block of columns whose width you want to change.
nCol2	Long	Identifies the last column in a block of columns whose width you want to change.
nWidth	Integer	The column width value.
bDefColWidth	Boolean	Specifies whether the default column width is changed. True specifies that the default width is set to *nWidth*, and the specified columns are set to the default width. In addition, any columns that use the default width are updated with the new default. False specifies that the default width is unchanged.

Remarks

Column width is specified in units equal to 1/256th of an average character's width in the default font or twips, depending on the setting of the **ColWidthUnits** property. If you want to set the column width in twips with out changing the units in which widths are stored and displayed, use the **ColWidthTwips** property or the **SetColWidthTwips** method.

Use the **ColWidth** property to set or return the width of a single column.

See Also

ColWidthDlg method

RowHeight property

SetColWidthAuto method

SetRowHeight method

SetColWidthAuto Method

Description

Sets the widths of the specified columns to automatically adjust to the largest column entry, including the header.

Syntax

F1Book1.**SetColWidthAuto** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *bSetDefaults*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Specifies the range containing the columns for which to set the width.
bSetDefaults	Boolean	If True, columns greater than or lest than the default column width that don't need to be are set to the default width. If text is wider than column, those columns are resized.

Remarks

SetColWidthAuto specifies that the widths of the columns in the specified range are automatically set to display the largest entry in the columns. The columns are set at least as wide as the default column width. If -1 is specified for *nRow1*, or all rows are specified, the columns are made wide enough for the column headers as well as the specified cells.

See Also

ColWidth property

ColWidthDlg method

SetColWidth method

SetColWidthTwips Method

Description

Changes the width of one or more columns to the specified number of twips.

Syntax

F1Book1.**SetColWidthTwips** *nCol1*, *nCol2*, *nWidth*, *bDefColWidth*

Part	Type	Description
nCol1	Long	Identifies the first column in block of columns whose width you want to change.
nCol2	Long	Identifies the last column in a block of columns whose width you want to change.
nWidth	Long	Indicates the column width value in twips.
bDefColWidth	Boolean	Specifies whether the default column width is changed. True specifies that the default width is set to *nWidth*, and the specified columns are set to the default width. In addition, any columns that use the default width are updated with the new default. False specifies that the default width is unchanged.

See Also

ColWidth property

ColWidthDlg method

RowHeight property

SetColWidth method

SetColWidthAuto method

SetRowHeight method

SetDefaultFont Method

Description

Sets the default font and font size for the specified workbook.

Syntax

F1Book1.**SetDefaultFont** *pFont*, *nSize*

Part	Type	Description
pFont	String	Indicates the default font name.
nSize	Integer	Indicates the default font size in twips. When setting the size, you can specify points or twips. When specifying size in points, use a positive number; use a negative number when specifying twips. When specifying twips, **SetDefaultFont** uses the absolute value of the number you provide for size.

Remarks

In addition to setting the font and font size used to display data in a workbook, the default font can affect the widths of worksheet columns. Column widths are set in units equal to 1/256th of the character 0 (zero) in the default font, or twips, depending on the setting of **ColWidthUnits**.

Because the basic unit for measuring columns can change when you change the default font, you may need to adjust the widths of columns – including the row header column – to achieve the desired appearance for your workbook.

Note By default, Formula One uses Arial as the default font. Be sure you always use a TrueType font as the default font in order for print and display scaling to work correctly.

See Also

ColWidthTwips property

FormatDefaultFontDlg method

SetColWidth method

SetFont Method

Description

Sets font information for all selected cells in all selected sheets.

Syntax

F1Book1.**SetFont** *pFont*, *nSize*, *bBold*, *bItalic*, *bUnderline*, *bStrikeout*, *crColor*, *bOutline*, *bShadow*

Part	Type	Description
pFont	String	Indicates the font name.
nSize	Integer	You can specify the font size in points or twips, but the font size is always returned in twips. When specifying *nSize* in points, use a positive number; use a negative number when specifying twips. When you specify twips, this method uses the absolute value of the number you provide for *nSize*.
bBold, *bItalic*, *bUnderline*, *bStrikeout*	Boolean	Indicates whether these attributes are turned on for the font. True means the font has the attribute; False means the font does not have the attribute
crColor	OLE_COLOR	Indicates the color used to display the font.
bOutline, *bShadow*	Boolean	Indicates whether these attributes are turned on for the font. These attributes are not supported in this version of Formula One, so this argument must be set to 0.

See Also

FormatFontDlg method

GetFont method

SetHdrSelection Method

Description

Sets whether the row and column headings are selected.

Syntax

F1Book1.**SetHdrSelection** *bTopLeftHdr*, *bRowHdr*, *bColHdr*

Part	Type	Description
bTopLeftHdr	Boolean	Sets the value of the top left header flag. If True, the cell at the intersection of the row and column headings is selected. If False, the cell is not selected.
bRowHdr	Boolean	Sets the value of the row header selection flag. If True, the row headings are selected. If False, the row headings are not selected.
bColHdr	Boolean	Sets the value of the column header selection flag. If True, the column headings are selected. If False, the column headings are not selected.

See Also

GetHdrSelection method

SetIteration Method

Description

Sets or returns iteration information.

Syntax

F1Book1.**SetIteration** *bIteration*, *nMaxIterations*, *nMaxChange*

Part	Type	Description
bIteration	Boolean	Indicates the state of the iteration flag. It True, iteration is enabled. If False, iteration is disabled.
nMaxIterations	Integer	Indicates the maximum number of iterations.
nMaxChange	Double	Indicates the maximum change value.

Remarks

Iteration can be used to solve circular references. The program calculates until it iterates the number of times specified by *nMaxIterations* or until all cells change by less than the amount specified in *nMaxChange*.

See Also

GetIteration method

SetLineStyle Method

Description

Returns the weight, color, and style for the selected line object or the line forming the border around the selected object.

Syntax

F1Book1.**SetLineStyle** *nStyle*, *crColor*, *nWeight*

Part	Type	Description
nStyle	Integer	Indicates the line style. Following are valid values for this argument:

Values	Description
0	Solid
1	Dashed
2	Ditted
3	Dash-ditted
4	Dash-dit-ditted
5	None

Part	Type	Description
crColor	OLE_COLOR	Indicates the line color.
nWeight	Integer	Indicates the line weight. Following are the valid values returned to this variable:

Values	Line weight
0	1/2 point (displayed as 1 point rule on low resolution monitors)
1	1 point
2	2 points
3	3 points

Remarks

Solid lines can assume any of the line weights; styled lines appear as solid lines when set larger than 1/2 point.

See Also

LineStyleDlg method

SetPattern Method

Description

Sets the pattern for all selected cells and objects in all selected sheets.

Syntax

F1Book1.**SetPattern** *nPattern*, *crFG*, *crBG*

Part	Type	Description
nPattern	Integer	Indicates the pattern number used to display the cells or objects. The pattern number can range from 0 (no pattern) to 18 and represents one of the 18 patterns.
crFG, *crBG*	OLE_COLOR	Indicate the foreground and background colors for the pattern.

See Also

FormatPatternDlg method

GetPattern method

SetPrintAreaFromSelection Method

Description

Sets the print range to the currently selected ranges.

Syntax

F1Book1.**SetPrintAreaFromSelection**

See Also

PrintArea property

SetPrintScale Method

Description

Sets the current print scale settings for the active worksheet.

Syntax

F1Book1.**SetPrintScale** *nScale*, *bFitToPage*, *nVPages*, *nHPages*

Part	Type	Description
nScale	Integer	Indicates the scale factor. Scale factor can range from 10 to 400.
bFitToPage	Boolean	Sets the fit to page flag. If the flag is False, the scale percentage returned in scale is used to print the workbook. If the flag is True, the workbook is printed on the number of vertical and horizontal pages returned by *vPages* and *hPages*.
nVPages	Long	Sets the number of vertical pages to which the print job it fit.
nHPages	Long	Sets the number of horizontal pages to which the print job is fit.

Remarks

Each print range is printed starting on a new page. If *bFitToPage* is true and multiple ranges are specified the scale is reduced until the largest of the print ranges fits within the specified number of pages.

To print a worksheet at the largest scale possible, set *nScale* to 400 when *bFitToPage* is True. Then, specify the number of pages on which you want the worksheet printed.

See Also

GetPrintScale method

PrintArea property

SetPrintTitlesFromSelection Method

Description

Specifies the current selection as print titles to be printed at the top or left of each page.

Syntax

F1Book1.**SetPrintTitlesFromSelection**

Remarks

SetPrintTitlesFromSelection sets the "Print_Titles" user-defined name to a formula referring to the current selection. You cannot select partial rows or columns to be used as print titles.

Print titles are row or column titles that are printed on each page. Row titles are printed at the top of each page; column titles are printed on the left of each page. The name defines the fixed columns and rows that are printed. If set to null (""), no titles are printed.

See Also

PrintTitles property

SetProtection Method

Description

Sets the protection status of all selected cells in all selected sheets.

Syntax

F1Book1.**SetProtection** *bLocked*, *bHidden*

Part	Type	Description
bLocked	Boolean	Sets the locked cell flag. If the locked cell flag is True, the active cell is locked.
bHidden	Boolean	Sets the hide formulas flag. If the hide formulas flag is True, the formulas are hidden (formula results are not hidden).

Remarks

After locking cells and hiding formulas, you must enable protection for the workbook before cell locking and formula hiding is enabled. Protection for a workbook is enabled using the **EnableProtection** property.

See Also

GetProtection method

ProtectionDlg method

SetRowHeight Method

Description

Sets the height for the specified range of rows in all selected sheets.

Syntax

F1Book1.**SetRowHeight** *nRow1*, *nRow2*, *nHeight*, *bDefRowHeight*

Part	Type	Description
nRow1, *nRow2*	Long	Indicate the first and last rows in a range of rows.
nHeight	Integer	Indicates the row height value in twips. A twip is 1/1440th of an inch.
bDefRowHeight	Boolean	Specifies whether the default row height is changed. True specifies that the default height is set to *nHeight*, and the specified rows are set to the default height. In addition, any rows that use the default height are updated with the new default. False specifies that the default height is unchanged.

See Also

ColWidth property

RowHeight property

SetRowHeightAuto Method

Description

Sets the height of the specified rows automatically.

Syntax

F1Book1.**SetRowHeightAuto** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *bSetDefaults*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Coordinates for the range containing the rows for which to set the height.
bSetDefaults	Boolean	Determines when the specified rows are resized. If True, all specified rows are adjusted automatically. If False, only rows in the specified row range that need to be larger than their current size are adjusted.

Remarks

SetRowHeightAuto specifies that the heights of the rows in the specified range are automatically set to display the tallest entry in the specified rows. The rows are set at least as tall as the default row height.

See Also

RowHeight property

RowHeightDlg method

SetRowHeight method

SetRowHidden Method

Description

Changes the display status of one or more rows.

Syntax

F1Book1.**SetRowHidden** *nRow1*, *nRow2*, *bRowHidden*

Part	Type	Description
nRow1	Long	Identifies the first row in a range.
nRow2	Long	Identifies the last row in a range.
bRowHidden	Boolean	Sets the status of the row hidden flag for all rows in the range identified by *nRow1* and *nRow2*. If the row hidden flag is true, the specified rows are not displayed. If the flag is False, the rows are displayed.

SetSelection Method

Description

Selects the specified range and moves the active cell to the top left cell in the range.

Syntax

F1Book1.**SetSelection** *nRow1*, *nCol1*, *nRow2*, *nCol2*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Identifies the row and column coordinates you want to select.

See Also

GetSelection method

ObjSetSelection method

SelectionCount property

SetTabbedText Method

Description

Places a block of tab-delimited text in a workbook.

Syntax

F1Book1.**SetTabbedText** *nStartRow*, *nStartCol*, *pRows*, *pCols*, *bValuesOnly*, *pText*

Part	Type	Description
nStartRow, *nStartCol*	Long	Identifies the starting row and column where the text block is placed.
pRows, *pCols*	Long	Variables, passed by reference, that receive the returned number of rows and columns contained in the text block.
bValuesOnly	Boolean	Determines if the text is placed as unformatted text (True) or as formatted text (False).
pText	String	Identifies the tab-delimited block of text. .

See Also

Clip property

ClipValues property

GetTabbedText method

Text property

TextRC property

Text property

SetValidationRule Method

Description

Sets the validation rule for the currently selected range of cells.

Syntax

F1Book1.**SetValidationRule** *pRule*, *pText*

Part	Type	Description
pRule	String	Formula used to test the entered value.
pText	String	Text to display in cell if validation fails.

Remarks

Validation rules can be used to validate data entered in a cell or a range of cells. A validation rule consists of a formula to test, and text to display if the validation fails. If the formula returns True, the value is entered. If the formula returns a text string, the string is displayed and the value is not entered. If the formula returns False, the value is not entered and the validation text is displayed.

You can use relative references in validation rules. These references are considered to be relative to the active cell. This allows a validation rule to be properly applied to an entire range.

See Also

GetValidationRule method

ValidationFailed event

ValidationRuleDlg method

Sheet Property

Description

Sets or returns the active worksheet.

Syntax

F1Book1.**Sheet** [= *sheet*]

Part	Type	Description
sheet	Long	Index number that identifies an existing worksheet within the workbook attached to the view. Worksheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.

Remarks

Set **Sheet** to make a specific worksheet current. This property returns the index number of the currently active worksheet.

See Also

NumSheets property

SheetName property

Example

The following example resets sheet names after inserting new sheets:

```
Dim i As Integer
For i = 1 to F1Book1.NumSheets
    F1Book1.SheetName ( i ) = "Sheet"& i
Next i
```

SheetName Property

Description

Assigns a name to a worksheet or returns the current name of the specified worksheet.

Syntax

F1Book1.**SheetName** (*nSheet*) [= *sheetName*]

Part	Type	Description
nSheet	Long	Index number that identifies an existing worksheet within the workbook attached to the view. Worksheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.
sheetName	String	Name associated with *nSheet*.

Remarks

Set **SheetName** to give the specified worksheet a name, or change its existing name. This property returns the name of the specified worksheet.

When you change a sheet name, formulas that reference the worksheet are updated to reference the new sheet name.

See Also

NumSheets property

Sheet property

SheetSelected Property

Description

Sets or returns the selection status of a worksheet.

Syntax

F1Book1.**SheetSelected** (*nSheet*) [= *boolean*]

Part	Type	Description
nSheet	Long	Index number that identifies an existing worksheet within the workbook attached to the view. Workheets are indexed from left to right beginning with 1. Do not confuse the index with the sheet name that appears on the sheet tab.

Remarks

Set **SheetSelected** to True to select a worksheet or False to deselect the worksheet.

See Also

NumSheets property

Sheet property

ShowActiveCell Method

Description

Positions the view to show the active cell if it is not currently displayed in the window.

Syntax

F1Book1.**ShowActiveCell**

Remarks

If the active cell is not displayed in the view window the active worksheet is scrolled until the active cell becomes visible.

See Also

GetActiveCell method

Row property

Col property

SetActiveCell method

ShowColHeading Property

Description

Sets or returns whether column headings are displayed.

Syntax

F1Book1.**ShowColHeading** [= *boolean*]

Remarks

If this property is True, column headings are displayed.

See Also

PrintColHeading property

ShowEditBar Property

Description

Sets or returns whether the edit bar is displayed.

Syntax

F1Book1.**ShowEditBar** [= *boolean*]

Remarks

If this property is True, the edit bar is displayed with the workbook.

Note The edit bar does not appear until the container makes it UI active (provides a window for it.) For example, scroll bars do not appear in Visual Basic design mode, but they do in edit mode and run mode.

See Also

ShowEditBarCellRef property

ShowEditBarCellRef Property

Description

Sets or returns whether the reference of the active cell appears with the edit bar.

Syntax

F1Book1.**ShowEditBarCellRef** [= *boolean*]

Remarks

If this property is True, and the **ShowEditBar** property is True, the cell reference of the active cell in the active worksheet is displayed with the edit bar. If **ShowEditBar** is False, **ShowEditBarCellRef** has no effect.

ShowFormulas Property

Description

Sets or returns whether formulas are displayed in place of cell values.

Syntax

F1Book1.**ShowFormulas** [= *boolean*]

If this property is True, formula text is displayed in cells instead of the values formulas produce. In order to accomplish this, column widths are shown at twice their actual size.

ShowGridLines Property

Description

Sets or returns whether grid lines are displayed.

Syntax

F1Book1.**ShowGridLines** [= *boolean*]

Remarks

If this property is True, grid lines are displayed.

ShowHScrollBar Property

Description

Sets or returns how the horizontal scroll bar is displayed.

Syntax

F1Book1.**ShowHScrollBar** [= *setting*]

Part	Type	Description
setting	Index	Controls the appearance and behavior of horizontal scroll bars. Following are the valid F1ShowOffOnAutoConstants:

Constant	Description
F1Off	The horizontal scroll bar is hidden.
F1On	The horizontal scroll bar is displayed.
F1Automatic	The horizontal scroll bar is displayed if the workbook is wider than the window and the workbook is active.

Remarks

The scroll bars do not appear on the control until the container makes them UI active (provides a window for them.) For example, scroll bars do not appear in Visual Basic design mode, but they do in edit mode and run mode.

See Also

ShowVScrollBar property

ShowRowHeading Property

Description

Sets or returns whether row headings are displayed.

Syntax

F1Book1.**ShowRowHeading** [= *boolean*]

Remarks

If the flag is True, row headings are displayed.

See Also

PrintRowHeading property

ShowSelections Property

Description

Sets or returns whether selections are displayed.

Syntax

F1Book1.**ShowSelections** [= *setting*]

Part	Type	Description
setting	Index	Controls whether selections are displayed on the workbook. Following are the valid F1ShowOffOnAutoConstants:

Constant	Description
F1Off	Do not display selections, and user cannot interactively make selections.
F1On	Display all selections at all times
F1Auto	Display selections in this control only when the control is active.

See Also

AllowSelections property

ShowTabs Property

Description

Sets or returns the display status and position of the sheet name tabs on a workbook.

Syntax

*F1Book1.***ShowTabs** [= *setting*]

Part	Type	Description
setting	Integer	Controls the display of the sheet name tabs. Following are the valid F1ShowTabsConstants

Constant	Description
F1TabsOff	Tabs are not displayed
F1TabsBottom	Tabs are displayed along the bottom of the workbook.
F1TabsTop	Tabs are displayed along the top of the workbook.

ShowTypeMarkers Property

Description

Sets or returns whether frames are displayed around cells to identify the cell types.

Syntax

*F1Book1.***ShowTypeMarkers** [= *boolean*]

Remarks

If this property is True, the following frame types are used to identify different types of cells:

Cell Type	Frame Marker
Empty cell	None
Blank formatted cell	Blue frame
Value cell (holds a number or text)	Green frame
Formula cell	Red frame

By default, type markers are not displayed.

ShowVScrollBar Property

Description

Sets or returns how the vertical scroll bar is displayed.

Syntax

F1Book1.**ShowVScrollBar** [= *setting*]

Part	Type	Description
setting	Index	Controls the appearance and behavior of vertical scroll bars. Following are the valid F1ShowOffOnAutoConstants:

Constant	Description
F1Off	The vertical scroll bar is hidden.
F1On	The vertical scroll bar is displayed.
F1Automatic	The vertical scroll bar is displayed if the workbook is taller than the window and the workbook is active.

Remarks

The scroll bars do not appear on the control until the container makes them UI active (provides a window for them.) For example, scroll bars do not appear in Visual Basic design mode, but they do in edit mode and run mode.

See Also

ShowHScrollBar property

ShowZeroValues Property

Description

Set or returns whether zero value cells are displayed.

Syntax

F1Book1.**ShowZeroValues** [= *boolean*]

Remarks

If this property is True, cells with zero values are displayed. If False, zero value cells are displayed as blanks.

Sort Method

Description

Specifies a range of data to be sorted and the keys by which to sort the data.

Syntax

F1Book1.**Sort** *nR1 nC1 nR2 nC2 bSortByRow Keys*

Part	Type	Description
nR1, *nC1*, *nR2*, *nC2*	Long	Coordinates of the range of data to be sorted.
bSortByRow	Boolean	Specifies how data is sorted. If True, data is sorted by rows; if False, data is sorted by columns.
Keys	Variant	Identifies the key or keys to use to sort the data. This argument can be a single integer or an array of integers.

Remarks

If the data is sorted by rows, each row of data in the specified range is considered a record and sorted together. If data is sorted by columns, each column in the specified range is considered a record.

When defining sort keys, specify the number of the row or column in the selected range that is to serve as a key. Use a positive number to define an ascending key; use a negative number to define a descending key.

See Also

Sort3 method

SortDlg method

Sort3 Method

Description

Specifies a range of data to be sorted and as many as three keys by which to sort the data.

Syntax

F1Book1.**Sort3** *nRow1*, *nCol1*, *nRow2*, *nCol2*, *bSortByRows*, *nKey1*, *nKey2*, *nKey3*

Part	Type	Description
nRow1, *nCol1*, *nRow2*, *nCol2*	Long	Specify the range of data to be sorted.
bSortByRows	Boolean	Specifies how data is sorted. If True, data is sorted by rows; if False, data is sorted by columns.
nKey1, *nKey2*, *nKey3*	Long	Specify the sort keys. If the data is sorted by rows, columns are specified as sort keys; rows are specified as sort keys if the data is sorted by columns. *nKey1* is the primary key, *nKey2* is the secondary key, and *nKey3* is the last sort key.

Remarks

If the data is sorted by rows, each row of data in the specified range is considered a record and sorted together. If data is sorted by columns, each column in the specified range is considered a record.

When defining sort keys, specify the number of the row or column in the selected range that is to serve as a key. Use a positive number to define an ascending key; use a negative number to define a descending key.

For example, to specify the second column in the selected range as a primary descending key, enter -2 for *nKey1*.

To sort on one key, *nKey2* must be zero. To sort on two keys, *nKey3* must be zero.

See Also

Sort method

SortDlg method

SortDlg Method

Description

Displays the Sort dialog box.

Syntax

F1Book1.**SortDlg**

Remarks

The Sort dialog box allows you to sort the data in the currently selected range. The dialog box allows you to specify sort keys, whether those sort keys are ascending or descending, and whether data is sorted by rows or columns.

See Also

Sort method

Sort3 method

SS Method

Description

Returns the handle to the workbook associated with the current view control.

Syntax

handle = *F1Book1.***SS**

Part	Type	Description
handle	Long	Variable that receives the returned workbook handle.

SSUpdate Method

Description

Updates all workbooks.

Syntax

*F1Book1.***SSUpdate**

Remarks

SSUpdate updates everything that might be delayed. This includes recalculating any workbooks with the **AutoRecalc** property set to True, updating the scroll bar position, and firing a Modified event if needed. An update happens automatically when the system is idle. You should not normally need to call this method.

SSVersion Method

Description

Returns the version number of the Formula One control.

Syntax

version = *F1Book1*.**SSVersion**

Part	Type	Description
version	Integer	Variable that receives the returned version number. The high byte is the major release number and the low byte is the minor number.

StartEdit Method

Description

Begins edit mode for the active cell.

Syntax

F1Book1.**StartEdit** *bClear*, *bInCellEditFocus*, *bArrowsExitEditMode*

Part	Type	Description
bClear	Boolean	Sets the clear edit bar flag. If True, the edit bar is cleared as edit mode commences.
bInCellEditFocus	Boolean	Sets the in cell edit flag. If True, editing focus is given to in-cell editing; if False, editing focus is given to the edit bar.
bArrowsExitEditMode	Boolean	Sets the arrows exit edit mode flag. If True, edit mode is exited if you press an arrow key on the keyboard.

Remarks

StartEdit starts edit mode for the active cell and allows you to specify how the cell is edited.

See Also

CancelEdit method

EndEdit method

SwapTables Method

Description

Exchanges the workbooks attached to two views.

Syntax

F1Book1.**SwapTables** *hSS2*

Part	Type	Description
hSS2	Long	Handle to a view whose workbook you want to swap with the current view.

Remarks

When the workbooks are exchanged, the view information is not swapped - only the workbooks are swapped. The following properties are not swapped when SwapTables is called: **Tablename**, all **Do** properties, and **FileName**. If you want to swap both workbooks and view information between two views, then you must swap these properties progammatically.

See Also

Attach method

AttachToSS method

TableName Property

Description

Sets or returns the name assigned to the workbook associated with the current view control.

Syntax

F1Book1.**TableName** [= *name*]

Part	Type	Description
name	String	Worksheet *name*.

Remarks

A number of methods, such as **Attach**, reference a workbook by its name instead of its handle. Set this property to change the workbook's name, and get the value of this property to determine the workbook's current name.

This is the name used to refer to the workbook in external references. For example, if you set **TableName** to MySheet, you can refer to Sheet1!A1 in that workbook by entering the formula [MySheet]Sheet1!A1 in another workbook.

TableName and **Title** are the same thing. Changing one property changes the other.

Text Property

Description

Enters text in the active cell of all selected sheets or returns the text value of the active cell in the active worksheet.

Syntax

*F1Book1.***Text** [= *text*]

Part	Type	Description
text	String	Text value.

Remarks

Set **Text** to place text into the active cell. If the active cell contains a formula, the formula is deleted when the *text* is placed in the cell.

Text returns the current text value of the active cell. If the cell contains a formula, the variable contains the result of the formula.

See Also

Entry property

Formula property

Number property

TextRC property

TextRC Property

Description

Enters text in the specified cell of all selected sheets or returns the text value of the specified cell.

Syntax

F1Book1.**TextRC** (*nRow*, *nCol*) [= *text*]

Part	Type	Description
nRow, *nCol*	Long	Row and column numbers of the cell that identify the specific cell.
text	String	Text value.

Remarks

Set **TextRC** to place *text* into the active cell. If the active cell contains a formula, the formula is deleted when the text is placed in the cell.

TextRC returns the current text value of the active cell. If the cell contains a formula, the variable contains the result of the formula.

See Also

EntryRC property

FormulaRC property

NumberRC property

Text property

Title Property

Description

Sets or returns the title of the workbook.

Syntax

*F1Book1.***Title** [= *title*]

Part	Type	Description
title	String	Title text. A title can be of any length.

Remarks

The title of a workbook can be used in external references to access multiple workbooks.

This is the name used to refer to the workbook in external references. For example, if you set **Title** to MySheet, you can refer to Sheet1!A1 in that workbook by entering the formula [MySheet]Sheet1!A1 in another workbook.

TableName and **Title** are the same thing. Changing one property changes the other.

See Also

Attach method

TopLeftText Property

Description

Sets or returns the text displayed at the intersection of the row and column headings in the top left corner. Setting this property affects all selected sheets.

Syntax

*F1Book1.***TopLeftText** [= *text*]

Part	Type	Description
text	String	Text displayed in the row and column heading intersection.

Remarks

The text placed in the top left corner of the spreadsheet is used for display purposes only. This text can be up to 254 characters long.

See Also

ColText property

RowText property

TopRow Property

Description

Sets or returns the top row displayed in the active worksheet.

Syntax

*F1Book1.***TopRow**[= *row*]

Part	Type	Description
row	Long	Number of first *row* displayed in the view.

Remarks

Set **TopRow** to change the first *row* displayed in the view. **TopRow** returns the number of the first row currently displayed in the view.

See Also

LeftCol property

TransactCommit Method

Description

Commits changes made during a transaction.

Syntax

*F1Book1.***TransactCommit**

Remarks

Transactions allow the performance of multiple operations with the ability to undo changes if all operations do not succeed. Every operation between the start of a transaction and the end of a transaction can be undone by calling TransactRollback. If all operations succeed, TransactCommit should be called to make the changes permanent and release resources associated with the transaction.

Every **TransactStart** method call must have a matching **TransactCommit** or **TransactRollback** method call.

TransactRollback Method

Description

Undoes all changes made to a table since the last TransactStart method was called.

Syntax

*F1Book1.***TransactRollback**

Remarks

Transactions allow the performance of multiple operations with the ability to undo changes if all operations do not succeed. Every operation between the start of a transaction and the end of a transaction can be undone by calling **TransactRollback**. If all operations succeed, **TransactCommit** method should be called to make the changes permanent and release resources associated with the transaction.

Every **TransactStart** method call must have a matching **TransactCommit** or **TransactRollback** call.

TransactStart Method

Description

Starts a transaction.

Syntax

*F1Book1.***TransactStart**

Remarks

Transactions allow the performance of multiple operations with the ability to undo changes if all operations do not succeed. Every operation between the start of a transaction and the end of a transaction can be undone by calling **TransactRollback** method. If all operations succeed, **TransactCommit** method should be called to make the changes permanent and release resources associated with the transaction.

Every **TransactStart** call must have a matching **TransactCommit** or **TransactRollback** call.

Note TransactStart makes a copy of the entire workbook and should probably not be used on very large workbooks.

TwipsToRC Method

Description

Converts a point in a worksheet, as specified by a set of coordinates, to the corresponding row and column at which the point is located.

Syntax

F1Book1.**TwipsToRC** *x*, *y*, *pRow*, *pCol*

Part	Type	Description
x	Long	Horizontal coordinate of the point to convert.
y	Long	Vertical coordinate of the point to convert.
pRow	Long	Variable, passed by reference, that receives the returned row number.
pCol	Long	Variable, passed by reference, that receives the returned column reference.

Remarks

The coordinates specified by this method are measured in twips from the upper left corner of the worksheet control. **TwipsToRC** can determine the row and column that corresponds to a point returned by the **DragDrop** and **DragOver** events.

pRow and *pCol* return 0 if the referenced point is located in a row or column heading.

See Also

RangeToTwips method

Type Property

Description

Returns the cell type of the active cell.

Syntax

[*pType* =] *F1Book1.***Type**

Part	Type	Description
pType	Integer	Variable, passed by reference, that receives the returned cell type. Following are the valid types that can be returned to this variable:

Value	Cell Type
-0	Empty
-1	Number
-1	Formula returning number
-2	Text
-2	Formula returning text
-3	Logical
-3	Formula returning logical
-4	Error
-4	Formula returning error

See Also

Entry property

Formula property

Number property

Text property

TypeRC property

TypeRC Property

Description

Returns the cell type of the specified cell.

Syntax

[*pType* =] *F1Book1.***TypeRC** (*nRow*, *nCol*)

Part	Type	Description
nRow, *nCol*	Long	Row and column number of the cell for which the type is returned.
pType	Integer	Variable, passed by reference, that receives the returned cell type. Following are the valid types that can be returned to this variable:

Value	Cell Type
-0	Empty
-1	Number
-1	Formula returning number
-2	Text
-2	Formula returning text
-3	Logical
-3	Formula returning logical
-4	Error
-4	Formula returning error

See Also

EntryRC property

FormulaRC property

NumberRC property

TextRC property

Type property

ValidationRuleDlg Method

Description

Displays the Validation Rule dialog box.

Syntax

*F1Book1.***ValidationRuleDlg**

Remarks

You can use the Validation Rule dialog box to set the validation rule for the current selection and apply it to all selected worksheets. Validation rules are used to validate data entered in a cell. A validation rule consists of a formula to test, and text to display if the validation fails.

See Also

GetValidationRule method

SetValidationRule method

ViewScale Property

Description

Sets or returns the current display scale for a workbook.

Syntax

*F1Book1.***ViewScale** [= *scale*]

Part	Type	Description
scale	Integer	Indicates the current display scale, ranging from 10 to 400 percent. 100 percent is normal display scale.

Visible Property

Description

Sets or returns whether the Formula One object is visible.

Syntax

*F1Book1.***Visible** [= *boolean*]

Write Method

Description

Saves the workbook to the specified file.

Syntax

F1Book1.**Write** *pPathName*, *nSaveType*

Part	Type	Description
pPathName	String	Name of the file to write. The name can include drive, path, and filename.
nSaveType	Index	Type of file to write. Following are the valid F1FileTypeConstants:

Constants	Description
F1FileFormulaOne	Formula One format
F1FileExcel4	Excel 4.0 format
F1FileTabbedText	Tab-delimited text file
F1FileExcel5	Excel 5.0 format
F1FileFormulaOne3	Formula One v.3 format
F1FileTabbedTextValuesOnly	Tab-delimited text file values only

See Also

Read method

SaveWindowInfo method

WriteToBlob method

Example

The following example displays the Save File dialog box, records the users selections, and writes the file to disk using this information:

```
Sub SaveFile ( )
    Dim fName As String, fType As Integer
    OnError Goto FileSaveError
    F1Book1.SaveFileDlg " ", fName, fType
    F1Book1.Write fname, fType
    Exit Sub
FileSaveError;
    If Err < > F1ErrorCancel then
    MsgBox "Formula One Error # & Err
End Sub
```

WriteToBlob Method

Description

Writes a worksheet to a blob variable.

Syntax

F1Book1.**WriteToBlob** *phBlob*, *nReservedBytes*

Part	Type	Description
phBlob	OLE_HANDLE	Reference to a blob variable in memory.
nResevedBytes	Integer	Size of the blob variable. Not implemented in this version and must be 0.

Remarks

Once a workbook is written to a blob, it can be stored in a database, or passed to **ReadFromBlob**.

See Also

Write method

Windows API

- GlobalAlloc
- GlobalFree
- GlobalLock
- GlobalUnlock

Using Events

Formula One provides a full range of events that allow you to track and monitor actions performed on a spreadsheet view control by users of your application. Events allow you to respond to user's actions and control the operations of the worksheet control.

Drilling for Data

Events and Other Controls

Capturing Mouse Movements

Drilling for Data

Many applications consist of summary forms backed up by detail forms. For example, you may have a sales management application that reports your company's sales by region. The summary screen shows the total sales for each region. Other worksheets show the various regions and their sales breakdowns. If the user double clicks one of the summary region columns, a second worksheet is displayed that shows the sales breakdown in that region.

This type of operation is referred to as drilling. Drilling is generally defined as the ability to see greater levels of detail by double clicking a worksheet. The area clicked defines the additional information that is displayed.

Drilling can be implemented using the DblClick event. The following example demonstrates how to catch the event.

```
Private Sub F1Book1_DblClick (nRow As Long, nCol As Long)
    If nCol = 3 Then
        F1Book1.Sheet = 3
    End If
End Sub
```

Formula One is adept at handling this type of model since it can handle multiple workbooks, worksheets, and supports formulas with external references.

Events and Other Controls

If is often necessary to use one of the Visual Basic controls during data entry. For example, you might use a pop up list box to display a list of items that can be entered in a worksheet. This is easily accomplished by designing a form with a list box and displaying it when the user clicks a cell.

The following example creates and fills a Visual Basic list box; then, it displays the list box when the user right clicks anywhere in the worksheet.

```
' Initialize list box items to available fonts and make it invisible.
Private Sub Form_Load ()
    Dim i As Integer
    List1.Visible = False
    For i = 0 To Screen.FontCount - 1
        List1.AddItem Screen.Fonts(i)
    Next i
End Sub
' Move the listbox to the active cell and make it visible.
Private Sub F1Book1_RClick (nRow As Long, nCol As Long)
    Dim ssError As Integer, shown As Integer
    Dim xOffset As Integer, yOffSet As Integer, thisWidth As Integer, thisHeight As Integer
    ' Get the cell clicked on in twips.
    F1Book1.RangeToTwips nRow, nCol, nRow, nCol, xOffSet, yOffSet, thisWidth, thisHeight,
                shown
    ' Set position and size of the listbox, move to click location and  make it visible
    List1.Move (F1Book1.Left + xOffset), (F1Book1.Top + yOffSet), (F1Book1.Left + 2 *
                thisWidth), (F1Book1.Top + 5 * thisHeight)
    List1.Visible = True
End Sub
' Set the current selection to the specified font. Default height  is 10 and the other
        attributes are not set.
Sub List1_Click ()
    F1Book1.SetFont List1.List(List1.ListIndex), 10, False, False, False, False, False,
                False, False)
    ' Make listbox invisible now that we are done with it.
    List1.Visible = False
End Sub
```

Other controls can be displayed in the same manner, including combo boxes and menus. In addition, many custom controls that are built in Visual Basic can be used in this manner.

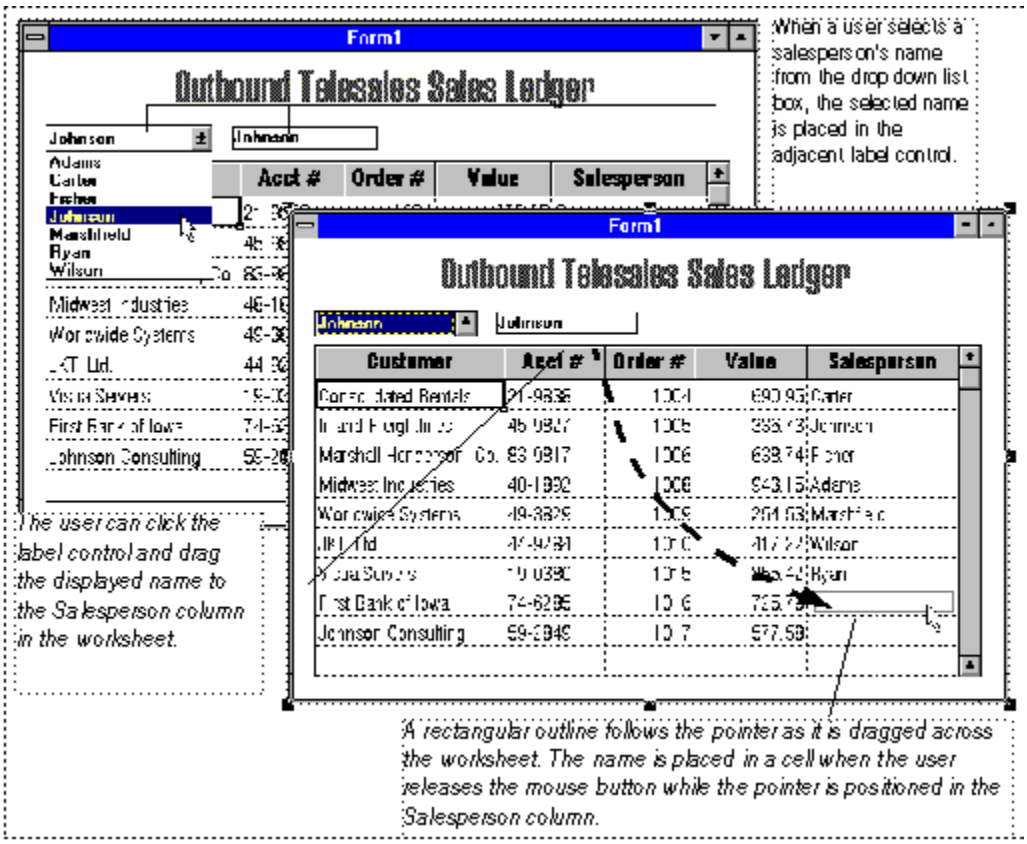
Capturing Mouse Movements

The **DragOver** and **DragDrop** events capture mouse movements within a worksheet and return locations within the worksheet as X and Y coordinates. The events can be used with methods that require coordinates in twips, such as the **TwipsToRC** method.

The following example uses the **TwipsToRC** in a **DragDrop** event. The method uses the X and Y coordinates returned by the event and converts the points to a row and column location in the worksheet. If the location returned by the event is located in column 5 of the worksheet, the **TextRC** property is called and places the caption contained in the Label2 control in the worksheet cell returned by TwipsToRC.

```
Private Sub F1Book1_DragDrop (Source As Control, X As Single, Y As Single)
        Dim row as long, col as long
        If strCompare ("Label2", Source, Name) =0 then
             F1Book1.TwipsToRC X,Y, row, col
                If col=5 then
             F1Book1.TextRC (row, col) = Label2.Caption
        End If
    End If
End Sub
```

The following illustration shows a form that employs the preceding example code. The code is attached to the Formula One control.



CancelEdit Event

Description

This event occurs when the user leaves edit mode without making changes or presses the Escape key.

Syntax

Private Sub F1Book1_**CancelEdit** ( )

Click Event

Description

The **Click** event occurs when the user presses and releases the left mouse button while the pointer is in the Formula One window and the window has focus.

Syntax

Private Sub F1Book1_**Click** (ByVal *nRow* as Long, ByVal *nCol* as Long)

Part	Description
nRow, *nCol*	Coordinates that identify the cell in which the user clicks. If a click occurs on a row heading, *nRow* is 0; *nCol* is 0 if the click occurs on a column heading. If a scroll bar is clicked, the event does not fire.

DblClick Event

Description

The **DblClick** event occurs when the user double clicks the left mouse button while the pointer is in the Formula One window and the window has focus.

Syntax

Private Sub F1Book1_**DblClick** (ByVal *nRow* as Long, ByVal *nCol* as Long)

Part	Description
nRow , *nCol*	Coordinates that identify the cell in which the user double clicks. If a double click occurs on a row heading, *nRow* is 0; *nCol* is 0 if the double click occurs on a column heading. If a scroll bar is double clicked, the event does not fire.

DragDrop Event

Description

This event occurs when a drag-drop operation is completed.

Syntax

Private Sub F1Book1_**DragDrop** (*Source* As Control, *x* As Single, *y* As Single)

DragOver Event

Description

This event occurs when a drag-drop operation is in process.

Syntax

Private Sub F1Book1_**DragOver** (*Source* As Control, *x* As Single, *y* As Single, *State* As Integer)

EndEdit Event

Description

This event occurs when an editing operation is completed.

Syntax

Private Sub F1Book1_**EndEdit** (*EditString* As String, *Cancel* As Integer)

Part	Description
EditString	Edited text to be entered in the active cell. This value can be changed to modify what is placed in the cell.
Cancel	Set to True to force edit mode to continue. This is often used for data validation when you do not want the user to exit edit mode until data is correct

EndRecalc Event

Description

This event occurs when the recalculation process is completed.

Syntax

Private Sub F1Book1_**EndRecalc** ( )

GotFocus Event

Description

The **GotFocus** event occurs when the Formula One window receives focus, either by clicking the object or changing the focus in code using the SetFocus method.

Syntax

Private Sub F1Book1_**GotFocus** ( )

KeyDown, KeyUp Events

Description

These events occur when the user presses (KeyDown) and releases (KeyUp) a key while the Formula One object has the focus.

Syntax

Private Sub F1Book1_**KeyDown** (*KeyCode* As Integer, *Shift* As Integer)

Private Sub F1Book1_**KeyUp** (*KeyCode* As Integer, *Shift* As Integer)

KeyPress Event

Description

Occurs when the user presses and releases a key that triggers an ANSI character.

Syntax

Private Sub F1Book1_**KeyPress** (*KeyAscii* As Integer)

LostFocusEvent

Description

The **LostFocus** event occurs when the Formula One window loses focus, either by clicking the object or changing the focus in code using the **SetFocus** method.

Syntax

Private Sub F1Book1_**LostFocus** ( )

Modified Event

Description

Occurs when a change is made in the workbook.

Syntax

Private Sub F1Book1_**Modified** ( )

Remarks

When a new workbook is created it's **Modified** property is set to FALSE. When something is changed the **Modified** property is set to TRUE and the **Modified** event is fired. Further changes do not cause the **Modified** event to fire until the **Modified** property is set to FALSE.

To cause the **Modified** event to be fired every time a change is made, set the **Modified** property to False within the **Modified** event.

MouseDown, MouseUp Events

Description

Occurs when the user presses a mouse button (MouseDown) or releases a mouse button (MouseUp.).

Syntax

Private Sub F1Book1_**MouseDown** (*Button* As Integer, *Shift* As Integer, *x* As Single, *y* As Single)

Private Sub F1Book1_**MouseUp** (*Button* As Integer, *Shift* As Integer, *x* As Single, *y* As Single)

Part	Description
Button	Identifies the pressed or released button that caused the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when button is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x, *y*	Coordinates of the current mouse location.

Remarks

Use the bitwise AND operator to check bits.

MouseMove Event

Description

Occurs when the user moves the mouse.

Syntax

Private Sub F1Book1_**MouseMove** (*Button* As Integer, *Shift* As Integer, *x* As Single, *y* As Single)

Part	Description
Button	Identifies whether any buttons where down when the mouse moved. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when *button* is pressed or released. A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x, y	Coordinates of the current mouse location.

Remarks

Use the bitwise AND operator to check bits.

ObjClick Event

Description

This event occurs when an object is clicked.

Syntax

Private Sub F1Book1_**ObjClick** (*ObjName* As String, *ObjID* As Long)

Part	Description
ObjName	Name of the clicked object.
ObjID	Identification number of the clicked object.

Remarks

An object must be named to receive an **ObjClick** event.

ObjDblClick Event

Description

This event occurs when an object is double clicked.

Syntax

Private Sub F1Book1_**ObjDblClick** (*ObjName* As String, By Val *ObjID* As Long)

Part	Description
ObjName	Name of the double-clicked object.
ObjID	Identification number of the double-clicked object.

Remarks

An object must be named to receive an **ObjDblClick** event.

ObjGotFocus Event

Description

This event occurs when a window based object such as a button, check box or list box gets the focus.

Syntax

Private Sub F1Book1_**ObjGotFocus** (*ObjName* As String, ByVal *ObjID* As Long)

Part	Description
ObjName	Name of the object that receives focus.
ObjID	Identification number of the object that receives focus.

Remarks

An object must be named to receive an **ObjGotFocus** event.

ObjLostFocus Event

Description

This event occurs when a window based object such as a button, check box, or list box loses the focus.

Syntax

Private Sub F1Book1_**ObjLostFocus** (*ObjName* As String, ByVal *ObjID* As Long)

Part	Description
ObjName	Name of the object that loses focus.
ObjID	Identification number of the object that loses focus.

Remarks

An object must be named to receive an **ObjLostFocus** event.

ObjValueChanged Event

Description

This event occurs when the value of a check box or list box changes.

Syntax

Private Sub F1Book1_**ObjValueChanged** (*ObjName* As String, ByVal *ObjID* As Long)

Part	Description
ObjName	Name of the object whose value changes.
ObjID	Identification number of the object whose value changes.

Remarks

An object must be named to receive an **ObjValueChanged** event.

RClick Event

Description

The **RClick** event occurs when the user presses and releases the right mouse button while the pointer is in the Formula One window and the window has focus.

Syntax

Private Sub F1Book1_**RClick** (ByVal *nRow* as Long, ByVal *nCol* as Long)

Part	Description
nRow, *nCol*	Coordinates that specify the cell in which the user right clicks. If a click occurs on a row heading, *nRow* is 0; *nCol* is 0 if the click occurs on a column heading. If a scroll bar is clicked, the event does not fire

RDblClick Event

Description

The **RDblClick** event occurs when the user double clicks the right mouse button while the pointer is in the Formula One window and the window has focus.

Syntax

Private Sub F1Book1_**RDblClick** (ByVal *nRow* as Long, ByVal *nCol* as Long)

Part	Description
nRow, *nCol*	Coordinates that specify the cell in which the user double clicks. If a double click occurs on a row heading, nRow is 0; nCol is 0 if the double click occurs on a column heading. If a scroll bar is double clicked, the event does not fire.

SelChange Event

Description

This event occurs when the active cell is changed or the current selection is changed.

Syntax

Private Sub F1Book1_**SelChange** ( )

Remarks

Use caution when performing actions that change the row and column selection (e.g., using the **Row** or **Col** properties) within this event. Recursive actions could overflow the stack.

The following example would not cause a problem because it would only occur once.

```
If F1Book1.SelStartRow < 10 then
    F1Book1.SelStartRow = 10
End if
```

However, the following example would not be appropriate in a **SelChange** Event because it would keep recursing.

```
F1Book1.SelStartRow = F1Book1.SelStartRow.1
```

StartEdit Event

Description

This event occurs when an editing operation is started.

Syntax

Private Sub F1Book1_**StartEdit** (*EditString* As String, *Cancel* As Integer)

Part	Description
EditString	Text to be edited.
Cancel	Can be set to True to cancel edit mode. In this case, edit mode is not entered.

StartRecalc Event

Description

This event occurs when the recalculation process is started.

Syntax

Private Sub F1Book1_**StartRecalc** ( )

TopLeftChanged Event

Description

This event occurs when the top left edge of any cell changes position. This event is fired every time any column or row is resized, a column or row is hidden or displayed, or the worksheet is scrolled in any direction. This event is also fired when a worksheet is first created. The execution of this event is deferred until the system is idle.

Syntax

Private Sub F1Book1_**TopLeftChanged** ( )

ValidationFailed Event

Description

Occurs when a user attempts to enter data into a cell and it fails to pass the validation test.

Syntax

Private Sub F1Book1_**ValidationFailed** (*pEntry* As String, ByVal *nSheet* As Long, ByVal *nRow* As Long, ByVal *nCol* As Long, *pShowMessage* As String, *pAction* As Integer)

Part	Description
pEntry	Entry the user attempted to make.
nSheet	Identifies the worksheet the user tried to enter data into.
nRow, *nCol*	Row and column numbers that identify the cell the user tried to enter data into.
pShowMessage	The message the user receives when the validation fails.
pAction	Action Formula One takes when the validation fails. Following are the valid settings for pAction:

Setting	Description
0	Return F1ErrorValidationFailed. This is the default action.
1	Display *pShowMessage* in an error dialog box and return F1ErrorValidationFailed.
2	Enter the value without checking validation.
3	Retry entering with validation. Another ValidationFailed event is fired if it still fails.

Remarks

You can change the value of *pEntry* within this event. The changed value is used if *pAction* is set to 3. Setting *pAction* to 3 without changing *pEntry* would be pointless, because the validation will still fail.

Performance Tuning

The following tips can help you make the most efficient use of memory and get the best performance from Formula One.

- **Avoid formatting blank cells.** It is more efficient to format an entire row or column because no cells are created. When you format a blank range, that does not consist of whole rows or whole columns, Formula One must create empty cells before it can apply the format.
- **Build worksheets by rows instead of columns.** Formula One allocates memory by rows. You can save memory by building tables a row at a time, rather than a column at a time. For example, fill cells in row 1 before moving to row 2, and so on, rather than filling cells in column A before moving to column B, and so on.
- **Build ranges from the lower right corner.** When building a table one cell at a time from code, it is faster and more efficient to start in the lower right corner of the area in which you are working. This ensures that the row pointers are allocated simultaneously instead of one at a time. Likewise, each row is allocated once instead of being reallocated as each cell is added.
- **Use values instead of formulas whenever possible.**
- **Avoid adding empty rows and columns for white space.** Adjust the row height or column width to create white space instead of adding empty rows or columns. If you must have additional white space on your worksheet, empty rows are more efficient than empty columns.
- **Disable repainting when performing a series of operations.** When performing a number of sequential operations on a worksheet, disable repainting with **Repaint** so the screen does not repaint after each operation. This increases the speed of the operation and avoids unnecessary screen flashing.
- **Use methods to copy and move data.** Use **EditCopyRight**, **EditCopyDown**, **CopyRange**, **CopyRangeEx**, and **MoveRange** to copy and move cells. These methods are much faster than using the clipboard. In addition, these methods update cell references to maintain the integrity of your formulas.

Specifications

The following table lists the technical specifications for the Formula One control.

Specifications	
Maximum worksheet size	16,384 Rows by 256 Columns
Column width	0 to 255 characters
Row height	0 to 409 points
Text length	255 characters
Formula length	1024 characters
Number precision	15 digits
Largest positive number	9.99999999999999E307
Largest negative number	-9.99999999999999E307
Smallest positive number	1E-307
Smallest negative number	-1E-307
Maximum number of iterations	32,767
Maximum number of colors	36
Maximum number of available colors	Limited by display card and monitor
Maximum number of fonts per sheet	256
Maximum number of selected ranges	2048
Maximum number of names per sheet	16384
Maximum length of name	255
Maximum number of function arguments	30
Maximum length of format string	255
Maximum number of tables	Limited by system resources (Windows and memory)
Excel file format version	Excel 4.0 and Excel 5.0