

# Table of Contents

[Copyright Information](#)      [Technical Support](#)      [Company Commitment](#)

[Event Reference](#)

[Property Reference](#)

[Function Call Reference](#)

[Worksheet Function Reference](#)

## Chapter 1

### [Getting Started](#)

[Introduction](#)

[Installing Formula One](#)

[Loading Formula One](#)

[Autoloading Formula One](#)

[Using Help](#)

[Basic Concepts](#)

[Placing Controls on Forms](#)

[Placing Edit Bars on Forms](#)

[Distributing Applications](#)

[Redistributing Files](#)

## Chapter 2

### [Programming Tools](#)

[Using Properties](#)

[Properties Summary](#)

[Using Function Calls](#)

[Calling Functions](#)

[Using Built-In Dialog Boxes](#)

[Trapping for Errors](#)

[Function Call Summary](#)

[Dialog Box Function Calls](#)

[Edit Bar Function Calls](#)

[Formatting Function Calls](#)

[Data Entry Function Calls](#)

[Printing Function Calls](#)

[Range Editing Function Calls](#)

[Recalculation Function Calls](#)

[Selection Function Calls](#)

[Worksheet Function Calls](#)

[Miscellaneous Function Calls](#)

[Using Events](#)

[Validating Data](#)

[Drilling for Data](#)

[Events and Other Controls](#)

[Event Summary](#)

## Chapter 3

### [Using Views and Worksheets](#)

[Working with Worksheets](#)

[Working with Views](#)

[Using View Information](#)

[Saving View Information](#)

[Attaching Views to Worksheets](#)

[One View with Multiple Worksheets](#)

[One Worksheet with Multiple Views](#)

[Saving Worksheets](#)

[Reading and Writing Files](#)

## Using the Worksheet Designer

### Chapter 4

#### Using Edit Bar Controls

Creating Edit Bar Controls

Edit Bar Properties

Edit Bar Events

Edit Bar Function Calls

### Chapter 5

#### Worksheet Fundamentals

Navigating through Worksheets

Using Keyboard Commands

Performing Mouse Actions

Selecting Cells

Selecting Cells with the Mouse

Selecting Cells with Properties

Using the Selection Property

Selecting Cells with Function Calls

Selecting Rows and Columns

Selecting Rows and Columns with Properties

### Chapter 6

#### Working with Data

Worksheet Data Entry

Entering Data with Properties

Entering Data with Function Calls

Limiting Data Entry

Limiting Formula Entry

Locking Cells

Worksheet Data Types

Entering Constant Values

Entering Formulas

Formula Operators

Operator Precedence

Cell References

Absolute and Relative References

External References

Automatically Entering Cell References

Worksheet Errors

Displaying Formulas

Custom Functions

Built-In Worksheet Functions

Understanding Functions

Entering Functions

Nesting Functions

Entering Arguments

Syntax Errors

Using Names

Calculating Worksheets

Setting Automatic Recalculation

Solving Circular References

### Chapter 7

#### Editing Worksheets

Cut, Copy, and Paste Function Calls

[Copying Data Across Ranges](#)  
[Copying Data Interactively](#)  
[Moving Data](#)  
[Moving Data Interactively](#)  
[Inserting Cells, Rows, and Columns](#)  
[Clearing and Deleting Cells, Rows, and Columns](#)  
[Sorting Worksheets](#)

## Chapter 8

### [Formatting Worksheets](#)

[Built-In Number Formats](#)  
[Formatting Rows and Columns](#)  
[Obtaining Formatted Text](#)  
[Custom Formatting](#)  
[Aligning Data](#)  
[Changing Row Heights and Column Widths](#)  
[Interactively Sizing Rows and Columns](#)  
[Sizing Rows and Columns with Function Calls](#)  
[Setting Cell Borders and Colors](#)  
[Formatting Row and Column Headings](#)

## Chapter 9

### [Printing Worksheets](#)

[Printing with Function Calls](#)  
[Specifying Print Areas](#)  
[Specifying Row and Column Print Titles](#)  
[Specifying Print Headers and Footers](#)  
[Specifying Page Breaks](#)  
[Page Break Function Calls](#)

## Chapter 10

### [Working with Databases](#)

[Accessing Databases](#)  
[Using Virtual Record Buffers](#)  
[DataChanged and DataFieldChanged Properties](#)  
[RowMode Property](#)  
[Deleting Records](#)  
[DataRowLoad and DataNewRow Events](#)  
[Specifying Database Column Display](#)  
[Calculating Database Formulas](#)  
[Displaying and Using Field Names](#)

## Chapter 11

### [Performance Tuning](#)

### [Worksheet Specifications](#)

## Chapter 12

### [Formula One and Visual C++](#)

[CVBControl Class](#)  
[Getting and Setting Properties with Visual C++](#)  
[Differences Between Visual Basic and Visual C++](#)

## Chapter 13

### [Worksheet Designer Overview](#)

[File Menu Commands](#)  
[Edit Menu Commands](#)

[View Menu Commands](#)  
[Format Menu Commands](#)  
[Window Menu Commands](#)

**Chapter 14**

[A-Z Event Reference](#)

**Chapter 15**

[A-Z Property Reference](#)

**Chapter 16**

[A-Z Function Call Reference](#)

**Chapter 17**

[A-Z Worksheet Function Reference](#)

## Company Commitment

The phenomenal increase in computing power in recent years has given rise to applications that are increasingly complex. This trend has made it all but impossible to develop applications from raw code. Instead, developers need high-quality tools if they are to build world class applications. VisualTools, Inc. was formed in February 1993, to supply those tools. Our guiding principles are:



Integrity is our highest concern



Customers are the reason we exist



Quality is not available for compromise



Employee involvement is our basic business model

## Introduction

Formula One is a high performance spreadsheet control that allows you to create, manipulate and print worksheets. It contains the tools needed to store, analyze, manipulate, and present your data. Its major features include:



**Excel Compatible.** Formula One reads and writes Excel 4.0 compatible worksheets. Developers can create applications that share information with all major Windows applications.



**Database Access.** Formula One is a bound control, allowing connection to Access databases.



**Multiple Worksheets.** With Formula One, you can work with multiple worksheets simultaneously.



**Drag-and-Drop.** Formula One allows you to move and copy data in ranges by dragging ranges.



**Superior Data Formatting.** Excel-style custom formatting is supported, allowing unlimited formatting options.



**Worksheet Designer.** The Worksheet Designer is an interactive program that allows you to design and format the worksheet for your application by pointing and clicking, and choosing format commands from menus.



**Ease of Use.** Formula One has many features that make it one of the easiest data aware controls to use, including built-in dialog boxes for your applications, the complete documentation available through on-line help, and endless flexibility for customizing the control.

## Getting Technical Support

The VisualTools technical support staff can help you with any problem you encounter installing or using Formula One. If you need assistance, contact VisualTools in any of the following ways:



**By telephone.** You can contact our technical support staff at (913)599-6500 on weekdays between 8:30 a.m. and 5:30 p.m., central time.



**By FAX.** You can contact us by FAX at (913)599-6597.



**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 72204,3521.



**By mail.** Address your correspondence to:  
Customer Service Department  
VisualTools, Inc.  
15721 College Blvd.  
Lenexa, Kansas 66219

## **Getting Started**

Before you can use the Formula One control, it must be installed on your system. The following sections provide instructions for installing the Formula One files, including the sample applications. Information is also provided about how to load the control in your Visual Basic tool box and place a control on a form.



## Installing Formula One

The Setup program creates new directories and copies the Formula One files to your hard disk.



### To install Formula One on your hard disk:

1. Start Microsoft Windows.
2. Insert the Formula One disk in your floppy drive.
3. From Program Manager, select the File menu and choose Run.
4. In the Run dialog box, type a:\setup or b:\setup, depending on where you placed the Formula One disk.

There are eight basic Formula One components:

<b>File</b>	<b>Description</b>
VTSS.VBX	Visual Basic control
VTSSDLL.DLL	Worksheet engine
VTSSAPP.EXE	Worksheet Designer application
VTSS.LIC	License file necessary to run in design mode
VTSS.HLP	Help file
VTSS.TXT	Text file containing all declarations and constants
VTSS.H	Header file containing all declarations and constants
VTSS.BAS	Declarations and constants in Basic form

In addition to the files listed in the preceding table, several sample applications are installed subdirectories in the VTFORM1 directory, unless you specify another location for installation.

The program group VisualTools is created in your Program Manager. The Worksheet Designer application and sample applications are installed in this program group.

## Loading Formula One

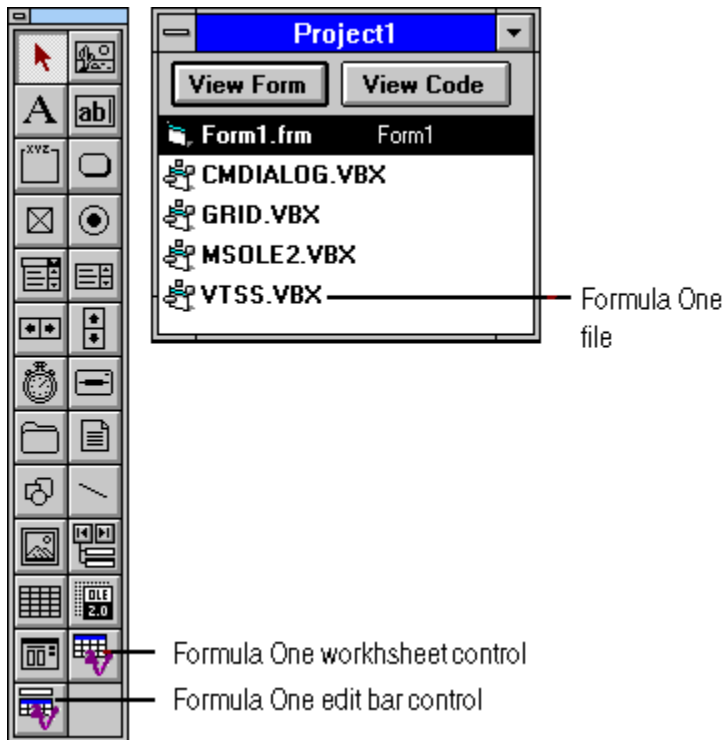
To use Formula One in Visual Basic, you must add the Formula One controls to the Visual Basic toolbox.



### To add the Formula One control to the Visual Basic toolbox:

1. Start Visual Basic.
2. From Visual Basic, select the File menu and choose Add File.
3. Select the WINDOWS\SYSTEM directory on your hard disk and double click VTSS.VBX. If you installed the Formula One files in a custom location, you must open the directory in which you installed VTSS.VBX.

The Formula One icons are added to the Visual Basic toolbox.



## Autoloading Formula One

You can configure Visual Basic to automatically load the Formula One controls when you start a new project in Visual Basic.



### **To configure Visual Basic to automatically load VTSS.VBX:**

1. Start Visual Basic and open AUTOLOAD.MAK.
2. Choose Add File from the File menu.

The Add File dialog box appears.

3. Select VTSS.VBX and choose OK.

The control is added to the project list.

4. Choose Save Project from the File menu.

## Using Help

Comprehensive on-line help is available to assist you as you learn and use the Formula One controls. The complete set of Formula One documentation is available through on-line help. In addition, you can receive context-sensitive help for properties.



### **To access the help index:**

1. Click the Formula One icon (on the toolbox).
2. Press F1.



### **To access context-sensitive help for properties:**

1. Select a Formula One control on your form.
2. Highlight a Formula One property in the Properties dialog box.
3. Press F1.

## Basic Concepts

Formula One is a Visual Basic Custom Control (VBX). It can be accessed directly by Microsoft's Visual Basic or Visual C++.

The Formula One control is a compatible subset of Microsoft's Excel spreadsheet application. You can design complex models either directly in Formula One or import data from Excel. Similarly, data collected and manipulated in an application using Formula One can be exported to Excel or other applications that read Excel 4.0 files.

Data, formulas, and formatting information can be entered in the Formula One control at design time or run time.

When using the Formula One control at design time, you have access to the Worksheet Designer application. With this application, you can manipulate the worksheet control just like it was a part of spreadsheet application.

The Worksheet Designer is accessed by double clicking the control with the right mouse button.

Worksheet controls can be saved with a form or in a separate file.

The Formula One control can be used as a bound control. It automatically handles adding, deleting, updating, and displaying data from Access databases. With this feature, building database applications is greatly simplified.

You can also store a complete worksheet in a single field of each record, allowing you to create a database of worksheets. Access databases refer to these fields as OLE fields.

## Placing Formula One Controls on Forms

Creating a new worksheet control and placing it on a form is as simple as point, click, and drag.



### **To place the Formula One control on a form:**

1. Select the Formula One tool in the Visual Basic tool box.
2. Position the mouse in the form at the location where you want to draw the control.
3. Click and drag to draw the outline of the worksheet on the form.

When you release the mouse, the new worksheet control is placed in the location you specified.

Once the worksheet is placed on the form, you can immediately use the control. Formula One's control defaults are set so you can add data and formulas, navigate through the worksheet, and access the Worksheet Designer without writing any code.

## Placing an Edit Bar Control on a Form

An edit bar is an auxiliary control that interacts with worksheet controls. An edit bar allows you to enter and edit data in worksheet controls. Edit bar controls are not required because you can edit data directly in a worksheet control. However, edit bars are convenient if you have long text or complex formulas to enter.

You can place an edit bar control on a form in the same manner as a worksheet control.



### **To place an edit bar control on a form:**

1. Select the edit bar tool in the Visual Basic tool box.
2. Position the mouse in the form at the location where you want to draw the edit bar.
3. Click and drag to draw the outline of the edit bar on the form.

When you release the mouse, the new edit bar control is placed in the location you specified.

## **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:

<b>File</b>	<b>Description</b>
VTSS.VBX	Visual Basic control
VTSSDLL.DLL	Worksheet engine

**Copyright © 1994 VisualTools, Inc. All rights reserved.**

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of VisualTools, Inc.

© 1994 VisualTools, Inc. All rights reserved.

Microsoft, MS, MS-DOS, and GW-BASIC are registered trademarks and Microsoft Access, QuickBasic, Visual Basic, and Windows are trademarks of Microsoft Corporation in the USA and other countries.

CompuServe is a registered trademark of CompuServe, Inc.

TrueType is a registered trademark of Apple Computer, Inc.

The VisualTools License Agreement, included with the product, specifies the permitted and prohibited uses of the product. Any unauthorized reproduction or use of the product, or breach of the terms and conditions of the License Agreement, is forbidden. The VisualTools License Agreement sets forth the only warranties applicable to the product and documentation. All warranty disclaimers and exclusions set forth therein apply to the information contained in this document.

Created and published by  
VisualTools, Inc.

15721 College Boulevard  
Lenexa, Kansas 66219  
913-599-6500  
FAX: 913-599-6597

## Programming Tools

Although many applications can be created "visually" via the Visual Basic interface and Worksheet Designer, your application may require more complex functionality. Formula One/ VB properties and function calls are programming tools that provide access to that functionality.



Formula One properties provide a subset of the complete control functionality that is sufficient for most applications. Properties are easy to use, yet powerful tools. Formula One properties include some standard Visual Basic properties.



For more complex applications, application developers can use function calls to access the complete range of the controls functionality. In addition, function calls allow you to access the suite of built-in dialog boxes.

## Using Properties

Formula One provides 112 properties. With properties, you can perform a variety of tasks, such as hiding and displaying elements of a worksheet, selecting cells and ranges, setting print margins, and counting the number of database records displayed in a worksheet. Many properties allow you to perform complex tasks with very little coding.

The following example shows the code required to read an Excel worksheet from disk using the read file property:

```
Sheet1.ReadFile = "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.

```
Sheet1.Row = 1
Sheet1.Col = 1
Sheet1.Number = 1
Sheet1.Row = 2
Sheet1.Number = 2
Sheet1.Row = 3
Sheet1.Formula = "sin(A1) + cos(A2)"
Sheet1.Row = 4
Sheet1.Col = 2
Sheet1.Text = "The End!"
```

The following illustration shows the result of the preceding example.

	A	B	C
1	1		
2	2		
3	0.425324		
4		The End!	
5			

## Properties Summary

The following table lists the properties available in Formula One

<b>Property</b>	<b>Description</b>
<a href="#"><u>AllowAppLaunch</u></a>	Determines if the Worksheet Designer is allowed to launch when the user double clicks the Formula One window.
<a href="#"><u>AllowArrows</u></a>	Determines if the arrow keys can move the active cell.
<a href="#"><u>AllowDelete</u></a>	Determines if the Delete key can delete the current record or clear the current selection.
<a href="#"><u>AllowEditHeaders</u></a>	Determines if row, column, and top left header text can be edited.
<a href="#"><u>AllowFillRange</u></a>	Determines if the user is allowed to fill a range by dragging a selection's fill handle.
<a href="#"><u>AllowFormulas</u></a>	Determines if the user can enter new formulas or edit existing formulas.
<a href="#"><u>AllowInCellEditing</u></a>	Determines if in-cell editing is allowed.
<a href="#"><u>AllowMoveRange</u></a>	Determines if the user can move a selection by dragging it to a new location.
<a href="#"><u>AllowResize</u></a>	Determines if the user is allowed to resize rows or columns.
<a href="#"><u>AllowSelections</u></a>	Determines if the user is allowed to make selections.
<a href="#"><u>AllowTabs</u></a>	Determines if the Tab and Shift Tab keys can move the active cell within the current selection.
<a href="#"><u>AutoRecalc</u></a>	Determines if automatic recalculation is enabled. Forces the worksheet to be recalculated immediately when set to True.
<a href="#"><u>BackColor</u></a>	Determines the background color of the Formula One window.
<a href="#"><u>BorderStyle</u></a>	Determines the border style for the Formula One window.
<a href="#"><u>Col</u></a>	Determines the active column in the worksheet. This is a run time only property.
<a href="#"><u>DataAutoAddNew</u></a>	Determines if the worksheet has an empty row at the end for adding new records.
<a href="#"><u>DataChanged</u></a>	Indicates that the data in the current record has changed.
<a href="#"><u>DataConnected</u></a>	Specifies whether the worksheet is connected to a data control. This can be useful for downloading data and disconnecting before performing analysis or operations on the data.
<a href="#"><u>DataField</u></a>	Binds the Formula One control to a database field. Used for storing an entire Formula One worksheet within a single field.
<a href="#"><u>DataFieldChanged</u></a>	Indicates whether the specified field has been changed by the user. This is a run time only property.
<a href="#"><u>DataFieldCount</u></a>	Returns the number of database fields displayed in the worksheet. This is a run time, read only property.
<a href="#"><u>DataFieldNumber</u></a>	Returns the database field number of the specified column. This is a run time, read only property.
<a href="#"><u>DataFields</u></a>	Binds the Formula One control to one or more database fields. Used for displaying one record per row.
<a href="#"><u>DataHdrField</u></a>	Allows a field's value to be specified as the row headers.
<a href="#"><u>DataRowBase</u></a>	Returns the row number of the record in row 1 of the worksheet. Used only when the virtual record mode is used. This number is invalid after a find or if other users add or delete records to the database. When valid, DataRowBase plus Row equals the database record number.
<a href="#"><u>DataRowCount</u></a>	Returns the number of database records currently loaded in the worksheet. This is a run time, read only property.
<a href="#"><u>DataRowsBuffered</u></a>	Specifies how many rows are held in memory at one time.
<a href="#"><u>DataSetColumnFormats</u></a>	Determines if formats for date, time, and currency fields are set automatically.

<b><u>DataSetColumnNames</u></b>	Determines if the column headings are replaced by the field names.
<b><u>DataSetColumnWidths</u></b>	Determines if the column widths are automatically set to the widest data in the column.
<b><u>DataSetMaxCol</u></b>	Determines if the MaxCol property is set to the number of columns containing fields.
<b><u>DataSetMaxRow</u></b>	Determines if the MaxRow property is set to the number of rows containing records.
<b><u>DataSource</u></b>	Determines the data control through which the current Formula One control is bound to a database. This is a design time only property.
<b><u>DoCancelEdit</u></b>	Determines if the CancelEdit event can be fired.
<b><u>DoClick</u></b>	Determines if the Click event can be fired.
<b><u>DoDataNewRow</u></b>	Specifies whether the DataNewRow event gets fired when the data control sends the AddNew message.
<b><u>DoDataRowLoad</u></b>	Specifies whether the DataRowLoad event gets fired after each row is loaded from the data control.
<b><u>DoDbClick</u></b>	Determines if the DbClick Event can be fired.
<b><u>DoEndEdit</u></b>	Determines if the EndEdit Event can be fired.
<b><u>DoEndRecalc</u></b>	Determines if the EndRecalc Event can be fired.
<b><u>DoSelChange</u></b>	Determines if the SelChange Event can be fired.
<b><u>DoStartEdit</u></b>	Determines if the StartEdit Event can be fired.
<b><u>DoStartRecalc</u></b>	Determines if the StartRecalc Event can be fired.
<b><u>DoTopLeftChanged</u></b>	Determines if the TopLeftChanged Event can be fired.
<b><u>DragIcon</u></b>	Determines the icon to be displayed in a drag-and-drop operation.
<b><u>DragMode</u></b>	Specifies manual or automatic dragging mode for drag-and-drop operations.
<b><u>EditName</u></b>	Determines the edit bar that is used with this worksheet.
<b><u>Enabled</u></b>	Determines if the Formula One object is enabled.
<b><u>EnableProtection</u></b>	Enables cell protection for the current worksheet.
<b><u>Entry</u></b>	Specifies the formatted contents of a cell.
<b><u>ExtraColor</u></b>	Determines the color of the Formula One window outside the active cell area.
<b><u>FileName</u></b>	Specifies the name from which a worksheet is loaded and to which it is saved. If this property is empty the worksheet is saved in the form.
<b><u>FixedCol</u></b>	Determines the starting fixed column in the Formula One window.
<b><u>FixedCols</u></b>	Determines how many columns to fix at the left edge of the worksheet.
<b><u>FixedRow</u></b>	Determines the starting fixed row in the Formula One window.
<b><u>FixedRows</u></b>	Determines how many rows to fix at the top of the worksheet.
<b><u>FormattedText</u></b>	Returns the formatted text of a cell. The text is the same as displayed in the worksheet, including all formatting.
<b><u>Formula</u></b>	Specifies a formula as a text string for the active cell. This is a run time only property.
<b><u>Height</u></b>	Determines the dimensions of an object.
<b><u>HelpContextID</u></b>	Determines the associated help context number for an object. Used to provide context sensitive help in an application.
<b><u>hWnd</u></b>	Specifies a handle to the control. This property is a run time, read only property.
<b><u>Index</u></b>	Specifies a unique number that identifies a control in a control array. This property is a run time, read only property.
<b><u>Left</u></b>	Determines the distance between the internal left edge of an object and the left edge of the

container.

<b><u>LeftCol</u></b>	Determines the leftmost column displayed in the worksheet window.
<b><u>MaxCol</u></b>	Specifies the last displayable column.
<b><u>MaxRow</u></b>	Specifies the last displayable row.
<b><u>MinCol</u></b>	Specifies the first displayable column.
<b><u>MinRow</u></b>	Specifies the first displayable row.
<b><u>MousePointer</u></b>	Determines the type of mouse pointer displayed when the cursor is in the Formula One control.
<b><u>Name</u></b>	Specifies the name by which the object can be referred in the program code. This name cannot be changed at run time.
<b><u>Number</u></b>	Specifies a numeric value for the active cell. This is a run time only property.
<b><u>Parent</u></b>	Specifies the form on which the control is located. This is a run time, read only property.
<b><u>PrintArea</u></b>	Specifies the worksheet ranges to be printed.
<b><u>PrintBottomMargin</u></b>	Determines the bottom page margin.
<b><u>PrintColHeading</u></b>	Determines if the worksheet column headings are printed.
<b><u>PrintFooter</u></b>	Determines the contents of the page footer.
<b><u>PrintGridLines</u></b>	Determines if the grid lines are printed.
<b><u>PrintHCenter</u></b>	Determines if the worksheet is horizontally centered on the page.
<b><u>PrintHeader</u></b>	Determines the contents of the page header.
<b><u>PrintLeftMargin</u></b>	Determines the left page margin.
<b><u>PrintLeftToRight</u></b>	Determines if the worksheet pages print from top to bottom or left to right.
<b><u>PrintNoColor</u></b>	Determines if the worksheet pages are printed in color.
<b><u>PrintRightMargin</u></b>	Determines the right page margin.
<b><u>PrintRowHeading</u></b>	Determines if the worksheet row headings are printed.
<b><u>PrintTitles</u></b>	Determines the rows and columns printed as titles on each page.
<b><u>PrintTopMargin</u></b>	Determines the top page margin.
<b><u>PrintVCenter</u></b>	Determines if the worksheet is vertically centered on the page.
<b><u>ReadFile</u></b>	Reads a worksheet from a file into the control.
<b><u>Repaint</u></b>	Determines if Formula One repaints after a change is made to the worksheet.
<b><u>Row</u></b>	Determines the active row in the worksheet. This is a run time only property.
<b><u>RowMode</u></b>	Specifies whether individual cells or entire rows can be selected.
<b><u>Selection</u></b>	Determines the current selection.
<b><u>SelEndCol</u></b>	Determines the ending column of a selected range.
<b><u>SelEndRow</u></b>	Determines the ending row of a selected range.
<b><u>SelStartCol</u></b>	Determines the starting column of a selected range.
<b><u>SelStartRow</u></b>	Determines the starting row of a selected range.
<b><u>ShowColHeading</u></b>	Determines if the column headings are displayed in the Formula One window.
<b><u>ShowGridLines</u></b>	Determines if the grid lines are displayed in the Formula One window.
<b><u>ShowHScrollBar</u></b>	Determines how the horizontal scroll bar is displayed.
<b><u>ShowRowHeading</u></b>	Determines if the row headings are displayed in the Formula One window.
<b><u>ShowSelections</u></b>	Determines how selections are displayed.

<b><u>ShowVScrollBar</u></b>	Determines how the vertical scroll bar is displayed.
<b><u>SS Property</u></b>	Specifies the handle to a worksheet view. This is a run time only, read only property.
<b><u>TabIndex</u></b>	Determines the tab order of the Formula One control within its parent form.
<b><u>TableName</u></b>	Specifies the name by which the worksheet is referred in formulas in other worksheets.
<b><u>TabStop</u></b>	Determines if the user can use the Tab key to set the focus to this control.
<b><u>Tag</u></b>	Stores any extra data needed by your application. This property is not used by Visual Basic or Formula One.
<b><u>Text</u></b>	Specifies a text string for the active cell. This is a run time only property.
<b><u>Top</u></b>	Determines the distance between the internal top edge of an object and the top edge of the container.
<b><u>TopRow</u></b>	Determines the top row displayed in the worksheet window.
<b><u>Visible</u></b>	Determines if the Formula One object is visible.
<b><u>Width</u></b>	Determines the width of a Formula One object.
<b><u>WriteExcel4</u></b>	Writes the current worksheet to the specified Excel 4.0 file.
<b><u>WriteFile</u></b>	Writes the current worksheet to the specified file.



## Using Function Calls

Function calls provide access to the complete set of Formula One functionality. In fact, function calls provide enough functionality to build a complete stand-alone spreadsheet, if desired.

Function calls can be easily accessed from Visual Basic, Visual C++, C, and many other Windows applications.



To access function calls from Visual Basic, include the file VTSS.TXT in your project.



To access function calls from Visual C++ or C, include the header files VTSS.H and SSERROR.H and the library VTSSDLL.LIB.

These files contain the function declarations and constants.

## Calling Functions

Function calls operate on a worksheet through a specific view. Each function call requires a handle to a view to tell it on which view and worksheet it is operating. This handle is available as the [SS Property](#) property. The handle is a run time only, read only property. You cannot change the setting of this property.

There are two basic types of function calls.



Some functions set values or perform operations and do not return data.



Other functions perform an operation and return a value or string to the caller.

Functions that do not return data are the simplest to call. An example of this type of function is [SSSetNumber](#). This function call places a number in the active cell. The following code uses this function call:

```
hSS = Sheet1.SS
SSERROR = SSSetNumber (hSS, 1234.56)
```

After calling a function, test SSERROR to determine if the operation succeeded. If its value is zero, the function succeeded. If the value is non-zero, an error occurred. The list of possible error values is provided in the section [Trapping Errors](#).

Function calls that return values or strings require a destination for the result. In the case of a value, you must supply a variable in which the value is returned. An example of a function call that returns a value is [SSGetNumber](#). This function call returns the number in the active cell. The following code uses this function call:

```
Dim TheNumber#
hSS = sheet1.SS
SSERROR = SSGetNumber (hSS, TheNumber#)
```

The value of the active cell is placed in the variable TheNumber. As in the previous example, SSERROR is tested to determine if the operation succeeded.

Function calls that return strings need both a buffer and a place in which to return the string. In addition, you must specify the size of the buffer, insuring that the buffer is not overrun. Space for the buffer must be allocated using Space\$(n), where n is the size of the buffer, prior to calling the function.

An example of a function call that returns a string is [SSGetText](#). This function call returns the text of the active cell. The following code uses this function call:

```
hssView = sheet1.SS
TheBuffer$ = Space$(50)
BuffSize = 50
SSERROR = SSGetText (hssView, TheBuffer$, BuffSize)
```

As in the previous examples, SSERROR is tested to determine if the operation succeeded.

## **Built-In Dialog Boxes**

To make application development easier, Formula One provides a suite of built-in dialog boxes that can be invoked by function calls. These dialog boxes provide yet another avenue for accessing the Formula One functionality.

The dialog boxes can be displayed by your application to allow the application user to provide input for the worksheet control. Refer to the [dialog box function call summary](#) for a list of the dialog box function calls.

## Trapping Errors

Formula One errors that occur during Visual Basic execution are handled like other Visual Basic errors. An error sets the Err error status function and forces an error. These errors can be caught with the On Error statement and tested with the Err function.

Function calls also return an error status. This status should be checked after each function call is executed. The following table lists the errors that can be generated by function calls. These error values are contained in VTSS.TXT for Visual Basic projects, and SSERROR.H for C and C++ projects.

**Note** Formula One error numbers are incremented by 20,000 to avoid conflict with Visual Basic error numbers.

---

Error Name	Error Number	Description
SSERROR_NONE	0	Function succeeded.
SSERROR_GENERAL	1	Function failed with a non-specific error.
SSERROR_BAD_ARGUMENT	2	One of the function arguments was invalid.
SSERROR_NO_MEMORY	3	Not enough memory to complete the task.
SSERROR_BAD_FORMULA	4	The formula syntax is incorrect.
SSERROR_BUF_TOO_SHORT	5	The returned result is longer than the return buffer size. A NULL string is placed in the buffer.
SSERROR_NOT_FOUND	6	Cannot find item for which function is looking.
SSERROR_BAD_RC	7	The row/column reference is invalid.
SSERROR_BAD_HSS	8	Invalid view handle passed.
SSERROR_TOO_MANY_HSS	9	Unable to create additional view handles.
SSERROR_NO_TABLE	10	No worksheet attached to the view.
SSERROR_UNABLE_TO_OPEN_FILE	11	Cannot open the specified file.
SSERROR_INVALID_FILE	12	Cannot read invalid file.
SSERROR_INSERT_SHIFT_OFF_TABLE	13	Insert pushes cells outside of worksheet bounds.
SSERROR_ONLY_ONE_RANGE	14	Specified command expects only one selected range.
SSERROR_NOTHING_TO_PASTE	15	Nothing to paste when a paste operation was requested.
SSERROR_BAD_NUMBER_FORMAT	16	Invalid custom format string.
SSERROR_TOO_MANY_FONTS	17	Cannot add fonts to the table.
SSERROR_TOO_MANY_SELECTED_RANGES	18	Cannot add selected ranges.
SSERROR_UNABLE_TO_WRITE_FILE	19	An error occurred while writing the file.
SSERROR_NO_TRANSACTION	20	<a href="#">SSTransactCommit</a> or <a href="#">SSTransactRollback</a> was called without first calling <a href="#">SSTransactStart</a> .
SSERROR_NOTHING_TO_PRINT	21	No data to print in the table or selected range.
SSERROR_PRINT_MARGINS_DONT_FIT	22	Print margins are out of range.
SSERROR_CANCEL	23	Returned if the user presses Cancel in a built-in dialog box.
SSERROR_UNABLE_TO_INITIALIZE_PRINTER	24	Cannot initialize the printer.
SSERROR_STRING_TOO_LONG	25	An argument to a C function specified a string that was too long.
SSERROR_FORMULA_TOO_LONG	26	Specified formula is too long.

SSERROR_UNABLE_TO_OPEN_CLIPBOARD	27	Cannot open the Windows clipboard.
SSERROR_PASTE_WOULD_OVERFLOW_SHEET	28	The paste operation extends beyond the last row or last column of the worksheet.
SSERROR_LOCKED_CELLS_CANNOT_BE_MODIFIED	29	Attempted to modify cells that are locked with protection enabled.
SSERROR_LOCKED_DOCUMENT_CANNOT_BE_MODIFIED	30	Attempted to modify a document that has protection enabled.
SSERROR_INVALID_NAME	31	Specified a user defined name that is invalid.
SSERROR_CANT_DELETE_NAME_IN_USE	32	Attempted to delete a user defined name that is currently in use by a formula.
SSERROR_UNABLE_TO_FIND_NAME	33	Could not find specified user defined name.

## Dialog Box Function Call Summary

Dialog box function calls invoke the Formula One built-in dialog boxes. The following table lists the dialog box function calls.

<b>Dialog Box Functions</b>	<b>Operation</b>
<a href="#"><u>SSCalculationDlg</u></a>	This dialog box allows you to enable and disable automatic recalculation and specify iteration values for calculating circular references.
<a href="#"><u>SSColorPaletteDlg</u></a>	This dialog box allows you to edit colors in the color palette, specify a default color, and use the default color palette.
<a href="#"><u>SSColWidthDlg</u></a>	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.
<a href="#"><u>SSDefinedNameDlg</u></a>	This dialog box allows you to add and delete user defined names.
<a href="#"><u>SSFilePageSetupDlg</u></a>	This dialog box allows you to define header and footer text, page margins, page print order, page centering, worksheet-related print options.
<a href="#"><u>SSFilePrintSetupDlg</u></a>	This dialog box allows you to select the printer to which the worksheet is sent, the page orientation, and paper size.
<a href="#"><u>SSFormatAlignmentDlg</u></a>	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.
<a href="#"><u>SSFormatBorderDlg</u></a>	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.
<a href="#"><u>SSFormatFontDlg</u></a>	This dialog box allows you to specify the font, point size, font style, and color of data in the selected range.
<a href="#"><u>SSFormatNumberDlg</u></a>	This dialog box allows you to define custom number formats for data in the selected range.
<a href="#"><u>SSFormatPatternDlg</u></a>	This dialog box allows you to specify the fill pattern and foreground and background colors for the selected range.
<a href="#"><u>SSGotoDlg</u></a>	This dialog box allows you to select the worksheet page to display.
<a href="#"><u>SSOpenFileDialog</u></a>	This dialog box allows you to open worksheets from disk.
<a href="#"><u>SSProtectionDlg</u></a>	This dialog box allows you to specify whether the cells in the selected range are locked and hidden.
<a href="#"><u>SSRowHeightDlg</u></a>	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.
<a href="#"><u>SSSaveFileDialog</u></a>	This dialog box allows you to save the current file in Formula One or Excel 4.0 format.
<a href="#"><u>SSSortDlg</u></a>	This dialog box allows you to set the sorting method and sort keys for data sorting.

## Edit Bar Function Call Summary

Edit bar function calls manipulate edit bar controls. The following table lists the edit bar function calls. These function calls are not normally called from Visual Basic.

<b>Edit bar Functions</b>	<b>Operation</b>
<a href="#"><u>SSEditBarDelete</u></a>	Deletes an edit bar.
<a href="#"><u>SSEditBarHeight</u></a>	Returns the default height of an edit bar.
<a href="#"><u>SSEditBarMove</u></a>	Moves an edit bar.
<a href="#"><u>SSEditBarNew</u></a>	Creates a new edit bar.

## Formatting Function Call Summary

Formatting function calls control the appearance of worksheets and the data they contain. The following table lists the formatting function calls.

<b>Formatting Functions</b>	<b>Operation</b>
<a href="#"><u>SSFormatCurrency0</u></a>	Formats selected ranges with currency format and no decimal places.
<a href="#"><u>SSFormatCurrency2</u></a>	Formats selected ranges with currency format and two decimal places.
<a href="#"><u>SSFormatFixed</u></a>	Formats selected ranges with fixed format and no decimal places.
<a href="#"><u>SSFormatFixed2</u></a>	Formats selected ranges with fixed format and two decimal places.
<a href="#"><u>SSFormatFraction</u></a>	Formats the selected ranges with the fraction format.
<a href="#"><u>SSFormatGeneral</u></a>	Formats the selected ranges with the general format.
<a href="#"><u>SSFormatHmampm</u></a>	Formats the selected ranges in 12-hour time format.
<a href="#"><u>SSFormatMdyy</u></a>	Formats the selected ranges with the date format.
<a href="#"><u>SSFormatPercent</u></a>	Formats the selected ranges in percent format.
<a href="#"><u>SSFormatScientific</u></a>	Formats the selected ranges in scientific format.
<a href="#"><u>SSGetAllowResize</u></a>	Returns the state of the resize flag.
<a href="#"><u>SSGetBackColor</u></a>	Returns the background color of the view.
<a href="#"><u>SSGetColWidth</u></a>	Returns the width of the specified column.
<a href="#"><u>SSGetExtraColor</u></a>	Returns the color outside the worksheet.
<a href="#"><u>SSGetMaxCol</u></a>	Returns the last displayable column.
<a href="#"><u>SSGetMaxRow</u></a>	Returns the last displayable row.
<a href="#"><u>SSGetMinCol</u></a>	Returns the first displayable column.
<a href="#"><u>SSGetMinRow</u></a>	Returns the first displayable row.
<a href="#"><u>SSGetRowHeight</u></a>	Returns the height of the specified row.
<a href="#"><u>SSGetShowColHeading</u></a>	Returns the show column heading flag.
<a href="#"><u>SSGetShowFormulas</u></a>	Returns the show formulas flag.
<a href="#"><u>SSGetShowGridLines</u></a>	Returns the show grid lines flag.
<a href="#"><u>SSGetShowHScrollBar</u></a>	Returns the show horizontal scroll bar flag.
<a href="#"><u>SSGetShowRowHeading</u></a>	Returns the show row heading flag.
<a href="#"><u>SSGetShowSelections</u></a>	Returns the show selections flag.
<a href="#"><u>SSGetShowVScrollBar</u></a>	Returns the show vertical scroll bar flag.
<a href="#"><u>SSGetShowZeroValues</u></a>	Returns the show zero values flag.
<a href="#"><u>SSSetAlignment</u></a>	Specifies data alignment for a selection.
<a href="#"><u>SSSetAllowResize</u></a>	Specifies whether resizing rows and columns by dragging is allowed.
<a href="#"><u>SSSetBackColor</u></a>	Specifies the background color of the worksheet.
<a href="#"><u>SSSetBorder</u></a>	Specifies the border for all selected cells.
<a href="#"><u>SSSetColText</u></a>	Specifies the text for a column header.
<a href="#"><u>SSSetColWidth</u></a>	Determines the width for the specified columns.
<a href="#"><u>SSSetColWidthAuto</u></a>	Specifies that column widths are set automatically.
<a href="#"><u>SSSetExtraColor</u></a>	Specifies the color of the view area outside the worksheet.
<a href="#"><u>SSSetFont</u></a>	Specifies the font information for all selected cells.



<b><u>SSSetHdrHeight</u></b>	Specifies the height of column headers.
<b><u>SSSetHdrWidth</u></b>	Specifies the width of row headers.
<b><u>SSSetLeftCol</u></b>	Specifies the leftmost column in the view.
<b><u>SSSetMaxCol</u></b>	Specifies the last displayable column.
<b><u>SSSetMaxRow</u></b>	Specifies the last displayable row.
<b><u>SSSetMinCol</u></b>	Specifies the first displayable column.
<b><u>SSSetMinRow</u></b>	Specifies the first displayable row.
<b><u>SSSetNumberFormat</u></b>	Specifies the number format for all selected cells.
<b><u>SSSetPattern</u></b>	Specifies the fill pattern and colors for the selected cells.
<b><u>SSSetRowHeight</u></b>	Specifies the height for the specified rows.
<b><u>SSSetRowHeightAuto</u></b>	Specifies that row heights are set automatically.
<b><u>SSSetRowText</u></b>	Specifies the text for a row header.
<b><u>SSSetShowColHeading</u></b>	Specifies whether column headings are displayed.
<b><u>SSSetShowFormulas</u></b>	Specifies whether formulas are displayed in place of cell values.
<b><u>SSSetShowGridLines</u></b>	Specifies whether grid lines are displayed.
<b><u>SSSetShowHScrollBar</u></b>	Determines how the horizontal scroll bar is displayed .
<b><u>SSSetShowRowHeading</u></b>	Specifies whether row heading are displayed.
<b><u>SSSetShowSelections</u></b>	Specifies how selections are displayed.
<b><u>SSSetShowVScrollBar</u></b>	Determines how the vertical scroll bar is displayed.
<b><u>SSSetShowZeroValues</u></b>	Determines whether zero value cells are displayed.
<b><u>SSSetTopLeftText</u></b>	Specifies the text for the top left header.
<b><u>SSSetTopRow</u></b>	Sets the top row displayed in the view.

## Data Entry Function Call Summary

Data entry function calls allow you to enter, edit, and obtain data. The following table lists the data entry function calls.

<b>Data Entry Functions</b>	<b>Operation</b>
<a href="#"><u>SSCancelEdit</u></a>	Aborts edit mode and leaves current cell unchanged.
<a href="#"><u>SSEndEdit</u></a>	Exits edit mode and commits changes to current cell.
<a href="#"><u>SSGetAllowEditHeaders</u></a>	Returns the state of the edit headers flag.
<a href="#"><u>SSGetAllowInCellEditing</u></a>	Returns the state of the in-cell editing flag.
<a href="#"><u>SSGetEntry</u></a>	Returns the value of the current cell in edit mode format.
<a href="#"><u>SSGetEntryRC</u></a>	Returns the value of the specified cell in edit mode format.
<a href="#"><u>SSGetFormattedText</u></a>	Returns the value of the current cell as it appears in the worksheet.
<a href="#"><u>SSGetFormattedTextRC</u></a>	Returns the value of the specified cell as it appears in the worksheet.
<a href="#"><u>SSGetFormula</u></a>	Returns the text version of the formula in the active cell.
<a href="#"><u>SSGetFormulaRC</u></a>	Returns the text version of the formula of the specified cell.
<a href="#"><u>SSGetLogicalRC</u></a>	Returns the logical (True or False) value of the specified cell.
<a href="#"><u>SSGetNumber</u></a>	Returns the numeric value of the active cell.
<a href="#"><u>SSGetNumberRC</u></a>	Returns the numeric value of the specified cell.
<a href="#"><u>SSGetText</u></a>	Returns the text value of the active cell.
<a href="#"><u>SSGetTextRC</u></a>	Returns the text value of the specified cell.
<a href="#"><u>SSSetAllowEditHeaders</u></a>	Specifies whether header text can be edited.
<a href="#"><u>SSSetAllowInCellEditing</u></a>	Specifies whether in-cell editing is allowed.
<a href="#"><u>SSSetEntry</u></a>	Sets the value of the current cell in edit mode format.
<a href="#"><u>SSSetEntryRC</u></a>	Sets the value of the specified cell in edit mode format.
<a href="#"><u>SSSetFormula</u></a>	Sets the formula of the active cell.
<a href="#"><u>SSSetFormulaRC</u></a>	Sets the formula of the specified cell.
<a href="#"><u>SSSetLogicalRC</u></a>	Sets the logical value of a specified cell.
<a href="#"><u>SSSetNumber</u></a>	Sets the numeric value of the active cell.
<a href="#"><u>SSSetNumberRC</u></a>	Sets the numeric value of a specified cell.
<a href="#"><u>SSSetText</u></a>	Sets the text value of the active cell.
<a href="#"><u>SSSetTextRC</u></a>	Sets the text of a specified cell.
<a href="#"><u>SSStartEdit</u></a>	Begins edit mode.

## Printing Function Call Summary

Printing function calls allow you to set printing specifications and print worksheets. The following table lists the printing function calls.

<b>Printing Functions</b>	<b>Operation</b>
<a href="#"><u>SSAddColPageBreak</u></a>	Adds a vertical page break adjacent to the current cell.
<a href="#"><u>SSAddPageBreak</u></a>	Adds vertical and horizontal page breaks adjacent to the current cell.
<a href="#"><u>SSAddRowPageBreak</u></a>	Adds a horizontal page break adjacent to the current cell.
<a href="#"><u>SSFilePrint</u></a>	Prints a worksheet.
<a href="#"><u>SSGetPrintArea</u></a>	Returns the current print area.
<a href="#"><u>SSGetPrintBottomMargin</u></a>	Returns the bottom page margin used during printing.
<a href="#"><u>SSGetPrintColHeading</u></a>	Returns the print column heading flag.
<a href="#"><u>SSGetPrintFooter</u></a>	Returns the page footer.
<a href="#"><u>SSGetPrintGridLines</u></a>	Returns the print grid lines flag.
<a href="#"><u>SSGetPrintHCenter</u></a>	Returns the horizontal center flag.
<a href="#"><u>SSGetPrintHeader</u></a>	Returns the page header.
<a href="#"><u>SSGetPrintLeftMargin</u></a>	Returns the left page margin used during printing.
<a href="#"><u>SSGetPrintLeftToRight</u></a>	Returns the left to right flag.
<a href="#"><u>SSGetPrintNoColor</u></a>	Returns the print no color flag.
<a href="#"><u>SSGetPrintRightMargin</u></a>	Returns the right page margin used during printing.
<a href="#"><u>SSGetPrintRowHeading</u></a>	Returns the print row heading flag.
<a href="#"><u>SSGetPrintTitles</u></a>	Returns the print titles.
<a href="#"><u>SSGetPrintTopMargin</u></a>	Returns the top page margin used during printing.
<a href="#"><u>SSGetPrintVCenter</u></a>	Returns the vertical center flag.
<a href="#"><u>SSNextColPageBreak</u></a>	Returns the next column where there is a page break.
<a href="#"><u>SSNextRowPageBreak</u></a>	Returns the next row where there is a page break.
<a href="#"><u>SSRemoveColPageBreak</u></a>	Removes a vertical page break adjacent to the current cell.
<a href="#"><u>SSRemovePageBreak</u></a>	Removes vertical and horizontal page breaks adjacent to the current cell.
<a href="#"><u>SSRemoveRowPageBreak</u></a>	Removes a horizontal page break adjacent to the current cell.
<a href="#"><u>SSSetPrintArea</u></a>	Specifies the print area.
<a href="#"><u>SSSetPrintAreaFromSelection</u></a>	Sets the print range to the currently selected ranges.
<a href="#"><u>SSSetPrintBottomMargin</u></a>	Specifies the bottom page margin used during printing.
<a href="#"><u>SSSetPrintColHeading</u></a>	Specifies whether column headings are printed.
<a href="#"><u>SSSetPrintFooter</u></a>	Specifies the footer to print at the bottom of each page.
<a href="#"><u>SSSetPrintGridLines</u></a>	Specifies whether grid lines are printed.
<a href="#"><u>SSSetPrintHCenter</u></a>	Specifies whether the worksheet is horizontally centered when printed.
<a href="#"><u>SSSetPrintHeader</u></a>	Specifies the header to print at the top of each page.
<a href="#"><u>SSSetPrintLeftMargin</u></a>	Specifies the left page margin used during printing.
<a href="#"><u>SSSetPrintLeftToRight</u></a>	Specifies the order in which worksheet pages are printed.
<a href="#"><u>SSSetPrintNoColor</u></a>	Specifies whether the worksheet is printed in color.
<a href="#"><u>SSSetPrintRightMargin</u></a>	Specifies the right page margin used during printing.

<b><u>SSetPrintRowHeading</u></b>	Specifies whether row headings are printed.
<b><u>SSetPrintTitles</u></b>	Specifies titles to be printed at the top or left of each page.
<b><u>SSetPrintTitlesFromSelection</u></b>	Specifies the current selection as print titles to be printed at the top or left of each page.
<b><u>SSetPrintTopMargin</u></b>	Specifies the top page margin used during printing.
<b><u>SSetPrintVCenter</u></b>	Specifies whether the worksheet is vertically centered when printed.

## Range Editing Function Call Summary

Range editing function calls perform editing operations on worksheet ranges and the data they contain. The following table lists the range editing function calls.

<b>Range Editing Functions</b>	<b>Operation</b>
<a href="#"><u>SSCanEditPaste</u></a>	Determines if there is something on the internal clipboard that can be pasted to the worksheet.
<a href="#"><u>SSClearRange</u></a>	Clears the specified range.
<a href="#"><u>SSCopyAll</u></a>	Copies the contents of one worksheet to another.
<a href="#"><u>SSCopyRange</u></a>	Copies a range from one worksheet to another.
<a href="#"><u>SSDeleteRange</u></a>	Deletes cells, rows, or columns from the specified range.
<a href="#"><u>SSEditClear</u></a>	Clears all cells in the selected ranges.
<a href="#"><u>SSEditCopy</u></a>	Copies the selected range to the internal clipboard.
<a href="#"><u>SSEditCopyDown</u></a>	Copies cells in the top row of a selection down.
<a href="#"><u>SSEditCopyRight</u></a>	Copies cells in the left column of a selection to columns to the right.
<a href="#"><u>SSEditCut</u></a>	Cuts the selected range to the internal clipboard.
<a href="#"><u>SSEditDelete</u></a>	Deletes cells, rows, or columns from the selected range.
<a href="#"><u>SSEditInsert</u></a>	Inserts cells, rows, or columns in the selected range.
<a href="#"><u>SSEditPaste</u></a>	Pastes the internal clipboard to the selected range.
<a href="#"><u>SSGetAllowArrows</u></a>	Returns the state of the allow arrows flag.
<a href="#"><u>SSGetAllowDelete</u></a>	Returns the state of the allow delete flag.
<a href="#"><u>SSGetAllowFillRange</u></a>	Returns the state of the fill range flag.
<a href="#"><u>SSGetAllowMoveRange</u></a>	Returns the state of the move range flag.
<a href="#"><u>SSGetAllowTabs</u></a>	Returns the state of the allow tabs flag.
<a href="#"><u>SSGetEnterMovesDown</u></a>	Returns the state of the enter moves down flag.
<a href="#"><u>SSInsertRange</u></a>	Inserts cells, rows, or columns in the specified range.
<a href="#"><u>SSMoveRange</u></a>	Moves a range.
<a href="#"><u>SSSetAllowArrows</u></a>	Specifies whether arrow keys can move the active cell.
<a href="#"><u>SSSetAllowDelete</u></a>	Specifies whether the delete key deletes the current selection or a record.
<a href="#"><u>SSSetAllowFillRange</u></a>	Specifies whether filling by dragging a range is allowed.
<a href="#"><u>SSSetAllowMoveRange</u></a>	Specifies whether moving ranges by dragging is allowed.
<a href="#"><u>SSSetAllowTabs</u></a>	Specifies whether the tab key can move the active cell through a selected range.
<a href="#"><u>SSSetEnterMovesDown</u></a>	Specifies whether the enter key moves the active cell down to the next cell, even if a range is not selected.

## Recalculation Function Call Summary

Recalculation function calls control the manner in which worksheets are recalculated. The following table lists the recalculation function calls.

<b>Recalculation Functions</b>	<b>Operation</b>
<a href="#"><u>SSCalculationDlg</u></a>	Displays the Calculations dialog box.
<a href="#"><u>SSCheckRecalc</u></a>	Recalculates the worksheet if needed.
<a href="#"><u>SSGetAllowFormulas</u></a>	Returns the state of the user formula flag.
<a href="#"><u>SSGetAutoRecalc</u></a>	Returns the state of the automatic recalc flag.
<a href="#"><u>SSGetIteration</u></a>	Returns the iteration information.
<a href="#"><u>SSRecalc</u></a>	Recalculates the worksheet attached to a view.
<a href="#"><u>SSSetAllowFormulas</u></a>	Specifies whether the user is allowed to enter formulas.
<a href="#"><u>SSSetAutoRecalc</u></a>	Specifies whether automatic recalculation is enabled.
<a href="#"><u>SSSetIteration</u></a>	Sets the iteration information.

## Selection Function Call Summary

Selection function calls select cells and ranges of cells in the worksheet. The following table lists the selection function calls.

<b>Selection Functions</b>	<b>Operation</b>
<a href="#"><u>SSAddSelection</u></a>	Adds a new selection to the current selection list.
<a href="#"><u>SSFormatRCNr</u></a>	Creates a string containing a formatted row and column reference.
<a href="#"><u>SSGetActiveCell</u></a>	Returns the row and column of the active cell.
<a href="#"><u>SSGetAllowSelections</u></a>	Returns the state of the select range flag.
<a href="#"><u>SSGetHdrSelection</u></a>	Returns the state of the header selection flags.
<a href="#"><u>SSGetRowMode</u></a>	Returns the state of the row mode flag.
<a href="#"><u>SSGetSelection</u></a>	Returns the start and end row and column of the specified selection.
<a href="#"><u>SSGetSelectionCount</u></a>	Returns the number of selected ranges.
<a href="#"><u>SSGetSelectionRef</u></a>	Returns the current selection as a formula.
<a href="#"><u>SSSetActiveCell</u></a>	Sets the active cell to the specified row and column.
<a href="#"><u>SSSetAllowSelections</u></a>	Specifies whether selecting ranges is allowed.
<a href="#"><u>SSSetHdrSelection</u></a>	Specifies whether the column, row, and top left header are selected.
<a href="#"><u>SSSetRowMode</u></a>	Specifies whether individual cells or entire rows are selected.
<a href="#"><u>SSSetSelection</u></a>	Selects the specified range and moves the active cell to the top left cell in the range.
<a href="#"><u>SSSetSelectionRef</u></a>	Sets the current selection from a formula.
<a href="#"><u>SSSetShowSelections</u></a>	Specifies whether selections are displayed.
<a href="#"><u>SSShowActiveCell</u></a>	Positions the view to show the active cell if it is not currently in the window.

## Worksheet Function Call Summary

Worksheet function calls create, manipulate, and delete worksheets. The following table lists the worksheet function calls.

<b>Worksheet Functions</b>	<b>Operation</b>
<a href="#"><u>SSAttach</u></a>	Searches for a worksheet with the given title and attaches it to a view.
<a href="#"><u>SSAttachToSS</u></a>	Attaches a worksheet from one view to another.
<a href="#"><u>SSCheckModified</u></a>	Checks to see if the view or worksheet has been modified since the last SSM_MODIFIED; sends an SSM_MODIFIED message if necessary.
<a href="#"><u>SSDelete</u></a>	Deletes a view and its worksheet.
<a href="#"><u>SSDeleteTable</u></a>	Deletes a worksheet.
<a href="#"><u>SSInitTable</u></a>	Initializes a view.
<a href="#"><u>SSNew</u></a>	Creates a new worksheet view.
<a href="#"><u>SSSwapTables</u></a>	Exchanges the worksheets attached to two views.
<a href="#"><u>SSUpdate</u></a>	Updates all worksheets.



## Miscellaneous Function Call Summary

Miscellaneous function calls provide varied functionality including controlling events, defining names, reading and writing files, protecting cells, and manipulating worksheet rows and columns. The following table lists the miscellaneous function calls.

<b>Misc. Functions</b>	<b>Operation</b>
<a href="#"><u>SSCallWindowProc</u></a>	Used to pass Windows messages to the view.
<a href="#"><u>SSClearClipboard</u></a>	Clears the clipboard.
<a href="#"><u>SSDeleteDefinedName</u></a>	Deletes a user defined name.
<a href="#"><u>SSErrorNumberToText</u></a>	Returns the error text corresponding to the specified error number.
<a href="#"><u>SSGetDefinedName</u></a>	Returns the range definition for a user defined name.
<a href="#"><u>SSGetEnableProtection</u></a>	Returns the state of the enable protection flag.
<a href="#"><u>SSGetFireEvent</u></a>	Returns the flag for whether a given event is enabled.
<a href="#"><u>SSGetFixedCols</u></a>	Returns the starting and number of fixed columns.
<a href="#"><u>SSGetFixedRows</u></a>	Returns the starting and number of fixed rows.
<a href="#"><u>SSGetLastCol</u></a>	Returns the column number of the last occupied column.
<a href="#"><u>SSGetLastColForRow</u></a>	Returns the number of the last occupied column for the specified row.
<a href="#"><u>SSGetLastRow</u></a>	Returns the row number of the last occupied row.
<a href="#"><u>SSGetLeftCol</u></a>	Returns the leftmost column displayed in the view.
<a href="#"><u>SSGetRepaint</u></a>	Returns the repaint status of the worksheet.
<a href="#"><u>SSGetTitle</u></a>	Returns the title of the worksheet.
<a href="#"><u>SSGetTopRow</u></a>	Returns the top row displayed in the view.
<a href="#"><u>SSGetTypeRC</u></a>	Returns the cell type of the specified cell.
<a href="#"><u>SSMaxCol</u></a>	Returns the maximum number of columns supported by this version.
<a href="#"><u>SSMaxRow</u></a>	Returns the maximum number of rows supported by this version.
<a href="#"><u>SSRangeToTwips</u></a>	Returns the offset, height, and width in twips of a specified range.
<a href="#"><u>SSRead</u></a>	Reads a worksheet from disk.
<a href="#"><u>SSReadIO</u></a>	Reads a worksheet using a user specified read function.
<a href="#"><u>SSSaveWindowInfo</u></a>	Saves the window specific information from a view to its worksheet.
<a href="#"><u>SSSetAppName</u></a>	Defines the application name that appears in the title bar of error dialog boxes.
<a href="#"><u>SSSetDefinedName</u></a>	Defines or changes a user defined name.
<a href="#"><u>SSSetDefWindowProc</u></a>	Sets the default window procedure for a worksheet view.
<a href="#"><u>SSSetDoSetCursor</u></a>	Specifies how the cursor is set.
<a href="#"><u>SSSetEnableProtection</u></a>	Specifies that protection for cells marked as locked or hidden is enabled.
<a href="#"><u>SSSetFireEvent</u></a>	Determines whether an event is allowed to fire.
<a href="#"><u>SSSetFixedCols</u></a>	Sets the number of fixed columns.
<a href="#"><u>SSSetFixedRows</u></a>	Sets the number of fixed rows.
<a href="#"><u>SSSetProtection</u></a>	Specifies the protection of the currently selected cells.
<a href="#"><u>SSSetRepaint</u></a>	Sets the repaint status of the worksheet.
<a href="#"><u>SSSetTitle</u></a>	Sets the title of the worksheet.
<a href="#"><u>SSSetTopRow</u></a>	Sets the top row displayed in the view.

<a href="#"><u>SSSort</u></a>	Sorts the selected range of data with an unlimited number of sort keys.
<a href="#"><u>SSSort3</u></a>	Sorts the selected range of date with as many as three sort keys.
<a href="#"><u>SSTransactCommit</u></a>	Commits all changes since transaction began.
<a href="#"><u>SSTransactRollback</u></a>	Undoes all changes since transaction began.
<a href="#"><u>SSTransactStart</u></a>	Starts a transaction.
<a href="#"><u>SSTwipsToRC</u></a>	Returns the row and column that correspond to a given point.
<a href="#"><u>SSVBXCopyCellsFromDoubleArray</u></a>	Copies a two-dimensional array of numbers to a range.
<a href="#"><u>SSVBXCopyCellsToDoubleArray</u></a>	Copies a range of numbers to a two-dimensional array.
<a href="#"><u>SSVersion</u></a>	Returns the version number of the Formula One control.
<a href="#"><u>SSWrite</u></a>	Saves the worksheet to a file.
<a href="#"><u>SSWriteIO</u></a>	Writes a worksheet using the specified write function.

## **Using Events**

Formula One provides a set of 17 events that allow you to track and monitor actions performed on a worksheet control by users of your application. Events allow you to respond to users actions and control the operations of the worksheet control.

The following sections describe situations in which events can be used.

## Validating Data

You can use the [EndEdit](#) event to validate cell entries. If the cell entry is out of range, the proper actions can be taken. The following example demonstrates checking column 1 for values between 0 and 1000.

```
Sub Sheet1_EndEdit (EditString As String, Cancel As Integer)
    Dim Thevalue#
    Thevalue# = Val(EditString)

    ' Demonstrates how to check entries for a valid range

    If sheet1.Col = 1 Then
        If Thevalue# < 0 Or Thevalue# > 1000 Then
            Beep
            MsgBox "Value must be between 0 and 1000."
            Cancel = True
        End If
    End If
End Sub
```

## 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.

```
Sub Sheet1_DblClick (nrow As Long, nCol As Long)
    If nCol = 3 Then
        ' Here you would bring up another worksheet!
        MsgBox "Drilling on row " & nrow
    End If
End Sub
```

Formula One is especially capable of handling this type of model since it can handle multiple worksheets and external references.

## Events and Other Controls

It 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 list box; then, it displays the list box when the user clicks column five.

```
' Fill list box with font names when loading form 2

Sub Form_Load ()
    Dim i
    List1.Move      50, 50, 2000, 1750
    For i = 0 To screen.FontCount - 1
        List1.AddItem Screen.Fonts(i)
    Next i
End Sub

' Get the user's selection and put the value in the clicked cell

Sub List1_DblClick()
    popup$ = List1.List (List1.ListIndex)
    form1.Sheet1.Text = popup$
    Unload form2
End Sub

' Catch the click on the column and display the list box form

Sub Sheet1_DblClick (nrow As Long, nCol As Long)
    If nCol = 4 Then
        form2.Show
    End If
End Sub
```

Other controls can be displayed in the same manner, including built-in controls such as combo boxes and menus. In addition, many custom controls that are built in Visual Basic can be used in this manner.

## Event Summary

The following table lists the events available in Formula One.

<b>Event</b>	<b>Description</b>
<a href="#"><u>CancelEdit</u></a>	Occurs if the user leaves edit mode without making changes or presses the Escape key.
<a href="#"><u>Click</u></a>	Occurs when the user presses and releases the mouse button while the mouse pointer is in the Formula One window.
<a href="#"><u>DataNewRow</u></a>	Occurs when a new record is created.
<a href="#"><u>DataRowLoad</u></a>	Occurs after a new row is loaded from the data control.
<a href="#"><u>DbClick</u></a>	Occurs when the user double clicks the mouse button while the mouse pointer is in the Formula One window.
<a href="#"><u>DragDrop</u></a>	Occurs when a Drag-Drop operation is completed.
<a href="#"><u>DragOver</u></a>	Occurs when a Drag-Drop operation is in process.
<a href="#"><u>EndEdit</u></a>	Occurs when an editing operation is completed.
<a href="#"><u>EndRecalc</u></a>	Occurs when the recalculation process is completed.
<a href="#"><u>GotFocus</u></a>	Occurs when the Formula One window receives focus, either by clicking the object or changing the focus in code using the SetFocus method.
<a href="#"><u>KeyDown</u></a>	Occurs when the user presses a key while the Formula One object has the focus.
<a href="#"><u>KeyUp</u></a>	Occurs when the user releases a key while the Formula One object has the focus.
<a href="#"><u>KeyPress</u></a>	Occurs when the user presses and releases an ANSI key.
<a href="#"><u>LostFocus</u></a>	Occurs when the Formula One window loses focus.
<a href="#"><u>SelChange</u></a>	Occurs when the active cell or selected range changes.
<a href="#"><u>StartEdit</u></a>	Occurs when an editing operation is started.
<a href="#"><u>StartRecalc</u></a>	Occurs when the recalculation process is started.
<a href="#"><u>TopLeftChanged</u></a>	Occurs when the cell displayed as the top left cell of the worksheet changes.

## Using Views and Worksheets

Views and worksheets are important parts of Formula One. With an understanding of how views and worksheets work and interact, you can more effectively use the Formula One control.



Worksheets are objects that are maintained by the Formula One engine.



A view is a window into a specific worksheet.



## Working with Worksheets

A worksheet and a view of that worksheet are created when you draw a Formula One control on a form. When you double-click a worksheet to launch the Worksheet Designer, a second view of the worksheet is created.

Worksheets include:



cell data



cell formulas



worksheet formatting information



worksheet specific information such as printing attributes and calculation attributes.

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



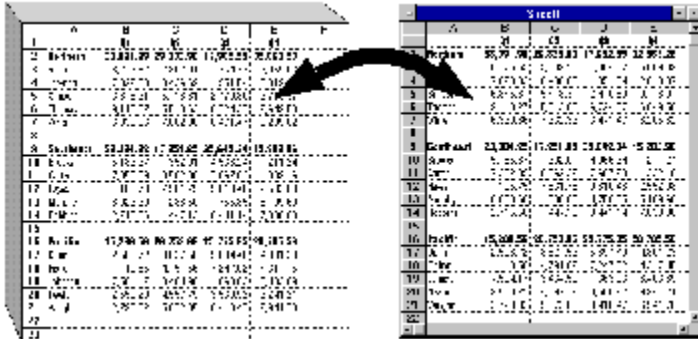
## Working with Views

Views provide access to and allow interaction with worksheets. Without a view, you cannot observe the work that you have performed in a worksheet.

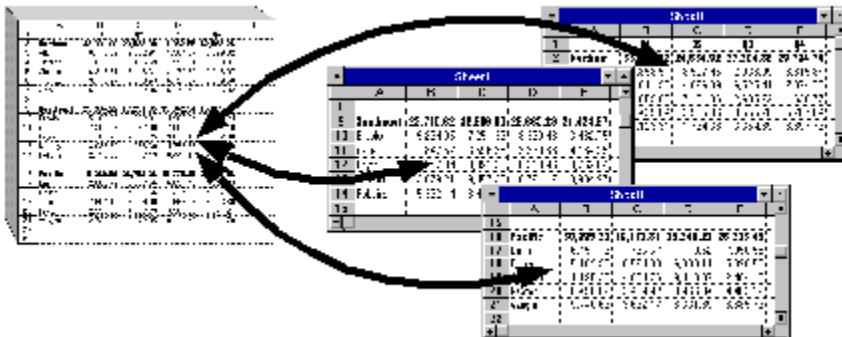
Each view has only one worksheet to which it is attached.

There can be multiple views into one worksheet.

When you place a Formula One control on a form, a view and a worksheet are automatically created. After a view is created, you can change the worksheet to which it is attached at any time.



*This figure illustrates the concept of one view and one worksheet.*



*This figure illustrates the concept of multiple views into one worksheet.*

## Using View Information

Associated with views is information that describes how the worksheet is displayed. Views contain information about:



grid line display



column and row heading display



fixed row and column specifications



maximum worksheet viewing size

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

Much of the information stored by the view can be accessed and changed through the control's properties.

## **Saving View Information**

When a worksheet is saved to a form or a file, the settings from the view that requested the save operation are saved with the worksheet. When a view is attached to a worksheet, the view settings are retrieved from the worksheet.

## Attaching Views to Worksheets

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

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



A view can be connected to only one worksheet.



A worksheet can have multiple views to which it is attached.

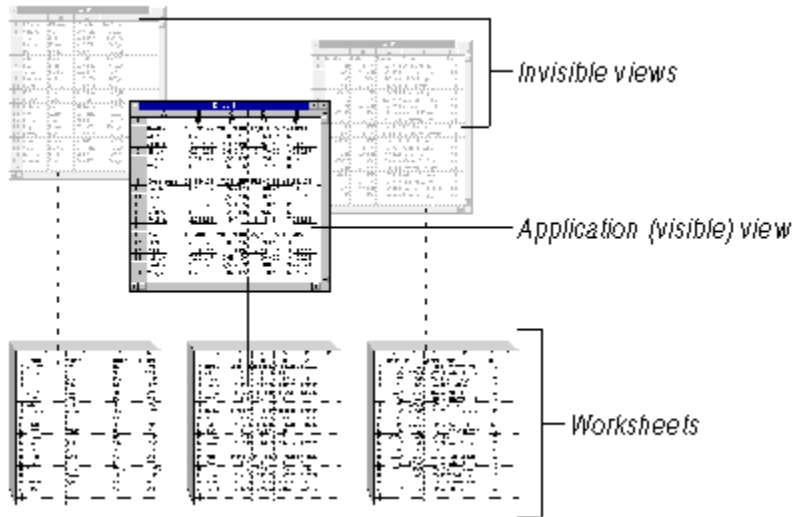


A worksheet must have at least one view to which it is attached. A worksheet ceases to exist if it is not attached to a view.

## One View with Multiple Worksheets

Because a view can be connected to only one worksheet, you must employ invisible views to accommodate an application that uses one view with multiple worksheets.

To accomplish this, set the **Visible** property to False for all views in your application except the view you want displayed. Then, use the **SSSwapTables** function call to connect any worksheet to the visible view.



## One Worksheet with Multiple Views

When multiple views are attached to a single worksheet, 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 worksheet.

The manner in which you attach multiple views to a single worksheet depends on whether the worksheet is stored as a file or on a form.



If you are loading a worksheet from disk, the views should have the same **FileName** property and the same **TableName** property. This causes the worksheet to load once, and the views are connected to the same worksheet.



If the worksheet is saved with a form, or if it is created dynamically, you can use the **SSAttachToSS** function call to attach views. The worksheet is attached to the current view; however, the worksheet also remains attached to any previous views to which it was attached.

## Saving Worksheets

Each worksheet control can be saved with the form on which it resides or to a separate file. The **FileName** property determines where worksheets are stored. The method you use for your application depends on which is more advantageous.



Saving worksheets with the form. This method reduces the number of files needed to run an application. It also reduces the potential that a worksheet can be separated from its application.

If the **FileName** property is blank, the worksheet is stored with the form.



Saving worksheets in separate files. This method allows files to be shared among multiple Formula One users as well as with other applications such as Excel. Worksheets saved to a file are also easier to create and modify.

If the **FileName** property contains a valid path and file name, the worksheet is stored in a file.



## Reading and Writing Files

Formula One can read and write two different file formats. The following table lists the formats and the associated file name extensions.

File extension	Description
.VTS	Formula One native format; an extension of Excels BIFF 4 format.
.XLS	BIFF 4 format

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 worksheet.

If the file you save contains features not supported by Excel, they are removed when the worksheet 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.

**Important** If you load an Excel worksheet that contains features not supported by Formula One, such as graphics, those features are ignored. If the imported worksheet is written from Formula One as an Excel worksheet 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.

Formula One applications read and write worksheets using the [ReadFile](#), [WriteFile](#) and [WriteExcel4](#) properties. The **WriteExcel4** property writes an Excel 4.0 compatible worksheet. In the following examples, a native worksheet and an Excel worksheet are read.

```
Sheet1.ReadFile = "c:\vtss\samples\amortize.vts" ' Reads Native
Sheet1.ReadFile = "c:\vtss\samples\amortize.xls" ' Reads Excel 4.0
```

In the following examples, a worksheet is written to a file twice, once as a native worksheet and once as an Excel worksheet.

```
Sheet1.WriteFile = "c:\vtss\samples\newone.vts" ' Write Native
Sheet1.WriteExcel4 = "c:\vtss\samples\newone.xls" ' Write Excel
```

## Using the Worksheet Designer

The Worksheet Designer is a Windows application that can be accessed directly from a Formula One control, either at design time or run time. The designer allows you to visually design worksheet controls for your application. With the designer, you can:



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



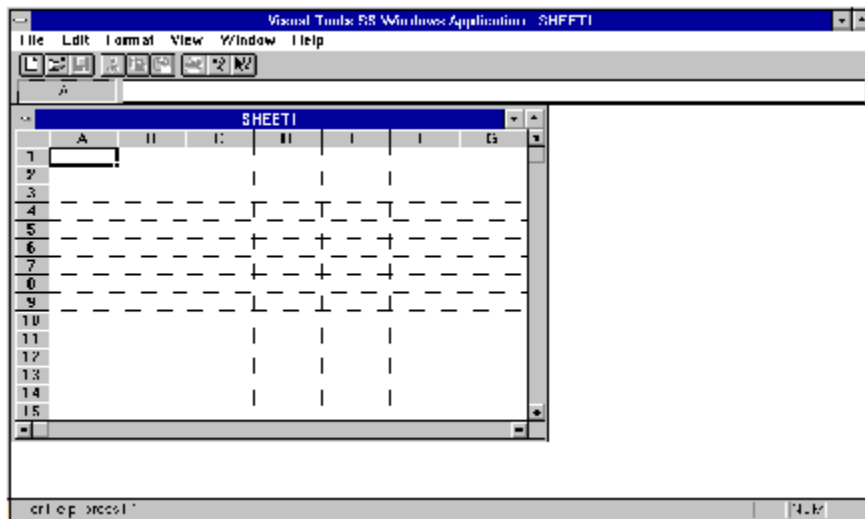
protect and hide cells



set user permissions



select items to be shown and hidden



*The Worksheet Designer appears and behaves much like a commercial spreadsheet application.*

The Worksheet Designer is accessed by double clicking a worksheet control with the right mouse button. When launched, a new window is displayed containing the Worksheet Designer.

The Worksheet Designer looks much like a commercial spreadsheet application. In fact, it is a stand-alone Windows application that accesses the Formula One engine. Any changes made to the worksheet in the Worksheet Designer are reflected in the Formula One worksheet control on your form. Using the Worksheet Designer greatly accelerates application development.

Refer [Worksheet Designer Overview](#) for information about the menus and commands available in the Worksheet Designer.

## Using Edit Bar Controls

The edit bar control is similar to the edit bar on most commercial spreadsheets such as Excel or 1-2-3. The edit bar is used to enter or edit data and formulas.

The edit bar is optional and is most applicable when in-cell editing is not appropriate. For example, long entries, particularly formulas, can be more easily entered and edited using an edit bar. If your application manipulates data and formulas using Visual Basic code or with in-cell editing, you do not need an edit bar.

=SUM(A1:A5)						
	A	B	C	D	E	↑
1	335.15	929.00	753.90			
2	834.49	226.41	313.56			
3	54.55	784.75	351.22			
4	955.35	568.03	844.71			
5	33.35	536.36	614.26			
6	SUM(A1:A5)					
7						
8						
9						
10						
+						→

## Creating Edit Bar Controls

The edit bar is a separate custom control that works in conjunction with the worksheet control. Each worksheet control has a property called **EditName** to specify with which edit bar control it is used.

If **EditName** is blank, the user cannot enter or edit data or formulas unless the AllowInCellEditing property is True.

If the **EditName** property of a worksheet control matches the **EditName** property of an existing edit bar, the existing edit bar interacts with the worksheet control.

In simple applications, you can use one edit bar with one worksheet control. For more complex applications, you can use one or more edit bars with multiple worksheet controls.

For example, if your application contains two or more worksheet controls on the same form, you can use one edit bar for all the worksheet controls. All worksheet controls can reference the same edit bar by setting the **EditName** property of the worksheet controls to match the name of the edit bar. Then, the edit bar interacts with the worksheet control that is active.

Accounts Receivable						
	A	B	C	D	E	
1	Accounts Payable					
2						
3						
4						
5						
6						
7						
8						
9						

	A	B	C	D	E	↑
1	Receivable					
2						
3						
4						
5						
6						
7						
8						↓
←						→

*This illustration shows one edit bar with two worksheet controls.*

When you place a new worksheet control on a form, its **EditName** property is set to the default name, SSEdit1. This is also the default setting of the **EditName** property for a newly created edit bar.

If you add more than one worksheet control to a form, they all have SSEdit1 as the default edit bar name. As a result, multiple worksheet controls work with one edit bar without any programmer intervention.

When you place a second edit bar on a form, its **EditName** property is also set to SSEdit1. To eliminate confusion, you should change the **EditName** property of the second edit bar and its corresponding worksheet control.

## Edit Bar Properties

The **EditName** property is the only non-standard Visual Basic property. When connecting an edit bar to a worksheet control, the **EditName** property in both controls must match.

The edit bar control contains the following standard Visual Basic properties:

### Properties

---

<a href="#"><u>DragIcon</u></a>	<a href="#"><u>HelpContextID</u></a>	<a href="#"><u>TabStop</u></a>
<a href="#"><u>DragMode</u></a>	<a href="#"><u>Index</u></a>	<a href="#"><u>Tag</u></a>
<a href="#"><u>EditName</u></a>	<a href="#"><u>Left</u></a>	<a href="#"><u>Top</u></a>
<a href="#"><u>Enabled</u></a>	<a href="#"><u>Name</u></a>	<a href="#"><u>Visible</u></a>
<a href="#"><u>Height</u></a>	<a href="#"><u>TabIndex</u></a>	<a href="#"><u>Width</u></a>

For additional information about edit bar properties, refer to the property descriptions in the Microsoft Visual Basic Language Reference Manual.

## Edit Bar Events

The edit bar control contains the following standard Visual Basic events:

### Events

---

[Click](#)

[DragDrop](#)

[DragOver](#)

## Edit Bar Function Calls

Formula One provides the following function calls that can be used with edit bars:

### Function calls

---

[SSEditBarDelete](#)    [SSEditBarMove](#)    [SSGetSSEdit](#)

[SSEditBarHeight](#)    [SSEditBarNew](#)    [SSSetSSEdit](#)



## Worksheet Fundamentals

Before you can successfully use a worksheet control, you must understand some basic concepts about the worksheet. You must understand how to select cells, ranges, rows, and columns, enter and delete data, and display specific sections of a worksheet.

The following sections discuss:



navigating through a worksheet with keyboard commands.



mouse actions executed in the worksheet control.



selecting cells and ranges with the mouse, properties, and function calls.



selecting entire rows and columns.

## Navigating through Worksheets

When working in the Worksheet Designer or in a worksheet at run time, you can navigate through a worksheet 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 worksheet



delete data from a selected cell or range

## Using Keyboard Commands

The tables in this section list the keyboard commands you can use when working in the Worksheet Designer or a worksheet 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 worksheet.

<b>Key</b>	<b>Description</b>
Enter	When in edit mode, accepts the current entry. When a range is selected, 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, accepts the current entry and moves active cell vertically to previous cell in selection.
Tab	When in edit mode, accepts the current entry. When a range is selected, accepts the current entry and moves active cell horizontally to next cell in selection.
Shift-Tab	When in edit mode, accepts the current entry. When a range is selected, accepts the current entry and moves active cell horizontally to previous cell selection.
F2	Enters edit mode.
F9	Recalculates worksheet.
Del	Clears current selection or deletes the current record depending on the setting of the <a href="#">AllowDelete</a> property.
Escape	Cancels current data entry or editing operation. If you are not editing and are currently in a database row, refreshes current database row.

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.

<b>Key</b>	<b>Description</b>
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.

<b>Key</b>	<b>Description</b>
Scroll lock	Causes the worksheet 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

Primarily the mouse is used to select items in a worksheet at run time. In addition, mouse actions at design time can select the worksheet control, display the control code window, and launch the Worksheet Designer. The following table lists the mouse actions you can perform at design time.

Action	Description
Left Click or Right Click	Selects the Formula One control.
Left Double Click	Displays the code window for the Formula One control.
Right Double Click	Launches and displays the Worksheet Designer application.

The following table lists the mouse actions you can perform at run time or in the Worksheet Designer.

Action	Description
Left Click	Moves the active cell to the pointer position.
Right Click	Does nothing.
Left Click in Row or Column Headings	Selects entire row or column.
Left Click in Top Left Corner	Selects entire worksheet.
Left Double Click in Top Left Corner, Row Headings, or Column Headings	Displays a dialog box that allows you to enter a label for the top left corner or the column or row heading that was double clicked. Available only in the Worksheet Designer.
Left Double Click	In the Worksheet Designer, invokes in-cell editing. At run time, if the <a href="#">DoDbClick</a> property is True, a <a href="#">DbClick</a> event is fired. If the property is False, in-cell editing is invoked (if the <a href="#">AllowInCellEditing</a> property is True).
Right Double Click	In the Worksheet Designer, does nothing. At run time, if the <a href="#">AllowAppLaunch</a> property is True, the Worksheet Designer application is launched.
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 worksheet.
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.

## Selecting Cells

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

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					

Single cell selection

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					

Single range selection

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					

Multiple range selection

## 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 with Properties

The first range selected is always reflected in the [SelStartRow](#), [SelStartCol](#), [SelEndRow](#), and [SelEndCol](#) properties. You can also set these properties to select a range. For example, to select the range C2:E4, use the following Visual Basic code:

```
Sheet1.SelStartRow = 2 ' Row 2
Sheet1.SelStartCol = 3 ' Column C
Sheet1.SelEndRow = 4 ' Row 4
Sheet1.SelEndCol = 5 ' Column E
```

These properties are the easiest way to determine the current selection, or to create a selection in preparation for performing a selection-based operation (e.g., copying data). For example, the following code selects a range and copies the top row down to the rows below.

```
Sheet1.SelStartRow = 2 ' Row 2
Sheet1.SelStartCol = 3 ' Column C
Sheet1.SelEndRow = 4 ' Row 4
Sheet1.SelEndCol = 5 ' Column E
sserror = SSEditCopyDown (Sheet1.SS)
```

Using properties to perform this task is much faster than performing the same operation with Visual Basic code.



## Using the Selection Property

All selections are reflected in the [Selection](#) property. This property contains a text representation of the cells selected in the worksheet. You can use this property to select a cell, range, or multiple ranges.

For example, to select the ranges A1:C3 and A11:C13, set the **Selection** property to "A1:C3, A11:C13". This property can be set to any Formula One formula that returns one or more ranges.

## Selecting Cells with Function Calls

Function calls can be used to select ranges.

**SSSetSelection** removes all current selections and selects a range.

**SSAddSelection** adds a selection to the current selection list. Continue calling **SSAddSelection** to create multiple selections.

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

```
sserror = SSSetSelection (Sheet1.SS, 1, 1, 4, 4)      ' Select A1:D4
sserror = SSAddSelection (Sheet1.SS, 5, 5, 8, 8)     ' Add E5:H8
```

In addition, **SSGetSelectionCount**, **SSGetSelection**, **SSGetSelectionRef** are function calls that help you work with multiple selections.

**SSGetSelectionCount** returns the number of selections. Use this function if a selection is made by the user and you need to determine how many ranges are selected.

**SSGetSelection** returns all current selections in the form of a formula (e.g. A1:D4,E5:H8) The formula returned by this function call is the same as the string contained by the **Selection** property.

**SSGetSelectionRef** returns the row and column reference for an individual selection.

## **Selecting Rows and Columns**

Entire rows and columns can be selected in the worksheet at run time or in the Worksheet 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 in the upper left corner of the worksheet and click.

## Selecting Rows and Columns with Properties

The [SelStartRow](#) and [SelStartCol](#) properties can be used to select entire rows and columns. To select an entire row, set **SelStartCol** to -1; to select an entire column, set **SelStartRow** to -1.

For example, if you want to select all of column 2 and column 3, use the following code:

```
Sheet1.SelStartRow = -1 ' Selects all rows
Sheet1.SelStartCol = 2 ' Selects starting column as 2
Sheet1.SelEndCol = 3 ' Selects ending column as 3
```

Notice that the [SelEndRow](#) property is not set in the preceding sample code. Since setting **SelStartRow** to -1 selects all rows, **SelEndRow** is set automatically. After the previous example, **SelStartRow** returns 1 and **SelEndRow** returns 16384.

**NOTE** You can also use -1 to select rows or columns in the [Row](#) and [Col](#) properties, and any function call that requires row and column parameters.

---

## Working with Data

Entering and manipulating data is the basis for nearly all work performed in a Formula One control. In a worksheet control, you can enter virtually any type of data and formula. With formulas and built-in functions, you can evaluate and calculate that data and make decisions based on the results of those operations.

The following help topics discuss:



how to enter data directly, with properties, and with function calls.



how to limit user data entry.



the types of constant values that can be entered.



how to construct and use formulas.



the suite of built-in worksheet functions



using names.



the methods for calculating worksheets.

## Worksheet Data Entry

One of the basic tasks encountered when working with a worksheet control 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 the worksheet control at run time. Or, you can enter data in the Worksheet Designer at design time.



**Properties.** Numbers, text, and formulas can be entered in the active cell via properties.



**Function calls.** Several function calls are provided that allow you to enter data in the active cell or a specified cell.

## Entering Data with Properties

The following table lists the properties that can enter data in the active cell.

Property	Description
<a href="#">Entry</a>	Specifies data of any type for the active cell.
<a href="#">Formula</a>	Specifies a formula, as a text string, for the active cell.
<a href="#">Number</a>	Specifies a numeric value for the active cell.
<a href="#">Text</a>	Specifies a text string for the active cell.

**Formula**, **Number**, and **Text** are run time only properties. In addition to placing data in a cell, all three properties can retrieve data from a cell.

When specifying a formula with the **Formula** property, the formula should be provided as a text string without the leading equal sign (=).

To specify the cell that is active, you can use the [Row](#) and [Col](#) properties. The **Row** property specifies the row containing the active cell; the **Col** property specifies the column containing the active cell.

The following example uses the **Formula** property to place the [RAND](#) function in columns 1 through 10 of row 1.

```
Dim TheCol%  
Sheet1.Row = 1  
For TheCol = 1 to 10  
    Sheet1.Col = TheCol  
    Sheet1.Formula = "RAND() "  
Next TheCol
```

## Entering Data with Function Calls

Formula One provides a full complement of function calls for entering data. In addition to entering data in the active cell, function calls allow you to enter data in a cell other than the active cell.

The following table lists the function calls that enter data.

<b>Function call</b>	<b>Description</b>
<a href="#"><u>SSSetActiveCell</u></a>	Sets the active cell in the worksheet.
<a href="#"><u>SSSetEntry</u></a>	Sets the value of the current cell in edit mode format.
<a href="#"><u>SSSetEntryRC</u></a>	Sets the value of the specified cell in edit mode format.
<a href="#"><u>SSSetFormula</u></a>	Sets the formula of the active cell.
<a href="#"><u>SSSetFormulaRC</u></a>	Sets the formula of the specified cell.
<a href="#"><u>SSSetLogicalRC</u></a>	Sets the logical value of the specified cell.
<a href="#"><u>SSSetNumber</u></a>	Specifies the numeric value of the active cell.
<a href="#"><u>SSSetNumberRC</u></a>	Sets the numeric value of the specified cell.
<a href="#"><u>SSSetText</u></a>	Sets the text value of the active cell.
<a href="#"><u>SSSetTextRC</u></a>	Sets the text of the specified cell.



## Limiting Data Entry

Some applications may require that the user not be allowed to enter or edit data. To prevent data entry, the **AllowInCellEditing** property must be False, and there must not be an edit bar whose **EditName** property matches the **EditName** property of the worksheet control. Data and formula entry and editing is thus prevented. Any data manipulation must be performed through program code.

## 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.

## Locking Cells

To set editing permissions on a per cell basis, set the locked attribute of each cell to the appropriate value with the Cell Protection command in the Format menu of the Worksheet Designer. Then, enable protection for the worksheet with the Enable Protection command.

You can also set the locked status of the currently selected cells with the [SSSetProtection](#) function call.

## 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

**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 displayed as a series of number signs across the cell (e.g., #####). You must widen the cell to display the number.

Use the **SSSetColWidthAuto** function call 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.

```
sserror = SSSetColWidthAuto (Sheet1.SS, 1, 10)
```

**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. To enter a number as text, precede it with a single quotation mark (').

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 data alignment 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?.

## 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 in the [Formula](#) property or the [SSetFormula](#) and [SSetFormulaRC](#) function calls, 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.

## 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).
	Space	Intersection - produces a reference that contains all cells common to the two references (e.g., A1:A10 A10:A20 returns A10).
	,	Union - produces one reference that includes the two references (e.g., A1:A10,C1:C10).

## 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
Space	Intersection
,	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.



## **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.

## 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 or moved, 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 or moved, the reference does not change. Absolute references are designated by placing a dollar sign (\$) in front of the row or 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.

<b>Reference</b>	<b>Type</b>
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.

## External References

References can point to cells in other worksheets. This type of reference is called an external reference. An external reference is created by placing a worksheet name before the cell reference, separated by an exclamation point. The following table shows examples of external references.

<b>Reference</b>	<b>Type</b>
Sales!A1	Relative reference pointing to cell A1 in the worksheet named Sales.
FY91!\$A\$1	Absolute reference pointing to cell A1 in the worksheet named FY91.
Q1!\$A1	Absolute column reference, relative row reference pointing to cell A1 in the worksheet named Q1.
Store1!A\$1	Relative column reference, absolute row reference pointing to cell A1 in the worksheet named Store1.

## 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.

## 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.

<b>Error</b>	<b>Cause</b>
#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.

## Displaying Formulas

It is often convenient to display formula text instead of the values they produce. [SSetShowFormulas](#) 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.

```
sserror = SSetShowFormulas (sheet1.SS, True) ' Displays formulas
sserror = SSetShowFormulas (sheet1.SS, False) ' Displays formula text
```

## Custom Functions

Formula One allows you to create custom functions. Custom functions must be supplied in a DLL. Use the worksheet function [CALL](#) to call custom functions.

Refer to the sample worksheet CUSTFUNC.VTS for an example of a custom function call in a worksheet. The sample worksheet calls a custom function from the CUSTFUNC.C DLL file. Both files are provided with your Formula One installation media.

## Built-In Worksheet Functions

Formula One contains a set of 125 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

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.

## **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.

## 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)
```

## Entering Arguments

The arguments for a function can be:



Numerical values



Logical values



Text strings



Error values



References to cells or ranges

Each argument requires a specific type of data. Refer to the [Worksheet Function Reference](#) to determine the type of data required for the function you are entering.

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.

## **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.

## 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 " $= \text{Sales} - \text{Expenses}$ " 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  $=\text{SQRT}(2)$ .

You can define names using the Worksheet Designer. Or, you can define names using the [SSSetDefinedName](#) function call. The following code uses this function call to define a name.

```
sserror = SSSetDefinedName (Sheet1.SS, "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.

<b>Built in Name</b>	<b>Purpose</b>
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.

## **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 insures that the formula results are always correct.

When the 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 the worksheet current, insuring that data is always valid.

## 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 or the [SSSetAutoRecalc](#) function call. After the completion of an operation, automatic recalculation can be enabled and the worksheet updated.



## 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 [SSSetIteration](#) and [SSCalculationDlg](#) function calls. These functions 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.*

## Editing Worksheets

Formula One provides a variety of methods for moving data within a worksheet.



Data can be cut, copied, and pasted using function calls.



You can interactively copy data in a worksheet by clicking and dragging the copy handle on a worksheet selection.



Data can be moved interactively by clicking and dragging the border of a worksheet selection.



Function calls allow you to insert data in and delete data from ranges, rows, and columns.



A selected range of data can be sorted according to keys that you specify.

## Cut, Copy, and Paste Function Calls

Ranges of data can be edited using one of several editing function calls. 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, formatting, and formulas; cell references are not maintained by the Windows clipboard.

The following table describes the function calls that interact with the clipboards.

<b>Function Name</b>	<b>Operation</b>
<a href="#"><u>SSClearClipboard</u></a>	Clears the internal clipboard.
<a href="#"><u>SSEditCopy</u></a>	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.
<a href="#"><u>SSEditCut</u></a>	Cuts the current selection to the internal clipboard. If there is more than one selection, only the first selection is cut.
<a href="#"><u>SSEditPaste</u></a>	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.
<a href="#"><u>SSCanEditPaste</u></a>	Determines if the internal clipboard or the Windows clipboard contains data.



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.

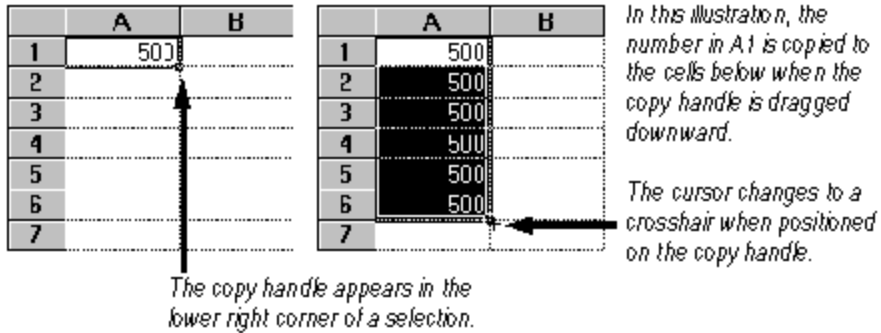
## Copying Data Across Ranges

Three function calls copy data within and between worksheets. The following table describes these function calls.

<b>Function Name</b>	<b>Operation</b>
<a href="#"><u>SSEditCopyDown</u></a>	Copies the top row of the selection down. Relative references are automatically adjusted.
<a href="#"><u>SSEditCopyRight</u></a>	Copies the left column of the selection right. Relative references are automatically adjusted.
<a href="#"><u>SSCopyRange</u></a>	Copies a range from one range to another, within the same worksheet or between worksheets.

## 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.



You can disable the user's ability to copy data by setting the [AllowFillRange](#) property to False. Or, you can call the [SSSetAllowFillRange](#) function to disable interactive data copying.

## Moving Data

Several methods can be used to move ranges of data. The easiest method uses the [SSMoveRange](#) function call. When you use this function call, 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 Visual Basic code to move the data. However, cell references are not adjusted using this technique.

## Moving Data Interactively

You can move data interactively by dragging a selection to a new location. This is accomplished by positioning the pointer on the border of the selection you want to move. When placed on the selection border, the pointer changes to an arrow. You can then drag the selection to a new location.

	A	B	C	D	E	F	G	H
1		Q1	Q2	Q3	Q4			
2	Americas	7023.05	5402.05	5577.19	4424.5			
3	Asia	7023.05	7700.54	5402.05	9067.54			
4	Europe	4734.65	4873.25	6907.21	3071.25			
5		18650.85	18191.35	21556.01	18351.25			
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

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

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

	A	B	C	D	E	F	G	H
1		Q1	Q2	Q3	Q4			
2	Americas	7023.05	5402.05	5577.19	4424.5			
3	Asia	7023.05	7700.54	5402.05	9067.54			
4	Europe	4734.65	4873.25	6907.21	3071.25			
5		18650.85	18191.35	21556.01	18351.25			
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

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

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11			Q1	Q2	Q3	Q4		
12		Americas	7023.05	5402.05	5577.19	4424.5		
13		Asia	7023.05	7700.54	5402.05	9067.54		
14		Europe	4734.65	4873.25	6907.21	3071.25		
15			18650.85	18191.35	21556.01	18351.25		
16								

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 the **AllowMoveRange** property to False. Or, use the **SSSetAllowMoveRange** function call to disable interactive data moving.

If you press the CONTROL key as you click and drag a selection border, a copy of the selected range is created and moved as you drag the pointer. The copied range is placed at the point where you release the mouse button. The original range is not moved.

## Inserting Cells, Rows, and Columns

The **SSInsertRange** function call inserts new cells in a worksheet. For this function call, 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.

```
sserror = SSInsertRange(Sheet1.SS, 2, 2, 3, 3, kShiftVertical)
```

The **SSEditInsert** function call can insert cells, rows, and columns. You specify whether rows, columns, or cells should be inserted. This function 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.

```
sserror = SSEditInsert (Sheet1.SS, kShiftRows)
```

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

```
sserror = SSEditInsert (Sheet1.SS, kShiftColumns)
```

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

```
sserror = SSEditInsert (Sheet1.SS, kShiftHorizontal)
```

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

```
sserror = SSEditInsert (Sheet1.SS, kShiftVertical)
```

The shift constants (e.g., kShiftRows, kShiftColumns, kShiftHorizontal, kShiftVertical) are defined in VTSS.H and VTSS.TXT.



## Clearing and Deleting Cells, Rows, and Columns

Several function calls delete and clear data. The following table lists these function calls.

Function Name	Operation
<a href="#"><u>SSEditDelete</u></a>	Deletes the current selection.
<a href="#"><u>SSDeleteRange</u></a>	Deletes the specified range.
<a href="#"><u>SSEditClear</u></a>	Clears the current selection.
<a href="#"><u>SSClearRange</u></a>	Clears the specified range.

**SSEditDelete** is similar to the [SSEditInsert](#) function call. For **SSEditDelete**, 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 Visual Basic code:

```
sserror = SSEditDelete (Sheet1.SS, kShiftRows)
```

If you delete cells (e.g., using **SSEditDelete** or **SSDeleteRange**) 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 **SSDeleteRange** function call. This function call allows you to explicitly specify the range to delete. The following code uses this function call.

```
sserror = SSDeleteRange (Sheet1.SS, 1, 1, 3, 3, kShiftRows)
```

Clearing a cell clears the value, format, and formula from that cell, but 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 **SSEditClear** or **SSClearRange** to clear a cell or range of cells. The following example clears the current selection.

```
sserror = SSEditClear (Sheet1.SS)
```

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

```
sserror = SSClearRange (Sheet1.SS, 1, 1, 3, 3)
```

## Sorting Worksheets

You can sort the data in a worksheet and specify the keys by which the data is sorted. [SSortDlg](#) 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 [SSort](#) function call. This function call provides the same functionality as the sort dialog box. Refer to Chapter 16, AZ Function Call Reference, for information about this function call.

For Visual Basic programmers, sorting data is made easier with the [SSort3](#) function call. This function call allows you to specify the sort keys directly in the function call parameters; **SSort** references an array to determine sort keys. However, **SSort3** limits you to three sort keys.

## **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

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

Category	Format	3	-3	.3
All	General	3	-3	.3
Fixed	0 3	-3	0	
	0.003.00	-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
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
Percentage	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

## **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 row containing formatted, empty cells consumes memory.

## Obtaining Formatted Text

You can obtain the formatted text from a cell by using the [FormattedText](#) property or the [SSGetFormattedText](#) or [SSGetFormattedTextRC](#) function calls. These function calls return text exactly as it is displayed in the worksheet.

## 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"
```

In the Worksheet Designer, custom number formats can be defined by choosing Custom Number in the Format menu. This command displays the Custom Format dialog box where you can enter custom formats.

If you want to specify a custom format by function call, use the [SSSetNumberFormat](#) function call. The following code uses **SSSetNumberFormat** to format numbers in the current selection with two decimal places and negative numbers with parentheses.

```
sserror = SSSetNumberFormat (Sheet1.SS, "#,##0.00_);(#,##0.00)")
```

The Custom Format dialog box can also be displayed by calling [SSFormatNumberDlg](#). 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.

```
sserror = SSFormatNumberDlg (Sheet1.SS)
```

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 0s 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.
[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.

If you are using the Worksheet Designer, data alignment can be set by choosing Alignment from the Format menu. This command displays 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.

The [SSSetAlignment](#) function call also allows you 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:

```
sserror = SSSetAlignment (Sheet1.SS, 2, False, 3, 0)
```

In the preceding example, 2 specifies that the cell data is left aligned, False specifies that word wrap is disabled, 3 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).

The Alignment dialog box can also be invoked by calling the [SSFormatAlignmentDlg](#) function call. The following code invokes the Alignment dialog box.

```
sserror = SSFormatAlignmentDlg (Sheet1.SS)
```

## **Changing Row Heights and Column Widths**

The width of columns and the height of rows can be changed interactively or set with function calls. Interactive column and row sizing can be performed in the Worksheet Designer at design time or in a worksheet control at run time.

## 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.

## Sizing Rows and Columns with Function Calls

The following table lists the function calls that allow you to size rows and columns.

Function Name	Operation
<a href="#"><u>SSSetColWidth</u></a>	Sets the width of the specified columns. Column width is specified in units of 1/256 of an average characters width in the default font.
<a href="#"><u>SSSetColWidthAuto</u></a>	Automatically sets the width of the specified columns to accommodate the largest data in the column.
<a href="#"><u>SSColWidthDlg</u></a>	Displays the Column Width dialog box.
<a href="#"><u>SSSetRowHeight</u></a>	Sets the height of the specified rows. Row height is specified in twips (one twip equals 1/1440 inch).
<a href="#"><u>SSSetRowHeightAuto</u></a>	Automatically sets the height of the specified rows to accommodate the tallest data in the row.
<a href="#"><u>SSRowHeightDlg</u></a>	Displays the Row Height dialog box.

SSSetRowHeight and SSSetColWidth 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.

```
sserror = SSSetRowHeight (Sheet1.SS, 1, 10, 720, FALSE)
sserror = SSSetColWidth (Sheet1.SS, 1, 10, 2560, FALSE)
```

SSSetColWidthAuto and SSSetRowHeightAuto 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).

```
sserror = SSSetRowHeightAuto (Sheet1.SS, 1, 1, 10, 10, True)
sserror = SSSetColWidthAuto (Sheet1.SS, 1, 1, 10, 10, True)
```

## Setting Cell Borders and Colors

Cells and ranges can be formatted with borders, colors, and patterns. These attributes can be set using the Worksheet Designer or function calls.

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.

From the Worksheet Designer:



Choose Border from the Format menu to specify the borders for the currently selected cells. The Borders dialog box is displayed when you choose this command.



To specify cell colors and patterns, choose Pattern from the Format menu. This command invokes the Pattern dialog box.

**SSSetBorder** is the function call that sets the border, outline, shading, and color for the selected cells. The following code uses this function call:

```
sserror = SSSetBorder (Sheet1.SS, 2, 1, 1, 1, 1, 1, 3, 4, 4, 4, 4)
```

**SSSetPattern** is the function call that sets the color and pattern for the selected cells. The following code uses this function call:

```
sserror = SSSetPattern (Sheet1.SS, 2, 128, 0)
```

To invoke the Borders dialog box by function call, call **SSFormatBorderDlg**. The following code displays this dialog box:

```
sserror = SSFormatBorderDlg (Sheet1.SS)
```

To invoke the Pattern dialog box by function call, call **SSFormatPatternDlg**. The following code displays this dialog box:

```
sserror = SSFormatPatternDlg (Sheet1.SS)
```

## Formatting Row and Column Headings

In addition to formatting worksheet cells, you can format the fonts and colors used for row and column headings. You can also specify the text used to label rows and columns.

Worksheet headings contain three pieces: the row headings, column headings, and the box in the upper left corner of the worksheet where the row and column headings intersect.

In the Worksheet Designer or a worksheet control at run time, you can select a heading area by pressing CTRL+Shift and clicking the heading. After a heading area is selected, you can use the Alignment, Font, Border, and Pattern commands in the Format menu to format the selected headings.

You can also change the row heading width and column heading height by dragging the right and bottom edges of the upper left corner heading. You can also set the heading sizes by calling [SSSetHdrWidth](#) and [SSSetHdrHeight](#).

To select a heading area by function call, use [SSSetHdrSelection](#). The following code selects the column headings.

```
sserror = SSSetHdrSelection (Sheet1.SS, False, False, True)
```

Several function calls and properties allow you to change the text that appears in the headings.

[SSSetColText](#) allows you to set the text heading for a specific column. Likewise, [SSSetRowText](#) and [SSSetTopLeftText](#) set row heading text and the text in the upper left corner box.

The following code sets the heading for column 10 to "Sales" instead of the default "J".

```
sserror = SSSetColText (Sheet1.SS, 10, "Sales")
```

If you use the worksheet as a bound control, you can specify that a row header contain the contents of a field by setting the [DataHdrField](#) property. Refer [Accessing Databases](#) for information about the **DataHdrField** property.

## Printing Worksheets

Formula One provides several options for printing worksheets and setting printing specifications.



Worksheets can be printed through the Worksheet Designer, either at design time or run time.



Function calls allow you to print a worksheet directly. You can also use function calls to display the page setup and printer setup dialog boxes.



Properties can be used to set printing attributes.



## Printing with Function Calls

Printing a worksheet is easily implemented using the **SSFilePrint** function call. The following code uses this function call to print a worksheet.

```
sserror = SSFilePrint (Sheet1.SS, True)
```

When you call **SSFilePrint**, the Print dialog box is displayed, allowing you to specify the pages to print, the number of copies to print, and other related items.

You can also use function calls to display dialog boxes for specifying page setup and printer setup. The Page Setup dialog box 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.

```
sserror = SSFilePageSetupDlg (Sheet1.SS)
```

When you 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.

```
sserror = SSFilePrintSetupDlg (Sheet1.SS)
```

## Specifying Print Areas

**SSFilePrint** prints the entire 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` name, you can:



set the **PrintArea** property.



call the **SSSetPrintArea** function.



launch the Worksheet Designer and select the ranges to print. Then, choose Print Area from the File menu.

The following example uses the **PrintArea** property and the **SSSetPrintArea** function call to set A1:D25 as the area to be printed.

```
Sheet1.PrintArea = "A1:D25"  
sserror = SSSetPrintArea (Sheet1.SS, "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.

The following example uses the **PrintArea** property and the **SSSetPrintArea** function call to set the ranges A1:D4 and F5:I8 as the areas to use as the print area.

```
Sheet1.PrintArea = "A1:D4,F5:I8"  
sserror = SSSetPrintArea (Sheet1.SS, "A1:D4,F5:I8")
```

## 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.

The Print\_Titles name holds the row and column titles specification. To set the Print\_Titles name, you can:



set the **PrintTitles** property.



call the **SSSetPrintTitles** function call.



launch the Worksheet Designer and select the cells to use as print titles. Then, choose Print Titles from the File menu.

**Important** When setting print titles, you must select entire rows and columns.

---

The following example uses the **PrintTitles** property and the **SSSetPrintTitles** function call to set A1:A10 as the area to use as print titles.

```
Sheet1.PrintTitles = "A1:A10"
```

```
sserror = SSSetPrintTitles (Sheet1.SS, "A1:A10")
```

## Specifying Print 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 and the [SSetPrintFooter](#) and [SSetPrintHeader](#) function calls.

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.

<b>Format Code</b>	<b>Description</b>
&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 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.

<b>Format Code</b>	<b>Description</b>
&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

The following example uses the PrintHeader property to specify the header text and center the header.

```
Sheet1.PrintHeader = "&CThis is a centered title"
```

## Specifying Page Breaks

Both horizontal and vertical page breaks can be specified on a worksheet. Page breaks can be specified interactively using the Worksheet Designer, or you can use function calls.

In the Worksheet Designer, page breaks are always placed adjacent to the active cell. When using function calls, 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.

## Page Break Function Calls

There are several categories of page break function calls. The [SSAddPageBreak](#) and [SSRemovePageBreak](#) function calls add page breaks adjacent to the active cell. The following example uses these function calls:

```
sserror = SSAddPageBreak (Sheet1.SS)
sserror = SSRemovePageBreak (Sheet1.SS)
```

[SSAddRowPageBreak](#), [SSAddColPageBreak](#), [SSRemoveRowPageBreak](#), and [SSRemoveColPageBreak](#) add and remove page breaks adjacent to the row or column that you specify in the function call. The following example uses the [SSAddRowPageBreak](#) function call:

```
sserror = SSAddRowPageBreak (Sheet1.SS, therow)
```

[SSNextRowPageBreak](#) returns the next page break below the row that you specify in the function call. [SSNextColPageBreak](#) returns the next page break to the right of the column that you specify in the function call. The following example uses the [SSNextRowPageBreak](#) function call:

```
sserror = SSNextRowPageBreak (Sheet1.SS, therow, nextbreak)
```

## **Working with Databases**

Database connectivity is one of Formula One's most powerful features. Formula One can access database information in two ways - in table mode or in BLOB mode. Visual Basic refers to controls with database access ability as bound controls.

## Accessing Databases

When used as a bound control in table mode, each worksheet row represents a record from the current database record set. Each worksheet column represents a database field. The worksheet is filled from the database when the data control is refreshed or rolled back. If data is changed, added, or deleted in the worksheet, the database is automatically updated to reflect the changes.

When used as a bound control in BLOB mode, a complete worksheet can be stored in a single database field. Therefore, you can have a database of worksheets. As an example, this can be useful in a real estate application where each record contains a picture of a property, information about the property, and a worksheet showing the financial history of the property.

Formula One also supports virtual mode for record buffering. Virtual mode allows only a subset of a large record set to be loaded in memory. Records not currently in memory are fetched automatically when needed. This feature allows you to easily create an application designed for maximum speed and minimum memory usage.

To use Formula One as a bound control, the database properties must be used. The following table lists the properties that control the worksheet when used as a bound control.

<b>Data Property</b>	<b>Operation</b>
<a href="#"><u>AllowDelete</u></a>	Determines whether the Delete key can delete records.
<a href="#"><u>DataAutoAddNew</u></a>	Determines whether the worksheet has an empty row at the end for adding new records.
<a href="#"><u>DataChanged</u></a>	Determines if the data has changed. May be set to force data to be rewritten.
<a href="#"><u>DataConnected</u></a>	Specifies whether the worksheet is connected to the data control. This property can be used to download data and then disconnect before performing analysis or operations on the data.
<a href="#"><u>DataField</u></a>	Specifies the field in which to put the table when used in BLOB mode.
<a href="#"><u>DataFieldChanged(#)</u></a>	This is a Boolean array that indicates whether the specified column has been changed by the user.
<a href="#"><u>DataFieldCount</u></a>	Returns the number of database fields in the table.
<a href="#"><u>DataFieldNumber(Column#)</u></a>	Returns the ordinal field number of the specified column. Used if the fields are displayed in a different order than they occur in the database.
<a href="#"><u>DataFields</u></a>	Specifies the fields to display when used in table mode. This is a list of as many as 256 semicolon-separated field names. The list can include blank columns(;;) and calculated columns (=formula).
<a href="#"><u>DataHdrField</u></a>	Allows a field's value to be specified as the row headers.
<a href="#"><u>DataRowBase</u></a>	Returns the row number of the record in row 1 of the worksheet. Used only when virtual record mode is enabled. This number may be invalid after a find or if other users are adding or deleting records to the database. When valid, <b>DataRowBase</b> plus <b>Row</b> equals the actual database record number.
<a href="#"><u>DataRowCount</u></a>	Returns the number of database rows in the table.
<a href="#"><u>DataRowsBuffered</u></a>	Specifies how many rows are held in memory simultaneously.
<a href="#"><u>DataSetColumnNames</u></a>	Determines if field names are used as column headings instead of the regular column headings. If True, when the data control is refreshed, the previous column headings are removed and the column headings are set to the field names for each column. If False, when the data control is refreshed, the column headings are not changed.
<a href="#"><u>DataSetColumnWidths</u></a>	Automatically sizes the column widths to fit the data in the columns.
<a href="#"><u>DataSetMaxCol</u></a>	If True, <b>MaxCol</b> is set to the number of fields displayed when the data control is refreshed. Otherwise, <b>MaxCol</b> is left unchanged when the data control is refreshed.



**DataSetMaxRow**

If True, **MaxRow** is set to the number of records returned from the database plus 1. The additional blank row is used for adding new records. **MaxRow** is also incremented by one each time a new record is added and decremented by one each time a record is deleted.

**DataSource**

Specifies to which data control to connect this worksheet.

**DoDataNewRow**

Specifies whether the **DataNewRow** event gets fired when the data control sends the AddNew message.

**DoDataRowLoad**

Specifies whether the **DataRowLoad** event gets fired after each row is loaded from the data control.

**RowMode**

Specifies whether only entire rows can be selected, or if individual cells can be selected.

**Note** The **DataFieldChanged**, **DataFieldCount**, **DataFieldNumber**, **DataRowBase**, and **DataRowCount** properties are valid only when a Formula One control is connected to a data control (e.g., the **DataConnected** property is True). The other database properties can be set when a Formula One control is disconnected from a data control; however, they have no effect until a Formula One control is connected to a data control.

---

The following examples illustrate how Formula One's data properties are used. The first example displays the fields from the Authors table in the Biblio database shipped with Visual Basic 3.0.



The column headings are replaced by field titles.



The number of rows in the table are limited to the number of records in the database plus one (for adding new data).



The number of columns are limited to the number of fields in the database table.



The columns are automatically sized to fit the data.

These specifications can be achieved just by setting Formula One's **DataSource** property to the name of the data control (Data1 is the default name). All other properties use their default value.

The following illustration shows the result of this example.

Browse data		
	Au_ID	Author
1	1	Amson, Robert, 1970-
2	2	Atre, Shaku.
3	3	Bamford, Carl.
4	4	Brackett, Michael H.
5	5	Brown, Kenyon.
6	6	Cornell, Gary.
7	7	Craig, John Clark.
8	8	Curtice, Robert M.
9	9	Date, C. J.
10	10	Dutka, Alan F.
11	11	Elmasri, Ramez.
12	12	Emerson, Sandra L.
13	13	Evans, J.D.
14	14	Flavin, Matt.
15	15	Fleming, Candace C.
16	16	Groff, James R.
17	17	Gruber, Martin.
18	18	Hawryszkiewycz, I. T.
19	19	Hergert, Douglas.

The second example demonstrates how a worksheet can be stored in a database field. This is useful for applications where each record must contain one or more tables. In the example, a real estate database is opened that contains information about houses for sale. This simple database has the following fields:

Field Name	Type	Length
Property Number	Counter	
Subdivision	Text	20
Address	Text	25
Sqft	Number	
Price	Number	
Picture	OLE Object	
Amortization Schedule	OLE Object	

The table is called Houses and has seven fields. The field Amortization Schedule holds the loan amortization spreadsheet for each house. Note that it is designed as an OLE Object even though it is used as a long binary data field. The field stores large, undefined pieces of data. In this case, the amortization table is stored in the field.

Creating a field that holds a table is accomplished by setting the **DataField** property. In our example, **DataField** is set to "Amortization Schedule" to tell Formula One that a worksheet is stored in the amortization field. The real estate application is shown in the following illustration.

# Westridge

*Real Estate Brokers*

**Address:** 12389 W. Parkway Plaza

**Property No.:** 48929

**Subdivision:** Westrock Hills

**Schools:** Westrock Elementary, Lost Springs Middle,  
Carter High School

**Bedrooms:** 4 **Baths:** 4 **Fireplace:** Y

**Kitchen:** 10 x 15 **Living Rm:** 25 x 41

**Master BR:** 20 x 25 **Dining Rm:** 15 x 20

**Price:** \$195,000 **Sq. Ft:** 3415



<b>Price</b>	\$195,000
<b>Down Payment</b>	\$20,000
<b>Loan Amount</b>	\$175,000
<b>Interest Rate</b>	7.00%
<b>Loan Term (Years)</b>	30
<b>Monthly Payment</b>	\$1,020.83

**Notes:**

New A/C, gas furnace '93, Master suite includes jacuzzi, 3 stall garage, 4 blocks from grade school, easy access to highways and westside business parks

## Using Virtual Record Buffers

When connected to large record sets, you may not want all the records loaded into memory simultaneously. Formula One allows you to load part of the database table in memory, maximizing speed and minimizing memory requirements.

The **DataRowsBuffered** property allows you to specify how many records to hold in memory at one time. If a record is needed that is not currently in the buffer, Formula One automatically retrieves that record and adjusts the buffer.

The default value for this property is 128 records. This setting is sufficient for applications that have a small number of records or for data browsing applications. In these situations, Formula One handles all the virtual operations transparently.

**Note** It is recommended that the number of rows specified in **DataRowsBuffered** be at least twice the number of rows displayed on the screen.

---

If your application performs a complex operation on a larger number of records, the **DataRowsBuffered** setting should be increased to get as many records into memory as possible. For example, if you perform a mathematical operation on each record in a large database, you should set this number as high as possible, reducing the number of times the program has to retrieve records from disk.

If you want to make the buffer large enough to hold all the records of the current record set, you can use the **RecordCount** property of the data control. This property indicates the number of records in the record set. However, remember that the number of records can change at any time in a multi-user environment. Refer to the **RecordCount** property description in the Visual Basic documentation for additional information about this property.

**Important** Formula One can hold a maximum of 16384 rows.

---

## DataChanged And DataFieldChanged Properties

If the user enters or edits data at run time, Formula One automatically updates the database. However, if the record data is changed through Visual Basic code, the programmer needs to set the [DataFieldChanged](#) array property for any modified column.

```
Sheet1.DataFieldChanged(1) = True
```

There is one array member for each field displayed by Formula One. The **DataFieldChanged** property array is based at 1. Setting a **DataFieldChanged** array member causes the modified column to be written back to disk before another record is fetched.

You can also read the **DataFieldChanged** array to determine if a field has been modified (e.g., modified by a user at run time). The following example displays the field number, field name, and modified flag for all currently displayed fields.

```
For i = 1 to Sheet1.DataFieldCount
    n = Sheet1.DataFieldNumber(i)
    s = "Column " + i + " Field Number " + Str$(n)
    s = s + " [" + data1.Recordset.Fields(n).Name + "]."
```

In this example, Formula One cycles through the fields from 1 to [DataFieldCount](#). **DataFieldCount** is the number of fields displayed in the table. The [DataFieldNumber](#) property returns the actual field number since it may not be the same as the column number (if you omitted some fields or displayed fields out of order in the worksheet). The **DataFieldChanged** property returns True or False depending on whether the field has been modified.

## RowMode Property

Many database applications handle data in complete records as opposed to specific cells or fields. These types of applications often require the current row be marked as the user moves through the database.

The **RowMode** property makes this type of operation easy. When this property is True, only entire rows can be marked. One row is always highlighted and represents the current record. In addition, the active cell within the record is highlighted so the user knows which field is current.

When **RowMode** is enabled, it is possible to select a group of records (assuming the **AllowSelections** property is True). This is useful for group edits. If you want to disallow multiple selected records, disable the **AllowSelections** property.

## Deleting Records

The [AllowDelete](#) property makes it easy for the user to delete records. When this property is True, the user is allowed to delete the current record by pressing the Delete key. A dialog box is displayed to confirm that the record should be deleted.

Rows can also be deleted using the Delete method of the data control.

## **DataRowLoad and DataNewRow Events**

Two important database specific events are the [DataRowLoad](#) and [DataNewRow](#) events. The **DataRowLoad** event is fired each time a new row is loaded from the data control. This allows the program to perform any processing on a record before it is made available to the user.

For example, certain fields may contain integer numbers that represent specific items. You may want to store the integers in the database to minimize disk usage. But when you present the integers to the user, you want full text explanations, not just numbers. The **DataRowLoad** event can expand these integers into text as the records are loaded.

You can also use the data control's **Validate** event to reverse this example. It allows you to make modifications to the data before the record is written back to disk.

The **DataNewRow** event is fired each time the program prepares to create a new record. When the user moves into the blank row at the end of a worksheet, the **DataNewRow** event is fired. The record is not written to the database until the user exits that row, or the UpdateRecord method of the data control is used. This event is useful for making changes to the record before it is written to the database. For example, you may want to place the current date and time in a cell each time a new record is created.



## Specifying Database Column Display

By default, when you connect Formula One to a data control and open a database, all fields are displayed in the worksheet. However, Formula One allows you to specify which database fields are displayed by entering a field list in the [DataFields](#) property.

The following example shows a typical field list:

```
Sheet1.DataFields = "Item;Qty;Price"
```

You can enter null field entries by using two semicolons with no field name between them. For example, if you wanted a blank column between the Item field and the Qty field, enter the following field list:

```
Sheet1.DataFields = "Item;;Qty;Price"
```

You can also enter formula columns that are automatically calculated for each record. To do this, place a formula between semicolons instead of a field name. For example, if you wanted to multiply Qty by Price for each record and display the result in a new column, add =Qty\*Price to the field list, as shown in the following example:

```
Sheet1.DataFields = "Item;;Qty;Price;=Qty*Price"
```

Column formulas can access all functions, cell references, and fields. Displayed fields can be referred to by name, as in the previous example. Fields accessed in this manner refer to the values in the Formula One control. If the user has changed the value of the field, the new data is used in the calculation, even though it has not yet been written to the database.

Formulas should only refer to fields by name and not cell reference. They should also avoid referring to cells in other rows. Otherwise, you may encounter unpredicted results.

You can refer to a database field by enclosing the field name in square brackets. This returns the field's current value in the database, regardless of the editing performed in the Formula One control. You can also use this method to refer to fields not displayed on the Formula One control. For example, to access a field not currently displayed, you could use the following column formula:

```
Sheet1.DataFields = "=Qty*Price+[Freight]"
```

## **Calculating Database Formulas**

Formulas work differently when the Formula One control is connected to a database. When a single record is changed only the calculations in that row are updated - not all formulas in the worksheet. When not connected to a database, all formulas in the worksheet are recalculated whenever a value is changed.

## Displaying and Using Field Names

The names of database fields can be used to label columns in your worksheet if the [DataSetColumnNames](#) property is set to True. By default, this property is True.

When you load a database in a worksheet, each field name becomes a user defined name in the worksheet. The user defined name identifies the column that the field occupies, allowing you to easily make formula references to those columns.

If a field name contains spaces, the spaces are replaced by underscores (\_) in the user defined name. Underscores are not used for spaces when labeling columns.

## 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, 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 down a worksheet by rows, rather than across a worksheet by columns. Any spacing between data blocks occupies less memory if the data blocks are underneath one another than if they are across from each other.

**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 insures 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.

**Disable repainting when performing a series of operations.** When performing a number of sequential operations on a worksheet, disable repainting with the **Repaint** property so the screen does not repaint after each operation. This increases the speed of the operation and avoids unnecessary screen flashing.

**Use function calls when setting and getting numbers, text and formulas.** When possible, use the function calls [SSGetNumberRC](#), [SSSetNumberRC](#), [SSGetTextRC](#), [SSSetTextRC](#), [SSGetFormulaRC](#), and [SSSetFormulaRC](#) rather than the [Number](#), [Text](#), and [Formula](#) properties. Function calls allow for greater numeric precision (64 bit, double precision) than properties (32 bit, single precision). In addition, these function calls allow you to obtain data from a cell and place data in a cell without changing the current selection, thus reducing processing.

**Save the [SS Property](#) property in a variable if it is used frequently.**

**Disable events not in use.** For example, set the [DoSelChange](#) property to False if you do not have a [SelChange](#) event. [SelChange](#) is usually the most costly event since it is fired every time [Row](#), [Col](#), [SelStartRow](#), [SelEndRow](#), [SelStartCol](#), or [SelEndCol](#) is changed.

**Use function calls to copy and move data.** Use [SSEditCopyRight](#), [SSEditCopyDown](#), [SSCopyRange](#), and [SSMoveRange](#) to copy and move cells. These functions are much faster than using the clipboard or copying the data yourself from Visual Basic. In addition, these function calls 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	9.999999999999999E307
Largest negative number	-9.999999999999999E307
Smallest positive number	1E-307
Smallest negative number	-1E-307
Maximum number of iterations	32,767
Maximum number of colors	16
Maximum number of available colors	Limited by your display card and monitor
Maximum number of fonts per sheet	256
Maximum number of selected ranges	2048
Maximum number of names per sheet	Limited by memory
Maximum length of name	255
Maximum number of function arguments	30
Maximum length of format string	255
Maximum number of tables	256
Excel file format version	BIFF4 - Excel 4.0

## Formula One and Visual C++

Before using Formula One with Visual C++, you should read the following chapters:



"Using Custom Controls," Chapter 3 in the Microsoft App Studio User's Guide.



"Programming with VBX Controls," Chapter 17 in the Microsoft Visual C++ Class Library User's Guide.



"CVBControl Class" in the Microsoft Visual C++ Class Library Reference. It is also documented in on-line help under "Visual Object Classes."



Technical Note 27 in MSVC\HELP\MFCNOTES.HLP



### **To add the Formula One Control to the Visual C++ Control Palette:**

1. Start Visual C++.
2. From the Tools menu, choose App Studio.
3. From the File menu in App Studio, choose Install Controls.
4. Select the WINDOWS\SYSTEM directory on your hard disk and double click VTSS.VBX.

## CVBControl Class

The CVBControl Class is a special class defined in the Microsoft Foundation class library. It is specifically designed to allow easy integration of Visual Basic Custom Controls into Visual C++ programs.

The CVBControl Class allows you to load controls, get their properties, set their properties, change their screen locations, and perform other operations. It also provides support for custom control events and methods. Within your application, every VBX control becomes an object of class CVBControl.

The easiest way to use a VBX control with Visual C++ is to load the control in App Studio. You can then drag the control to a dialog box, set properties, and connect code.

You can also use the Formula One Control with C++ without the aid of the App Studio. To use a VBX Control in Visual C++ without App Studio, you must create a VBX-control object and load the Formula One control.

The following code performs this function:

```
CVBControl *pSSvb = new CVBControl;  
pSSvb->Create("VTSS.VBX;SSView;MySheet", NULL, rect, pParentWnd, nID, NULL,  
TRUE);
```

This code creates a CVBControl object named pSSVB and loads the Formula One control. Then, it creates a control named SS with window text of SpreadSheet.

## Getting and Setting Properties in Visual C++

Within the CVBControl Class there is a set of functions designed to access VBX properties. These functions are called the Property Access Member Functions. You must select the proper function depending on the data type you are accessing. Examples for each of the Formula One properties are shown in Chapter 15, A-Z Property Reference.

The following example code enables the Formula One gridlines:

```
pSSVB->SetNumProperty ("ShowGridLines", True);
```



## Differences between Visual Basic and Visual C++

Visual C++ supports only version 1.0 VBXs. The functionality supported by Visual Basic 3.0 is not supported by Visual C++ 1.0. This functionality includes the **HelpContextID**, **HWnd**, and **Data...** properties.

## **Worksheet Designer Overview**

The Worksheet Designer is an interactive program that allows you to design and format the worksheet for your application by pointing and clicking, and choosing format commands from menus. The Worksheet Designer allows you to manipulate a worksheet control just like it was a part of spreadsheet application.

## File Menu Commands

The following table lists the commands available in the File menu

<b>Command</b>	<b>Description</b>
New	Creates a new worksheet. When you create a new worksheet with this command, it does not create a new worksheet control on a form. The new worksheet can be saved to disk.
Open	Opens a worksheet file from disk. Files saved in Formula One format (.VTS files) or Excel 4.0 format (.XLS files) can be opened.
Close	Closes the current worksheet.
Save	Saves the current worksheet. Files can be saved in Formula One format (.VTS files) or Excel 4.0 format (.XLS files).
Save As	Allows you to save the current worksheet with a different name or format.
Print Area	Defines the currently selected range as the Print_Area user-defined name.
Print Titles	Defines the currently selected range 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.
Print	Prints the 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.
Exit	Exits the Worksheet Designer application.

## Edit Menu Commands

The following table lists the commands available in the Edit menu

<b>Command</b>	<b>Description</b>
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.
Clear	Displays a submenu that allows you to clear data from the current selection. You can clear only formats, only values (including formulas), or both formats and values.
Insert	<p>Inserts cells at the location of the current selection. Cells adjacent to the insertion are shifted to make room for the new cells.</p> <p>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.</p>
Delete	<p>Deletes the current selection. Cells adjacent to the deleted cells are shifted to fill the space left by the vacated cells.</p> <p>If 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.</p>
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 select the worksheet page to display.
Recalc	Recalculates the worksheet.
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.

## View Menu Commands

The following table lists the commands available in the View menu

Command	Description
Show	Displays the Show dialog box. This dialog box allows you to determine if formulas, gridlines, row headings, column headings, and zero values are shown or hidden. In addition, you can specify the display status of the vertical and horizontal scroll bars and worksheet selections. You can also set the maximum number of rows and columns displayed.
Allow	Displays the Allow dialog box. This dialog box allows you to determine whether interactive actions such as row and column resizing, range filling, range moving, formula entry, in-cell editing, and range selection can be performed. In addition, you can enable and disable the actions of the arrow, tab, delete, and return keys.
Fix Rows/Columns	If rows are selected, they are fixed at the left edge of the worksheet; if columns are selected, they are fixed at the top edge of the worksheet.  Fixed rows and columns do not scroll out of view.
Unfix Rows/Columns	This command replaces Fix Rows/Columns if the worksheet contains fixed rows or columns. Returns fixed rows and columns to their normal, scrollable status.
Toolbar	When enabled, the tool bar at the top of the Worksheet Designer, including the entry bar, is displayed; when disabled, it is hidden.
Status Bar	When enabled, the status bar at the bottom of the Worksheet Designer is displayed; when disabled, it is hidden.

## Format Menu Commands

The following table lists the commands available in the Format menu

<b>Command</b>	<b>Description</b>
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.
Enable Protection	Enables protection for protected cells in the worksheet.
Disable Protection	Disables protection for protected cells in the worksheet.
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.
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.
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.

## Window Menu Commands

The following table lists the commands available in the Window menu

<b>Command</b>	<b>Description</b>
New Window	Creates an additional window that displays the current worksheet.
Cascade	If multiple worksheet windows are displayed, the windows are placed in a cascading arrangement in the Formula One window.
Tile	If multiple worksheet windows are displayed, the windows are tiled in the Formula One window so that each worksheet is displayed.
Arrange Icons	Arranges the icons of minimized worksheets in the Formula One window.

## A-Z Event Reference

See also [Event Summary](#)

This section provides a complete alphabetical reference for the Formula One events. Refer to [Using Events](#) for additional information about using events. The events listed in this section are:

[CancelEdit Event](#)

[Click Event](#)

[DataNewRow Event](#)

[DataRowLoad Event](#)

[DbClick Event](#)

[DragDrop Event](#)

[DragOver Event](#)

[EndEdit Event](#)

[EndRecalc Event](#)

[GotFocus Event](#)

[KeyDown, KeyUp Events](#)

[KeyPress Event](#)

[LostFocus Event](#)

[SelChange Event](#)

[StartEdit Event](#)

[StartRecalc Event](#)

[TopLeftChanged](#)



## CancelEdit Event

See also [A-Z Event List](#)

**Description** This event occurs when the user leaves edit mode without making changes or presses the Escape key.

**Syntax** **Sub Sheet1\_CancelEdit** ([Index As *Integer*])

## Click Event

See also [A-Z Event List](#)

<b>Description</b>	The <b>Click</b> event occurs when the user presses and releases the mouse button while the pointer is in the Formula One window.
<b>Syntax</b>	<b>Sub Sheet1_Click</b> ([Index As <i>Integer</i> ,] <i>nRow</i> as Long, <i>nCol</i> as Long)
<b>Remarks</b>	<i>nRow</i> and <i>nCol</i> specify the cell in which the user clicks. If a click does not occur on a cell, <i>nRow</i> and <i>nCol</i> are zero.  For additional information, refer to the description of the <b>Click</b> event in the Microsoft Visual Basic Language Reference Manual.

## DataNewRow Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when a new record is added.
<b>Syntax</b>	<b>Sub Sheet1_DataNewRow</b> ( <i>[Index As Integer,] nRow as Long</i> )
<b>Remarks</b>	<p>The <b>DataNewRow</b> event occurs when the data control sends the AddNew message to Formula One. This includes when the user enters the empty row at the end of a worksheet. <i>nRow</i> is the worksheet row number in which the new record is stored; it is not the row number of the record set.</p> <p>This event does not fire unless the worksheet control is bound to a data control.</p>

## DataRowLoad Event

See also [A-Z Event List](#)

**Description** This event occurs after a new row is loaded from the data control.

**Syntax** **Sub Sheet1\_DataRowLoad** ([*Index* As Integer,] *nRow* as Long)

**Remarks** *nRow* is the worksheet row number in which the data is stored; it is not the row number of the data set.

This event does not fire unless the worksheet control is bound to a data control.

**Important** The code in this event must not change the active cell or selection using properties or function calls. It is permissible to use the function calls that take a row and column number directly but do not change them (e.g., `SSSetTextRC`).

---

## DbiClick Event

See also [A-Z Event List](#)

<b>Description</b>	The <b>DbiClick</b> event occurs when the user double clicks the mouse button while the pointer is in the Formula One window.
<b>Syntax</b>	<b>Sub Sheet1_DbiClick</b> ( <i>[Index As Integer,]</i> <i>nRow</i> as Long, <i>nCol</i> as Long)
<b>Remarks</b>	<i>nRow</i> and <i>nCol</i> specify the cell in which the user double clicks. If a double click does not occur on a cell, <i>nRow</i> and <i>nCol</i> are zero.  For additional information, refer to the description of the <b>DbiClick</b> event in the Microsoft Visual Basic Language Reference Manual.

## DragDrop Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when a drag-drop operation is completed.
<b>Syntax</b>	<b>Sub Sheet1_DragDrop</b> ( <i>[Index As Integer,] Source As Control, X As Single, Y As Single</i> )
<b>Remarks</b>	For additional information, refer to the description of the <b>DragDrop</b> event in the Microsoft Visual Basic Language Reference Manual.

## DragOver Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when a drag-drop operation is in process.
<b>Syntax</b>	<b>Sub Sheet1_DragOver</b> ( <i>[Index As Integer,] Source As Control, X As Single, Y As Single, State As Integer</i> )
<b>Remarks</b>	For additional information, refer to the description of the <b>DragOver</b> event in the Microsoft Visual Basic Language Reference Manual.

## EndEdit Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when an editing operation is completed.
<b>Syntax</b>	<b>Sub Sheet1_EndEdit</b> ([ <i>Index</i> As Integer,] <i>EditString</i> As String, <i>Cancel</i> As Integer)
<b>Remarks</b>	<b>EditString</b> is the edited text to be entered in the active cell. This value can be changed to modify what is placed in the cell.  <i>Cancel</i> can be 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

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when the recalculation process is completed.
<b>Syntax</b>	<b>Sub Sheet1_EndRecalc</b> ( <i>[Index As Integer]</i> )
<b>Remarks</b>	The <b>EndRecalc</b> event occurs after a worksheet has been recalculated.

## GotFocus Event

See also [A-Z Event List](#)

<b>Description</b>	The <b>GotFocus</b> event occurs when the Formula One window receives focus, either by clicking the object or changing the focus in code using the SetFocus method.
<b>Syntax</b>	<b>Sub Sheet1_GotFocus</b> ( <i>[Index As Integer]</i> )
<b>Remarks</b>	For additional information, refer to the description of the <b>GotFocus</b> event in the Microsoft Visual Basic Language Reference Manual.

## KeyDown,KeyUp Events

See also [A-Z Event List](#)

<b>Description</b>	These events occur when the user presses ( <b>KeyDown</b> ) and releases ( <b>KeyUp</b> ) a key while the Formula One object has the focus.
<b>Syntax</b>	<b>Sub Sheet1_KeyDown</b> ( <i>[Index As Integer,] KeyCode As Integer, Shift As Integer</i> ) <b>Sub Sheet1_KeyUp</b> ( <i>[Index As Integer,] KeyCode As Integer, Shift As Integer</i> )
<b>Remarks</b>	For additional information, refer to the descriptions of the <b>KeyDown</b> and <b>KeyUp</b> events in the Microsoft Visual Basic Language Reference Manual.

## KeyPress Event

See also [A-Z Event List](#)

<b>Description</b>	Occurs when the user presses and releases an ANSI key.
<b>Syntax</b>	<b>Sub Sheet1_KeyPress</b> ( <i>[Index As Integer,] KeyAscii As Integer</i> )
<b>Remarks</b>	For additional information, refer to the description of the <b>KeyPress</b> event in the Microsoft Visual Basic Language Reference Manual.

## LostFocus Event

See also [A-Z Event List](#)

<b>Description</b>	The <b>LostFocus</b> event occurs when the Formula One window loses focus, either by clicking the object or changing the focus in code using the SetFocus method.
<b>Syntax</b>	<b>Sub Sheet1_LostFocus</b> ( <i>Index As Integer</i> )
<b>Remarks</b>	For additional information, refer to the description of the <b>LostFocus</b> event in the Microsoft Visual Basic Language Reference Manual.

## SelChange Event

See also [A-Z Event List](#)

**Description** This event occurs when the active cell is changed or the current selection is changed.

**Syntax** **Sub Sheet1\_SelChange** ([*Index As Integer*])

**Important** Actions that change the row and column selection (e.g., using the **Row** or **Col** properties) should not be used within this event as you will encounter unexpected results.

---

## StartEdit Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when an editing operation is started.
<b>Syntax</b>	<b>Sub Sheet1_StartEdit</b> ([ <i>Index</i> As Integer,] <i>EditString</i> As String, <i>Cancel</i> As Integer)
<b>Remarks</b>	<b>EditString</b> is the text to be edited. <i>Cancel</i> can be set to True to cancel edit mode. In this case, edit mode is not entered.

## StartRecalc Event

See also [A-Z Event List](#)

<b>Description</b>	This event occurs when the recalculation process is started.
<b>Syntax</b>	<b>Sub Sheet1_StartRecalc</b> ([ <i>Index</i> As Integer])
<b>Remarks</b>	The <b>StartRecalc</b> event occurs when the worksheet is about to be recalculated.



## TopLeftChanged Event

See also [A-Z Event List](#)

**Description** This event occurs when the cell that is displayed as the top left cell of the worksheet changes (e.g., when the user scrolls the worksheet). The execution of this event is deferred until the system is idle.

**Syntax** **Sub Sheet1\_TopLeftChanged** ([*Index* As Integer])

## A-Z Property Reference

See also [Property Summary](#)

This chapter provides a complete alphabetical reference for the Formula One properties. Refer to [Using Properties](#) for additional information about using properties.

<a href="#">AllowAppLaunch Property</a>	<a href="#">Formula Property</a>
<a href="#">AllowArrows Property</a>	<a href="#">Height Property</a>
<a href="#">AllowDelete Property</a>	<a href="#">HelpContextID Property</a>
<a href="#">AllowEditHeaders Property</a>	<a href="#">hWnd Property</a>
<a href="#">AllowFillRange Property</a>	<a href="#">Index Property</a>
<a href="#">AllowFormulas Property</a>	<a href="#">Left Property</a>
<a href="#">AllowInCellEditing Property</a>	<a href="#">LeftCol Property</a>
<a href="#">AllowMoveRange Property</a>	<a href="#">MaxCol Property</a>
<a href="#">AllowResize Property</a>	<a href="#">MaxRow Property</a>
<a href="#">AllowSelections Property</a>	<a href="#">MinCol Property</a>
<a href="#">AllowTabs Property</a>	<a href="#">MinRow Property</a>
<a href="#">AutoRecalc Property</a>	<a href="#">MousePointer Property</a>
<a href="#">BackColor Property</a>	<a href="#">Name Property</a>
<a href="#">BorderStyle Property</a>	<a href="#">Number Property</a>
<a href="#">Col Property</a>	<a href="#">Parent Property</a>
<a href="#">DataAutoAddNew Property</a>	<a href="#">PrintArea Property</a>
<a href="#">DataChanged Property</a>	<a href="#">PrintBottomMargin Property</a>
<a href="#">DataConnected Property</a>	<a href="#">PrintColHeading Property</a>
<a href="#">DataField Property</a>	<a href="#">PrintFooter Property</a>
<a href="#">DataFieldChanged Property</a>	<a href="#">PrintGridLines Property</a>
<a href="#">DataFieldCount Property</a>	<a href="#">PrintHCenter Property</a>
<a href="#">DataFieldNumber Property</a>	<a href="#">PrintHeader Property</a>
<a href="#">DataFields Property</a>	<a href="#">PrintLeftMargin Property</a>
<a href="#">DataHdrField Property</a>	<a href="#">PrintLeftToRight Property</a>
<a href="#">DataRowBase Property</a>	<a href="#">PrintNoColor Property</a>
<a href="#">DataRowCount Property</a>	<a href="#">PrintRightMargin Property</a>
<a href="#">DataRowsBuffered Property</a>	<a href="#">PrintRowHeading Property</a>
<a href="#">DataSetColumnFormats Property</a>	<a href="#">PrintTitles Property</a>
<a href="#">DataSetColumnNames Property</a>	<a href="#">PrintTopMargin Property</a>
<a href="#">DataSetColumnWidths Property</a>	<a href="#">PrintVCenter Property</a>
<a href="#">DataSetMaxCol Property</a>	<a href="#">ReadFile Property</a>
<a href="#">DataSetMaxRow Property</a>	<a href="#">Repaint Property</a>
<a href="#">DataSource Property</a>	<a href="#">Row Property</a>
<a href="#">DoCancelEdit Property</a>	<a href="#">RowMode Property</a>
<a href="#">DoClick Property</a>	<a href="#">Selection Property</a>
<a href="#">DoDataNewRow Property</a>	<a href="#">SelEndCol Property</a>
<a href="#">DoDataRowLoad Property</a>	<a href="#">SelEndRow Property</a>
<a href="#">DoDbfClick Property</a>	<a href="#">SelStartCol Property</a>
<a href="#">DoEndEdit Property</a>	<a href="#">SelStartRow Property</a>
<a href="#">DoEndRecalc Property</a>	<a href="#">ShowColHeading Property</a>
<a href="#">DoSelChange Property</a>	<a href="#">ShowGridLines Property</a>
<a href="#">DoStartEdit Property</a>	<a href="#">ShowHScrollBar Property</a>
<a href="#">DoStartRecalc Property</a>	<a href="#">ShowRowHeading Property</a>
<a href="#">DoTopLeftChanged Property</a>	<a href="#">ShowSelections Property</a>
<a href="#">DragIcon Property</a>	<a href="#">ShowVScrollBar Property</a>
<a href="#">DragMode Property</a>	<a href="#">SS Property</a>
<a href="#">EditName Property</a>	<a href="#">TabIndex Property</a>
<a href="#">Enabled Property</a>	<a href="#">TableName Property</a>
<a href="#">EnableProtection Property</a>	<a href="#">TabStop Property</a>
<a href="#">Entry Property</a>	<a href="#">Tag Property</a>
<a href="#">ExtraColor Property</a>	<a href="#">Text Property</a>

[FileName Property](#)  
[FixedCol Property](#)  
[FixedCols Property](#)  
[FixedRow Property](#)  
[FixedRows Property](#)  
[FormattedText Property](#)

[Top Property](#)  
[TopRow Property](#)  
[Visible Property](#)  
[Width Property](#)  
[WriteExcel4 Property](#)  
[WriteFile Property](#)

## AllowAppLaunch Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <a href="#">Worksheet Designer</a> is allowed to launch at run time when the user double clicks the Formula One window with the right mouse button.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowAppLaunch [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowAppLaunch")</code> <code>pSSVB-&gt;SetNumProperty("AllowAppLaunch", {True False})</code>
<b>Remarks</b>	<p>When True, this property allows the user to invoke the Worksheet Designer at run time by double clicking the Formula One window with the right mouse button. Double click events can be received when the left mouse button is double clicked.</p> <p>When False, the Worksheet Designer cannot be launched at run time. The Worksheet Designer can always be launched at design time by double clicking the Formula One window with the right mouse button.</p>
<b>Data Type</b>	Integer (Boolean)
<b>Example</b>	<pre>Sheet1.AllowAppLaunch = True ' Allow user to launch Worksheet Designer</pre>

## AllowArrows Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the arrow keys can move the active cell.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowArrows [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowArrows")</code> <code>pSSVB-&gt;SetNumProperty("AllowArrows", {True False})</code>
<b>Remarks</b>	When True, this property allows the user to move the active cell using the four arrow keys.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">AllowTabs</a> property and <a href="#">SSGetAllowArrows</a> and <a href="#">SSSetAllowArrows</a> functions
<b>Example</b>	<code>Sheet1.AllowArrows = True ' Allow arrows to move active cell</code>

## AllowDelete Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the Delete key can delete the current record or clear the current selection.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowDelete [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowDelete")</code> <code>pSSVB-&gt;SetNumProperty("AllowDelete", {True False})</code>
<b>Remarks</b>	When not connected to a data control, this property allows the user to specify whether the Delete key can clear the current selection. When connected to a data control, this property allows the user to specify whether the Delete key can delete the current record, if the whole row is marked, or clear the current selection if less than a row is marked.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAllowDelete</a> and <a href="#">SSSetAllowDelete</a> functions
<b>Example</b>	<code>Sheet1.AllowDelete = True ' Allow Delete key to delete record</code>

## AllowEditHeaders Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user is allowed to edit row, column, and top left headers by double clicking a header.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowEditHeaders [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowEditHeaders")</code> <code>pSSVB-&gt;SetNumProperty("AllowEditHeaders", {True False})</code>
<b>Remarks</b>	<p>When this property is True, the names displayed in row, column, and top left headers can be edited by double clicking the header to be edited. The Header Name dialog box is displayed, allowing you to enter a new header name.</p> <p>If False, editing of headers is not allowed and a <b>DbClick</b> event is passed when a header is double clicked.</p>
<b>Return Value</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAllowEditHeaders</a> and <a href="#">SSSetAllowEditHeaders</a> functions and <a href="#">DbClick</a> event.
<b>Example</b>	<code>Sheet1.AllowEditHeaders = False ' Disallow header editing</code>

## AllowFillRange Property

See also [A-Z Property List](#)

**Description** Determines if the user is allowed to fill a range by dragging a selection's copy handle.

**Syntax (VB)** `[form.][control.]AllowFillRange [= {True|False}]`

**Syntax (VC++)** `pSSVB->GetNumProperty("AllowFillRange")`  
`pSSVB->SetNumProperty("AllowFillRange", {True|False})`

**Remarks** When True, this property allows the user to automatically copy a selection by dragging the copy handle on a selection. The copy handle is the small knob at the lower right corner of a selection.

**Data Type** Integer (Boolean)

**See Also** [SSGetAllowFillRange](#) and [SSSetAllowFillRange](#) functions

**Example** `Sheet1.AllowFillRange = True ' Allow automatic filling`

## AllowFormulas Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user can enter new formulas or edit existing formulas.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>AllowFormulas</b> [= {True False}]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("AllowFormulas") pSSVB-> <b>SetNumProperty</b> ("AllowFormulas", {True False})
<b>Remarks</b>	When True, the user is allowed to enter new formulas in the worksheet. When False, the user cannot enter new formulas. However, existing formulas are retained and new formulas can be added by program code.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">Formula</a> property and <a href="#">SSGetAllowFormulas</a> , <a href="#">SSSetAllowFormulas</a> , and <a href="#">SSSetFormula</a> functions
<b>Example</b>	Sheet1.AllowFormulas = True ' Allow the user to enter formulas



## AllowInCellEditing Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if in-cell editing is allowed.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowInCellEditing [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowInCellEditing")</code> <code>pSSVB-&gt;SetNumProperty("AllowInCellEditing", {True False})</code>
<b>Remarks</b>	When True, this property allows data and formulas to be entered or edited directly in a cell without using an edit bar. However, if entering long data or formulas, it is often more convenient to use an edit bar.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAllowInCellEditing</a> and <a href="#">SSSetAllowInCellEditing</a> functions
<b>Example</b>	<code>Sheet1.AllowInCellEditing = True ' Allow user to edit data within the cell</code>

## AllowMoveRange Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user can move a selection by dragging it to a new location.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowMoveRange [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowMoveRange")</code> <code>pSSVB-&gt;SetNumProperty("AllowMoveRange", {True False})</code>
<b>Remarks</b>	When True, this property allows the user to move the current selection to another area by dragging the border surrounding the selection.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAllowMoveRange</a> and <a href="#">SSSetAllowMoveRange</a> functions
<b>Example</b>	<code>Sheet1.AllowMoveRange = True ' Allow drag and drop moving</code>

## AllowResize Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user can resize rows or columns.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowResize [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowResize")</code> <code>pSSVB-&gt;SetNumProperty("AllowResize", {True False})</code>
<b>Remarks</b>	When True, this property allows the user to resize rows or columns by dragging the line to the right of a column heading or at the bottom of a row heading.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAllowResize</a> and <a href="#">SSSetAllowResize</a> functions
<b>Example</b>	<code>Sheet1.AllowResize = True ' Allow resizing</code>

## AllowSelections Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user can make selections.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowSelections [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowSelections")</code> <code>pSSVB-&gt;SetNumProperty("AllowSelections", {True False})</code>
<b>Remarks</b>	When True, this property allows the user to select a cell or range of cells. When False, selections cannot be made with the mouse or keyboard.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">ShowSelections</a> property and <a href="#">SSGetAllowSelections</a> and <a href="#">SSSetAllowSelections</a> functions
<b>Example</b>	<code>Sheet1.AllowSelections = True ' Allow selections</code>

## AllowTabs Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the Tab and Shift Tab keys can move the active cell within the current selection.
<b>Syntax (VB)</b>	<code>[form.][control.]AllowTabs [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AllowTabs")</code> <code>pSSVB-&gt;SetNumProperty("AllowTabs", {True False})</code>
<b>Remarks</b>	When True, this property allows the Tab and Shift Tab keys to move the active cell within the current selection. Tab moves the cell right through the selected range, wrapping to the left if it reaches the right edge of the selection. Shift Tab moves the active cell in the opposite direction in the selected range.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">AllowArrows</a> and <a href="#">AllowSelections</a> properties and <a href="#">SSGetAllowTabs</a> and <a href="#">SSSetAllowTabs</a> functions
<b>Example</b>	<code>Sheet1.AllowTabs = True ' Tabs can move the active cell</code>

## AutoRecalc Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if automatic recalculation is enabled. Forces the worksheet to be recalculated immediately, if needed, when set to True.
<b>Syntax (VB)</b>	<code>[form.][control.]AutoRecalc [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("AutoRecalc")</code> <code>pSSVB-&gt;SetNumProperty("AutoRecalc", {True False})</code>
<b>Remarks</b>	<p>When True, this property enables automatic recalculation and recalculates the worksheet if needed. Thereafter, any change to the worksheet causes all formulas to be recalculated.</p> <p>Notice that the screen may not be updated if the worksheet is manipulated with a tight Visual Basic loop. If you want the screen to be updated in this circumstance, you must call <code>SSUpdate</code> after each cell is processed. This slows worksheet processing significantly.</p>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetAutoRecalc</a> , <a href="#">SSSetAutoRecalc</a> , and <a href="#">SSUpdate</a> functions
<b>Example</b>	<code>Sheet1.AutoRecalc = True ' Automatic recalc is on</code>

## BackColor Property

See also [A-Z Property List](#)

**Description** Determines the background color of the Formula One window.

**Syntax (VB)** [*form.*][*control.*]**BackColor** [= *color*]

**Syntax (VC++)** pSSVB->**GetNumProperty**("BackColor")  
pSSVB->**SetNumProperty**("BackColor", *color*)

**Remarks** This property expects a color in the standard Windows environment RGB scheme. The Formula One window is displayed in this color.

This value can be one of the following:



**Normal RGB Colors.** These colors are specified using the color palette, or by using the RGB or QBColor functions.



**System default colors.** System color constants are specified in the Visual Basic CONSTANT.TXT file.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

For additional information, refer to the description of the **BackColor** property in the Microsoft Visual Basic Language Reference Manual.

**Data Type** Long

**See Also** [SSSetBackColor](#) function

**Example**  
`Sheet1.BackColor = QBColor(Rnd * 15) ' Random Color`  
`Sheet1.BackColor = RGB(0, 255, 255) ' Cyan`

## BorderStyle Property

See also [A-Z Property List](#)

**Description** Determines the border style for the Formula One window.

**Syntax (VB)** `[form.][control.]BorderStyle = {0|1}`

**Syntax (VC++)** `pSSVB->GetNumProperty("BorderStyle")`  
`pSSVB->SetNumProperty("BorderStyle", {0|1})`

**Remarks** The BorderStyle property settings are:

Setting	Description
---------	-------------

---

0	None
---	------

1	Fixed Single
---	--------------

**Data Type** Integer (Enumerated)

**Example** `Sheet1.BorderStyle = 1 ' Single line border`



## Col Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the active column in the worksheet. This is a run time only property.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>Col</b> [= <i>Column</i> ]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("Col") pSSVB-> <b>SetNumProperty</b> ("Col", <i>Column</i> )
<b>Remarks</b>	<p>The <b>Col</b> property is used with the <b>Row</b> property to set the active cell in the worksheet. The <b>Col</b> property is automatically changed if a range is selected using the <b>SelStart...</b> and <b>SelEnd...</b> properties.</p> <p>You can specify -1 as the row and column number to indicate all rows or all columns. For example, setting <b>Row</b> to 1 and <b>Col</b> to -1 causes all columns in row 1 to be selected. Setting both <b>Row</b> and <b>Col</b> to -1 selects the entire worksheet.</p>
<b>Data Type</b>	Long
<b>See Also</b>	<a href="#">Row</a> , <a href="#">SelStart...</a> , and <a href="#">SelEnd...</a> properties and <a href="#">SSSetActiveCell</a> function
<b>Example</b>	<pre>Sheet1.Col = 5 ' Select row 3 column 5 as the active cell Sheet1.Row = 3 Sheet1.Col = -1 ' Select all of row one Sheet1.Row = 1</pre>

## DataAutoAddNew Property

See also [A-Z Property List](#)

**Description** Determines if the worksheet has an empty row at the end for adding new records.

**Syntax (VB)** [*form.*][*control.*]**DataAutoAddNew** [= {**True**|**False**}]

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataAutoAddNew")  
pSSVB->**SetNumProperty**("DataAutoAddNew", {**True**|**False**})

**Remarks** The **DataAutoAddNew** property determines if an empty row is placed at the end of the worksheet. The empty row is used to add a new record to the database table. When the user enters this row, Formula One calls the AddNew method of the attached data control. After each new record is added, a new blank row is placed at the bottom for the next new record.

**Data Type** Integer (Boolean)

**Example** Sheet1.DataAutoAddNew = True ' A blank row is added to the end

## DataChanged Property

See also [A-Z Property List](#)

**Description** Indicates that the data in the current record has changed. This is a run time only property.

**Syntax (VB)** [*form.*][*control.*]**DataChanged** [ = {**True|False**}]

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataChang\*ed")  
pSSVB->**SetNumProperty**("DataChanged", {**True|False**})

**Remarks** The **DataChanged** property indicates whether data in the current record has changed. If True, the data in the control is not the same as in the current record. When the cursor moves out of the current row (record), the data is written to the database.

The **DataChanged** property is automatically set if a change is made to the current record by the user at run time. It is not set if changes are made to the record using Visual Basic code. In this case, the programmer should set the **DataFieldChanged** property to indicate modified fields. This causes the **DataChanged** property to be automatically set and the changed fields to be written to the database.

**Data Type** Integer (Boolean)

**See Also** [DataFieldChanged](#) property

**Example** AnyChange = Sheet1.DataChanged

## DataConnected Property

See also [A-Z Property List](#)

**Description** Specifies whether the worksheet is connected to a data control.

**Syntax (VB)** `[form.][control.]DataConnected [= {True|False}]`

**Syntax (VC++)** `pSSVB->GetNumProperty("DataConnected")`  
`pSSVB->SetNumProperty("DataConencted", {True|False})`

**Remarks** The **DataConnected** property determines if the Formula One control is currently connected to the data control specified in the [DataSource](#) property. When True, the control responds to Refresh messages sent by the data control. When False, the worksheet is disconnected from the data control and all future Refresh messages are ignored. The contents of the worksheet is still available. To reconnect, set **DataConnected** to True and refresh the data control.

Use this property when you want to disconnect a data control and perform data analysis on static data.

Note that only rows that are currently buffered when the control is disconnected are available in the worksheet. If you want all rows available in the worksheet, set the [DataRowsBuffered](#) property to a number large enough to hold all the records and refresh the data control.

**Data Type** Integer (Boolean)

**Example** `Sheet1.DataConnected = False ' Disconnect from the data control`

## DataField Property

See also [A-Z Property List](#)

**Description** Binds the Formula One control to a database field. Used for storing an entire Formula One worksheet in a single field.

**Syntax (VB)** `[form.][control.]DataField [= fieldname]`

**Syntax (VC++)** `pSSVB->GetStrProperty("DataField")`  
`pSSVB->SetStrProperty("DataField", fieldname)`

**Remarks** The **DataField** property is used when a complete worksheet is stored in a database field. This field must be a long binary field; however, Access refers to these fields as OLE fields. Each record in the database contains a unique instance of a Formula One control.

If you want the Formula One control to display one record from the data control in each row of the Formula One control, leave this property blank. To control which fields appear in which columns when displaying one record per row, use the **DataFields** property.

**Data Type** String

**See Also** [DataFields](#) property

**Example** `Sheet1.DataField = "Financials"`

## DataFieldChanged Property

See also [A-Z Property List](#)

**Description** Indicates whether the specified field has been changed by the user. This is a run time only property; it is valid only when a worksheet control is connected to a data control.

**Syntax (VB)** `[form.][control.]DataFieldChanged(column number) [= {True|False}]`

**Remarks** The **DataFieldChanged** property is a boolean array that indicates whether the specified field has been changed by the user. If any member of **DataFieldChanged** is True, the **DataChanged** property is automatically set to True, causing the database to be updated.

If any field is modified by Visual Basic code, the **DataFieldChanged** array member must be set to True, or this change is not reflected in the database.

**Data Type** Integer (Boolean)

**See Also** [DataChanged](#) property

**Example** `Sheet1.DataFieldChanged(1) = True`

## DataFieldCount Property

See also [A-Z Property List](#)

**Description** Returns the number of database fields displayed in the worksheet. This is a run time, read only property; it is valid only when a worksheet control is connected to a data control

**Syntax (VB)** [*form.*][*control.*]**DataFieldCount**

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataFieldCount")

**Remarks** The **DataFieldCount** property returns the number of fields displayed in the worksheet. It returns 0 if no database is attached, or if there were no fields specified that the worksheet could handle (e.g., memo fields).

**Data Type** Integer

**See Also** [DataRowCount](#) property

**Example** NumFields = Sheet1.DataFieldCount

## DataFieldNumber Property

See also [A-Z Property List](#)

**Description** Returns the number of the database field in the specified column. This is a read only, run time only property; it is valid only when a worksheet control is connected to a data control

**Syntax (VB)** `[form.][control.]DataFieldNumber(Column)`

**Remarks** The **DataFieldNumber** property is an array of field numbers representing the number of the field to which the specified column is pointing. The property is used when the fields are displayed in a different order than they exist in the database. This property is not valid for bound and calculated fields.

It is important to note that field numbers are based at 0 and column numbers are based at 1. Fields not supported in Formula One (e.g., memo fields) are skipped.

**Data Type** Integer

**See Also** [DataFields](#) property

**Example** `OrdNum = Sheet1.DataFieldNumber(1)`



## DataFields Property

See also [A-Z Property List](#)

**Description** Binds the Formula One control to one or more database fields. This property is used for displaying one record per row.

**Syntax (VB)** [*form.*][*control.*]**DataFields** [= *fieldnames*]

**Syntax (VC++)** pSSVB->**GetStrProperty**("DataFields")  
pSSVB->**SetStrProperty**("DataFields", [= *fieldnames*])

**Remarks** The DataFields property specifies the fields with which to fill the worksheet. If the property is blank (and the DataField property is blank), all the Formula One supported fields are brought into the worksheet. Fields not supported are skipped. The number of fields is limited to 256.

If you do not want to display all the fields, or do not want them in the default order, a semi-colon separated list of field names can be used to specify the field to display in each column. The first field is placed in column 1, the second field in column 2, and so on.

This property is referenced when the worksheet gets a refresh or rollback message from the data control.

You can enter null field entries by specifying two semicolons with no field name between them. For example, if you want a blank column between the Item field and the Qty field, enter the following field list.

```
Sheet1.DataFields = "Item;;Qty;Price"
```

You can also enter formula columns that are automatically calculated for each record. To accomplish this, place a formula between semicolons instead of a field name. For example, to multiply Qty by Price for each record and display the result in a new column, enter the following field list.

```
Sheet1.DataFields = "Item;;Qty;Price;=Qty*Price"
```

Column formulas can access all functions, operators, and fields. Displayed fields can be referred to by name, as in the previous example. Fields accessed in this manner refer to the values in the Formula One control. If the user has changed the value of the field, the new data is used in the calculation, even though it is not yet written to the database.

Formulas should only refer to fields by name and not by cell reference. They should also avoid referring to cells in other rows. Otherwise, you may encounter unpredicted results. To refer to a column that does not contain a field, you must use a reference to the entire column. The following example puts the result of Qty\*Price in column 3 and multiplies this value by 0.065 for column 4.

```
Sheet1.DataFields = "Qty;Price; =Qty*Price; =C1:C16384*.065"
```

You can also refer to a field by enclosing the field name in square brackets. This returns the field's current value in the database, regardless of editing performed in the Formula One control. You can also use this method to refer to fields that are not displayed on the Formula One control. For example, to access a field not currently displayed, use the following column formula.

```
Sheet1.DataFields = "=Qty*Price+[Freight]"
```

Formulas work differently when the Formula One control is connected to a database. When a single record is changed only the calculations in that row are updated, not all formulas in the worksheet. When not connected to a database, all formulas in a

worksheet are recalculated when a value changes.

**Data Type** String

**See Also** [DataField](#) and [DataFieldNumber](#) properties

**Example** `Sheet1.DataFields = "Name;Address;City;State;Zip"`

## DataHdrField Property

See also [A-Z Property List](#)

<b>Description</b>	Allows a fields data to be specified as the row header names.
<b>Syntax (VB)</b>	<code>[form.][control.]DataHdrField [ = <i>fieldname</i>]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("DataHdrField")</code> <code>pSSVB-&gt;SetStrProperty("DataHdrField", <i>fieldname</i>)</code>
<b>Remarks</b>	The <b>DataHdrField</b> property displays a field's data as the row header names. The field's data replaces the row numbers normally displayed at the left of each row.
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">DataFields</a> property
<b>Example</b>	<code>Sheet1.DataHdrField = "Name"</code>

## DataRowBase Property

See also [A-Z Property List](#)

**Description** Returns the row number of the record in row 1 of the worksheet. This is a run time, read only property; it is valid only when a worksheet control is connected to a data control

**Syntax (VB)** [*form.*][*control.*]**DataRowBase**

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataRowBase")

**Remarks** The **DataRowBase** property returns the row number of the record in row 1 of the worksheet. This number is only relevant when using the virtual record mode since the first record in the worksheet may not be the first record in the record set. This number is based at 0; thus, when a table is first loaded, the number is 0. This number, when added to the number in the Row property, is equal to the actual record number in the database.

This number becomes invalid after a find, or if multiple applications add or delete records in the database. You can force the number to become valid by executing a MoveFirst or MoveLast method.

**Data Type** Integer

**See Also** [DataRowsBuffered](#) property

**Example** `FirstRow = Sheet1.DataRowBase`

## **DataRowCount Property**

See also [A-Z Property List](#)

**Description** Returns the number of database records displayed in the worksheet. This is a run time, read only property; it is valid only when a worksheet control is connected to a data control

**Syntax (VB)** `[form.][control.]DataRowCount`

**Syntax (VC++)** `pSSVB->GetNumProperty("DataRowCount")`

**Data Type** Integer

**See Also** [DataFieldCount](#) property

**Example** `NumRows = Sheet1.DataRowCount`

## DataRowsBuffered Property

See also [A-Z Property List](#)

**Description** Specifies how many database rows are kept in memory simultaneously.

**Syntax (VB)** `[form.][control.]DataRowsBuffered [= Rows]`

**Syntax (VC++)** `pSSVB->GetNumProperty("DataRowsBuffered")`  
`pSSVB->SetNumProperty("DataRowsBuffered", Rows)`

**Remarks** The **DataRowsBuffered** property determines how many database rows are held in memory simultaneously. If there are more rows in the data set than specified in **DataRowsBuffered**, new rows are automatically brought into the buffer as needed. The default is 128 rows.

**Note** It is recommended that the number of rows specified in **DataRowsBuffered** be at least twice the number of rows displayed on the screen. The minimum number of rows that can be specified is 32.

---

The worksheet is limited to as many rows as specified in **DataRowsBuffered**. For example, if you specify 1000 rows for the buffer, there are only 1000 rows displayed in your worksheet. When you move beyond 1000 records, all current records are scrolled up and the top record is scrolled out of the buffer.

The [DataRowBase](#) property determines the offset in the worksheet. It indicates how many records are scrolled off the top of the worksheet.

To turn off virtual mode, you must set **DataRowsBuffered** greater than or equal to the number of records in the database.

When in virtual mode, the **Row** property does not work as it normally would. For example, the following code fragment does not work because records are shuffled through the first 'DataRowsBuffered' rows of the worksheet.

```
nTotal# = 0#  
Sheet1.Col = 1 ' Column with Quantity field  
For nRow = 1 To Data1.RecordSet.RecordCount  
    Sheet1.Row = nRow  
    nTotal# = nTotal# + Sheet1.Number  
Next nRow
```

The following code is correct.

```
nTotal# = 0#  
Sheet1.Col = 1 ' Column with Quantity field  
data1.Recordset.MoveFirst  
While Not data1.Recordset.EOF
```

## DataSetColumnFormats Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if formats for date, time, and currency fields are set automatically.
<b>Syntax (VB)</b>	<code>[form.][control.]DataSetColumnFormats [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DataSetColumnFormats") pSSVB-&gt;SetNumProperty("DataSetColumnFormats", {True False})</code>
<b>Remarks</b>	The <b>DataSetColumnFormats</b> property determines if the formats for date, time, and currency fields are set automatically when data is placed in a spreadsheet control. If True, formats for columns containing these fields are set automatically. If False, you must set the formats for these columns manually in the Worksheet Designer or in code after the data control is refreshed.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DataFields</a> property
<b>Example</b>	<code>Sheet1.DataSetColumnFormats = False</code>

## DataSetColumnNames Property

See also [A-Z Property List](#)

**Description** Determines if the column headings are replaced by field names.

**Syntax (VB)** [*form.*][*control.*]**DataSetColumnNames** [= {**True**|**False**}]

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataSetColumnNames")  
pSSVB->**SetNumProperty**("DataSetColumnNames", {**True**|**False**})

**Remarks** The **DataSetColumnNames** property determines if the column headings are replaced by 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).

This property is referenced when the worksheet gets a refresh or rollback message from the data control.

**Data Type** Integer (Boolean)

**See Also** [DataFields](#) property

**Example** Sheet1.DataSetColumnNames = True



## DataSetColumnWidths Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if column widths are automatically set to accommodate the widest data in the column.
<b>Syntax (VB)</b>	<code>[form.][control.]DataSetColumnWidths [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DataSetColumnWidths")</code> <code>SSVB-&gt;SetNumProperty("DataSetColumnWidths", {True False})</code>
<b>Remarks</b>	<p>If True, this property automatically sets the width of each column to be wide enough to display the widest data in the column.</p> <p>When the worksheet gets a refresh or rollback message from the data control the column widths are updated based on the data in the columns.</p>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DataFields</a> property and <a href="#">SSSetColWidthAuto</a> function
<b>Example</b>	<code>Sheet1.DataSetColumnWidths = True</code>

## DataSetMaxCol Property

See also [A-Z Property List](#)

**Description** Determines if the maximum number of worksheet columns is set to the number of fields currently displayed.

**Syntax (VB)** [*form.*][*control.*]**DataSetMaxCol** [= {**True**|**False**}]

**Syntax (VC++)** pSSVB->**GetNumProperty**("DataSetMaxCol")  
pSSVB->**SetNumProperty**("DataSetMaxCol", {**True**|**False**})

**Remarks** If True, this property sets the [MaxCol](#) property to limit the number of displayed columns to the number of fields currently loaded from the database.

This property is referenced when the worksheet gets a refresh or rollback message from the data control.

**Data Type** Integer (Boolean)

**See Also** [DataSetMaxRow](#) property and [SSSetMaxCol](#) function

**Example** Sheet1.DataSetMaxCol = True

## DataSetMaxRow Property

See also [A-Z Property List](#)

**Description** Determines if the [MaxRow](#) property is set to the number of records currently loaded from the database.

**Syntax (VB)** `[form.][control.]DataSetMaxRow [= {True|False}]`

**Syntax (VC++)** `pSSVB->GetNumProperty("DataSetMaxRow")`  
`pSSVB->SetNumProperty("DataSetMaxRow", {True|False})`

**Remarks** If True, this property sets the **MaxRow** property to limit the number of displayed rows to the number of records currently loaded from the database. If **DataAutoAddNew** is True, an additional row is allowed. The additional row is the blank row at the end of the worksheet for adding new records.

This property is referenced when the worksheet gets a refresh or rollback message from the data control and when a record is added to or deleted from the data set.

**Data Type** Integer (Boolean)

**See Also** [DataSetMaxCol](#) property and [SSSetMaxRow](#) function

**Example** `Sheet1.DataSetMaxRow = True`

## **DataSource Property**

See also [A-Z Property List](#)

**Description** Determines the data control through which the current Formula One control is bound to a database. The property allows read and write capabilities at design time; it is not available at run time.

**Remarks** For additional information, refer to the description of the **DataSource** property in the Microsoft Visual Basic Language Reference Manual.

## DoCancelEdit Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>CancelEdit</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoCancelEdit [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoCancelEdit")</code> <code>pSSVB-&gt;SetNumProperty("DoCancelEdit", {True False})</code>
<b>Remarks</b>	If True, this property allows the CancelEdit event to be fired when the user aborts editing a cell.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">CancelEdit</a> , <a href="#">EndEdit</a> and <a href="#">StartEdit</a> events, <a href="#">DoEndEdit</a> and <a href="#">DoStartEdit</a> properties, and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoCancelEdit = True</code>

## DoClick Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>Click</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoClick [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoClick")</code> <code>pSSVB-&gt;SetNumProperty("DoClick", {True False})</code>
<b>Remarks</b>	If True, this property allows the <b>Click</b> event to be fired when the user clicks the Formula One control. If False, the event is not fired.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">Click</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoClick = True</code>

## DoDataNewRow Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>DataNewRow</b> event can be fired when the data control sends the AddNew message.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>DoDataNewRow</b> [= {True False}]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("DoDataNewRow") pSSVB-> <b>SetNumProperty</b> ("DoDataNewRow", {True False})
<b>Remarks</b>	If True, this property allows the <b>DataNewRow</b> event to be fired when the data control issues an AddNew message. The event is also fired when the user enters the empty row at the end of a worksheet or any call is made to the AddNew method.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DataNewRow</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	Sheet1.DoDataNewRow = True

## DoDataRowLoad Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>DataRowLoad</b> event can be fired after each row is loaded from the data control.
<b>Syntax (VB)</b>	<code>[form.][control.]DoDataRowLoad [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoDataRowLoad")</code> <code>pSSVB-&gt;SetNumProperty("DoDataRowLoad", {True False})</code>
<b>Remarks</b>	<p>If True, the <b>DoDataRowLoad</b> property allows the <b>DataRowLoad</b> event to be fired each time a new record is loaded from the data control.</p> <p>This event is often used when performing processes or calculations on a record before it is displayed.</p>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DataRowLoad</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoDataRowLoad = True</code>



## DoDbClick Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>DbClick</b> event can be fired.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>DoDbClick</b> [= { <b>True False</b> }]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("DoDbClick") pSSVB-> <b>SetNumProperty</b> ("DoDbClick", { <b>True False</b> })
<b>Remarks</b>	<p>If True, this property allows the <b>DbClick</b> 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.</p> <p>The default for this property is True.</p> <p><b>Note</b> Double clicking a Formula One control with the right mouse button always launches the Worksheet Designer.</p> <hr/>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DbClick</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	Sheet1.DoDbClick = False

## DoEndEdit Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>EndEdit</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoEndEdit [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoEndEdit")</code> <code>pSSVB-&gt;SetNumProperty("DoEndEdit", {True False})</code>
<b>Remarks</b>	If True, this property allows the <b>EndEdit</b> event to be fired when the user finishes editing a cell.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">CancelEdit</a> , <a href="#">EndEdit</a> and <a href="#">StartEdit</a> events, <a href="#">DoCancelEdit</a> and <a href="#">DoStartEdit</a> properties, and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoEndEdit = True</code>

## DoEndRecalc Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>EndRecalc</b> event can be fired.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>DoEndRecalc</b> [= { <b>True</b>   <b>False</b> }]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("DoEndRecalc") pSSVB-> <b>SetNumProperty</b> ("DoEndRecalc", { <b>True</b>   <b>False</b> })
<b>Remarks</b>	If True, this property allows the <b>EndRecalc</b> event to be fired when the worksheet finishes recalculation. If you disable this event, processing is accelerated when you perform large operations on a worksheet with Visual Basic code.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">AutoRecalc</a> and <a href="#">DoStartRecalc</a> properties, <a href="#">EndRecalc</a> event, and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	Sheet1.DoEndRecalc = True

## DoSelChange Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>SelChange</b> event can be fired.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>DoSelChange</b> [= { <b>True</b>   <b>False</b> }]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("DoSelChange") pSSVB-> <b>SetNumProperty</b> ("DoSelChange", { <b>True</b>   <b>False</b> })
<b>Remarks</b>	If True, this property allows the <b>SelChange</b> event to be fired when the current selection changes. If you disable this event, processing is accelerated when you perform large operations on a worksheet with Visual Basic code.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SelChange</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	Sheet1.DoSelChange = True

## DoStartEdit Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>StartEdit</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoStartEdit [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoStartEdit")</code> <code>pSSVB-&gt;SetNumProperty("DoStartEdit", {True False})</code>
<b>Remarks</b>	If True, this property allows the <b>StartEdit</b> event to be fired when the current cell enters edit mode.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">CancelEdit</a> , <a href="#">EndEdit</a> and <a href="#">StartEdit</a> events, <a href="#">DoCancelEdit</a> and <a href="#">DoEndEdit</a> properties, and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoStartEdit = True</code>

## DoStartRecalc Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>StartRecalc</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoStartRecalc [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoStartRecalc")</code> <code>pSSVB-&gt;SetNumProperty("DoStartRecalc", {True False})</code>
<b>Remarks</b>	If True, this property allows the <b>StartRecalc</b> event to be fired when the worksheet begins recalculation. If you disable this event, processing is accelerated when you perform large operations on a worksheet with Visual Basic code.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">DoEndRecalc</a> property, <a href="#">StartRecalc</a> event, and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoStartRecalc = True</code>

## DoTopLeftChanged Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the <b>TopLeftChanged</b> event can be fired.
<b>Syntax (VB)</b>	<code>[form.][control.]DoTopLeftChanged[ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("DoTopLeftChanged")</code> <code>pSSVB-&gt;SetNumProperty("DoTopLeftChanged", {True False})</code>
<b>Remarks</b>	If True, this property allows the <b>TopLeftChanged</b> event to be fired when the cell that is displayed as the top left cell of the worksheet changes (e.g., when the user scrolls the worksheet). The execution of this event is deferred until the system is idle.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">TopLeftChanged</a> event and <a href="#">SSSetFireEvent</a> function
<b>Example</b>	<code>Sheet1.DoTopLeftChanged = True</code>

## DragIcon Property

See also [A-Z Property List](#)

**Description** Determines the icon displayed in a drag-and-drop operation.

**Syntax (VB)** `[form.][control.]DragIcon [ = icon]`

**Syntax (VC++)** `pSSVB->GetPictureProperty("DragIcon")`  
`pSSVB->SetPictureProperty("DragIcon", icon)`

**Remarks** The **DragIcon** property settings are:

<b>Setting</b>	<b>Description</b>
(None)	Default Windows Icon.
Icon	A custom mouse pointer. See Microsoft documentation.

For additional information, refer to the description of the **DragIcon** property in the Microsoft Visual Basic Language Reference Manual.

**Data Type** Integer

**See Also** [DragMode](#) property



## DragMode Property

See also [A-Z Property List](#)

**Description** Determines the dragging mode for drag-and-drop operations.

**Syntax (VB)** `[form.][control.]DragMode [ = mode]`

**Syntax (VC++)** `pSSVB->GetNumProperty("DragMode")`  
`pSSVB->SetNumProperty("DragMode", mode)`

**Remarks** The **DragMode** property settings are:

Setting	Description
---------	-------------

0 (Default)	Manual: Requires the drag method to initiate dragging.
-------------	--

1	Automatic: Clicking the source control initiates dragging.
---	--

For additional information, refer to the description of the **DragMode** property in the Microsoft Visual Basic Language Reference Manual.

**Data Type** Integer (Enumerated)

**See Also** [DragIcon](#) property

## EditName Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the edit bar to be used with this worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]EditName [= Editname]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("EditName")</code> <code>pSSVB-&gt;SetStrProperty("EditName", Editname)</code>
<b>Remarks</b>	<p>This property allows you to connect an edit bar control to a worksheet control. Using an edit bar to enter and edit formulas and values is an alternative to in-cell editing. Without the edit bar, all cell values must be manipulated through in-cell editing or with Visual Basic code.</p> <p>To connect a worksheet to an edit bar, the <b>EditName</b> property of the worksheet must match the <b>EditName</b> property of the edit bar. By default both are set to SSEdit1.</p>
<b>Data Type</b>	String
<b>Example</b>	<code>Sheet1.EditName = "editbar1" ' editbar1 is used</code>

## Enabled Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the Formula One object is enabled.
<b>Syntax (VB)</b>	<code>[form.][control.]Enabled [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Enabled")</code> <code>pSSVB-&gt;SetNumProperty("Enabled", {True False})</code>
<b>Remarks</b>	When True, this property enables the Formula One object; when False, the Formula One object is disabled.
<b>Data Type</b>	Integer (Boolean)
<b>Example</b>	<code>Sheet1.Enabled = True ' This worksheet object is enabled</code>

## EnableProtection Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies whether cell protection is enabled in a worksheet control.
<b>Syntax (VB)</b>	<code>[form.][control.]EnableProtection [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("EnableProtection")</code> <code>pSSVB-&gt;SetNumProperty("EnableProtection", {True False})</code>
<b>Remarks</b>	When True, cell protection is enabled in the worksheet control; when False, cell protection is disabled.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSProtectionDlg</a> , <a href="#">SSSetEnableProtection</a> , and <a href="#">SSSetProtection</a> functions
<b>Example</b>	<code>Sheet1.EnableProtection = True ' Cell protection is enabled</code>

## Entry Property

See also [A-Z Property List](#)

**Description** Specifies the formatted contents of a cell.

**Syntax (VB)** `[form.][control.]Entry [= Entry]`

**Syntax (VC++)** `pSSVB->GetStrProperty("Entry")`  
`pSSVB->SetStrProperty("Entry", Entry)`

**Remarks** This property allows you to enter information in a cell as a user would enter information. The program automatically determines the type of data entered (e.g., number, text, formula). It also recognizes date, time, percentage, currency, fraction, and scientific formats.

The Entry property also returns the value of a cell. The value is returned in the same format as it is displayed in the worksheet while entering or editing a cell.

**Data Type** String

**See Also** [Text](#) and [Number](#) properties

**Example** `Sheet1.Entry = "10%" ' Enter 10% (0.10) into current cell`

## ExtraColor Property

See also [A-Z Property List](#)

**Description** Determines the color of the Formula One window outside the active cell area.

**Syntax (VB)** `[form.][control.]ExtraColor [= color]`

**Syntax (VC++)** `pSSVB->GetNumProperty("ExtraColor")`  
`pSSVB->SetNumProperty("ExtraColor", color)`

**Remarks** This property expects a color in the standard Windows environment RGB scheme. The Formula One window outside the active cell area is displayed in this color.

This value can be one of the following:



**Normal RGB Colors.** These colors are specified using the color palette, or by using the RGB or QBColor functions.



**System default colors.** System color constants are specified in the Visual Basic CONSTANT.TXT file.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

**Data Type** Long

**See Also** [BackColor](#) property and [SSSetExtraColor](#) function

**Example**

```
Sheet1.ExtraColor = QBColor(Rnd * 15) ' Random Color  
Sheet1.ExtraColor = RGB(0, 255, 255) ' Cyan
```

## FileName Property

See also [A-Z Property List](#)

**Description** Specifies the name by which a worksheet is loaded and saved.

**Syntax (VB)** `[form.][control.]FileName [ = Filename]`

**Syntax (VC++)** `pSSVB->GetStrProperty("FileName")`  
`pSSVB->SetStrProperty("FileName", Filename)`

**Remarks** 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 is loaded. The file can be a Formula One file or an Excel 4.0 file. When the form is saved, the worksheet is saved in the file specified in the **FileName** property as a Formula One file.

If the **FileName** property is blank, the worksheet is saved with the form instead of in a separate file.

**Caution** If you set **FileName** to the name of an Excel 4.0 file, this file is overwritten with a Formula One file when the form is saved. Excel 4.0 features not supported in Formula One are lost.

---

**Data Type** String

**See Also** [ReadFile](#), [WriteFile](#), and [WriteExcel4](#) properties

**Example** `Sheet1.FileName = "c:\vtss\samples\mysheet.vts" ' Save the worksheet in mysheet.vts`

## FixedCol Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the starting fixed column in the Formula One window.
<b>Syntax (VB)</b>	<code>[form.][control.]FixedCol [= Column]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("FixedCol")</code> <code>pSSVB-&gt;SetNumProperty("FixedCol", Column)</code>
<b>Remarks</b>	The <b>FixedCol</b> property is used with the <b>FixedCols</b> 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.
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">FixedCols</a> , <a href="#">FixedRow</a> , and <a href="#">FixedRows</a> properties and <a href="#">SSSetFixedCols</a> function
<b>Example</b>	<pre>Sheet1.FixedCols = 2 ' Fix 2 columns Sheet1.FixedCol = 10 ' Starting with column 10 at the left of the window</pre>



## FixedCols Property

See also [A-Z Property List](#)

<b>Description</b>	Determines how many columns to fix at the left edge of the worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]FixedCols [= Columns]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("FixedCols")</code> <code>pSSVB-&gt;SetNumProperty("FixedCols", Columns)</code>
<b>Remarks</b>	The <b>FixedCols</b> property is used with the <b>FixedCol</b> 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.
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">FixedCol</a> , <a href="#">FixedRow</a> , and <a href="#">FixedRows</a> properties and <a href="#">SSSetFixedCols</a> function
<b>Example</b>	<pre>Sheet1.FixedCols = 5 ' Fix 5 columns Sheet1.FixedCol = 1 ' Starting with column 1</pre>

## FixedRow Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the starting fixed row in the Formula One window.
<b>Syntax (VB)</b>	<code>[form.][control.]FixedRow [= Row]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("FixedRow")</code> <code>pSSVB-&gt;SetNumProperty("FixedRow", Row)</code>
<b>Remarks</b>	The <b>FixedRow</b> property is used with the <b>FixedRows</b> 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.
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">FixedCol</a> , <a href="#">FixedCols</a> , and <a href="#">FixedRows</a> properties and <a href="#">SSSetFixedRows</a> function
<b>Example</b>	<pre>Sheet1.FixedRows = 1 ' Fix 1 row Sheet1.FixedRow = 3 ' Starting with row 3 at the top of the window</pre>

## FixedRows Property

See also [A-Z Property List](#)

<b>Description</b>	Determines how many rows to fix at the top of the worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]FixedRows [= Rows]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("FixedRows")</code> <code>pSSVB-&gt;SetNumProperty("FixedRows", Rows)</code>
<b>Remarks</b>	The <b>FixedRows</b> property is used with the <b>FixedRow</b> 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.
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">FixedCol</a> , <a href="#">FixedCols</a> , and <a href="#">FixedRow</a> properties and <a href="#">SSSetFixedRows</a> function
<b>Example</b>	<code>Sheet1.FixedRows = 2 ' Fix 2 rows at the top of the window</code> <code>Sheet1.FixedRow = 1 ' Starting with row 1</code>

## FormattedText Property

See also [A-Z Property List](#)

<b>Description</b>	Returns the formatted text string from the active cell. This is a run time, read only property.
<b>Syntax (VB)</b>	<code>[form.][control.]FormattedText</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("FormattedText")</code>
<b>Remarks</b>	The <b>FormattedText</b> property returns the formatted text of the active cell. The text returned is the same as displayed on the screen, with the exception of fill characters created by the '*' character in a custom number format.
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">Text</a> property and <a href="#">SSGetFormattedText</a> and <a href="#">SSGetFormattedTextRC</a> functions
<b>Example</b>	<pre>TheText = Sheet1.FormattedText ' Get the formatted text of the active cell</pre>

## Formula Property

See also [A-Z Property List](#)

<b>Description</b>	Returns or specifies a formula in the active cell as a text string. This is a run time only property.
<b>Syntax (VB)</b>	<code>[form.][control.]Formula [= Formula]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("Formula")</code> <code>pSSVB-&gt;SetStrProperty("Formula", Formula)</code>
<b>Remarks</b>	The <b>Formula</b> property sets or gets the formula of the active cell. If the cell does not contain a formula, the Formula property is blank.
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">Number</a> and <a href="#">Text</a> properties and <a href="#">SSSetFormula</a> function
<b>Example</b>	<code>Sheet1.Formula = "sum(a1:a10)"</code> ' Set the active cell's formula to sum a1:a10 <code>TheFormula\$ = Sheet1.Formula</code> ' Get the current formula of the active cell

## Height Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the dimensions of an object.
<b>Syntax (VB)</b>	<code>[form.][control.]Height [ = height]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetFloatProperty("Height")</code> <code>pSSVB-&gt;SetFloatProperty("Height", height)</code>
<b>Remarks</b>	<p>Measurements are calculated from the center of the control's border. This property uses the scale units of the control's container.</p> <p>For additional information, refer to the description of the <b>Height</b> property in the Microsoft Visual Basic Language Reference Manual.</p>
<b>Data Type</b>	Single
<b>Example</b>	<code>Sheet1.Height = 3375 ' Set the height of the worksheet control</code>

## HelpContextID Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the associated help context number for an object. This property is used to provide context sensitive help in an application.
<b>Syntax (VB)</b>	<code>[form.][control.]HelpContextID [= helpid]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("HelpContextID")</code> <code>pSSVB-&gt;SetNumProperty("HelpContextID", helpid)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>HelpContextID</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Long

## hWnd Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies a handle to the control. This is a run time, read only property.
<b>Syntax (VB)</b>	<i>[form.]</i> <i>[control.]</i> <b>hWnd</b>
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("hWnd")
<b>Remarks</b>	For additional information, refer to the description of the <b>hWnd</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer



## Index Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies a unique number that identifies a control in a control array. This is a run time, read only property.
<b>Syntax (VB)</b>	<i>[form.]</i> <i>[control(i).]</i> <b>Index</b>
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("Index")
<b>Remarks</b>	For additional information, refer to the description of the <b>Index</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer

## Left Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the distance between the internal left edge of an object and the left edge of the container.
<b>Syntax (VB)</b>	<code>[form.][control.]Left [= x]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetFloatProperty("Left")</code> <code>pSSVB-&gt;SetFloatProperty("Left", x)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>Left</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Single
<b>See Also</b>	<a href="#">Top</a> property
<b>Example</b>	<code>Sheet1.Left = 900 ' Sets object 900 units from left edge of container</code>

## LeftCol Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the leftmost column displayed in the worksheet window.
<b>Syntax (VB)</b>	<code>[form.][control.]LeftCol [= Column]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("LeftCol")</code> <code>pSSVB-&gt;SetNumProperty("LeftCol", Column)</code>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">TopRow</a> property and <a href="#">SSSetLeftCol</a> function
<b>Example</b>	<code>Sheet1.LeftCol = 3 ' Display column 3 as the leftmost column</code>

## MaxCol Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the last displayable column in a worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]MaxCol [= Column]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("MaxCol")</code> <code>pSSVB-&gt;SetNumProperty("MaxCol", Column)</code>
<b>Remarks</b>	<p>This property determines the last column that can be displayed in a worksheet. Extra space in the Formula One window is displayed as a solid color using the <a href="#">ExtraColor</a> property.</p> <p>Columns not displayed can still be accessed so an application can use them for data storage and calculations.</p>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">MaxRow</a> , <a href="#">MinCol</a> , and <a href="#">MinRow</a> properties and <a href="#">SSSetMaxCol</a> function
<b>Example</b>	<code>Sheet1.MaxCol = 10 ' End worksheet display with column J</code>

## MaxRow Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the last displayable row in a worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]MaxRow [= Row]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("MaxRow")</code> <code>pSSVB-&gt;SetNumProperty("MaxRow", Row)</code>
<b>Remarks</b>	<p>This property determines the last row that can be displayed in a worksheet. Extra space in the Formula One window is displayed as a solid color using the <a href="#">ExtraColor</a> property.</p> <p>Rows not displayed can still be accessed so an application can use them for data storage and calculations.</p>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">MaxCol</a> , <a href="#">MinCol</a> , and <a href="#">MinRow</a> properties and <a href="#">SSSetMaxRow</a> function
<b>Example</b>	<code>Sheet1.MaxRow = 15 ' End worksheet display with row 15</code>

## MinCol Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the first column that can be displayed in a worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]MinCol [= Column]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("MinCol")</code> <code>pSSVB-&gt;SetNumProperty("MinCol", Column)</code>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">MaxCol</a> , <a href="#">MaxRow</a> , and <a href="#">MinRow</a> properties and <a href="#">SSSetMinCol</a> function
<b>Example</b>	<code>Sheet1.MinCol = 3 ' Start worksheet display with column C</code>

## MinRow Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the first row that can be displayed in a worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]MinRow [= Row]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("MinRow")</code> <code>pSSVB-&gt;SetNumProperty("MinRow", Row)</code>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">MaxCol</a> , <a href="#">MaxRow</a> , and <a href="#">MinCol</a> properties and <a href="#">SSSetMinRow</a> function
<b>Example</b>	<code>Sheet1.MinRow = 5</code> ' Start worksheet display with row 5

## MousePointer Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the type of mouse pointer displayed when the pointer is in the Formula One control.
<b>Syntax (VB)</b>	<code>[form.][control.]MousePointer [= setting]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("MousePointer")</code> <code>pSSVB-&gt;SetNumProperty("MousePointer", setting)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>MousePointer</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">SSSetDoSetCursor</a> function



## Name Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the name by which the object can be referred in the program code. This is a design time only property.
<b>Remarks</b>	<p>The default name is the object type plus a unique integer. For example, the first Formula One object created is Sheet1.</p> <p>For additional information about this property as it refers to objects, refer to the description of the <b>Name</b> property in the Microsoft Visual Basic Language Reference Manual.</p>
<b>Data Type</b>	String

## Number Property

See also [A-Z Property List](#)

**Description** Specifies the numeric value of the active cell. This is a run time only property.

**Syntax (VB)** `[form.][control.]Number [ = Number]`

**Syntax (VC++)** `pSSVB->GetNumProperty("Number")`  
`pSSVB->SetNumProperty("Number", Number)`

**Remarks** The **Number** property can set or get the numeric value of the active cell. If the cell contains text, the property attempts to convert the text to a number.

**Data Type** Long

**See Also** [Formula](#) and [Text](#) properties and [SSSetNumber](#) function

**Examples**

```
Sheet1.Number = 10 ' Set the active cell to 10
TheNumber = Sheet1.Number ' Get the current value of the active
cell
```

## Parent Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the form on which the control is located. This is a run time, read only property.
<b>Syntax (VB)</b>	<code>[control.]Parent</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>Parent</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Form

## PrintArea Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the worksheet areas to be printed.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>PrintArea</b> [= <i>Areas</i> ]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetStrProperty</b> ("PrintArea") pSSVB-> <b>SetStrProperty</b> ("PrintArea", " <i>Areas</i> ")
<b>Remarks</b>	The <b>PrintArea</b> property sets the Print_Area user defined name to the worksheet regions specified in Areas. This name defines the worksheet range(s) to be printed. It can contain multiple ranges (e.g., A1:C3,A11:C13). If PrintArea is set to Null (""), the worksheet is printed from A1 to the last row and last column containing data.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> and <a href="#">SSSetPrintArea</a> functions
<b>Example</b>	Sheet1.PrintArea = "A1:C3,A11:C13"

## PrintBottomMargin, PrintLeftMargin, PrintRightMargin, and PrintTopMargin Properties

See also [A-Z Property List](#)

<b>Description</b>	Determines the top, left, bottom, and right page margins.
<b>Syntax (VB)</b>	<code>[form.][control.]Print...Margin [= Margin]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetFloatProperty("Print...Margin")</code> <code>pSSVB-&gt;SetFloatProperty("Print...Margin", Margin)</code>
<b>Remarks</b>	These properties specify the page margins when printing. They are used to increase or decrease the amount of white space between the worksheet and the edge of the paper. The margins are specified in inches and can be a maximum of 48 inches.
<b>Data Type</b>	Single
<b>See Also</b>	<a href="#">SSFilePrint</a> , <a href="#">SSSetPrintArea</a> , <a href="#">SSSetPrintBottomMargin</a> , <a href="#">SSSetPrintLeftMargin</a> , <a href="#">SSSetPrintRightMargin</a> , and <a href="#">SSSetPrintTopMargin</a> functions
<b>Examples</b>	<code>Sheet1.PrintTopMargin = .75 ' .75" top margin</code> <code>Sheet1.PrintLeftMargin = 1.5 ' 1.5" left margin</code> <code>Sheet1.PrintBottomMargin = 1 ' 1" bottom margin</code> <code>Sheet1.PrintRightMargin = .75 ' .75" right margin</code>

## PrintColHeading Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the worksheet column headings are printed.
<b>Syntax (VB)</b>	<code>[form.][control.]PrintColHeading [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("PrintColHeading")</code> <code>pSSVB-&gt;SetNumProperty("PrintColHeading", {True False})</code>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> and <a href="#">SSSetPrintArea</a> functions
<b>Example</b>	<code>Sheet1.PrintColHeading = True ' Print the column heading</code>

## PrintFooter, PrintHeader Properties

See also [A-Z Property List](#)

**Description** Determines the contents of the page headers and footers.

**Syntax (VB)** `[form.][control.]Print... [= String]`

**Syntax (VC++)** `pSSVB->GetStrProperty("Print...")`  
`pSSVB->SetStrProperty("Print...", String)`

**Remarks** These properties specify the contents of the header and footer. You can enter text with the following special codes:

<b>Format Code</b>	<b>Description</b>
&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 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.

<b>Format Code</b>	<b>Description</b>
&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

**Data Type** String

**See Also** [SSFilePrint](#), [SSSetPrintArea](#), [SSSetPrintFooter](#), and [SSSetPrintHeader](#) functions

**Examples**

```
Sheet1.PrintHeader = "Page &p of &n" ' Print page and total number of pages  
Sheet1.PrintFooter = "&d &t &f" ' Print date, time, and filename
```

## PrintGridLines Property

See also [A-Z Property List](#)


<b>Description</b>	Determines if the grid lines are printed.
<b>Syntax (VB)</b>	<code>[form.][control.]PrintGridLines [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("PrintGridLines")</code> <code>pSSVB-&gt;SetNumProperty("PrintGridLines", {True False})</code>
<b>Remarks</b>	The <b>PrintGridLines</b> property determines if the grid lines are included when you print a worksheet. This property does not affect the display of grid lines in the worksheet.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> and <a href="#">SSSetPrintArea</a> functions
<b>Example</b>	<code>Sheet1.PrintGridLines = True ' Print the grid lines</code>




## PrintHCenter, PrintVCenter Properties

See also [A-Z Property List](#)

<b>Description</b>	Determines how the worksheet is centered on a page.
<b>Syntax (VB)</b>	<code>[form.][control.]Print...Center [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Print...Center")</code> <code>pSSVB-&gt;SetNumProperty("Print...Center", {True False})</code>
<b>Remarks</b>	The <b>PrintHCenter</b> and <b>PrintVCenter</b> properties determine how the worksheet is centered on a page.

 If **PrintHCenter** is True, the printed area of each page is centered between the left and right margins. If False, the printed area is positioned against the left margin.

 If **PrintVCenter** is True, the printed area of each page is centered between the top and bottom margins. If False, the printed area is positioned against the top margin.

**Data Type** Integer (Boolean)

**See Also** [SSFilePrint](#), [SSSetPrintArea](#), [SSSetPrintHCenter](#), and [SSSetPrintVCenter](#) functions

**Examples**

```
Sheet1.PrintHCenter = True ' Center horizontally
Sheet1.PrintVCenter = True ' Center vertically
```

## PrintHeader Property

See also [A-Z Property List](#)

See [PrintFooter](#) property.

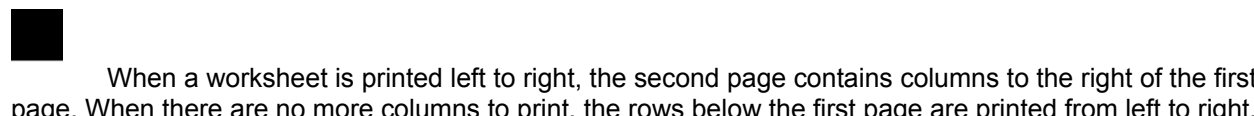
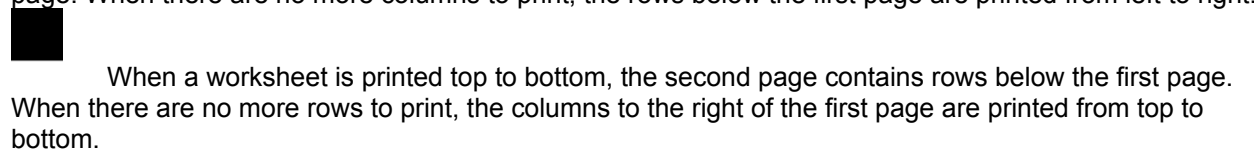
## PrintLeftMargin Property

See also [A-Z Property List](#)

See [PrintBottomMargin](#) property.

## PrintLeftToRight Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the worksheet pages print from top to bottom or left to right.
<b>Syntax (VB)</b>	<code>[form.][control.]PrintLeftToRight [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("PrintLeftToRight")</code> <code>pSSVB-&gt;SetNumProperty("PrintLeftToRight", {True False})</code>
<b>Remarks</b>	Setting the <b>PrintLeftToRight</b> property to True causes the worksheet pages to be printed left to right. Setting the property to False causes the pages to be printed top to bottom.
	 <p>When a worksheet is printed left to right, the second page contains columns to the right of the first page. When there are no more columns to print, the rows below the first page are printed from left to right.</p>
	 <p>When a worksheet is printed top to bottom, the second page contains rows below the first page. When there are no more rows to print, the columns to the right of the first page are printed from top to bottom.</p>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> , <a href="#">SSSetPrintArea</a> , and <a href="#">SSSetPrintLeftToRight</a> functions
<b>Example</b>	<code>Sheet1.PrintLeftToRight = True ' Print the pages left to right</code>

## PrintNoColor Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if worksheet pages are printed in color or black and white.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>PrintNoColor</b> [ = { <b>True</b>   <b>False</b> }]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("PrintNoColor") pSSVB-> <b>SetNumProperty</b> ("PrintNoColor", { <b>True</b>   <b>False</b> })
<b>Remarks</b>	Setting the <b>PrintNoColor</b> property to True causes the worksheet pages to print in black and white only. Setting the property to False causes the pages to be printed in color.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> , <a href="#">SSSetPrintArea</a> , and <a href="#">SSSetPrintNoColor</a> functions
<b>Example</b>	Sheet1.PrintNoColor = True ' Print the pages in black and white only

## **PrintRightMargin Property**

See also [A-Z Property List](#)

See [PrintBottomMargin](#) property.

## PrintRowHeading Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the worksheet row headings are printed.
<b>Syntax (VB)</b>	<code>[form.][control.]PrintRowHeading [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("PrintRowHeading")</code> <code>pSSVB-&gt;SetNumProperty("PrintRowHeading", {True False})</code>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSFilePrint</a> and <a href="#">SSSetPrintArea</a> functions
<b>Example</b>	<code>Sheet1.PrintRowHeading = True ' Print the row heading</code>

## PrintTitles Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the row and column titles printed on each page.
<b>Syntax (VB)</b>	<code>[form.][control.]PrintTitles [= row/col titles]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("PrintTitles")</code> <code>pSSVB-&gt;SetStrProperty("PrintTitles", row/col titles)</code>
<b>Remarks</b>	The <b>PrintTitles</b> property specifies the rows and columns printed on each new page. You must specify a row or column reference. In addition, you must specify entire rows and columns.
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">SSFilePrint</a> and <a href="#">SSSetPrintArea</a> functions
<b>Examples</b>	<code>Sheet1.PrintTitles = "\$A\$1:\$IV\$1" ' Print row 1 on each page</code> <code>Sheet1.PrintTitles = "\$A\$1:\$A\$16384" ' Print column 1 on each page</code> <code>Sheet1.PrintTitles = "\$A\$1:\$IV\$1;\$A\$1:\$A\$16384" ' Print row 1 and column 1 on each page</code>



## PrintTopMargin Property

See also [A-Z Property List](#)

See [PrintBottomMargin](#) property.

## PrintVCenter Property

See also [A-Z Property List](#)

See [PrintHCenter](#) property.

## ReadFile Property

See also [A-Z Property List](#)

<b>Description</b>	Reads a worksheet from a file into a control.
<b>Syntax (VB)</b>	<code>[form.][control.]ReadFile [= Filename]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("ReadFile")</code> <code>pSSVB-&gt;SetStrProperty("ReadFile", Filename)</code>
<b>Remarks</b>	<p>The <b>ReadFile</b> property causes the specified worksheet file to be loaded in the control. The property does not establish a permanent reference to the worksheet. Any changes made to the file on disk are not automatically reflected in the file loaded by the <b>ReadFile</b> property.</p> <p>To establish a permanent reference to a worksheet file, use the <b>FileName</b> property.</p> <p>Formula One can read native files (.VTS extension), Excel 4.0 files (.XLS extension), and tab-delimited text files.</p>
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">FileName</a> , <a href="#">WriteExcel4</a> , and <a href="#">WriteFile</a> properties and <a href="#">SSRead</a> function
<b>Examples</b>	<pre>Sheet1.ReadFile = "c:\vtss\samples\amort.vts" ' Read a Formula One file Sheet1.ReadFile = "c:\vtss\samples\amort.xls" ' Read an Excel file</pre>

## Repaint Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the Formula One control is repainted after a change is made to the worksheet.
<b>Syntax (VB)</b>	<code>[form.][control.]Repaint [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Repaint")</code> <code>pSSVB-&gt;SetNumProperty("Repaint", {True False})</code>
<b>Remarks</b>	Setting the <b>Repaint</b> property to False does not allow the Formula One control to repaint when a change is made to the worksheet. This is useful when several operations are performed on the worksheet and you do not want the worksheet to continually repaint during the process. Setting this property to True causes the control to be refreshed.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetRepaint</a> and <a href="#">SSSetRepaint</a> functions
<b>Example</b>	<code>Sheet1.Repaint = False ' Turn repainting off</code>

## Row Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the active row in the worksheet. This is a run time only property.
<b>Syntax (VB)</b>	<code>[form.][control.]Row [= Row]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Row")</code> <code>pSSVB-&gt;SetNumProperty("Row", Row)</code>
<b>Remarks</b>	<p>The <b>Row</b> property is used along with the <b>Col</b> property to set the active cell in the worksheet. The <b>Row</b> property is automatically changed if a range is selected using the <b>SelStart...</b> and <b>SelEnd...</b> properties.</p> <p>You can specify -1 as the row and column number to indicate all rows or all columns. For example, setting <b>Row</b> to -1 and <b>Col</b> to 1 causes all rows in column 1 to be selected. Setting both <b>Row</b> and <b>Col</b> to -1 selects the entire worksheet.</p>
<b>Data Type</b>	Long
<b>See Also</b>	<a href="#">Col</a> , <a href="#">SelStart...</a> , and <a href="#">SelEnd...</a> properties and <a href="#">SSSetActiveCell</a> function
<b>Examples</b>	<pre>Sheet1.Row = 10 ' Select row 10 column 1 as the active cell Sheet1.Col = 1 Sheet1.Row = -1 ' Select all of column one Sheet1.Col = 1</pre>

## RowMode Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies whether individual cells or entire rows can be selected.
<b>Syntax (VB)</b>	<code>[form.][control.]RowMode [= {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("RowMode")</code> <code>pSSVB-&gt;SetNumProperty("RowMode", {True False})</code>
<b>Remarks</b>	<p>The <b>RowMode</b> property determines how rows are marked. If True, any selection in a row causes the entire row to be selected. This is especially useful when using the control as a database browsing tool.</p> <p>If False, individual cells can be selected. This is the default mode.</p>
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">SSGetRowMode</a> and <a href="#">SSSetRowMode</a> functions
<b>Example</b>	<code>Sheet1.RowMode = True ' Enable row mode</code>

## Selection Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the current selection.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>Selection</b> [= <i>Range</i> ]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetStrProperty</b> ("Selection") pSSVB-> <b>SetStrProperty</b> ("Selection", <i>Range</i> )
<b>Remarks</b>	<p>Selection contains the string representation of the range specified in <b>SelEndCol</b>, <b>SelEndRow</b>, <b>SelStartCol</b>, and <b>SelStartRow</b>. Alternately, you can specify a selection using the <b>Selection</b> property. When a range is selected in this manner, <b>SelEndCol</b>, <b>SelEndRow</b>, <b>SelStartCol</b>, and <b>SelStartRow</b> are automatically updated. Use these properties to select a range before performing operations such as copying or deleting data.</p> <p>To select more than one range, separate the ranges with a comma.</p> <p>The <b>Selection</b> property can also be set to a formula that returns one or more ranges. However, it always returns a simple reference. For example:</p> <pre>Sheet1.Selection = "OFFSET(A1:C1, 1, 0)" ' Select cells A2:C2 MsgBox Sheet1.Selection ' Displays "A2:C2" instead of the                         original formula</pre>
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">SelEndCol</a> , <a href="#">SelEndRow</a> , <a href="#">SelStartCol</a> , and <a href="#">SelStartRow</a> properties and <a href="#">SSSetActiveCell</a> and <a href="#">SSSetSelection</a> functions
<b>Examples</b>	<pre>Sheet1.Selection = "A1:J10" ' Select a 10 by 10 range Sheet1.Selection = "A1:C1,A3:C3" ' Select ranges A1:C1 and A3:C3</pre>

## **SelEndCol, SelEndRow, SelStartCol, and SelStartRow Properties**

See also [A-Z Property List](#)

<b>Description</b>	These properties determine the starting column, starting row, ending column, and ending row of a selected range.
<b>Syntax (VB)</b>	<code>[form.][control.]Sel...Col [= Column]</code> <code>[form.][control.]Sel...Row [= Row]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Sel...Col")</code> <code>pSSVB-&gt;SetNumProperty("Sel...Col", Column)</code>  <code>pSSVB-&gt;GetNumProperty("Sel...Row")</code> <code>pSSVB-&gt;SetNumProperty("Sel...Row", Row)</code>
<b>Remarks</b>	<p><b>SelEndCol</b>, <b>SelEndRow</b>, <b>SelStartCol</b>, and <b>SelStartRow</b> 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.</p> <p>If you need to select multiple ranges, you must use the <b>Selection</b> property or one of the selection function calls.</p>
<b>Data Type</b>	Integer
<b>See Also</b>	<a href="#">Selection</a> property and <a href="#">SSSetSelection</a> function
<b>Examples</b>	<pre>Sheet1.SelStartCol = 1 ' Select a 10 by 10 range Sheet1.SelEndCol = 10 ' Starting at row 1 column 1 Sheet1.SelStartRow = 1 ' Ending at row 10 column 10 Sheet1.SelEndRow = 10</pre>



## **SelEndRow Property**

See also [A-Z Property List](#)

See [SelEndCol](#) property.

## **SelStartCol Property**

See also [A-Z Property List](#)

See [SelEndCol](#) property.

## **SelStartRow Property**

See also [A-Z Property List](#)

See [SelEndCol](#) property.

## ShowColHeading Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if column headings are displayed in the Formula One window.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>ShowColHeading</b> [ = {True False}]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("ShowColHeading") pSSVB-> <b>SetNumProperty</b> ("ShowColHeading", {True False})
<b>Remarks</b>	When True, this property displays column headings in the Formula One window. When False, no column headings are displayed.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">ShowRowHeading</a> properties and <a href="#">SSSetShowColHeading</a> function
<b>Example</b>	Sheet1.ShowColHeading = False ' Disable the column heading display

## ShowGridLines Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the grid lines are displayed in the Formula One window.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>ShowGridLines</b> [= {True False}]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("ShowGridLines") pSSVB-> <b>SetNumProperty</b> ("ShowGridLines", {True False})
<b>Remarks</b>	When True, this property displays gridlines in the Formula One window. When False, no grid lines are displayed.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">PrintGridLines</a> property and <a href="#">SSSetShowGridLines</a> function
<b>Example</b>	Sheet1.ShowGridLines = False ' Disable grid lines display

## ShowHScrollBar Property

See also [A-Z Property List](#)

**Description** Determines how the horizontal scroll bar is displayed.

**Syntax (VB)** `[form.][control.]ShowHScrollBar = {0|1|2}`

**Syntax (VC++)** `pSSVB->GetNumProperty("ShowHScrollBar")`  
`pSSVB->SetNumProperty("ShowHScrollBar", {0|1|2})`

**Remarks** The **ShowHScrollBar** property settings are:

Setting	Description
---------	-------------

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

Setting this property to 0 removes the horizontal scroll bar. Setting this property to 1 displays the horizontal scroll bar. Setting this property to 2 causes the horizontal scroll bar to be displayed if the worksheet is larger than the window and the worksheet is active.

**Data Type** Integer (Enumerated)

**See Also** [ShowVScrollBar](#) property and [SSSetShowHScrollBar](#) function

**Example** `Sheet1.ShowHScrollBar = 2 ' Set scroll bar display to automatic`

## ShowRowHeading Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if row headings are displayed in the Formula One window.
<b>Syntax (VB)</b>	[ <i>form.</i> ][ <i>control.</i> ] <b>ShowRowHeading</b> [ = {True False}]
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("ShowRowHeading") pSSVB-> <b>SetNumProperty</b> ("ShowRowHeading", {True False})
<b>Remarks</b>	When True, this property displays row headings in the Formula One window. When False, no row headings are displayed.
<b>Data Type</b>	Integer (Boolean)
<b>See Also</b>	<a href="#">ShowColHeading</a> property and <a href="#">SSSetShowRowHeading</a> function
<b>Example</b>	Sheet1.ShowRowHeading = False ' Disable the row heading display

## ShowSelections Property

See also [A-Z Property List](#)

**Description** Determines how selections are displayed.

**Syntax (VB)** `[form.][control.]ShowSelections = {0|1|2}`

**Syntax (VC++)** `pSSVB->GetNumProperty("ShowSelections")`  
`pSSVB->SetNumProperty("ShowSelections", {0|1|2})`

**Remarks** The **ShowSelections** property settings are:

Setting	Description
0	Do not display selections
1	Display all selections
2	Display selections in this control only

Setting this property to 0 disables the display of all selections. Setting this property to 1 forces the display of all selections. Setting this property to 2 causes the display of selections in the Formula One control only when the control is active.

**Data Type** Integer (Enumerated)

**See Also** [AllowSelections](#) property and [SSSetShowSelections](#) function

**Example** `Sheet1.ShowSelections = 2 ' Display selections in this control only`



## ShowVScrollBar Property

See also [A-Z Property List](#)

**Description** Determines how the vertical scroll bar is displayed.

**Syntax (VB)** `[form.][control.]ShowVScrollBar = {0|1|2}`

**Syntax (VC++)** `pSSVB->GetNumProperty("ShowVScrollBar")`  
`pSSVB->SetNumProperty("ShowVScrollBar", {0|1|2})`

**Remarks** The **ShowVScrollBar** property settings are:

Setting	Description
---------	-------------

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

Setting this property to 0 removes the vertical scroll bar. Setting this property to 1 displays the vertical scroll bar. Setting this property to 2 causes the vertical scroll bar to be displayed if the worksheet is larger than the window and the control is active.

**Data Type** Integer (Enumerated)

**See Also** [ShowHScrollBar](#) property and [SSSetShowVScrollBar](#) function

**Example** `Sheet1.ShowVScrollBar = 2 ' Set scroll bar display to automatic`

## SS Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies the handle to a worksheet view. This is a run time, read only property.
<b>Syntax (VB)</b>	<i>[form.]</i> [ <i>control.</i> ] <b>SS</b>
<b>Syntax (VC++)</b>	pSSVB-> <b>GetNumProperty</b> ("SS")
<b>Remarks</b>	The <b>SS</b> property specifies the handle to a worksheet view. The handle is needed as an argument for all the worksheet related function calls.
<b>Data Type</b>	Long
<b>Example</b>	<pre>sserror = SSFilePrint(Sheet1.SS, True) ' Print the worksheet</pre>

## TabIndex Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the tab order of the Formula One control within its parent form.
<b>Syntax (VB)</b>	<code>[form.][control.]TabIndex [= index]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("TabIndex")</code> <code>pSSVB-&gt;SetNumProperty("TabIndex", index)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>TabIndex</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer

## TableName Property

See also [A-Z Property List](#)

**Description** Specifies the name by which the worksheet is referred in formulas in other worksheets.

**Syntax (VB)** [*form.*][*control.*]**TableName** [= *string*]

**Syntax (VC++)** pSSVB->GetStrProperty("**TableName**")  
pSSVB->SetStrProperty("**TableName**", *string*)

**Remarks** This property defaults to the value in the [Name](#) property when a view is first created. For example, if this worksheet is named Sales, the formula Sales.A1 in another worksheet returns the value from A1 in this worksheet.

It is helpful to name worksheet something meaningful instead of using the default name (e.g., Sheet1, Sheet2, etc.)

Normally a worksheet can be accessed by multiple users. If you want exclusive access to a worksheet, you can set the TableName property to Null (""). This prohibits someone from attaching to this worksheet or referencing it in a formula.

**Data Type** String

**Example** Sheet1.Formula = "Sales.A1" ' Enter a formula referring to an external worksheet

## TabStop Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the user can use the Tab key to set the focus to this control.
<b>Syntax (VB)</b>	<code>[form.][control.]TabStop [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("TabStop")</code> <code>pSSVB-&gt;SetNumProperty("TabStop", {True False})</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>TabStop</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer (Boolean)

## Tag Property

See also [A-Z Property List](#)

<b>Description</b>	Stores any extra data needed by your application. This property is not used by Visual Basic.
<b>Syntax (VB)</b>	<code>[form.][control.]Tag [ = <i>string</i>]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("Tag")</code> <code>pSSVB-&gt;SetStrProperty("Tag", <i>string</i>)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>Tag</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	String

## Text Property

See also [A-Z Property List](#)

<b>Description</b>	Specifies a text string for the active cell. This is a run time only property.
<b>Syntax (VB)</b>	<code>[form.][control.]Text [ = Text]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("Text")</code> <code>pSSVB-&gt;SetStrProperty("Text", Text)</code>
<b>Remarks</b>	The <b>Text</b> property can set or get the text of the active cell. If the cell does not contain text, the Text property is blank.
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">FormattedText</a> , <a href="#">Formula</a> , and <a href="#">Number</a> properties and <a href="#">SSSetText</a> function
<b>Examples</b>	<code>Sheet1.Text = "Total Sales" ' Set the active cell's text to "Total Sales"</code> <code>TheText = Sheet1.Text ' Get the current text of the active cell</code>

## Top Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the distance between the internal top edge of an object and the top edge of the container.
<b>Syntax (VB)</b>	<code>[form.][control.]Top [= y]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetFloatProperty("Top")</code> <code>pSSVB-&gt;SetFloatProperty("Top", y)</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>Top</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Single
<b>See Also</b>	<a href="#">Left</a> property
<b>Example</b>	<code>Sheet1.Top = 900 ' Sets object 900 units from top edge of container</code>



## TopRow Property

See also [A-Z Property List](#)

**Description** Determines the top row displayed at the top edge of the Formula One window.

**Syntax (VB)** `[form.][control.]TopRow [ = Row]`

**Syntax (VC++)**  
pSSVB->GetNumProperty("TopRow")  
pSSVB->SetNumProperty("TopRow", Row)

**Data Type** Integer

**See Also** [LeftCol](#) property and [SSSetTopRow](#) function

**Example** `Sheet1.TopRow = 10 ' Top row displayed is row 10`

## Visible Property

See also [A-Z Property List](#)

<b>Description</b>	Determines if the Formula One object is visible.
<b>Syntax (VB)</b>	<code>[form.][control.]Visible [ = {True False}]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetNumProperty("Visible")</code> <code>pSSVB-&gt;SetNumProperty("Visible", {True False})</code>
<b>Remarks</b>	For additional information, refer to the description of the <b>Visible</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Integer (Boolean)

## Width Property

See also [A-Z Property List](#)

<b>Description</b>	Determines the width of a Formula One object.
<b>Syntax (VB)</b>	<code>[form.][control.]Width [ = width]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetFloatProperty("Width")</code> <code>pSSVB-&gt;SetFloatProperty("Width", width)</code>
<b>Remarks</b>	Measurements are calculated from the center of the control's border. This property uses the scale units of the control's container.  For additional information, refer to the description of the <b>Width</b> property in the Microsoft Visual Basic Language Reference Manual.
<b>Data Type</b>	Single

## WriteExcel4 Property

See also [A-Z Property List](#)

<b>Description</b>	Writes the current worksheet to the specified file in Excel 4.0 file format.
<b>Syntax (VB)</b>	<code>[form.][control.]WriteExcel4 [= Filename]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("WriteExcel4")</code> <code>pSSVB-&gt;SetStrProperty("WriteExcel4", Filename)</code>
<b>Remarks</b>	<p>The <b>WriteExcel4</b> property causes the current worksheet to be written to disk in Excel 4.0 file format using the specified filename.</p> <p>This does not establish a permanent reference to the file on disk. To establish a permanent reference to a file, use the <code>FileName</code> property.</p>
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">FileName</a> , <a href="#">ReadFile</a> , and <a href="#">WriteFile</a> properties and <a href="#">SSWrite</a> function
<b>Example</b>	<pre>Sheet1.WriteExcel4 = "c:\vtss\samples\new.xls" ' Write an Excel 4.0 file</pre>

## WriteFile Property

See also [A-Z Property List](#)

<b>Description</b>	Writes the current worksheet to the specified file in Formula One format.
<b>Syntax (VB)</b>	<code>[form.][control.]WriteFile [= Filename]</code>
<b>Syntax (VC++)</b>	<code>pSSVB-&gt;GetStrProperty("WriteFile")</code> <code>pSSVB-&gt;SetStrProperty("WriteFile", Filename)</code>
<b>Remarks</b>	<p>The <b>WriteFile</b> property causes the current worksheet to be written to disk using the specified filename.</p> <p>This does not establish a permanent reference to the file on disk. To establish a permanent reference to a file, use the <code>FileName</code> property.</p>
<b>Data Type</b>	String
<b>See Also</b>	<a href="#">FileName</a> , <a href="#">ReadFile</a> , and <a href="#">WriteExcel4</a> properties and <a href="#">SSWrite</a> function
<b>Example</b>	<code>Sheet1.WriteFile = "c:\vtss\samples\new.vts" ' Write Formula One file</code>

## A-Z Worksheet Function Reference

This chapter provides a complete alphabetical reference for the Formula One worksheet functions. Refer to [Built-In Worksheet Functions](#), for additional information about using worksheet functions.

<a href="#">ABS</a>	<a href="#">INDEX</a>	<a href="#">PROPER</a>
<a href="#">ACOS</a>	<a href="#">INDIRECT</a>	<a href="#">PV</a>
<a href="#">ACOSH</a>	<a href="#">INT</a>	<a href="#">RAND</a>
<a href="#">ADDRESS</a>	<a href="#">IPMT</a>	<a href="#">RATE</a>
<a href="#">AND</a>	<a href="#">IRR</a>	<a href="#">REPLACE</a>
<a href="#">ASIN</a>	<a href="#">ISBLANK</a>	<a href="#">REPT</a>
<a href="#">ASINH</a>	<a href="#">ISERR</a>	<a href="#">RIGHT</a>
<a href="#">ATAN</a>	<a href="#">ISERROR</a>	<a href="#">ROUND</a>
<a href="#">ATAN2</a>	<a href="#">ISLOGICAL</a>	<a href="#">ROW</a>
<a href="#">ATANH</a>	<a href="#">ISNA</a>	<a href="#">ROWS</a>
<a href="#">AVERAGE</a>	<a href="#">ISNONTEXT</a>	<a href="#">SEARCH</a>
<a href="#">CALL</a>	<a href="#">ISNUMBER</a>	<a href="#">SECOND</a>
<a href="#">CEILING</a>	<a href="#">ISREF</a>	<a href="#">SIGN</a>
<a href="#">CHAR</a>	<a href="#">ISTEXT</a>	<a href="#">SIN</a>
<a href="#">CHOOSE</a>	<a href="#">LEFT</a>	<a href="#">SINH</a>
<a href="#">CLEAN</a>	<a href="#">LEN</a>	<a href="#">SLN</a>
<a href="#">CODE</a>	<a href="#">LN</a>	<a href="#">SQRT</a>
<a href="#">COLUMN</a>	<a href="#">LOG</a>	<a href="#">STDEV</a>
<a href="#">COLUMNS</a>	<a href="#">LOG10</a>	<a href="#">STDEVP</a>
<a href="#">COS</a>	<a href="#">LOOKUP</a>	<a href="#">SUBSTITUTE</a>
<a href="#">COSH</a>	<a href="#">LOWER</a>	<a href="#">SUM</a>
<a href="#">COUNT</a>	<a href="#">MATCH</a>	<a href="#">SUMSQ</a>
<a href="#">COUNTA</a>	<a href="#">MAX</a>	<a href="#">SYD</a>
<a href="#">DATE</a>	<a href="#">MID</a>	<a href="#">T</a>
<a href="#">DATEVALUE</a>	<a href="#">MIN</a>	<a href="#">TAN</a>
<a href="#">DAY</a>	<a href="#">MINUTE</a>	<a href="#">TANH</a>
<a href="#">DB</a>	<a href="#">MIRR</a>	<a href="#">TEXT</a>
<a href="#">DDB</a>	<a href="#">MOD</a>	<a href="#">TIME</a>
<a href="#">DOLLAR</a>	<a href="#">MONTH</a>	<a href="#">TIMEVALUE</a>
<a href="#">ERROR.TYPE</a>	<a href="#">N</a>	<a href="#">TODAY</a>
<a href="#">EVEN</a>	<a href="#">NA</a>	<a href="#">TRIM</a>
<a href="#">EXACT</a>	<a href="#">NOT</a>	<a href="#">TRUE</a>
<a href="#">EXP</a>	<a href="#">NOW</a>	<a href="#">TRUNC</a>
<a href="#">FACT</a>	<a href="#">NPER</a>	<a href="#">TYPE</a>
<a href="#">FALSE</a>	<a href="#">NPV</a>	<a href="#">UPPER</a>
<a href="#">FIND</a>	<a href="#">ODD</a>	<a href="#">VALUE</a>
<a href="#">FIXED</a>	<a href="#">OFFSET</a>	<a href="#">VAR</a>
<a href="#">FLOOR</a>	<a href="#">OR</a>	<a href="#">VARP</a>
<a href="#">FV</a>	<a href="#">PI</a>	<a href="#">VDB</a>
<a href="#">HLOOKUP</a>	<a href="#">PMT</a>	<a href="#">VLOOKUP</a>
<a href="#">HOUR</a>	<a href="#">PPMT</a>	<a href="#">WEEKDAY</a>
<a href="#">IF</a>	<a href="#">PRODUCT</a>	<a href="#">YEAR</a>

## ABS

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the absolute value of a number.
<b>Syntax</b>	<b>ABS</b> ( <i>number</i> )  <i>number</i> is any integer.
<b>Remarks</b>	An absolute value does not display a positive or negative sign.
<b>See Also</b>	<a href="#">SIGN</a> function
<b>Examples</b>	ABS (-1) returns 1 ABS (1) returns 1

## ACOS

See also [A-Z Worksheet Function List](#)

**Description** Returns the arc cosine of a number.

**Syntax** **ACOS**(*number*)

*number* is the cosine of the angle. The cosine can range from 1 to -1.

**Remarks** The resulting angle is returned in radians (from 0 to  $\pi$ ).

**See Also** [COS](#) and [PI](#) functions

**Examples**  
ACOS (.5) returns 1.05  
ACOS (-.2) returns 1.77



## ACOSH

See also [A-Z Worksheet Function List](#)

**Description** Returns the inverse hyperbolic cosine of a number.

**Syntax** **ACOSH**(*number*)

*number* is any number equal to or greater than 1.

**See Also** [ASINH](#), [ATANH](#), and [COSH](#) functions

**Examples** ACOSH (1.2) returns .62

ACOSH (3) returns 1.76

## ADDRESS

See also [A-Z Worksheet Function List](#)

**Description** Creates a cell address as text.

**Syntax** **ADDRESS**(*row*, *column*, *ref\_type* [, *a1*] [, *sheet*])

*row* is the row number for the cell address.

*column* is the column number for the cell address.

*ref\_type* is the cell reference type. The following table lists the values for this argument.

<b>Argument</b>	<b>Reference type</b>
-----------------	-----------------------

---

1	Absolute
2	Absolute row, relative column
3	Relative row, absolute column
4	Relative

*a1* is the reference format. This argument must be TRUE() to represent an A1 reference format; Formula One does not support the R1C1 reference format.

*sheet* is the name of an external spreadsheet. Omitting this argument assumes that the reference exists in the current spreadsheet.

**See Also** [COLUMN](#), [OFFSET](#), and [ROW](#) functions

**Examples**  
ADDRESS (5, 6, 1) returns \$F\$5  
ADDRESS (5, 6, 4, TRUE (), SALES.VTS) returns SALES.VTS!F5

## AND

See also [A-Z Worksheet Function List](#)

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

**Syntax** **AND**(*logical\_list*)

*logical\_list* is 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, #VALUE! is returned.

**See Also** [IF](#), [NOT](#), and [OR](#) functions

**Examples**  
AND(1+1=2, 5+5=10) returns True because both arguments are true.  
AND(TRUE(), FALSE()) returns False

## ASIN

See also [A-Z Worksheet Function List](#)

**Description** Returns the arcsine of a number.

**Syntax** **ASIN**(*number*)

*number* is the sine of the resulting angle, ranging from -1 to 1.

**Remarks** The resulting angle is returned in radians (ranging from  $-\pi/2$  to  $\pi/2$ ).

**See Also** [ASINH](#), [PI](#), and [SIN](#) functions

**Examples** ASIN(-1) returns -1.57

ASIN(.4) returns .41

## ASINH

See also [A-Z Worksheet Function List](#)

**Description** Returns the inverse hyperbolic sine of a number.

**Syntax** ASINH(*number*)

*number* is any number.

**See Also** [ACOSH](#), [ASIN](#), [ATANH](#), and [SINH](#) functions

**Examples** ASINH(5.3) returns 2.37

ASINH(-4) returns -2.09

## ATAN

See also [A-Z Worksheet Function List](#)

**Description** Returns the arctangent of a number.

**Syntax** **ATAN**(*number*)

*number* is the tangent of the angle.

**Remarks** The resulting angle is returned in radians, ranging from  $-\pi/2$  to  $\pi/2$ . To find the result in degrees, multiply the result by 180/PI().

**See Also** [ATAN2](#), [ATANH](#), [PI](#), and [TAN](#) functions

**Examples**  
ATAN(3.5) returns 1.29  
ATAN(-4) returns -1.33

## ATAN2

See also [A-Z Worksheet Function List](#)

**Description** Returns the arctangent of the specified coordinates.

**Syntax** **ATAN2**(*x*, *y*)

*x* is the x coordinate.

*y* is 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  $\pi$ , excluding  $-\pi$ .

**See Also** [ATAN](#), [ATANH](#), [PI](#), and [TAN](#) functions

**Examples**  
ATAN2 (3, 6) returns 1.11  
ATAN2 (-1, .1) returns 3.04

## ATANH

See also [A-Z Worksheet Function List](#)

**Description** Returns the inverse hyperbolic tangent of a number.

**Syntax** **ATANH**(*number*)

*number* is a number between -1 and 1, excluding -1 and 1.

**See Also** [ACOS](#), [ASINH](#), and [TANH](#) functions

**Examples**  
ATANH (.5) returns .55  
ATANH (-.25) returns -.26



## AVERAGE

See also [A-Z Worksheet Function List](#)

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

**Syntax** **AVERAGE**(*number\_list*)

*number\_list* is 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.

**See Also** [MIN](#) and [MAX](#) functions

**Examples**  
AVERAGE(5, 6, 8, 14) returns 8.25  
AVERAGE(C15:C17) returns 134; C15:C17 contains 24,144, and 234

## CALL

See also [A-Z Worksheet Function List](#)

**Description** Calls a custom function in a dynamic linked library (DLL).

**Syntax** **CALL**(*file\_name*, *func\_name*, *data\_type*, *argument\_list*)

*file\_name* is the name of the DLL that contains the custom function. The file name should be provided as a quoted text string. You can also provide the path for the file.

*func\_name* is the name of the custom function to be called from the DLL. The function name should be provided as a quoted text string.

*argument\_list* is the list of arguments supplied to the custom function.

*data\_type* is the data type, as a quoted text string, of the arguments and return value of the custom function. The following table lists the data type codes that can be used for this argument.

Data type	Description	Pass by	C declaration
A	Logical (False =0, True =1)	Value	short int
B	IEEE 8-byte floating point number	Value	double
C	Null-terminated string (255 characters maximum)	Reference	char*
D	Byte-counted string (first byte contains string length; 255 characters maximum)	Reference	unsigned char *
E	IEEE 8-byte floating point number	Reference	double*
F	Null-terminated string (255 characters maximum)	Reference	char*
G	Byte-counted string (first byte contains string length; 255 characters maximum)	Reference	unsigned char*
H	Unsigned 2-byte integer	Value	unsigned short int
I	Signed 2-byte integer	Value	short int
J	Signed 4-byte integer	Value	long int
L	Logical (False=0, True =1)	Reference	short int*
M	Signed 2-byte integer	Reference	short int*
N	Signed 4-byte integer	Reference	long int*

### Remarks

For declarations made in C, it is assumed that your compiler defaults to 8-byte doubles, 2-byte short integers, and 4-byte long integers. In the Windows programming environment, all pointers should be far pointers.

Pascal calling conventions are used for all functions called from DLLs. For most C compilers, you must add the `--Pascal` keyword to the function declaration.

If the return value for your custom function uses a pass-by-reference data type, a null pointer can be passed as the return value. The null pointer is interpreted as the `#NUM!` error value.

For F and G data types, a custom function can modify an allocated string buffer. If the

return value type code is F or G, the value returned by the function is ignored. The list of function arguments is searched for the first data type that corresponds to the return value type. The current contents of the allocated string buffer is taken for the return value. 256 bytes is allocated for the argument; therefore, a function can return a larger string than it receives.

You can use a single digit (n), with a value from 1 to 9, as the code for data\_type. The variable in the location pointed to by the nth argument is modified instead of the return value; this process is referred to as modifying in place. The nth argument must be a pass-by-reference data type. In addition, you must declare the function void. For most C compilers, you can add the Void keyword to the function declaration.

**Example**

```
CALL ("\VTFORM1\DEMO4\CUSTFUNC.DLL", "Quotient", "BBB", 3, 2)
```

## CEILING

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Rounds a number up to the nearest multiple of a specified significance.
<b>Syntax</b>	<b>CEILING</b> ( <i>number</i> , <i>significance</i> )  <i>number</i> is the value to round. <i>significance</i> is the multiple to which to round.
<b>Remarks</b>	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, #VALUE! is returned. When the arguments have opposite signs, #NUM! is returned.
<b>See Also</b>	<a href="#">EVEN</a> , <a href="#">FLOOR</a> , <a href="#">INT</a> , <a href="#">ODD</a> , <a href="#">ROUND</a> , and <a href="#">TRUNC</a> functions
<b>Examples</b>	CEILING(1.23459, .05) returns 1.25 CEILING(-148.24, -2) returns -150

## CHAR

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns a character that corresponds to the supplied ANSI code.
<b>Syntax</b>	<b>CHAR</b> ( <i>number</i> )  number is a value between 1 and 255 that specifies an ANSI character.
<b>Remarks</b>	The character and associated numeric code are defined by Windows in the ANSI character set.
<b>See Also</b>	<a href="#">CODE</a> function
<b>Examples</b>	CHAR (70) returns F CHAR (35) returns #

## CHOOSE

See also [A-Z Worksheet Function List](#)

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

**Syntax** **CHOOSE**(*index*, *item\_list*)

*index* is a number that refers to an item in *item\_list*.



*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.

*item\_list* is 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.

**See Also** [INDEX](#) function

**Examples** CHOOSE(2, Q1, Q2, Q3, Q4) returns Q2

AVERAGE(CHOOSE(1, A1:A10, B1:B10, C1:C10)) returns the average of the contents of range A1:A10.

## CLEAN

See also [A-Z Worksheet Function List](#)

**Description** Removes all non-printable characters from the supplied text.

**Syntax** **CLEAN**(*text*)

*text* is any worksheet information.

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

**See Also** [CHAR](#) and [TRIM](#) functions

**Example** `CLEAN(Payments & CHAR(8) & Due)` returns Payments Due because the character returned by CHAR(8) is non-printable.

## CODE

See also [A-Z Worksheet Function List](#)

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

**Syntax** **CODE**(*text*)

*text* is any string.

**Remarks** The numeric code and associated string are defined in your computers character set. The character set used by Windows is the ANSI character set.

**See Also** [CHAR](#) function

**Examples**  
CODE (A) returns 65  
CODE (b) returns 98



## COLUMN

See also [A-Z Worksheet Function List](#)

**Description** Returns the column number of the supplied reference.

**Syntax** **COLUMN**(*reference*)

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

**See Also** [COLUMNS](#) and [ROW](#) functions

**Examples**  
COLUMN (B3) returns 2  
COLUMN () returns 4 if the function is entered in cell D2.

## COLUMNS

See also [A-Z Worksheet Function List](#)

**Description** Returns the number of columns in a range reference.

**Syntax** **COLUMNS**(*range*)

*range* is a reference to a range of cells.

**See Also** [COLUMN](#) and [ROWS](#) functions

**Example** COLUMNS (A1 : D5) returns 4

## COS

See also [A-Z Worksheet Function List](#)

**Description** Returns the cosine of an angle.

**Syntax** **COS**(*number*)

*number* is any number.

**See Also** [ACOS](#), [ASINH](#), [ATANH](#), [COSH](#), and [PI](#) functions

**Examples**  
COS (1.444) returns .126  
COS (5) returns .28

## COSH

See also [A-Z Worksheet Function List](#)

**Description** Returns the hyperbolic cosine of a number.

**Syntax** **COSH**(*number*)

*number* is any number.

**See Also** [ASINH](#), [ATANH](#), and [COS](#) functions

**Examples**  
COSH (2.10) returns 4.14  
COSH (.24) returns 1.03

## COUNT

See also [A-Z Worksheet Function List](#)

**Description** Returns the number of values in the supplied list.

**Syntax** **COUNT**(*value\_list*)

*value\_list* is a list of values. The list can contain as many as 30 values.

**Remarks** **COUNT** only numerates numbers or numerical values (e.g., 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.

**See Also** [AVERAGE](#), [COUNTA](#), and [SUM](#) functions

**Examples**  
COUNT(5, 6, Q2) returns 2  
COUNT(03/06/94, 06/21/94, 10/19/94) returns 3

## COUNTA

See also [A-Z Worksheet Function List](#)

**Description** Returns the number of non-blank values in the supplied list.

**Syntax** **COUNTA**(*expression\_list*)

*expression\_list* is 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.

**See Also** [AVERAGE](#), [COUNT](#), [PRODUCT](#), and [SUM](#) functions

**Examples** COUNTA(32, 45, Earnings, "") returns 4

COUNTA(C38:C40) returns 0 when the specified range contains empty cells

## DATE

See also [A-Z Worksheet Function List](#)

**Description** Returns the serial number of the supplied date.

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

*year* is 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* is a number representing the month (e.g., 12 represents December). If a number greater than 12 is supplied, the number is added to the to the first month of the specified year.

*day* is 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.

**See Also** [DATEVALUE](#), [DAY](#), [MONTH](#), [NOW](#), [TIMEVALUE](#), [TODAY](#), and [YEAR](#) functions

**Examples**  
DATE (94, 6, 21) returns 34506  
DATE (99, 3, 6) returns 36225

## DATEVALUE

See also [A-Z Worksheet Function List](#)

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

**Syntax** **DATEVALUE**(*text*)

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

**See Also** [NOW](#), [TIMEVALUE](#), and [TODAY](#) functions

**Examples**  
DATEVALUE (3/6/94) returns 34399  
DATEVALUE (12/25/95) returns 35058



## DAY

See also [A-Z Worksheet Function List](#)

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

**Syntax** **DAY**(*serial\_number*)

*serial\_number* is a date represented as a serial number or as text (e.g., 06-21-94 or 21-Jun-94).

**See Also** [HOUR](#), [MINUTE](#), [MONTH](#), [NOW](#), [SECOND](#), [TODAY](#), [WEEKDAY](#), and [YEAR](#) functions

**Examples**  
DAY (34399) returns 6  
DAY (06-21-94) returns 21

## DB

See also [A-Z Worksheet Function List](#)

**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*])

*cost* is the initial cost of the asset.

*salvage* is the salvage value of the asset.

*life* is the number of periods in the useful life of the asset.

*period* is the period for which to calculate the depreciation. The time units used to determine period and life must match.

*months* is the number of months in the first year of the items life. Omitting this argument assumes there are 12 months in the first year.

**See Also** [DDB](#), [SLN](#), [SYD](#), and [VDB](#) functions

**Example** DB(10000, 1000, 7, 3) returns 1451.52

## DDB

See also [A-Z Worksheet Function List](#)

**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*])

*cost* is the initial cost of the asset.

*salvage* is the salvage value of the asset.

*life* is the number of periods in the useful life of the asset.

*period* is the period for which to calculate the depreciation. The time units used to determine period and life must match.

*factor* is 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.

**See Also** [DB](#) , [SLN](#), [SYD](#), and [VDB](#) functions

**Example** DDB (10000,1000, 7, 3) returns 1457.73

## DOLLAR

See also [A-Z Worksheet Function List](#)

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

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

*number* is a number, a formula that evaluates to a number, or a reference to a cell that contains a number.

*precision* is a value representing the number of decimal places to the right of the decimal point. Omitting this argument assumes two decimal places.

**See Also** [FIXED](#), [TEXT](#), and [VALUE](#) functions

**Examples**  
DOLLAR(1023.789) returns \$1023.79  
DOLLAR(495.301, -2) returns \$500

## ERROR.TYPE

See also [A-Z Worksheet Function List](#)

**Description** Returns a number corresponding to an error.

**Syntax** **ERROR.TYPE**(*error\_ref*)

*error\_ref* is a cell reference.

**Remarks** The following table lists the error text and associated error numbers returned by this function.

<b>Number</b>	<b>Error text</b>
1	#NULL!
2	#DIV/0!
3	#VALUE!
4	#REF!
5	#NAME?
6	#NUM!
7	#N/A
#N/A	Other

**See Also** [ISERR](#) and [ISERROR](#) functions

**Example** `ERROR.TYPE(A1)` returns 2 if the formula in cell A1 attempts to divide by zero.

## **EVEN**

See also [A-Z Worksheet Function List](#)

**Description** Rounds the specified number up to the nearest even integer.

**Syntax** **EVEN**(*number*)

*number* is any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

**See Also** [CEILING](#), [FLOOR](#), [INT](#), [ODD](#), [ROUND](#), and [TRUNC](#) functions

**Examples**  
EVEN (2 . 5) returns 4  
EVEN (2030 . 45) returns 2032

## EXACT

See also [A-Z Worksheet Function List](#)

**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*)

*expression1* is any text.

*expression2* is any text.

**See Also** [LEN](#) and [SEARCH](#) functions

**Examples** EXACT (Match, Match) returns True  
EXACT (Match, match) returns False

## EXP

See also [A-Z Worksheet Function List](#)

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

**Syntax** **EXP**(*number*)

*number* is any number as the exponent.

**See Also** [LN](#) and [LOG](#) functions

**Examples**  
EXP (2.5) returns 12.18  
EXP (3) returns 20.09



## FACT

See also [A-Z Worksheet Function List](#)

**Description** Returns the factorial of a specified number.

**Syntax** **FACT**(*number*)

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

**See Also** [PRODUCT](#) function

**Examples**  
FACT (2 . 5) returns 2  
FACT (6) returns 720

## **FALSE**

See also [A-Z Worksheet Function List](#)

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

**Syntax** **FALSE()**

**See Also** [TRUE](#) function

## FIND

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Searches for a string of text within another text string and returns the character position at which the search string first occurs.
<b>Syntax</b>	<b>FIND</b> ( <i>search_text</i> , <i>text</i> [, <i>start_position</i> ])  <i>search_text</i> is the text to find. If you specify an empty string (""), <b>FIND</b> matches the first character in <i>text</i> .  <i>text</i> is the text to be searched.  <i>start_position</i> is the character position in <i>text</i> where the search begins. The first character in <i>text</i> is character number 1. When you omit this argument, the default starting position is character number 1.
<b>Remarks</b>	<b>FIND</b> is case-sensitive. You cannot use wildcard characters in the <i>search_text</i> .
<b>See Also</b>	<a href="#">EXACT</a> , <a href="#">LEN</a> , <a href="#">MID</a> , and <a href="#">SEARCH</a> functions
<b>Examples</b>	FIND(time, Theres no time like the present) returns 12 FIND(4, Aisle 4, Part 123-4-11, 9) returns 19

## FIXED

See also [A-Z Worksheet Function List](#)

**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*])

*number* is any number.

*precision* is 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 (e.g., 1,000.00).

**See Also** [DOLLAR](#), [ROUND](#), [TEXT](#), and [VALUE](#) functions

**Examples**  
FIXED(2000.5, 3) returns 2,000.500  
FIXED(2009.5, -1, 1) returns 2010

## FLOOR

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Rounds a number down to the nearest multiple of a specified significance.
<b>Syntax</b>	<b>FLOOR</b> ( <i>number</i> , <i>significance</i> )  <i>number</i> is the value to round. <i>significance</i> is the multiple to which to round.
<b>Remarks</b>	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.
<b>See Also</b>	<a href="#">CEILING</a> , <a href="#">EVEN</a> , <a href="#">INT</a> , <a href="#">ODD</a> , <a href="#">ROUND</a> , and <a href="#">TRUNC</a> functions
<b>Examples</b>	FLOOR(1.23459, .05) returns 1.2 FLOOR(-148.24, -2) returns -148

## FV

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the future value of an annuity based on regular payments and a fixed interest rate.
<b>Syntax</b>	<b>FV</b> ( <i>interest</i> , <i>nper</i> , <i>payment</i> [, <i>pv</i> ] [, <i>type</i> ])  <i>interest</i> is the fixed interest rate. <i>nper</i> is the number of payments in an annuity. <i>payment</i> is the fixed payment made each period. <i>pv</i> is 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. <i>type</i> 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.
<b>Remarks</b>	The units used for interest must match those used for <i>nper</i> . For example, if the annuity has an 8% annual interest rate over a period of 5 years, specify 8%/12 for interest and 5*12 for <i>nper</i> .  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.
<b>See Also</b>	<a href="#">IPMT</a> , <a href="#">NPER</a> , <a href="#">PMT</a> , <a href="#">PPMT</a> , <a href="#">PV</a> , and <a href="#">RATE</a> functions
<b>Examples</b>	FV(5%, 8, -500) returns 4,774.55 FV(10%/12, 240, -700, 1) returns 531,550.86

## HLOOKUP

See also [A-Z Worksheet Function List](#)

**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*)

*search\_item* is 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* is 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 (e.g., -2, -1, 0, 2...A through Z, False, True). Text searches are not case-sensitive.

*row\_index* is 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, #VALUE! is returned.

When *row\_index* is greater than the number of rows in the table, #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*, #REF! is returned.

**See Also** [INDEX](#), [LOOKUP](#), [MATCH](#), and [VLOOKUP](#) functions

	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

**Examples** In the preceding worksheet:

HLOOKUP(Northeast, B1:E5, 3) returns 22.63

HLOOKUP(Pacific, B1:E5, 7) returns #REF!

## HOUR

See also [A-Z Worksheet Function List](#)

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

**Syntax** **HOUR**(*serial\_number*)

*serial\_number* is 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).

**See Also** [DAY](#), [MINUTE](#), [MONTH](#), [NOW](#), [SECOND](#), [WEEKDAY](#), and [YEAR](#) functions

**Examples**  
HOUR(34259.4) returns 9  
HOUR(34619.976) returns 23



## IF

See also [A-Z Worksheet Function List](#)

**Description** Tests the condition and returns the specified value.

**Syntax** **IF**(*condition*, *true\_value*, *false\_value*)

*condition* is any logical expression.

*true\_value* is the value to be returned if condition evaluates to True.

*false\_value* is the value to be returned if condition evaluates to False.

**See Also** [AND](#), [FALSE](#), [NOT](#), [OR](#) , and [TRUE](#) functions

**Example** **IF**(A1>10, Greater, Less) returns Greater if the contents of A1 is greater than 10 and Less if the contents of A1 is less than 10.

## INDEX

See also [A-Z Worksheet Function List](#)

**Description** Returns the contents of a cell from a specified range.

**Syntax** **INDEX**(*reference* [, *row*] [, *column*] [, *range\_number*])

*reference* is 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 (e.g., (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 (e.g., INDEX(A1:A15, 3,, 1)).

*row* is the row number in *reference* from which to return data.

*column* is 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*.

**See Also** [CHOOSE](#), [HLOOKUP](#), [LOOKUP](#), [MATCH](#), and [VLOOKUP](#) functions

	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

**Examples** In the preceding worksheet:

INDEX(A2:B6, 2, 2) returns \$1415.35

INDEX((A2:B6, D2:E6), 4, 2, 2) returns \$1634.58

## INDIRECT

See also [A-Z Worksheet Function List](#)

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

**Syntax** **INDIRECT**(*ref\_text* [, *a1*])

*ref\_text* is a reference to a cell that references a third cell. If *ref\_text* is not a valid reference, #REF! is returned.

*a1* is the reference format. This argument must be TRUE() to represent an A1 reference format; Formula One does not support the R1C1 reference format.

**See Also** [OFFSET](#) function

**Example** `INDIRECT(C1)` returns the contents of the cell that C1 references. If C1 contains D1, the contents of D1 is returned by INDIRECT.

## INT

See also [A-Z Worksheet Function List](#)

**Description** Rounds the supplied number down to the nearest integer.

**Syntax** **INT**(*number*)

*number* is any real number.

**See Also** [CEILING](#), [FLOOR](#), [MOD](#), [ROUND](#), and [TRUNC](#) functions

**Examples**  
INT(10.99) returns 10  
INT(-10.99) returns -11

## IPMT

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.
<b>Syntax</b>	<b>IPMT</b> ( <i>interest</i> , <i>per</i> , <i>nper</i> , <i>pv</i> , [ <i>fv</i> ], [ <i>type</i> ])  <i>interest</i> is the fixed periodic interest rate.  <i>per</i> is the period for which to return the interest payment. This number must be between 1 and <i>nper</i> .  <i>nper</i> is the number of payments.  <i>pv</i> is the present value, or the lump sum amount the annuity is currently worth.  <i>fv</i> is the future value, or the value after all payments are made. If this argument is omitted, the future value is assumed to be 0.  <i>type</i> 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.
<b>Remarks</b>	The units used for interest must match those used for <i>nper</i> . For example, if the annuity has an 8% annual interest rate over a period of 5 years, specify 8%/12 for interest and 5*12 for <i>nper</i> .  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.
<b>See Also</b>	<a href="#">FV</a> , <a href="#">PMT</a> , <a href="#">PPMT</a> , and <a href="#">RATE</a> functions
<b>Examples</b>	IPMT(8%/12, 2, 48, 18000) returns -117.87 IPMT(8%/12, 2, 48, 18000, 0, 1) returns -117.09

## IRR

See also [A-Z Worksheet Function List](#)

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

**Syntax** **IRR**(*cash\_flow* [, *guess*])

*cash\_flow* is 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* is 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*.

**See Also** [MIRR](#), [NPV](#), and [RATE](#) functions

	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

**Examples** In the preceding worksheet:

IRR (B1 : B6) returns 3.72%

IRR (B1 : B3, -20%) returns -49.26%

## ISBLANK

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified cell is blank.
<b>Syntax</b>	<b>ISBLANK</b> ( <i>reference</i> )  <i>reference</i> is a reference to any cell.
<b>Remarks</b>	If the referenced cell is blank, True is returned. False is returned if the cell is not blank.
<b>See Also</b>	<a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Example</b>	ISBLANK (A1) returns True if A1 is a blank cell.

## ISERR

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression returns an error value.
<b>Syntax</b>	<b>ISERR</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns any error except #N/A!, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Example</b>	ISERR (A1) returns True if A1 contains a formula that returns an error (e.g., #NUM!).



## ISERROR

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression returns an error value.
<b>Syntax</b>	<b>ISERROR</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns any error value (e.g., #N/A!, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!), True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Examples</b>	ISERROR (4/0) returns True ISERROR (A1) returns False if A1 contains a formula that does not return an error.

## ISLOGICAL

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression returns a logical value.
<b>Syntax</b>	<b>ISLOGICAL</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the expression returns a logical value, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Example</b>	ISLOGICAL ( ISBLANK ( A1 ) ) returns True because ISBLANK returns a logical value.

## ISNA

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression returns the value not available error.
<b>Syntax</b>	<b>ISNA</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns the #N/A! error, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNONTTEXT</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Example</b>	ISNA (A1) returns True if cell A1 contains the NA() function or returns the error value #N/A!.

## ISNONTEXT

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression is not text.
<b>Syntax</b>	<b>ISNONTEXT</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns any value that is not text, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNUMBER</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Examples</b>	ISNONTEXT (F3) returns True if cell F3 contains a number or is a blank cell. ISNONTEXT (text) returns False.

## ISNUMBER

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression is a number.
<b>Syntax</b>	<b>ISNUMBER</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns a number, True is returned. Otherwise, False is returned. If <i>expression</i> returns a number represented as text (e.g., 12), False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISREF</a> , and <a href="#">ISTEXT</a> functions
<b>Examples</b>	ISNUMBER (123.45) returns True ISNUMBER (123) returns False

## ISREF

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression is a range reference.
<b>Syntax</b>	<b>ISREF</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns a range reference, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , and <a href="#">ISTEXT</a> functions
<b>Example</b>	ISREF (A3) returns True

## ISTEXT

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines if the specified expression is text.
<b>Syntax</b>	<b>ISTEXT</b> ( <i>expression</i> )  <i>expression</i> is any expression.
<b>Remarks</b>	If the <i>expression</i> returns text, True is returned. Otherwise, False is returned.
<b>See Also</b>	<a href="#">ISBLANK</a> , <a href="#">ISERR</a> , <a href="#">ISERROR</a> , <a href="#">ISLOGICAL</a> , <a href="#">ISNA</a> , <a href="#">ISNONTEXT</a> , <a href="#">ISNUMBER</a> , and <a href="#">ISREF</a> functions
<b>Example</b>	ISTEXT(2nd Quarter) returns True

## LEFT

See also [A-Z Worksheet Function List](#)

**Description** Returns the leftmost characters from the specified text string.

**Syntax** **LEFT**(*text* [, *num\_chars*])

*text* is any text string.

*num\_chars* is 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.

**See Also** [MID](#) and [RIGHT](#) functions

**Examples**  
LEFT(2nd Quarter) returns 2  
LEFT(2nd Quarter, 3) returns 2nd



## LEN

See also [A-Z Worksheet Function List](#)

**Description** Returns the number of characters in the supplied text string.

**Syntax** **LEN**(*text*)

*text* in any text string. Spaces in the string are counted as characters.

**See Also** [EXACT](#) and [SEARCH](#) functions

**Examples**  
LEN(3rd Quarter) returns 11  
LEN(1-3) returns 3

## LN

See also [A-Z Worksheet Function List](#)

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

**Syntax** LN(*number*)

*number* is any positive real number.

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

**See Also** [EXP](#), [LOG](#), and [LOG10](#) functions

**Examples** LN(12.18) returns 2.50

LN(20.09) returns 3.00

## LOG

See also [A-Z Worksheet Function List](#)

**Description** Returns the logarithm of a number to the specified base.

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

*text* is any positive real number.

*base* is the base of the logarithm. Omitting this argument assumes a base of 10.

**See Also** [EXP](#), [LN](#), and [LOG10](#) functions

**Examples**  
LOG(1) returns 0  
LOG(10) returns 1

## LOG10

See also [A-Z Worksheet Function List](#)

**Description** Returns the base-10 logarithm of a number.

**Syntax** **LOG10**(*number*)

*number* is any positive real number.

**See Also** [EXP](#), [LN](#), and [LOG](#) functions

**Examples**  
LOG10 (260) returns 2.41  
LOG10 (100) returns 2

## LOOKUP

See also [A-Z Worksheet Function List](#)

**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*)

*lookup\_value* is the value for which to search in the first range.

*lookup\_range* is 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 (e.g., -2, -1, 0, 1, 2...A through Z, False, True). The search is not case-sensitive.

*result\_range* is 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.

**See Also** [HLOOKUP](#), [INDEX](#), and [VLOOKUP](#) functions

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

**Examples** In the preceding worksheet:

LOOKUP(North, A2:A7, B2:B7) returns Detroit  
LOOKUP(Alabama, A2:A7, B2:B7) returns #N/A

## LOWER

See also [A-Z Worksheet Function List](#)

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

**Syntax** **LOWER**(*text*)

*text* is any string.

**See Also** [PROPER](#) and [UPPER](#) functions

**Examples**  
LOWER(3rd Quarter) returns 3rd quarter  
LOWER(JOHN DOE) returns john doe

## MATCH

See also [A-Z Worksheet Function List](#)

**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)

*lookup\_value* is 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* is the range to search and contains only one row or one column. The range can contain numbers, text, or logical values.

*comparison* is 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 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 (e.g., ...-2, -1, 0, 1, 2..., A through Z, False, True).

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 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 (e.g., 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.

**See Also** [HLOOKUP](#), [INDEX](#), [LOOKUP](#), and [VLOOKUP](#) functions

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

**Examples** In the preceding worksheet:

MATCH(7600, B2:B7,1) returns 5

MATCH("D\*", A2:A7,0) returns 2

## MAX

See also [A-Z Worksheet Function List](#)

**Description** Returns the largest value in the specified list of numbers.

**Syntax** **MAX**(*number\_list*)

*number\_list* is 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.

**See Also** [AVERAGE](#) and [MIN](#) functions

**Examples**  
MAX(50, 100, 150, 500, 200) returns 500  
MAX(A1:F12) returns the largest value in the range



## MID

See also [A-Z Worksheet Function List](#)

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

**Syntax** **MID**(text, start\_position, num\_chars)

*text* is the string from which to return characters.

*start\_position* is 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* is 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.

**See Also** [CODE](#), [FIND](#), [LEFT](#), [RIGHT](#), and [SEARCH](#) functions

**Examples**  
MID(Travel Expenses, 8, 8) returns Expenses  
MID(Part #45-7234, 7, 2) returns 45

## MIN

See also [A-Z Worksheet Function List](#)

**Description** Returns the smallest value in the specified list of numbers.

**Syntax** **MIN**(*number\_list*)

*number\_list* is 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.

**See Also** [AVERAGE](#) and [MAX](#) functions

**Examples**  
MIN(50, 100, 150, 500, 200) returns 50  
MIN(A1:F12) returns the smallest value in the range

## MINUTE

See also [A-Z Worksheet Function List](#)

**Description** Returns the minute that corresponds to the supplied date.

**Syntax** **MINUTE**(*serial\_number*)

*serial\_number* is 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.

**See Also** [DAY](#), [HOUR](#), [MONTH](#), [NOW](#), [SECOND](#), [WEEKDAY](#), and [YEAR](#) functions

**Examples**  
MINUTE (34506.4) returns 36  
MINUTE (34399.825) returns 48

## MIRR

See also [A-Z Worksheet Function List](#)

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

**Syntax** **MIRR**(*cash\_flows*, *finance\_rate*, *reinvest\_rate*)

*cash\_flow* is 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.

During calculation, **MIRR** uses the order in which the values appear to determine the order of cash flow.

Values that represent cash received should be positive; negative values represent cash paid.

Text, logical values, and empty cells in the range are ignored.

*finance\_rate* is the interest rate paid on money used in the cash flow.

*reinvest\_rate* is 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.

**See Also** [IRR](#), [NPV](#), and [RATE](#) functions

	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

**Examples** In the preceding worksheet:

MIRR(B1:B6, 12%, 8%) returns 5.20%

MIRR(B1:B3, 12%, 8%) returns -40.93%

## MOD

See also [A-Z Worksheet Function List](#)

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

**Syntax** **MOD**(*number*, *divisor*)

*number* is any number.

*divisor* is any non-zero number. If *divisor* is 0, #DIV/0! is returned.

**See Also** [INT](#), [ROUND](#), and [TRUNC](#) functions

**Examples**  
MOD(-23, 3) returns 1  
MOD(-23, -3) returns -2

## MONTH

See also [A-Z Worksheet Function List](#)

**Description** Returns the month that corresponds to the supplied date.

**Syntax** **MONTH**(*serial\_number*)

*serial\_number* is the date as a serial number or as text (e.g., 06-21-94 or 21-Jun-94).

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

**See Also** [DAY](#), [HOUR](#), [MINUTE](#), [NOW](#), [SECOND](#), [TODAY](#), [WEEKDAY](#), and [YEAR](#) functions

**Examples**  
MONTH (06-21-94) returns 6  
MONTH (34626) returns 10

## N

See also [A-Z Worksheet Function List](#)

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

**Syntax** **N**(*value*)

*value* is 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.

**See Also** [I](#) and [VALUE](#) functions

**Examples**  
N(32467) returns 32467  
N(A4) returns 1 if A4 contains the logical function True

## NA

See also [A-Z Worksheet Function List](#)

**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](#) function



## NOT

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns a logical value that is the opposite of its value.
<b>Syntax</b>	<b>NOT</b> ( <i>logical</i> )  <i>logical</i> is an expression that returns a logical value (e.g., True or False).
<b>Remarks</b>	If <i>logical</i> is false, <b>NOT</b> returns True. Conversely, if <i>logical</i> is true, <b>NOT</b> returns False.
<b>See Also</b>	<a href="#">AND</a> , <a href="#">IF</a> , and <a href="#">OR</a> functions
<b>Examples</b>	NOT (TRUE ()) returns False NOT (MONTH (12/25/94) = 12) returns False

## NOW

See also [A-Z Worksheet Function List](#)

**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](#), and [YEAR](#) functions

## NPER

See also [A-Z Worksheet Function List](#)

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

**Syntax** **NPER**(*interest*, *pmt*, *pv* [, *fv*] [, *type*])

*interest* is the fixed interest rate.

*pmt* is the fixed payment made each period. Generally, *pmt* includes the principle and interest, not taxes or other fees.

*pv* is the present value, the lump-sum amount that a series of future payments is currently worth.

*fv* is 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.

**See Also** [FV](#) , [IPMT](#) , [PMT](#) , [PPMT](#) , [PV](#) , and [RATE](#) functions

**Examples**  
NPER(12%/12, -350, -300, 16000, 1) returns 36.67  
NPER(1%, -350, -300, 16000) returns 36.98

## NPV

See also [A-Z Worksheet Function List](#)

**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*)

*discount\_rate* is the rate of discount for one period.

*value\_list* is a list of as many as 29 arguments or a reference to a range that contains values that represent payments and income.

flow.

During calculation, **NPV** uses the order in which the values appear to determine the order of cash

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 (e.g., 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*.

**See Also** [FV](#) , [IRR](#), and [PV](#) functions

**Example** NPV(8%, -12000, 3000, 3000, 3000, 7000) returns 811.57

## ODD

See also [A-Z Worksheet Function List](#)

**Description** Rounds the specified number up to the nearest odd integer.

**Syntax** **ODD**(*number*)

*number* is any number, a formula that evaluates to a number, or a reference to a cell that contains a number.

**See Also** [CEILING](#), [EVEN](#), [FLOOR](#), [INT](#), [ROUND](#), and [TRUNC](#) functions

**Examples**  
ODD (3 . 5) returns 5  
ODD (6) returns 7

## OFFSET

See also [A-Z Worksheet Function List](#)

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

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

*reference* is a reference to a cell from which the offset reference is based. If you specify a range reference, #VALUE! is returned.

*rows* is 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* is 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* is a positive number representing the number of rows to include in the offset range. Omitting this argument assumes a single row .

*width* is 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.

**See Also** [COLUMN](#), [INDIRECT](#), and [ROW](#) functions

**Examples**  
OFFSET (B1, 3, 2, 1, 1) returns the contents of cell D4  
SUM(OFFSET (A1, 2, 4, 3, 2)) equals the sum of the range E3:F5

## OR

See also [A-Z Worksheet Function List](#)

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

**Syntax** **OR**(*logical\_list*)

*logical\_list* is 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.

**See Also** [AND](#), [IF](#), and [NOT](#) functions

**Example** `OR(1 + 1 = 1, 5 + 5 = 10)` returns True because one of the arguments is true.

## PI

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the value of pi ( $\pi$ ), which is approximately 3.14159265358979 when calculated to 15 significant digits.
<b>Syntax</b>	<b>PI()</b>
<b>Remarks</b>	Although <b>PI</b> does not use arguments, you must supply the empty parentheses to correctly reference the function.
<b>See Also</b>	<a href="#">COS</a> , <a href="#">SIN</a> , and <a href="#">TAN</a> functions



## PMT

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.
<b>Syntax</b>	<b>PMT</b> ( <i>interest</i> , <i>nper</i> , <i>pv</i> [, <i>fv</i> ] [, <i>type</i> ])  <i>interest</i> is the fixed periodic interest rate. <i>nper</i> is the number of periods in the annuity. <i>pv</i> is the present value, or the amount the annuity is currently worth. <i>fv</i> is the future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed. <i>type</i> 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.
<b>Remarks</b>	<b>PMT</b> 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 <i>nper</i> . For example, if the annuity has an 8% annual interest rate over a period of 5 years, specify 8%/12 for interest and 5*12 for <i>nper</i> .  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.
<b>See Also</b>	<a href="#">FV</a> , <a href="#">IPMT</a> , <a href="#">NPER</a> , <a href="#">PPMT</a> , <a href="#">PV</a> , and <a href="#">RATE</a> functions
<b>Examples</b>	PMT (8%/12, 48, 18000) returns -439.43 PMT (8%/12, 48, 18000, 0, 1) returns -436.52

## PPMT

See also [A-Z Worksheet Function List](#)

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

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

*interest* is the fixed periodic interest rate.

*per* is the period for which to return the principle.

*nper* is the number of periods in the annuity.

*pv* is the present value, or the amount the annuity is currently worth.

*fv* is 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% annual interest rate over a period of 5 years, specify 8%/12 for interest and 5\*12 for *nper*.

**See Also** [FV](#) , [IPMT](#) , [NPER](#) , [PMT](#) , [PV](#) , and [RATE](#) functions

**Examples**  
PPMT(8%/12, 2, 48, 18000) returns -321.56  
PPMT(8%/12, 2, 48, 18000, 0, 1) returns -319.43

## PRODUCT

See also [A-Z Worksheet Function List](#)

**Description** Multiplies a list of numbers and returns the result.

**Syntax** **PRODUCT**(*number\_list*)

*number\_list* is 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.

**See Also** [FACT](#) and [SUM](#) functions

**Example** `PRODUCT(1, 2, 3, 4)` returns 24

## PROPER

See also [A-Z Worksheet Function List](#)

**Description** Returns the specified string in proper-case format.

**Syntax** **PROPER**(*text*)

*text* is 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**.

**See Also** [LOWER](#) and [UPPER](#) functions

**Examples** **PROPER(3rd Quarter)** returns 3Rd Quarter  
**PROPER(JOHN DOE)** returns John Doe

## PV

See also [A-Z Worksheet Function List](#)

**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*])

*interest* is the fixed periodic interest rate.

*nper* is the number of payment periods in the investment.

*pmt* is the fixed payment made each period.

*fv* is 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% annual interest rate over a period of 5 years, specify 8%/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.

**See Also** [FV](#) , [IPMT](#) , [NPER](#) , [PMT](#) , [PPMT](#) , and [RATE](#) functions

**Examples**  
PV(8%/12, 48, 439.43) returns -17999.89  
PV(8%/12, 48, -439.43) returns 17999.89

## RAND

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1.
<b>Syntax</b>	<b>RAND()</b>
<b>Remarks</b>	Although <b>RAND</b> does not use arguments, you must supply the empty parentheses to correctly reference the function.
<b>Example</b>	<code>RAND () *10</code> returns a random number greater than or equal to 0 and less than 10.

## RATE

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the interest rate per period of an annuity, given a series of constant cash payments made over a regular payment period.
<b>Syntax</b>	<b>RATE</b> ( <i>nper</i> , <i>pmt</i> , <i>pv</i> [, <i>fv</i> ] [, <i>type</i> ] [, <i>guess</i> ])  <i>nper</i> is the number of periods in the annuity.  <i>pmt</i> is the fixed payment made each period. Generally, <i>pmt</i> includes only principle and interest, not taxes or other fees.  <i>pv</i> is the present value of the annuity.  <i>fv</i> is the future value, or the amount the annuity will be worth. When you omit this argument, a future value of 0 is assumed.  <i>type</i> 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.  <i>guess</i> is your estimate of the interest rate. If no argument is supplied, a value of .1 (10%) is assumed.
<b>Remarks</b>	<b>RATE</b> 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 <i>guess</i> .
<b>See Also</b>	<a href="#">FV</a> , <a href="#">IPMT</a> , <a href="#">NPER</a> , <a href="#">PMT</a> , <a href="#">PPMT</a> , and <a href="#">PV</a> functions
<b>Example</b>	RATE(48, -439.43, 18000) returns .0067 (rounded to 4 decimals), which is the monthly interest rate. The annual interest rate (.0067 multiplied by 12) is 8%.

## REPLACE

See also [A-Z Worksheet Function List](#)

**Description** Replaces part of a text string with another text string.

**Syntax** **REPLACE**(*orig\_text*, *start\_position*, *num\_chars*, *repl\_text*)

*orig\_text* is the original text string.

*start\_position* is 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* is the number of characters to replace. If this argument is negative, #VALUE! is returned.

*repl\_text* is the replacement text string.

**See Also** [MID](#), [SEARCH](#), and [TRIM](#) functions

**Examples** REPLACE(For the year: 1993, 18, 1, 4) returns For the year: 1994



## REPT

See also [A-Z Worksheet Function List](#)

**Description** Repeats a text string the specified number of times.

**Syntax** **REPT**(*text*, *number*)

*text* is any text string.

*number* is 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** REPT(error-, 3) returns error-error-error-

## RIGHT

See also [A-Z Worksheet Function List](#)

**Description** Returns the rightmost characters from the given text string.

**Syntax** **RIGHT**(*text* [, *num\_chars*])

*text* is any text string.

*num\_chars* is 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.

**See Also** [LEFT](#) and [MID](#) functions

**Examples**  
RIGHT(2nd Quarter) returns r  
RIGHT(2nd Quarter, 7) returns Quarter

## ROUND

See also [A-Z Worksheet Function List](#)

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

**Syntax** **ROUND**(*number*, *precision*)

*number* is any value.

*precision* is 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.

**See Also** [CEILING](#), [FLOOR](#), [INT](#), [MOD](#), and [TRUNC](#) functions

**Examples**  
ROUND (123.456, 2) returns 123.46  
ROUND (9899.435, -2) returns 9900

## ROW

See also [A-Z Worksheet Function List](#)

**Description** Returns the row number of the supplied reference.

**Syntax** **ROW**(*reference*)

*reference* is a cell or range reference. Omitting this argument returns the row number of the cell in which **ROW** is entered.

**See Also** [COLUMN](#) and [ROWS](#) function

**Examples** ROW (B3) returns 3

## ROWS

See also [A-Z Worksheet Function List](#)

**Description** Returns the number of rows in a range reference.

**Syntax** **ROWS**(*range*)

*range* is a reference to a range of cells.

**See Also** [COLUMNS](#) and [ROW](#) functions

**Examples**  
ROWS (A1 : D5) returns 5  
ROWS (C30 : F35) returns 6

## SEARCH

See also [A-Z Worksheet Function List](#)

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

**Syntax** **SEARCH**(*search\_text*, *text* [, *start\_position*])

*search\_text* is the text to find.

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.

To search for an asterisk or question mark, include a tilde (~) before the character.

*text* is the text to be searched.

*start\_position* is 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.

**See Also** [FIND](#), [MID](#), [REPLACE](#), and [SUBSTITUTE](#) functions

**Examples**  
SEARCH(?5, Bin b45) returns 6  
SEARCH(b, Bin b45, 4) returns 5

## SECOND

See also [A-Z Worksheet Function List](#)

**Description** Returns the second that corresponds to the supplied date.

**Syntax** **SECOND**(*serial\_number*)

*serial\_number* is the time as a serial number. The decimal portion of the number represents time as a fraction of the day.

**See Also** [DAY](#), [HOUR](#), [MINUTE](#), [MONTH](#), [NOW](#), [WEEKDAY](#), and [YEAR](#) functions

**Examples**  
SECOND (.259) returns 58  
SECOND (34657.904) returns 46

## SIGN

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Determines the sign of the specified number.
<b>Syntax</b>	<b>SIGN</b> ( <i>number</i> )  <i>number</i> is any number.
<b>Remarks</b>	<b>SIGN</b> returns 1 if the specified number is positive, -1 if it is negative, and 0 if it is 0.
<b>See Also</b>	<a href="#">ABS</a> function
<b>Examples</b>	SIGN(-123) returns -1 SIGN(123) returns 1



## SIN

See also [A-Z Worksheet Function List](#)

**Description** Returns the sine of the supplied angle.

**Syntax** **SIN**(*number*)

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

**See Also** [ASIN](#) and [PI](#) functions

**Examples**  
SIN (45) returns .85  
SIN (90) returns .89

## SINH

See also [A-Z Worksheet Function List](#)

**Description** Returns the hyperbolic sine of the specified number.

**Syntax** **SINH**(*number*)

*number* is any number.

**See Also** [ASINH](#) and [PI](#) functions

**Examples**  
SINH (1) returns 1.18  
SINH (3) returns 10.02

## SLN

See also [A-Z Worksheet Function List](#)

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

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

*cost* is the initial cost of the asset.

*salvage* is the salvage value of the asset.

*life* is the number of periods of the useful life of the asset.

**See Also** [DDB](#), [SYD](#), and [VDB](#) functions

**Example** SLN(10000, 1000, 7) returns 1285.71

## SQRT

See also [A-Z Worksheet Function List](#)

**Description** Returns the square root of the specified number.

**Syntax** **SQRT**(*number*)

*number* is any positive number. If you specify a negative number, #NUM! is returned.

**See Also** [SUMSQ](#) function

**Examples**  
SQRT (9) returns 3  
SQRT (2.5) returns 1.58

## STDEV

See also [A-Z Worksheet Function List](#)

**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*)  
*number\_list* is 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.

**See Also** [STDEVP](#), [VAR](#), and [VARP](#) functions

**Example** STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5) returns .56

## STDEVP

See also [A-Z Worksheet Function List](#)

**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*)

*number\_list* is 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.

**See Also** [STDEV](#), [VAR](#), and [VARP](#) functions

**Example** STDEVP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5) returns .52

## SUBSTITUTE

See also [A-Z Worksheet Function List](#)

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

**Syntax** **SUBSTITUTE**(*text*, *old\_text*, *new\_text* [, *instance*])

*text* is a text string that contains the text to replace. You can also specify a reference to a cell that contains text.

*old\_text* is the text string to be replaced.

*new\_text* is the replacement text.

*instance* specifies the occurrence of *old\_text* to replace. If this argument is omitted, every instance of *old\_text* is replaced.

**See Also** [REPLACE](#) and [TRIM](#) functions

**Examples** `SUBSTITUTE(First Quarter Results, First, Second)` returns **Second Quarter Results**

`SUBSTITUTE(Shipment 45, Bin 45, 45, 52, 2)` returns **Shipment 45, Bin 52**

## SUM

See also [A-Z Worksheet Function List](#)

**Description** Returns the sum of the supplied numbers.

**Syntax** **SUM**(*number\_list*)

*number\_list* is 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.

**See Also** [AVERAGE](#), [COUNT](#), [COUNTA](#), [PRODUCT](#), and [SUMSQ](#) functions

**Examples** SUM(1000, 2000, 3000) returns 6000

SUM(A10:D10) returns 4000 when each cell in the range contains 1000



## SUMSQ

See also [A-Z Worksheet Function List](#)

**Description** Squares each of the supplied numbers and returns the sum of the squares.

**Syntax** **SUMSQ**(*number\_list*)

*number\_list* is 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.

**See Also** [SUM](#) function

**Example** SUMSQ(9, 10, 11) returns 302

## SYD

See also [A-Z Worksheet Function List](#)

**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*)

*cost* is the initial cost of the asset.

*salvage* is the salvage value of the asset.

*life* is the number of periods in the useful life of the asset.

*period* is the period for which to calculate the depreciation. The time units used to determine period and life must match.

**See Also** [DDB](#), [SLN](#), and [VDB](#) functions

**Example** SYD(10000, 1000, 7, 3) returns 1607.14

## T

See also [A-Z Worksheet Function List](#)

**Description** Tests the supplied value and returns the value if it is text.

**Syntax** T(*value*)

*value* is the value to test.

**Remarks** Empty text ("") is returned for any value that is not text.

**See Also** [N](#) , and [VALUE](#) functions

**Examples** T(Report) returns Report

T(A4) returns empty text ("") if A4 contains a number

## TAN

See also [A-Z Worksheet Function List](#)

**Description** Returns the tangent of the specified angle.

**Syntax** **TAN**(*number*)

*number* is the angle in radians. To convert a number expressed as degrees to radians, multiply the degrees by 180/PI().

**See Also** [ATAN](#), [ATAN2](#), [PI](#) , and [TANH](#) functions

**Examples**  
TAN (45) returns 1.62  
TAN (90) returns -2.00

## TANH

See also [A-Z Worksheet Function List](#)

**Description** Returns the hyperbolic tangent of a number.

**Syntax** **TANH**(*number*)

*number* is any number.

**See Also** [\*\*ATANH\*\*](#), [\*\*COSH\*\*](#), [\*\*SINH\*\*](#), and [\*\*TAN\*\*](#) functions

**Examples**  
TANH (-2) returns -.96  
TANH (1.2) returns .83

## TEXT

See also [A-Z Worksheet Function List](#)

**Description** Returns the given number as text, using the specified formatting.

**Syntax** **TEXT**(*number*, *format*)

*number* is any value, a formula that evaluates to a number, or a reference to a cell that contains a value.

*format* is 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.

**See Also** [DOLLAR](#), [FIXED](#), [I](#), and [VALUE](#) functions

**Examples**  
TEXT(123.62, 0.000) returns 123.620  
TEXT(34626.2, MM/DD/YY) returns 10/19/94

## TIME

See also [A-Z Worksheet Function List](#)

**Description** Returns a serial number for the supplied time.

**Syntax** **TIME**(*hour, minute, second*)

*hour* is a number from 0 to 23.

*minute* is a number from 0 to 59.

*second* is a number from 0 to 59.

**See Also** [HOUR](#), [MINUTE](#), [NOW](#), [SECOND](#), and [TIMEVALUE](#) functions

**Examples**  
TIME(12, 26, 24) returns .52  
TIME(1, 43, 34) returns .07

## TIMEVALUE

See also [A-Z Worksheet Function List](#)

**Description** Returns a serial number for the supplied text representation of time.

**Syntax** **TIMEVALUE**(*text*)

*text* is a time in text format.

**See Also** [HOUR](#), [MINUTE](#), [NOW](#), [SECOND](#), and [TIME](#) functions

**Examples**  
TIMEVALUE(1:43:43 am) returns .07  
TIMEVALUE(14:10:07) returns .59



## TODAY

See also [A-Z Worksheet Function List](#)

<b>Description</b>	Returns the current date as a serial number.
<b>Syntax</b>	<b>TODAY()</b>
<b>Remarks</b>	This function is updated only when the worksheet is recalculated.
<b>See Also</b>	<a href="#">DATE</a> , <a href="#">DAY</a> , and <a href="#">NOW</a> functions

## TRIM

See also [A-Z Worksheet Function List](#)

**Description** Removes all spaces from text except single spaces between words.

**Syntax** **TRIM**(*text*)

*text* is 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.

**See Also** [CLEAN](#), [MID](#), [REPLACE](#), and [SUBSTITUTE](#) functions

**Example** TRIM( Level 3, Gate 45 ) returns Level 3, Gate 45

## TRUE

See also [A-Z Worksheet Function List](#)

**Description** Returns the logical value True. This function always requires the trailing parentheses.

**Syntax** TRUE()

**See Also** [FALSE](#) function

## TRUNC

See also [A-Z Worksheet Function List](#)

**Description** Truncates the given number to an integer.

**Syntax** **TRUNC**(*number* [, *precision*])

*number* is any value.

*precision* is 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.

**See Also** [CEILING](#), [FLOOR](#), [INT](#), [MOD](#), and [ROUND](#) functions

**Examples**  
TRUNC (123.456, 2) returns 123.45  
TRUNC (9899.435, -2) returns 9800

## TYPE

See also [A-Z Worksheet Function List](#)

**Description** Returns the argument type of the given expression.

**Syntax** **TYPE**(*expression*)

*expression* is any expression.

**Remarks** The following table lists the *expression* types and numbers.

<b>Expression type</b>	<b>Number</b>
Number	1
Text string	2
Logical value	4
Error value	16

**See Also** [ISBLANK](#), [ISERR](#), [ISERROR](#), [ISLOGICAL](#), [ISNA](#), [ISNONTEXT](#), [ISNUMBER](#), [ISREF](#), and [ISTEXT](#) functions

**Examples**  
TYPE (A1) returns 1 if cell A1 contains a number.  
TYPE (Customer) returns 2

## UPPER

See also [A-Z Worksheet Function List](#)

**Description** Changes the characters in the specified string to uppercase characters.

**Syntax** **UPPER**(*text*)

*text* is any string.

**Remarks** Numeric characters in the string are not changed.

**See Also** [LOWER](#) and [PROPER](#) functions

**Examples**  
UPPER(3rd Quarter) returns 3RD QUARTER  
UPPER(JOHN DOE) returns JOHN DOE

## VALUE

See also [A-Z Worksheet Function List](#)

**Description** Returns the specified text as a number.

**Syntax** **VALUE**(*text*)

*text* is 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 (e.g., M/DD/YY for dates or H:MM AM/PM for time). If the format is not recognized, #VALUE! is returned.

**See Also** [DOLLAR](#), [FIXED](#), and [TEXT](#) functions

**Examples**  
VALUE (9800) returns 9800  
VALUE (123) returns 123

## VAR

See also [A-Z Worksheet Function List](#)

**Description** Returns the variance of a population based on a sample of values.

**Syntax** **VAR**(*number\_list*)

*number\_list* is 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.

**See Also** [STDEV](#), [STDEVP](#), and [VARP](#) functions

**Example** VAR(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5) returns .31



## **VARP**

See also [A-Z Worksheet Function List](#)

**Description** Returns the variance of a population based on an entire population of values.

**Syntax** **VARP**(*number\_list*)

*number\_list* is 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.

**See Also** [STDEV](#), [STDEVP](#), and [VAR](#) functions

**Example** VARP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5) returns .27

## VDB

See also [A-Z Worksheet Function List](#)

**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*])

*cost* is the initial cost of the asset.

*salvage* is the salvage value of the asset.

*life* is the number of periods in the useful life of the asset.

*start\_period* is the beginning period for which to calculate the depreciation. The time units used to determine *start\_period* and *life* must match.

*end\_period* is the ending period for which to calculate the depreciation. The time units used to determine *end\_period* and *life* must match.

*factor* is the rate at which the balance declines. Omitting this argument assumes a default of 2, which is the double-declining balance factor.

*method* is 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.

**See Also** [DDB](#), [SLN](#), and [SYD](#) functions

**Examples** VDB(10000, 1000, 7, 3, 4) returns 1041.23

## VLOOKUP

See also [A-Z Worksheet Function List](#)

**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*)

*search\_item* is 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* is 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 (e.g., -2, -1, 0, 2...A through Z, False, True). Text searches are not case-sensitive.

*column\_index* is 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.

**See Also** [HLOOKUP](#), [INDEX](#), [LOOKUP](#), and [MATCH](#) functions

	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

**Examples** In the preceding worksheet:

VLOOKUP("Clark", A2:E9, 4) returns \$28,700

VLOOKUP("Lee", A2:E9, 3) returns 3961

## WEEKDAY

See also [A-Z Worksheet Function List](#)

**Description** Returns the day of the week that corresponds to the supplied date.

**Syntax** **WEEKDAY**(*serial\_number*)

*serial\_number* is the date as a serial number or as text (e.g., 06-21-94 or 21-Jun-94).

**Remarks** **WEEKDAY** returns a number ranging from 1 (Sunday) to 7 (Saturday).

**See Also** [DAY](#), [NOW](#), [TEXT](#), and [TODAY](#) functions

**Examples**  
WEEKDAY (34399.92) returns 1, indicating Sunday  
WEEKDAY (06/21/94) returns 3, indicating Tuesday

## YEAR

See also [A-Z Worksheet Function List](#)

**Description** Returns the year that corresponds to the supplied date.

**Syntax** **YEAR**(*serial\_number*)

*serial\_number* is the date as a serial number or as text (e.g., 06-21-94 or 21-Jun-94).

**See Also** [DAY](#), [HOUR](#), [MINUTE](#), [MONTH](#), [NOW](#), [SECOND](#), [TODAY](#), and [WEEKDAY](#) functions

**Examples**  
YEAR(34328) returns 1993  
YEAR(06/21/94) returns 1994

## A-Z Function Call Reference

See also the function call summaries

[Dialog Box Function Call Summary](#)

[Edit Bar Function Call Summary](#)

[Formatting Function Call Summary](#)

[Data Entry Function Call Summary](#)

[Printing Function Call Summary](#)

[Range Editing Function Call Summary](#)

[Recalculation Function Call Summary](#)

[Selection Function Call Summary](#)

[Worksheet Function Call Summary](#)

[Miscellaneous Function Call Summary](#)

This chapter provides a complete alphabetical reference for the Formula One function calls. Refer to [Using Function Calls](#) for additional information about using function calls.

[SSAddColPageBreak](#)

[SSAddPageBreak](#)

[SSAddRowPageBreak](#)

[SSAddSelection](#)

[SSAttach](#)

[SSAttachToSS](#)

[SSCalculationDlg](#)

[SSCallWindowProc](#)

[SSCancelEdit](#)

[SSCanEditPaste](#)

[SSCheckModified](#)

[SSCheckRecalc](#)

[SSClearClipboard](#)

[SSClearRange](#)

[SSColorPaletteDlg](#)

[SSColWidthDlg](#)

[SSCopyAll](#)

[SSCopyRange](#)

[SSDefinedNameDlg](#)

[SSDelete](#)

[SSDeleteDefinedName](#)

[SSDeleteRange](#)

[SSDeleteTable](#)

[SSEditBarDelete](#)

[SSEditBarHeight](#)

[SSEditBarMove](#)

[SSEditBarNew](#)

[SSEditClear](#)

[SSEditCopy](#)

[SSEditCopyDown](#)

[SSEditCopyRight](#)

[SSEditCut](#)

[SSEditDelete](#)

[SSEditInsert](#)

[SSEditPaste](#)

[SSEndEdit](#)

[SSErrorNumberToText](#)

[SSFilePageSetupDlg](#)

[SSFilePrint](#)

[SSFilePrintSetupDlg](#)

[SSFormatAlignmentDlg](#)

[SSFormatBorderDlg](#)

[SSFormatCurrency0](#)

[SSFormatCurrency2](#)

[SSFormatFixed](#)

[SSFormatFixed2](#)

[SSFormatFontDlg](#)

[SSFormatFraction](#)

[SSFormatGeneral](#)

[SSFormatHmmpm](#)

[SSFormatMdy](#)

[SSFormatNumberDlg](#)

[SSGetLastCol](#)

[SSGetLastColForRow](#)

[SSGetLastRow](#)

[SSGetLeftCol](#)

[SSGetLogicalRC](#)

[SSGetMaxCol](#)

[SSGetMaxRow](#)

[SSGetMinCol](#)

[SSGetMinRow](#)

[SSGetNumber](#)

[SSGetNumberRC](#)

[SSGetPrintArea](#)

[SSGetPrintBottomMargin](#)

[SSGetPrintColHeading](#)

[SSGetPrintFooter](#)

[SSGetPrintGridLines](#)

[SSGetPrintHCenter](#)

[SSGetPrintHeader](#)

[SSGetPrintLeftMargin](#)

[SSGetPrintLeftToRight](#)

[SSGetPrintNoColor](#)

[SSGetPrintRightMargin](#)

[SSGetPrintRowHeading](#)

[SSGetPrintTitles](#)

[SSGetPrintTopMargin](#)

[SSGetPrintVCenter](#)

[SSGetRepaint](#)

[SSGetRowHeight](#)

[SSGetRowMode](#)

[SSGetSelection](#)

[SSGetSelectionCount](#)

[SSGetSelectionRef](#)

[SSGetShowColHeading](#)

[SSGetShowFormulas](#)

[SSGetShowGridLines](#)

[SSGetShowHScrollBar](#)

[SSGetShowRowHeading](#)

[SSGetShowSelections](#)

[SSGetShowVScrollBar](#)

[SSGetShowZeroValues](#)

[SSGetSSEdit](#)

[SSGetText](#)

[SSGetTextRC](#)

[SSGetTitle](#)

[SSGetTopRow](#)

[SSGetTypeRC](#)

[SSGotoDlg](#)

[SSInitTable](#)

[SSInsertRange](#)

[SSMaxCol](#)

[SSMaxRow](#)

[SSMoveRange](#)

[SSSetColWidthAuto](#)

[SSSetDefinedName](#)

[SSSetDefWindowProc](#)

[SSSetDoSetCursor](#)

[SSSetEnableProtection](#)

[SSSetEnterMovesDown](#)

[SSSetEntry](#)

[SSSetEntryRC](#)

[SSSetExtraColor](#)

[SSSetFireEvent](#)

[SSSetFixedCols](#)

[SSSetFixedRows](#)

[SSSetFont](#)

[SSSetFormula](#)

[SSSetFormulaRC](#)

[SSSetHdrHeight](#)

[SSSetHdrSelection](#)

[SSSetHdrWidth](#)

[SSSetIteration](#)

[SSSetLeftCol](#)

[SSSetLogicalRC](#)

[SSSetMaxCol](#)

[SSSetMaxRow](#)

[SSSetMinCol](#)

[SSSetMinRow](#)

[SSSetNumber](#)

[SSSetNumberFormat](#)

[SSSetNumberRC](#)

[SSSetPattern](#)

[SSSetPrintArea](#)

[SSSetPrintAreaFromSelection](#)

[SSSetPrintBottomMargin](#)

[SSSetPrintColHeading](#)

[SSSetPrintFooter](#)

[SSSetPrintGridLines](#)

[SSSetPrintHCenter](#)

[SSSetPrintHeader](#)

[SSSetPrintLeftMargin](#)

[SSSetPrintLeftToRight](#)

[SSSetPrintNoColor](#)

[SSSetPrintRightMargin](#)

[SSSetPrintRowHeading](#)

[SSSetPrintTitles](#)

[SSSetPrintTitlesFromSelection](#)

[SSSetPrintTopMargin](#)

[SSSetPrintVCenter](#)

[SSSetProtection](#)

[SSSetRepaint](#)

[SSSetRowHeight](#)

[SSSetRowHeightAuto](#)

[SSSetRowMode](#)

[SSSetRowText](#)

[SSFormatPatternDlg](#)  
[SSFormatPercent](#)  
[SSFormatRCNr](#)  
[SSFormatScientific](#)  
[SSGetActiveCell](#)  
[SSGetAllowArrows](#)  
[SSGetAllowDelete](#)  
[SSGetAllowEditHeaders](#)  
[SSGetAllowFillRange](#)  
[SSGetAllowFormulas](#)  
[SSGetAllowInCellEditing](#)  
[SSGetAllowMoveRange](#)  
[SSGetAllowResize](#)  
[SSGetAllowSelections](#)  
[SSGetAllowTabs](#)  
[SSGetAutoRecalc](#)  
[SSGetBackColor](#)  
[SSGetColWidth](#)  
[SSGetDefinedName](#)  
[SSGetEnableProtection](#)  
[SSGetEnterMovesDown](#)  
[SSGetEntry](#)  
[SSGetEntryRC](#)  
[SSGetExtraColor](#)  
[SSGetFireEvent](#)  
[SSGetFixedCols](#)  
[SSGetFixedRows](#)  
[SSGetFormattedText](#)  
[SSGetFormattedTextRC](#)  
[SSGetFormula](#)  
[SSGetFormulaRC](#)  
[SSGetHdrSelection](#)  
[SSGetIteration](#)

[SSNew](#)  
[SSNextColPageBreak](#)  
[SSNextRowPageBreak](#)  
[SSOpenFileDialog](#)  
[SSProtectionDlg](#)  
[SSRangeToTwips](#)  
[SSRead](#)  
[SSReadIO](#)  
[SSRecalc](#)  
[SSRemoveColPageBreak](#)  
[SSRemovePageBreak](#)  
[SSRemoveRowPageBreak](#)  
[SSRowHeightDlg](#)  
[SSSaveFileDialog](#)  
[SSSaveWindowInfo](#)  
[SSSetActiveCell](#)  
[SSSetAlignment](#)  
[SSSetAllowArrows](#)  
[SSSetAllowDelete](#)  
[SSSetAllowEditHeaders](#)  
[SSSetAllowFillRange](#)  
[SSSetAllowFormulas](#)  
[SSSetAllowInCellEditing](#)  
[SSSetAllowMoveRange](#)  
[SSSetAllowResize](#)  
[SSSetAllowSelections](#)  
[SSSetAllowTabs](#)  
[SSSetAppName](#)  
[SSSetAutoRecalc](#)  
[SSSetBackColor](#)  
[SSSetBorder](#)  
[SSSetColText](#)  
[SSSetColWidth](#)

[SSSetSelection](#)  
[SSSetSelectionRef](#)  
[SSSetShowColHeading](#)  
[SSSetShowFormulas](#)  
[SSSetShowGridLines](#)  
[SSSetShowHScrollBar](#)  
[SSSetShowRowHeading](#)  
[SSSetShowSelections](#)  
[SSSetShowVScrollBar](#)  
[SSSetShowZeroValues](#)  
[SSSetSSEdit](#)  
[SSSetText](#)  
[SSSetTextRC](#)  
[SSSetTitle](#)  
[SSSetTopLeftText](#)  
[SSSetTopRow](#)  
[SSShowActiveCell](#)  
[SSSort](#)  
[SSSort3](#)  
[SSSortDlg](#)  
[SSStartEdit](#)  
[SSSwapTables](#)  
[SSTransactCommit](#)  
[SSTransactRollback](#)  
[SSTransactStart](#)  
[SSTwipsToRC](#)  
[SSUpdate](#)  
[SSVBXCopyCellsFromDoubleArray](#)  
[SSVBXCopyCellsToDoubleArray](#)  
[SSVersion](#)  
[SSWrite](#)  
[SSWriteIO](#)

## SSAddColPageBreak

See also [A-Z Function Call List](#)

**Description** Adds a vertical page break adjacent to the left edge of the specified column.

**Syntax (VB)** **SSAddColPageBreak**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nCol*%)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSAddColPageBreak** (HSS *hSS*, RC *nCol*)

*hSS* is a handle to a view.

*nCol* is the column where the page break is added.

**Return Value** Integer

**See Also** [SSAddPageBreak](#), [SSAddRowPageBreak](#), [SSNextColPageBreak](#), [SSNextRowPageBreak](#), [SSRemoveColPageBreak](#), [SSRemovePageBreak](#), and [SSRemoveRowPageBreak](#) functions

**Example** `sserror = SSAddColPageBreak(Sheet1.SS, 2)`



## SSAddPageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Adds a horizontal and vertical page break adjacent to the active cell.
<b>Syntax (VB)</b>	<b>SSAddPageBreak</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSAddPageBreak</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	When page breaks are added adjacent to the active cell, the horizontal page break is added adjacent to the cells top edge; the vertical page break is added adjacent to the cells left edge.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddRowPageBreak</a> , <a href="#">SSNextColPageBreak</a> , <a href="#">SSNextRowPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , <a href="#">SSRemovePageBreak</a> , and <a href="#">SSRemoveRowPageBreak</a> functions
<b>Example</b>	<pre>sserror = SSAddPageBreak(sheet1.SS)</pre>

## SSAddRowPageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Adds a horizontal page break adjacent to the top edge of the specified row.
<b>Syntax (VB)</b>	<b>SSAddRowPageBreak</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSAddRowPageBreak</b> (HSS <i>hSS</i> , RC <i>nRow</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> is the row where the page break is added.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddPageBreak</a> , <a href="#">SSNextColPageBreak</a> , <a href="#">SSNextRowPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , <a href="#">SSRemovePageBreak</a> , and <a href="#">SSRemoveRowPageBreak</a> functions
<b>Example</b>	<pre>sserror = SSAddRowPageBreak(sheet1.SS, 2)</pre>

## SSAddSelection

See also [A-Z Function Call List](#)

<b>Description</b>	Adds a new selection to the current selection list.
<b>Syntax (VB)</b>	<b>SSAddSelection</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSAddSelection</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nRow2</i> , <i>nCol1</i> and <i>nCol2</i> are the row and column numbers of the selection to add to the selection list. If <i>nRow1</i> is -1, all rows are included in the selection; if <i>nCol1</i> is -1, all columns are included.
<b>Remarks</b>	Multiple selections allow operations such as formatting or clearing to be performed on non-contiguous areas.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetSelection</a> function and <a href="#">Selection</a> property
<b>Example</b>	<pre>sserror = SSAddSelection(sheet1.SS, 1, 1, 4, 4)</pre>

## SSAttach

See also [A-Z Function Call List](#)

**Description** Searches for a worksheet with the given title and attaches it to a view.

**Syntax (VB)** **SSAttach**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pTitle*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSAttach (HSS FAR *hSS*, LPCSTR *pTitle*)

*hSS* is a handle to a view.

*pTitle* is a string containing the name of the worksheet for which to search.

**Remarks** **SSAttach** searches for a worksheet with the given title. If a worksheet is found, it is attached to the specified view. If there was already a worksheet attached to the view, it is deleted as the specified worksheet is attached.

To delete the worksheet when it is no longer needed, call **SSDelete**.

**Return Value** Integer

**See Also** [SSAttachToSS](#), [SSDelete](#), [SSGetTitle](#), [SSNew](#), and [SSSetTitle](#) functions

**Example** `sserror = SSAttach (Sheet1.SS, "Sheet2")`

## SSAttachToSS

See also [A-Z Function Call List](#)

<b>Description</b>	Attaches a worksheet from one view to another.
<b>Syntax (VB)</b>	<b>SSAttachToSS</b> % Lib "VTSSDLL.DLL" (ByVal <i>hDstSS</i> &, ByVal <i>hSrcSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSAttachToSS</b> (HSS <i>hDstSS</i> , HSS <i>hSrcSS</i> )  <i>hDstSS</i> is a handle to the destination view. <i>hSrcSS</i> is a handle to the source view.
<b>Remarks</b>	<b>SSAttachToSS</b> attaches the worksheet of the source view to the destination view. If there was already a worksheet attached to the destination view, it is deleted as the source view is attached. After calling <b>SSAttachToSS</b> , both <i>hSrcSS</i> and <i>hDstSS</i> display the same worksheet.  To delete the worksheet when it is no longer needed, call <b>SSDelete</b> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> , <a href="#">SSDelete</a> , <a href="#">SSGetTitle</a> , <a href="#">SSNew</a> , and <a href="#">SSSetTitle</a> functions
<b>Example</b>	<pre>sserror = SSAttachToSS (Sheet1.SS, Sheet2.SS)</pre>

## SSCalculationDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Calculation dialog box.
<b>Syntax (VB)</b>	<b>SSCalculationDlg%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSCalculationDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSCalculationDlg</b> displays the Calculation dialog box. This dialog box allows you to enable and disable automatic recalculation and specify iteration values for calculating circular references.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAutoRecalc</a> , <a href="#">SSGetIteration</a> , <a href="#">SSSetAutoRecalc</a> , and <a href="#">SSSetIteration</a> functions and <a href="#">AutoRecalc</a> property
<b>Example</b>	<pre>sserror = SSCalculationDlg(Sheet1.SS)</pre>

## SSCallWindowProc

See also [A-Z Function Call List](#)

<b>Description</b>	Passes Windows messages to the view.
<b>Syntax (VB)</b>	<b>SSCallWindowProc%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>message%</i> , ByVal <i>wParam%</i> , ByVal <i>lParam</i> &)
<b>Syntax (VC++)</b>	LRESULT SSEXPORTAPI <b>SSCallWindowProc</b> (HSS <i>hSS</i> , UINT <i>message</i> , WPARAM <i>wParam</i> , LPARAM <i>lParam</i> )  <i>hSS</i> is a handle to a view.  <i>message</i> is the message being passed.  <i>wParam</i> and <i>lParam</i> are standard Windows procedure parameters passed to all Windows procedures.
<b>Remarks</b>	Window messages are passed to the worksheet view by calling <b>SSCallWindowProc</b> . All window messages should be passed to <b>SSCallWindowProc</b> for the worksheet view to work properly.  Messages not used by the worksheet view are passed to the function provided via <b>SSSetDefWindowProc</b> . If a valid callback procedure is not specified by <b>SSSetDefWindowProc</b> , the Windows API function DefWindowProc is called.  This function is not normally called from Visual Basic.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetDefWindowProc</a> function

## SSCancelEdit

See also [A-Z Function Call List](#)

<b>Description</b>	Cancels edit mode and leaves the contents of the active cell unchanged.
<b>Syntax (VB)</b>	<b>SSCancelEdit</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSCancelEdit</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSCancelEdit</b> aborts cell editing and exits edit mode without altering the contents of the active cell.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEndEdit</a> and <a href="#">SSStartEdit</a> functions
<b>Example</b>	<code>sserror = SSCancelEdit(Sheet1.SS)</code>



## SSCanEditPaste

See also [A-Z Function Call List](#)

<b>Description</b>	Determines if the internal clipboard or Windows clipboard contain items that can be pasted to the worksheet.
<b>Syntax (VB)</b>	<b>SSCanEditPaste</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pCanEditPaste</i> %)
<b>Syntax (VC++)</b>	BOOL SSEXPORTAPI <b>SSCanEditPaste</b> (HSS <i>hSS</i> , BOOL FAR * <i>pCanEditPaste</i> )  <i>hSS</i> is a handle to a view.  <i>pCanEditPaste</i> is a reference to a boolean that indicates if anything is in the clipboard.
<b>Remarks</b>	<b>SSCanEditPaste</b> returns True in the boolean referred to by <i>pCanEditPaste</i> if there is something in the internal clipboard or the Windows clipboard that can be pasted to the worksheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditCopy</a> , <a href="#">SSEditCut</a> , and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<pre>sserror = SSCanEditPaste(Sheet1.SS, canpaste)</pre>

## SSCheckModified

See also [A-Z Function Call List](#)

<b>Description</b>	Checks to see if the view or worksheet has been modified since the last SSM_MODIFIED message was sent.
<b>Syntax (VB)</b>	<b>SSCheckModified%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSCheckModified</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSCheckModified</b> checks to see if the view or its worksheet has been modified since the last SSM_MODIFIED message was sent. If so, then another SSM_MODIFIED message is sent, causing the <b>DataChanged</b> property to be updated.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">DataChanged</a> property
<b>Example</b>	<pre>sserror = SSCheckModified(Sheet1.SS)</pre>

## SSCheckRecalc

See also [A-Z Function Call List](#)

<b>Description</b>	Recalculates the worksheet if needed.
<b>Syntax (VB)</b>	<b>SSCheckRecalc</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSCheckRecalc</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSCheckRecalc</b> determines if the worksheet needs to be recalculated as a result of a change. If so, the worksheet is recalculated.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAutoRecalc</a> and <a href="#">SSSetAutoRecalc</a> functions and <a href="#">AutoRecalc</a> property
<b>Example</b>	<pre>sserror = SSCheckRecalc(Sheet1.SS)</pre>

## SSClearClipboard

See also [A-Z Function Call List](#)

<b>Description</b>	Clears the internal clipboard.
<b>Syntax (VB)</b>	<b>SSClearClipboard</b> % Lib "VTSSDLL.DLL" ()
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSClearClipboard</b> ();
<b>Remarks</b>	<b>SSClearClipboard</b> clears the contents of the internal clipboard and releases all resources associated with it.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCanEditPaste</a> and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<pre>sserror = SSClearClipboard()</pre>

## SSClearRange

See also [A-Z Function Call List](#)

**Description** Clears the specified range.

**Syntax (VB)** **SSClearRange**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow1*%, ByVal *nCol1*%, ByVal *nRow2*%, ByVal *nCol2*%, ByVal *nClearType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSClearRange (HSS *hSS*, RC *nRow1*, RC *nCol1*, RC *nRow2*, RC *nCol2*, int *nClearType*)

*hSS* is a handle to a view.

*nRow1*, *nCol1*, *nRow2*, and *nCol2* 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* determines what is cleared from the specified range. The following table lists the options for this argument.

Option	Description
0	Displays the Clear dialog box. This dialog box allows the user to specify whether only formats, only values, or both formats and values are cleared.
1	All (values and formats)
2	Formats only
3	Values only (including formulas)

**Return Value** Integer

**See Also** [SSDeleteRange](#) function

**Example** `sserror = SSClearRange(Sheet1.SS, 1, 1, 10, 10, 0)`

## SSColorPaletteDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Color Palette dialog box.
<b>Syntax (VB)</b>	<b>SSColorPaletteDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSColorPaletteDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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 worksheet specific.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetBackColor</a> , <a href="#">SSSetBorder</a> , <a href="#">SSSetExtraColor</a> , <a href="#">SSSetFont</a> , and <a href="#">SSSetPattern</a> functions
<b>Example</b>	<pre>sserror = SSColorPaletteDlg(Sheet1.SS)</pre>

## SSColWidthDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Column Width dialog box.
<b>Syntax (VB)</b>	<b>SSColWidthDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSColWidthDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Column Width 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetColWidth</a> and <a href="#">SSSetColWidthAuto</a> functions
<b>Example</b>	<pre>sserror = SScolWidthDlg(Sheet1.SS)</pre>

## SSCopyAll

See also [A-Z Function Call List](#)

<b>Description</b>	Copies the contents of one worksheet to another worksheet.
<b>Syntax (VB)</b>	<b>SSCopyAll</b> % Lib "VTSSDLL.DLL" (ByVal <i>hDstSS</i> &, ByVal <i>hSrcSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SScopyAll (HSS <i>hDstSS</i> , HSS <i>hSrcSS</i> )  <i>hDstSS</i> is a handle to the destination view. <i>hSrcSS</i> is a handle to the source view.
<b>Remarks</b>	<b>SSCopyAll</b> copies an entire worksheet from <i>hSrcSS</i> view to <i>hDstSS</i> view.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCopyRange</a> and <a href="#">SSMoveRange</a> functions
<b>Example</b>	<pre>sserror = SScopyAll(Sheet1.SS, Sheet2.SS)</pre>



## SSCopyRange

See also [A-Z Function Call List](#)

<b>Description</b>	Copies a range within a worksheet or from one worksheet to another.
<b>Syntax (VB)</b>	<b>SSCopyRange</b> % Lib "VTSSDLL.DLL" (ByVal <i>hDstSS</i> &, ByVal <i>nDstR1</i> %, ByVal <i>nDstC1</i> %, ByVal <i>nDstR2</i> %, ByVal <i>nDstC2</i> %, ByVal <i>hSrcSS</i> &, ByVal <i>nSrcR1</i> %, ByVal <i>nSrcC1</i> %, ByVal <i>nSrcR2</i> %, ByVal <i>nSrcC2</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSCopyRange</b> (HSS <i>hDstSS</i> , RC <i>nDstR1</i> , RC <i>nDstC1</i> , RC <i>nDstR2</i> , RC <i>nDstC2</i> , HSS <i>hSrcSS</i> , RC <i>nSrcR1</i> , RC <i>nSrcC1</i> , RC <i>nSrcR2</i> , RC <i>nSrcC2</i> )  <i>hDstSS</i> is a handle to the destination view. <i>nDstR1</i> , <i>nDstC1</i> , <i>nDstR2</i> , and <i>nDstC2</i> define the destination range. <i>hSrcSS</i> is a handle to the source view. <i>nSrcR1</i> , <i>nSrcC1</i> , <i>nSrcR2</i> , and <i>nSrcC2</i> define the source range.
<b>Remarks</b>	<b>SSCopyRange</b> copies the specified range from the <i>hSrcSS</i> view to the <i>hDstSS</i> view. The source and the destination ranges can be in different views, allowing ranges to be copied between worksheets. The copy operation is the same as if a copy and paste operation had occurred. Cell references are adjusted appropriately in the destination range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCopyAll</a> , <a href="#">SSEditCopy</a> , and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<pre>sserror = SScopyRange(Sheet1.SS, 1, 1, 10, 10, Sheet1.SS, 10, 10, 50, 50)</pre>

## SSDefinedNameDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Define Name dialog box.
<b>Syntax (VB)</b>	<b>SSDefinedNameDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSDefinedNameDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Define Name dialog box allows you to add and delete user defined names.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSDeleteDefinedName</a> , <a href="#">SSGetDefinedName</a> , and <a href="#">SSSetDefinedName</a> functions
<b>Example</b>	<pre>sserror = SSDefinedNameDlg(Sheet1.SS)</pre>

## SSDelete

See also [A-Z Function Call List](#)

<b>Description</b>	Deletes a view.
<b>Syntax (VB)</b>	SSDelete% Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bSendDeleteTableMsg</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSDelete</b> (HSS <i>hSS</i> , BOOL <i>bSendDeleteTableMsg</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSDelete</b> deletes the specified view. If no other views are attached to the worksheet, the worksheet is also deleted. If <i>bSendDeleteTableMsg</i> is True, a message is sent to all other views accessing this worksheet. The message tells the views that the worksheet is to be deleted.  <b>Important</b> This function should not be called from Visual Basic.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> , <a href="#">SSAttachToSS</a> , <a href="#">SSNew</a> , and <a href="#">SSInitTable</a> functions

## SSDeleteDefinedName

See also [A-Z Function Call List](#)

**Description** Deletes the specified user-defined name.

**Syntax (VB)** **SSDeleteDefinedName**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pName*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSDeleteDefinedName** (HSS *hSS*, LPCSTR *pName*)

*hSS* is a handle to a view.

*pName* is the user defined name to delete.

**Return Value** Integer

**See Also** [SSDefinedNameDlg](#), [SSGetDefinedName](#), and [SSSetDefinedName](#) functions

**Example** `sserror = SSDeleteDefinedName(Sheet1.SS, "Gross_Sales")`

## SSDeleteRange

See also [A-Z Function Call List](#)

**Description** Deletes cells, rows, or columns from the specified range.

**Syntax (VB)** **SSDeleteRange**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow1*%, ByVal *nCol1*%, ByVal *nRow2*%, ByVal *nCol2*%, ByVal *nShiftType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSDeleteRange** (HSS *hSS*, RC *nRow1*, RC *nCol1*, RC *nRow2*, RC *nCol2*, int *nShiftType*)

*hSS* is a handle to a view.

*nRow1*, *nCol1*, *nRow2*, and *nCol2* 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* determines how the delete should occur.

**Remarks** **SSDeleteRange** deletes cells, rows, or columns from the given range. *nShiftType* specifies how the delete occurs. The following table lists the settings for *nShiftType*. These values are defined in VTSS.H and VTSS.TXT.

Setting	Number	Description
kShiftHorizontal	1	Cells to the right of the range are shifted left to fill the vacated space
kShiftVertical	2	Cells below the range are shifted up to fill the vacated space
kShiftRows	3	Rows in which the range resides are deleted and lower rows are shifted up to fill the vacated space
kShiftCol	4	Columns in which the range resides are deleted and the rightmost columns are shifted left to fill the vacated space

**Return Value** Integer

**See Also** [SSClearRange](#) and [SSEditDelete](#) functions

**Example**  
`sserror = SSDeleteRange(Sheet1.SS, 1, 1, 10, 10, kShiftHorizontal)`

## SSDeleteTable

See also [A-Z Function Call List](#)

<b>Description</b>	Deletes a worksheet.
<b>Syntax (VB)</b>	<b>SSDeleteTable</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSDeleteTable</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSDeleteTable</b> detaches a view from a worksheet and deletes the worksheet if no other views are attached to the worksheet. A view must be reattached to a worksheet before it can be used.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> , <a href="#">SSAttachToSS</a> , <a href="#">SSNew</a> , and <a href="#">SSInitTable</a> functions
<b>Example</b>	<pre>sserror = SSDeleteTable(Sheet1.SS)</pre>

## SSEditBarDelete

See also [A-Z Function Call List](#)

**Description** Deletes the specified edit bar.

**Syntax (VB)** **SSEditBarDelete**% Lib "VTSSDLL.DLL" (ByVal *hSSEdit*&)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSEditBarDelete** (HSSEdit *hSSEdit*)

*hSSEdit* is a handle to an edit bar.

**Important** This function should not be called from Visual Basic.

---

**Return Value** Integer

**See Also** [SSEditBarNew](#) function

## SSEditBarHeight

See also [A-Z Function Call List](#)

**Description** Returns the default height of an edit bar.

**Syntax (VB)** `SSEditBarHeight% Lib "VTSSDLL.DLL" ()`

**Syntax (VC++)** `int SSEXPORTAPI SSEditBarHeight ()`

**Important** This function should not be called from Visual Basic.

---

**Return Value** Integer

**See Also** [SSEditBarNew](#) function



## SSEditBarMove

See also [A-Z Function Call List](#)

<b>Description</b>	Moves an edit bar.
<b>Syntax (VB)</b>	<b>SSEditBarMove</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSSEdit</i> &, ByVal <i>x</i> %, ByVal <i>y</i> %, ByVal <i>cx</i> %, ByVal <i>cy</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSEditBarMove</b> (HSSEdit <i>hSSEdit</i> , int <i>x</i> , int <i>y</i> , int <i>cx</i> , int <i>cy</i> )  <i>hSSEdit</i> is a handle to an edit bar. <i>x</i> and <i>y</i> are the coordinates of the upper left corner of the edit bar. <i>cx</i> is the width of the edit bar. <i>cy</i> is the height of the edit bar. <b>Important</b> This function should not be called from Visual Basic.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditBarNew</a> function

---

## SSEditBarNew

See also [A-Z Function Call List](#)

**Description** Creates a new edit bar.

**Syntax (VB)** **SSEditBarNew%** Lib "VTSSDLL.DLL" (ByVal *hWndParent%*, *hSSEdit&*)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSEditBarNew** (HWND *hWndParent*, HSSEdit FAR *\*hSSEdit*)

*hWndParent* is a handle to a parent window.

*hSSEdit* is the handle for the new edit bar.

**Remarks** **SSEditBarNew** creates a new edit bar. *hWndParent* is the parent window of the edit bar.

**Important** This function should not be called from Visual Basic.

---

**Return Value** Integer

**See Also** [SSEditBarDelete](#), [SSEditBarMove](#), and [SSEditBarHeight](#) functions

## SSEditClear

See also [A-Z Function Call List](#)

**Description** Clears all cells in the selected ranges.

**Syntax (VB)** **SSEditClear**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nClearType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSEditClear** (HSS *hSS*, int *nClearType*)

*hSS* is a handle to a view.

*nClearType* determines what is cleared from the selected range. The following table lists the options for this argument.

Option	Description
0	Displays the Clear dialog box. This dialog box allows the user to specify whether only formats, only values, or both formats and values are cleared.
1	All (values and formats)
2	Formats only
3	Values only (including formulas)

**Remarks** **SSEditClear** clears cells in all selected ranges. Non-contiguous ranges can be cleared simultaneously if **SSAddSelection** was used to select multiple ranges.

**Return Value** Integer

**See Also** [SSClearRange](#) and [SSEditDelete](#) functions

**Example** `sserror = SSEditClear(Sheet1.SS, 1)`

## SSEditCopy

See also [A-Z Function Call List](#)

<b>Description</b>	Copies the selected range to the clipboard.
<b>Syntax (VB)</b>	<b>SSEditCopy</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSEditCopy</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSEditCopy</b> copies the selected range to the clipboard. Only one range can be selected. If more than one range is selected, the error SSERROR_ONLY_ONE_RANGE is returned.  Cells copied to the internal clipboard retain their formulas, formatting, and data. When cells are copied to the Windows clipboard and the internal clipboard is deleted, the cells become formatted text representations of the data they contain. This usually happens when the application is exited.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditCut</a> and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<pre>sserror = SSEditCopy(Sheet1.SS)</pre>

## SSEditCopyDown

See also [A-Z Function Call List](#)

**Description** Copies cells in the top row of a selection to the other rows in the selected range.

**Syntax (VB)** **SSEditCopyDown**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSEditCopyDown (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** **SSEditCopyDown** copies data in the top row of a selection to the other rows in the selection and adjusts relative cell references appropriately.

**Return Value** Integer

**See Also** [SSEditCopyRight](#) function

**Example** `sserror = SSEditCopyDown(Sheet1.SS)`

## SSEditCopyRight

See also [A-Z Function Call List](#)

<b>Description</b>	Copies cells in the left column of a selection to the other columns in the selected range.
<b>Syntax (VB)</b>	<b>SSEditCopyRight%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSEditCopyRight</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSEditCopyRight</b> copies data in the left column of a selection to the other columns in the selection and adjusts relative cell references appropriately.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditCopyDown</a> function
<b>Example</b>	<code>sserror = SSEditCopyRight(Sheet1.SS)</code>

## SSEditCut

See also [A-Z Function Call List](#)

<b>Description</b>	Cuts the selected range to the clipboard.
<b>Syntax (VB)</b>	<b>SSEditCut</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSEditCut</b> ( <i>HSS hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<p><b>SSEditCut</b> cuts the selected range to the internal clipboard and clears it from the worksheet. Only one range can be selected. If more than one range is selected, the error SSERROR_ONLY_ONE_RANGE is returned.</p> <p>Cells cut to the internal clipboard retain their formulas, formatting, and data. When cells are copied to the Windows clipboard and the internal clipboard is deleted, the cells become formatted text representations of the data they contain. This usually happens when the application is exited.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditCopy</a> and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<pre>sserror = SSEditCut(Sheet1.SS)</pre>

## SSEditDelete

See also [A-Z Function Call List](#)

**Description** Deletes cells, rows, or columns from the selected range.

**Syntax (VB)** **SSEditDelete**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nShiftType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSEditDelete** (HSS *hSS*, int *nShiftType*)

*hSS* is a handle to a view.

*nShiftType* determines how the delete should occur.

**Remarks** **SSEditDelete** deletes cells, rows, or columns from the selected range. The following table lists the settings for *nShiftType*. These values are defined in VTSS.H and VTSS.TXT.

Setting	Number	Description
kShiftHorizontal	1	Cells to the right of the range are shifted left to fill the vacated space
kShiftVertical	2	Cells below the range are shifted up to fill the vacated space
kShiftRows	3	Rows in which the range resides are deleted and lower rows are shifted up to fill the vacated space
kShiftCols	4	Columns in which the range resides are deleted and the rightmost columns are shifted left to fill the vacated space

**Return Value** Integer

**See Also** [SSDeleteRange](#) and [SSEditInsert](#) functions

**Example** `sserror = SSEditDelete(Sheet1.SS, kShiftHorizontal)`



## SSEditInsert

See also [A-Z Function Call List](#)

**Description** Insert cells, rows, or columns in the selected range.

**Syntax (VB)** **SSEditInsert**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nShiftType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSEditInsert** (HSS *hSS*, int *nShiftType*)

*hSS* is a handle to a view.

*nShiftType* determines how the insert should occur.

**Remarks** **SSEditInsert** inserts cells, rows, or columns in the selected range. The following table lists the settings for *nShiftType*. These values are defined in VTSS.H and VTSS.TXT.

<b>Setting</b>	<b>Number</b>	<b>Description</b>
kShiftHorizontal	1	Cells of the selected range are shifted right to make room for the inserted cells.
kShiftVertical	2	Cells of the selected range are shifted down to make room for the inserted cells.
kShiftRows	3	Rows in which the range resides are shifted down to make room for the inserted cells.
kShiftCols	4	Columns in which the range resides are shifted right to make room for the inserted cells.

**Return Value** Integer

**See Also** [SSEditDelete](#) and [SSInsertRange](#) functions

**Example** `sserror = SSEditInsert(Sheet1.SS, kShiftHorizontal)`

## SSEditPaste

See also [A-Z Function Call List](#)

**Description** Pastes the contents of the clipboard to the selected range.

**Syntax (VB)** **SSEditPaste**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSEditPaste** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** **SSEditPaste** pastes information from the clipboard to the selected range in the 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.

**Return Value** Integer

**See Also** [SSCanEditPaste](#), [SSEditCopy](#), and [SSEditCut](#) functions

**Example** `sserror = SSEditPaste(Sheet1.SS)`

## SSEndEdit

See also [A-Z Function Call List](#)

<b>Description</b>	Ends edit mode and applies changes to the active cell.
<b>Syntax (VB)</b>	<b>SSEndEdit</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSEndEdit</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSEndEdit</b> 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, SSERROR_GENERAL is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCancelEdit</a> and <a href="#">SSStartEdit</a> functions
<b>Example</b>	<pre>sserror = SSEndEdit(Sheet1.SS)</pre>

## SSErrorNumberToText

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the error text corresponding to the specified error number.
<b>Syntax (VB)</b>	<b>SSErrorNumberToText%</b> Lib "VTSSDLL.DLL" (ByVal <i>nError%</i> , ByVal <i>pBuf\$</i> , ByVal <i>nBufSize%</i> )
<b>Syntax (VC++)</b>	<b>SSERROR SSEXPORTAPI SSErrorNumberToText</b> (SSERROR <i>nError</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>nError</i> is the number of the error for which to return error text.  <i>pBuf</i> is a string in which the error text is returned. The string must be of sufficient length to hold the returned text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must allocate space for the <i>pBuf</i> string using Space\$(n), where n is the <i>nBufSize</i> value you pass to the function.
<b>Return Value</b>	Integer
<b>Example</b>	<pre>sserror = SSErrorNumberToText(ErrorNumber, Buffer, Size)</pre>

## SSFilePageSetupDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Page Setup dialog box.
<b>Syntax (VB)</b>	<b>SSFilePageSetupDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFilePageSetupDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Page Setup dialog box allows you to define header and footer text, page margins, page print order, page centering, worksheet-related print options.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFilePrint</a> function
<b>Example</b>	<pre>sserror = SSFilePageSetupDlg(Sheet1.SS)</pre>

## SSFilePrint

See also [A-Z Function Call List](#)

<b>Description</b>	Prints a worksheet.
<b>Syntax (VB)</b>	<b>SSFilePrint</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bShowPrintDlg</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSFilePrint</b> (HSS <i>hSS</i> , BOOL <i>bShowPrintDlg</i> )  <i>hSS</i> is a handle to a view. <i>bShowPrintDlg</i> sets the show print dialog flag.
<b>Remarks</b>	<b>SSFilePrint</b> prints the worksheet or selections as directed by the user. If <i>bShowPrintDlg</i> 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.  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 worksheet is printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFilePageSetupDlg</a> and <a href="#">SSFilePrintSetupDlg</a> functions
<b>Example</b>	<pre>sserror = SSFilePrint(Sheet1.SS)</pre>

## SSFilePrintSetupDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the standard Windows Print Setup dialog box.
<b>Syntax (VB)</b>	<b>SSFilePrintSetupDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFilePrintSetupDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Print Setup dialog box allows you to select the printer to which the worksheet is sent, the page orientation, and paper size.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFilePageSetupDlg</a> and <a href="#">SSFilePrint</a> functions
<b>Example</b>	<pre>sserror = SSFilePrintSetupDlg(Sheet1.SS)</pre>

## SSFormatAlignmentDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Alignment dialog box.
<b>Syntax (VB)</b>	<b>SSFormatAlignmentDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatAlignmentDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAlignment</a> function
<b>Example</b>	<code>sserror = SSFormatAlignmentDlg(Sheet1.SS)</code>



## SSFormatBorderDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Border dialog box.
<b>Syntax (VB)</b>	<b>SSFormatBorderDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatBorderDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetBorder</a> function
<b>Example</b>	<code>sserror = SSFormatBorderDlg(Sheet1.SS)</code>

## SSFormatCurrency0

See also [A-Z Function Call List](#)

<b>Description</b>	Formats selected ranges with currency format and no decimal places.
<b>Syntax (VB)</b>	<b>SSFormatCurrency0%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatCurrency0</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	Currency (0) format displays numbers with a leading dollar sign and no decimal places.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFormatCurrency2</a> and <a href="#">SSSetNumberFormat</a> functions
<b>Example</b>	<code>sserror = SSFormatCurrency0 (Sheet1.SS)</code>

## SSFormatCurrency2

See also [A-Z Function Call List](#)

<b>Description</b>	Formats selected ranges with currency format and two decimal places.
<b>Syntax (VB)</b>	<b>SSFormatCurrency2</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatCurrency2</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	Currency (2) format displays numbers with a leading dollar sign and two decimal places.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFormatCurrency0</a> and <a href="#">SSSetNumberFormat</a> functions
<b>Example</b>	<code>sserror = SSFormatCurrency2 (Sheet1.SS)</code>

## SSFormatFixed

See also [A-Z Function Call List](#)

**Description** Formats selected ranges with fixed format and no decimal places.

**Syntax (VB)** **SSFormatFixed**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSFormatFixed** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** Fixed format includes thousands separators (commas).

**Return Value** Integer

**See Also** [SSSetNumberFormat](#) function

**Example** `sserror = SSFormatFixed(Sheet1.SS)`

## SSFormatFixed2

See also [A-Z Function Call List](#)

**Description** Formats selected ranges with fixed format and two decimal places.

**Syntax (VB)** **SSFormatFixed2**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSFormatFixed2** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** Fixed format includes thousands separators (commas).

**Return Value** Integer

**See Also** [SSSetNumberFormat](#) function

**Example** `sserror = SSFormatFixed2(Sheet1.SS)`

## SSFormatFontDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Font dialog box.
<b>Syntax (VB)</b>	<b>SSFormatFontDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatFontDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Font dialog box allows you to specify the font, point size, font style, and color of data in the selected range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetFont</a> function
<b>Example</b>	<pre>sserror = SSFormatFontDlg(Sheet1.SS)</pre>

## SSFormatFraction

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges with the fraction format.
<b>Syntax (VB)</b>	<b>SSFormatFraction%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSFormatFraction</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<code>sserror = SSFormatFraction(Sheet1.SS)</code>

## SSFormatGeneral

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges with the general format.
<b>Syntax (VB)</b>	<b>SSFormatGeneral</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSFormatGeneral</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The general format displays numbers with as many decimal places as necessary; thousands separators (commas) are not used.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<code>sserror = SSFormatGeneral(Sheet1.SS)</code>



## SSFormatHmampm

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges with the 12-hour time format.
<b>Syntax (VB)</b>	<b>SSFormatHmampm</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatHmampm</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	All selected ranges are formatted with the h:mm am/pm format (e.g., 1:00 am).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<pre>sserror = SSFormatHmampm(Sheet1.SS)</pre>

## SSFormatMdyy

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges with the date format.
<b>Syntax (VB)</b>	<b>SSFormatMdyy</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatMdyy</b> ( <i>HSS hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	All selected ranges are formatted with the m/d/yy format (e.g., 12/31/93).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<code>sserror = SSFormatMdyy(Sheet1.SS)</code>

## SSFormatNumberDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Custom Number dialog box.
<b>Syntax (VB)</b>	<b>SSFormatNumberDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatNumberDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Custom Number dialog box allows you to define custom number formats for data in the selected range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<code>sserror = SSFormatNumberDlg (Sheet1.SS)</code>

## SSFormatPatternDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Pattern dialog box.
<b>Syntax (VB)</b>	<b>SSFormatPatternDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatPatternDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	The Pattern dialog box allows you to specify the fill pattern and foreground and background colors for the selected range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPattern</a> function
<b>Example</b>	<code>sserror = SSFormatPatternDlg (Sheet1.SS)</code>

## SSFormatPercent

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges in percent format.
<b>Syntax (VB)</b>	<b>SSFormatPercent%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatPercent</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	Percent format displays numbers with a trailing percent sign and no decimal places.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<pre>sserror = SSFormatPercent(Sheet1.SS)</pre>

## SSFormatRCNr

See also [A-Z Function Call List](#)

<b>Description</b>	Creates a string containing a formatted row and column reference.
<b>Syntax (VB)</b>	<b>SSFormatRCNr</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>bDoAbsolute</i> %, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSFormatRCNr</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , BOOL <i>bDoAbsolute</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view, though it is not used in this version of the program. <i>nRow</i> and <i>nCol</i> are the row and column references to format. <i>bDoAbsolute</i> specifies whether absolute or relative cell references are used. Use True for absolute references, False for relative references. <i>pBuf</i> is a string in which the reference is returned. This string must be of sufficient length to hold the returned reference. <i>nBufSize</i> if the size of the return buffer. If the string is larger than <i>nBufSize</i> , an error is returned and an empty string is put in <i>pBuf</i> . <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSFormatRCNr</b> formats a row and column reference and creates a string containing the reference. The string is returned in <i>pBuf</i> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">Selection</a> property
<b>Example</b>	<pre>sserror = SSFormatRCNr (Sheet1.SS, 2, 3, True, buff\$, 10) ' Puts the null terminated string "\$C\$2" into buff\$.</pre>

## SSFormatScientific

See also [A-Z Function Call List](#)

<b>Description</b>	Formats the selected ranges in scientific format.
<b>Syntax (VB)</b>	<b>SSFormatScientific%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSFormatScientific</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumberFormat</a> function
<b>Example</b>	<code>sserror = SSFormatScientific(Sheet1.SS)</code>

## SSGetActiveCell

See also [A-Z Function Call List](#)

**Description** Returns the row and column of the active cell.

**Syntax (VB)** `SSGetActiveCell% Lib "VTSSDLL.DLL" (ByVal hSS&, pRow%, pCol%)`

**Syntax (VC++)** `SSERROR SSEXPORTAPI SSGetActiveCell (HSS hSS, LPRC pRow, LPRC pCol)`

*hSS* is a handle to a view.

*pRow* and *pCol* return the row and column of the active cell.

**Remarks** The active cell is the cell on which the cursor is currently located.

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					

Cell A1 is the active cell in this worksheet. The active cell is highlighted by a heavy border.

**Return Value** Integer

**See Also** [SSSetActiveCell](#) function and [Col](#) and [Row](#) properties

**Example** `sserror = SSGetActiveCell(Sheet1.SS, therow, thecol)`



## SSGetAllowArrows

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the allow arrows flag.
<b>Syntax (VB)</b>	<b>SSGetAllowArrows</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowArrows</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowArrows</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowArrows</i> )  <i>hSS</i> is a handle to a view. <i>pAllowArrows</i> is the destination of the allow arrows flag.
<b>Remarks</b>	<b>SSGetAllowArrows</b> returns the state of the allow arrows flag. If the flag is True, the arrow keys on your keyboard can move the active cell in the spreadsheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowArrows</a> function and <a href="#">AllowArrows</a> property
<b>Example</b>	<code>sserror = SSGetAllowArrows(Sheet1.SS, allowarrows)</code>

## SSGetAllowDelete

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the allow delete flag.
<b>Syntax (VB)</b>	<b>SSGetAllowDelete</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowDelete</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowDelete</b> (HSS <i>hSS</i> , BOOL FAR <i>pAllowDelete</i> )  <i>hSS</i> is a handle to a view. <i>pAllowDelete</i> is the destination of the allow delete flag.
<b>Remarks</b>	If the allow delete flag is True and data browsing mode is enabled, 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.  If the flag is False, the delete key does not delete records or clear selections. By default, the allow delete flag is True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowDelete</a> function and <a href="#">AllowDelete</a> property
<b>Example</b>	<code>sserror = SSGetAllowDelete(Sheet1.SS, allowdelete)</code>

## SSGetAllowEditHeaders

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the edit headers flag.
<b>Syntax (VB)</b>	<b>SSGetAllowEditHeaders</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowEditHeaders</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowEditHeaders</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowEditHeaders</i> )  <i>hSS</i> is a handle to a view. <i>pAllowEditHeaders</i> is the destination of the edit headers flag.
<b>Remarks</b>	<p>If the edit headers flag is True, the names displayed in row, column, and top left headers can be edited by double clicking the header to be edited. The Header Name dialog box is displayed, allowing you to enter a new header name.</p> <p>If the flag is False, editing of headers is not allowed and a <b>DbiClick</b> event is passed when a header is double clicked.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowEditHeaders</a> function, <a href="#">AllowEditHeaders</a> property, and <a href="#">DbiClick</a> event
<b>Example</b>	<pre>sserror = SSGetAllowEditHeaders(Sheet1.SS, alloweditheaders)</pre>

## SSGetAllowFillRange

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the fill range flag.
<b>Syntax (VB)</b>	<b>SSGetAllowFillRange%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowFillRange</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowFillRange</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowFillRange</i> )  <i>hSS</i> is a handle to a view. <i>pAllowFillRange</i> is the destination of the fill range flag.
<b>Remarks</b>	If the fill range flag is True, filling ranges by dragging a selection is allowed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowFillRange</a> function and <a href="#">AllowFillRange</a> property
<b>Example</b>	<code>sserror = SSGetAllowFillRange(Sheet1.SS, allowfillrange)</code>

## SSGetAllowFormulas

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the user formulas flag.
<b>Syntax (VB)</b>	<b>SSGetAllowFormulas</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowFormulas</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowFormulas</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowFormulas</i> )  <i>hSS</i> is a handle to a view. <i>pAllowFormulas</i> is the destination of the user formulas flag.
<b>Remarks</b>	If the user formulas flag is True, formulas can be added by the user at run time.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowFormulas</a> function and <a href="#">AllowFormulas</a> property
<b>Example</b>	<code>sserror = SSGetAllowFormulas(Sheet1.SS, allowformulas)</code>

## SSGetAllowInCellEditing

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the in-cell editing flag.
<b>Syntax (VB)</b>	<b>SSGetAllowInCellEditing</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowInCellEditing</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowInCellEditing</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowInCellEditing</i> )  <i>hSS</i> is a handle to a view.  <i>pAllowInCellEditing</i> is the destination of the in-cell editing flag.
<b>Remarks</b>	If the in-cell editing flag is True, in-cell editing is active and data can be entered and edited in a cell without the use of the edit bar.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowInCellEditing</a> function and <a href="#">AllowInCellEditing</a> property
<b>Example</b>	<pre>sserror = SSGetAllowInCellEditing(Sheet1.SS, allowincellediting)</pre>

## SSGetAllowMoveRange

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the move range flag.
<b>Syntax (VB)</b>	<b>SSGetAllowMoveRange</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowMoveRange</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowMoveRange</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowMoveRange</i> )  <i>hSS</i> is a handle to a view. <i>pAllowMoveRange</i> is the destination of the move range flag.
<b>Remarks</b>	If the move range flag is True, moving ranges by dragging a cell is allowed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowMoveRange</a> function and <a href="#">AllowMoveRange</a> property
<b>Example</b>	<code>sserror = SSGetAllowMoveRange(Sheet1.SS, allowmoverange)</code>

## SSGetAllowResize

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the resize flag.
<b>Syntax (VB)</b>	<b>SSGetAllowResize</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowResize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowResize</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowResize</i> )  <i>hSS</i> is a handle to a view. <i>pAllowResize</i> is the destination of the resize flag.
<b>Remarks</b>	If the resize flag is True, rows and columns can be sized by dragging row or column heading borders.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowResize</a> function and <a href="#">AllowResize</a> property
<b>Example</b>	<code>sserror = SSGetAllowResize(Sheet1.SS, allowresize)</code>



## SSGetAllowSelections

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the select range flag.
<b>Syntax (VB)</b>	<b>SSGetAllowSelections%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowSelections</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowSelections</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAllowSelections</i> )  <i>hSS</i> is a handle to a view. <i>pAllowSelections</i> is the destination of the select range flag.
<b>Remarks</b>	If the select range flag is True, ranges can be selected with the keyboard and by clicking and dragging with the mouse.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowSelections</a> function and <a href="#">AllowSelections</a> property
<b>Example</b>	<pre>sserror = SSGetAllowSelections(Sheet1.SS, allowselections)</pre>

## SSGetAllowTabs

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the allow tabs flag.
<b>Syntax (VB)</b>	<b>SSGetAllowTabs</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAllowTabs</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetAllowTabs</b> (HSS <i>hSS</i> , BOOL FAR * <i>pAllowTabs</i> )  <i>hSS</i> is a handle to a view. <i>pAllowTabs</i> is the destination of the allow tabs flag.
<b>Remarks</b>	If the allow tabs flag is True, the tab key can 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAllowTabs</a> function and <a href="#">AllowTabs</a> property
<b>Example</b>	<pre>sserror = SSGetAllowTabs(Sheet1.SS, allowtabs)</pre>

## SSGetAutoRecalc

<b>Description</b>	Returns the state of the automatic recalc flag.
<b>Syntax (VB)</b>	<b>SSGetAutoRecalc</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pAutoRecalc</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetAutoRecalc</b> (HSS <i>hSS</i> , BOOL FAR <i>*pAutoRecalc</i> )  <i>hSS</i> is a handle to a view. <i>pAutoRecalc</i> is the destination of the automatic recalc flag.
<b>Remarks</b>	If the automatic recalculation flag is True, automatic recalculation is enabled.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRecalc</a> and <a href="#">SSSetAutoRecalc</a> functions and <a href="#">AutoRecalc</a> property
<b>Example</b>	<code>sserror = SSGetAutoRecalc(Sheet1.SS, autorecalc)</code>

## SSGetBackColor

<b>Description</b>	Returns the background color of the view.
<b>Syntax (VB)</b>	<b>SSGetBackColor</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pBackColor</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetBackColor</b> (HSS <i>hSS</i> , COLORREF FAR <i>*pBackColor</i> )  <i>hSS</i> is a handle to a view.  <i>pBackColor</i> is the destination of the background color value.
<b>Remarks</b>	<b>SSGetBackColor</b> returns the background color of the worksheet attached to the specified view. All cells within the view are set to the background color except those with patterns.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetColor</a> function and <a href="#">BackColor</a> property
<b>Example</b>	<pre>sserror = SSGetBackColor(Sheet1.SS, backcolor)</pre>

## SSGetColWidth

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the width of the specified column.
<b>Syntax (VB)</b>	<b>SSGetColWidth%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nCol</i> %, <i>pWidth</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetColWidth</b> (HSS <i>hSS</i> , RC <i>nCol</i> , int FAR * <i>pWidth</i> )  <i>hSS</i> is a handle to a view. <i>nCol</i> is the column number for which to return the width. <i>pWidth</i> is the destination of the column width value.
<b>Remarks</b>	<b>SSGetColWidth</b> returns the width of the specified column in units equal to 1/256th of an average character's width in the default font.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetRowHeight</a> and <a href="#">SSSetColWidth</a> functions
<b>Example</b>	<pre>sserror = SSGetColWidth(Sheet1.SS, 1, colwidth)</pre>

## SSGetDefinedName

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the range definition for a user-defined range name.
<b>Syntax (VB)</b>	<b>SSGetDefinedName</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pName</i> %, ByVal <i>pBuf</i> %, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetDefinedName (HSS <i>hSS</i> , LPCSTR <i>pName</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view. <i>pName</i> is the range name for which the range definition is returned. <i>pBuf</i> is the destination buffer of the range definition. <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> <b>Important</b> Before calling this function, you must initialize a string to Space\$(n), where n is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetDefinedName</b> returns a string containing the range of the specified user-defined name. For example, if the range B10:F10 is named TotalSales, a string is returned containing the range reference B10:F10.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSDefinedNameDlg</a> , <a href="#">SSDeleteDefinedName</a> , and <a href="#">SSSetDefinedName</a> functions
<b>Example</b>	<pre>sserror = SSGetDefinedName(Sheet1.SS, name\$, namebuf\$, bufsize)</pre>

## SSGetEnableProtection

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the enable protection flag.
<b>Syntax (VB)</b>	<b>SSGetEnableProtection%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pEnableProtection</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetEnableProtection</b> (HSS <i>hSS</i> , BOOL FAR <i>*pEnableProtection</i> )  <i>hSS</i> is a handle to a view. <i>pEnableProtection</i> is the destination of the enable protection flag.
<b>Remarks</b>	If the enable protection flag is True, worksheet protection is enabled and worksheet cells are locked and formulas are hidden. Cells can be marked as locked and hidden using the <b>SSSetProtection</b> and <b>SSProtectionDlg</b> function calls.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSProtectionDlg</a> , <a href="#">SSSetEnableProtection</a> , and <a href="#">SSSetProtection</a> functions and <a href="#">EnableProtection</a> property
<b>Example</b>	<pre>sserror = SSGetEnableProtection(Sheet1.SS, enableprotection)</pre>

## SSGetEnterMovesDown

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the enter moves down flag.
<b>Syntax (VB)</b>	<b>SSGetEnterMovesDown</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pEnterMovesDown</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetEnterMovesDown</b> (HSS <i>hSS</i> , BOOL FAR <i>*pEnterMovesDown</i> )  <i>hSS</i> is a handle to a view. <i>pEnterMovesDown</i> is the destination of the enter moves down flag.
<b>Remarks</b>	If the enter moves down flag is True, the enter key moves the active cell down to the next row, even if no range is selected. If False, the enter key does not advance the active cell.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetEnterMovesDown</a> function
<b>Example</b>	<pre>sserror = SSGetEnterMovesDown(Sheet1.SS, entermovesdown)</pre>



## SSGetEntry

See also [A-Z Function Call List](#)

**Description** Returns the text value of the active cell in the same format as displayed while in edit mode.

**Syntax (VB)** **SSGetEntry**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pBuf*\$, ByVal *nBufSize*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSGetEntry (HSS *hSS*, LPSTR *pBuf*, int *nBufSize*)

*hSS* is a handle to a view.

*pBuf* is the destination buffer of the returned text.

*nBufSize* is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in *pBuf*.

**Important** Before calling this function, you must initialize a string to Space\$(*n*), where *n* is the *nBufSize* value you pass to the function.

---

**Remarks** The text returned 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.

**Return Value** Integer

**See Also** [SSGetEntryRC](#) and [SSSetEntry](#) functions and [Entry](#) and [Text](#) properties

**Example** `sserror = SSGetEntry(Sheet1.SS, textbuf$, bufsize)`

## SSGetEntryRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the text value of the specified cell in the same format as displayed while in edit mode.
<b>Syntax (VB)</b>	<b>SSGetEntryRC%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetEntryRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> );  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which to return the text.  <i>pBuf</i> is the destination buffer of the returned text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i>  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	The text returned 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEntry</a> and <a href="#">SSSetEntryRC</a> functions and <a href="#">Entry</a> and <a href="#">Text</a> properties
<b>Example</b>	<pre>sserror = SSGetEntryRC(Sheet1.SS, 1, 1, textbuf\$, bufsize)</pre>

## SSGetExtraColor

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the extra color.
<b>Syntax (VB)</b>	<b>SSGetExtraColor</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pExtraColor</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetExtraColor</b> (HSS <i>hSS</i> , COLORREF FAR <i>*pExtraColor</i> )  <i>hSS</i> is a handle to a view. <i>pExtraColor</i> is the destination of the extra color value.
<b>Remarks</b>	<b>SSGetExtraColor</b> returns the extra color that is used to fill the space in the view window not covered by the worksheet. This space occurs when the worksheet is smaller than the view window.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetExtraColor</a> function and <a href="#">ExtraColor</a> property
<b>Example</b>	<pre>sserror = SSGetExtraColor(Sheet1.SS, extracolor)</pre>

## SSGetFireEvent

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the flag that indicates whether the given event is enabled.
<b>Syntax (VB)</b>	<b>SSGetFireEvent%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nEvent</i> %, <i>pFireIt</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetFireEvent</b> (HSS <i>hSS</i> , UINT <i>nEvent</i> , BOOL FAR <i>*pFireIt</i> )  <i>hSS</i> is a handle to a view. <i>nEvent</i> is the event to test. <i>pFireIt</i> is the destination of the event enabled value.
<b>Remarks</b>	See the event list in <b>SSSetFireEvent</b> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetFireEvent</a> function
<b>Example</b>	<pre>sserror = SSGetFireEvent(Sheet1.SS, SSM_STARTEDIT, fireit)</pre>

## SSGetFixedCols

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the starting fixed column and the number of fixed columns in a view.
<b>Syntax (VB)</b>	<b>SSGetFixedCols%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pCol1</i> %, <i>pCols</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetFixedCols</b> (HSS <i>hSS</i> , LPRC <i>pCol1</i> , LPRC <i>pCols</i> )  <i>hSS</i> is a handle to a view. <i>pCol1</i> is the destination of the starting fixed column. <i>pCols</i> is the destination of the number of fixed columns.
<b>Remarks</b>	Fixed columns are columns that do not scroll. The columns are fixed at the left edge of the worksheet window.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetFixedCols</a> function and <a href="#">FixedCol</a> and <a href="#">FixedCols</a> properties
<b>Example</b>	<code>sserror = SSGetFixedCols(Sheet1.SS, startcol, numbercols)</code>

## SSGetFixedRows

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the starting fixed row and the number of fixed rows in a view.
<b>Syntax (VB)</b>	SSGetFixedRows% Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pRow1</i> %, <i>pRows</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetFixedRows (HSS <i>hSS</i> , LPRC <i>pRow1</i> , LPRC <i>pRows</i> )  <i>hSS</i> is a handle to a view. <i>pRow1</i> is the destination of the starting fixed row. <i>pRows</i> is the destination of the number of fixed rows.
<b>Remarks</b>	Fixed rows are rows that do not scroll. The rows are fixed at the top of the worksheet window.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetFixedRows</a> function and <a href="#">FixedRow</a> and <a href="#">FixedRows</a> properties
<b>Example</b>	<pre>sserror = SSGetFixedRows(Sheet1.SS, startrow, numberrows)</pre>

## SSGetFormattedText

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the formatted text value of the active cell.
<b>Syntax (VB)</b>	<b>SSGetFormattedText</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	<b>SSERROR SSEXPORTAPI SSGetFormattedText</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> );  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer of the returned text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetFormattedText</b> returns the text as it is seen in the spreadsheet, including all formatting.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEntry</a> , <a href="#">SSGetFormattedTextRC</a> , and <a href="#">SSSetEntry</a> functions and <a href="#">Entry</a> , <a href="#">FormattedText</a> , and <a href="#">Text</a> properties
<b>Example</b>	<pre>sserror = SSGetFormattedText(Sheet1.SS, textbuf\$, bufsize)</pre>

## SSGetFormattedTextRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the formatted text value of the specified cell.
<b>Syntax (VB)</b>	<b>SSGetFormattedTextRC%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetFormattedTextRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> );  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which to return the text.  <i>pBuf</i> is the destination buffer of the returned text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetFormattedTextRC</b> returns the text as it is seen in the spreadsheet, including all formatting.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEntryRC</a> , <a href="#">SSGetFormattedText</a> , and <a href="#">SSSetEntryRC</a> functions and <a href="#">Entry</a> , <a href="#">FormattedText</a> , and <a href="#">Text</a> properties
<b>Example</b>	<pre>sserror = SSGetFormattedTextRC(Sheet1.SS, 1, 1, textbuf\$, bufsize)</pre>



## SSGetFormula

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the text version of the formula in the active cell.
<b>Syntax (VB)</b>	<b>SSGetFormula</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetFormula</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view. <i>pBuf</i> is the destination buffer of the returned formula text. <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> . <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFormulaRC</a> , <a href="#">SSSetFormula</a> and <a href="#">SSSetFormulaRC</a> functions and <a href="#">Formula</a> property
<b>Example</b>	<pre>sserror = SSGetFormula(Sheet1.SS, formulabuf\$, bufsize)</pre>

## SSGetFormulaRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the text version of the formula of the specified cell.
<b>Syntax (VB)</b>	<b>SSGetFormulaRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetFormulaRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which to return the formula text.  <i>pBuf</i> is the destination buffer of the returned formula text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFormula</a> and <a href="#">SSSetFormulaRC</a> functions and <a href="#">Formula</a> property
<b>Example</b>	<pre>sserror = SSGetFormulaRC(Sheet1.SS, 1, 1, formulabuf\$, bufsize)</pre>

## SSGetHdrSelection

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the state of the header selection flags.
<b>Syntax (VB)</b>	<b>SSGetHdrSelection</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pTopLeftHdr</i> %, <i>pRowHdr</i> %, <i>pColHdr</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetHdrSelection</b> (HSS <i>hSS</i> , BOOL FAR <i>*pTopLeftHdr</i> , BOOL FAR <i>*pRowHdr</i> , BOOL FAR <i>*pColHdr</i> )  <i>hSS</i> is a handle to a view. <i>pTopLeftHdr</i> is the destination of the top left header selection flag. <i>pRowHdr</i> is the destination of the row header selection flag. <i>pColHdr</i> is the destination of the column header selection flag.
<b>Remarks</b>	<b>SSGetHdrSelection</b> returns the states of the header selection flags. The flags determine if the row headings, column headings, and the cell at the intersection of the row and column headings are selected. If a flag is True, the corresponding heading is selected. If False, the heading is not selected.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetHdrSelection</a> function
<b>Example</b>	<pre>sserror = SSGetHdrSelection(Sheet1.SS, toplefthdr, rowhdr, colhdr)</pre>

## SSGetIteration

See also [A-Z Function Call List](#)

<b>Description</b>	Returns iteration information.
<b>Syntax (VB)</b>	<b>SSGetIteration</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pIteration</i> %, <i>pMaxIterations</i> %, <i>pMaxChange</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetIteration</b> (HSS <i>hSS</i> , BOOL FAR * <i>pIteration</i> , int FAR * <i>pMaxIterations</i> , double FAR * <i>pMaxChange</i> )  <i>hSS</i> is a handle to a view. <i>pIteration</i> is the destination of the iteration flag. <i>pMaxIterations</i> is the destination of the maximum iterations value. <i>pMaxChange</i> is the destination of the maximum change value.
<b>Remarks</b>	<b>SSGetIteration</b> returns the iteration flag, maximum number of iterations, and the maximum change value. Iteration can be used to solve circular references. The program calculates until it iterates the number of times specified by <i>pMaxIterations</i> or until all cells change by less than the amount specified in <i>pMaxChange</i> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetIteration</a> function
<b>Example</b>	<pre>sserror = SSGetIteration(Sheet1.SS, iteration, maxiterations, maxchange)</pre>

## SSGetLastCol

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the number of the last occupied column.
<b>Syntax (VB)</b>	<b>SSGetLastCol</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pLastCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetLastCol</b> (HSS <i>hSS</i> , LPRC <i>pLastCol</i> )  <i>hSS</i> is a handle to a view. <i>pLastCol</i> is the destination for the last column value.
<b>Remarks</b>	<b>SSGetLastCol</b> returns the last column that is not empty, including columns that contain only formatting.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetLastRow</a> and <a href="#">SSGetLastColForRow</a> functions
<b>Example</b>	<code>sserror = SSGetLastCol(Sheet1.SS, lastcol)</code>

## SSGetLastColForRow

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the last occupied column in the specified row.
<b>Syntax (VB)</b>	<b>SSGetLastColForRow%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, <i>pLastColForRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetLastColForRow</b> (HSS <i>hSS</i> , RC <i>nRow</i> , LPC <i>pLastColForRow</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> is the row number for which to set the tag.  <i>pLastColForRow</i> is the destination of the last column value.
<b>Remarks</b>	<b>SSGetLastColForRow</b> returns the last column in the specified row that is not empty, including columns that contain only formatting.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetLastCol</a> and <a href="#">SSGetLastRow</a> functions
<b>Example</b>	<pre>sserror = SSGetLastColForRow(Sheet1.SS, 1, lastcol)</pre>

## SSGetLastRow

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the number of the last occupied row.
<b>Syntax (VB)</b>	<b>SSGetLastRow%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pLastRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetLastRow</b> (HSS <i>hSS</i> , LPRC <i>pLastRow</i> )  <i>hSS</i> is a handle to a view. <i>pLastRow</i> is the destination for the last row value.
<b>Remarks</b>	<b>SSGetLastRow</b> returns the last row that is not empty, including rows that contain only formatting.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetLastCol</a> and <a href="#">SSGetLastColForRow</a> functions
<b>Example</b>	<code>sserror = SSGetLastRow(Sheet1.SS, lastrow)</code>

## SSGetLeftCol

See also [A-Z Function Call List](#)

**Description** Returns the leftmost column displayed in the view.

**Syntax (VB)** **SSGetLeftCol**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, *pLeftCol*%)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSGetLeftCol** (HSS *hSS*, LPRC *pLeftCol*)

*hSS* is a handle to a view.

*pLeftCol* is the destination of the left column value.

**Return Value** Integer

**See Also** [SSSetLeftCol](#) function and [LeftCol](#) property

**Example** `sserror = SSGetLeftCol(Sheet1.SS, leftcol)`



## SSGetLogicalRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the logical (True or False) value of the specified cell.
<b>Syntax (VB)</b>	<b>SSGetLogicalRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, <i>plsTrue</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetLogicalRC (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , BOOL FAR <i>*plsTrue</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which the logical value is returned.  <i>plsTrue</i> is the destination of the cell's logical value.
<b>Remarks</b>	<b>SSGetLogicalRC</b> returns the logical value of the specified cell. If the cell contains a number, True is returned 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, return False.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetLogicalRC</a> function
<b>Example</b>	<code>sserror = SSGetLogicalRC(Sheet1.SS, 1, 1, logicalvalue)</code>

## SSGetMaxCol

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the last displayable column in a view.
<b>Syntax (VB)</b>	<b>SSGetMaxCol%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pMaxCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetMaxCol</b> (HSS <i>hSS</i> , LPRC <i>pMaxCol</i> )  <i>hSS</i> is a handle to a view. <i>pMaxCol</i> is the destination of the maximum column value.
<b>Remarks</b>	Columns beyond the last column are not displayed but can be used to hold data and formulas.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetMaxCol</a> function and <a href="#">MaxCol</a> property
<b>Example</b>	<pre>sserror = SSGetMaxCol(Sheet1.SS, maxcol)</pre>

## SSGetMaxRow

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the last displayable row in a view.
<b>Syntax (VB)</b>	<b>SSGetMaxRow%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pMaxRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetMaxRow</b> (HSS <i>hSS</i> , LPRC <i>pMaxRow</i> )  <i>hSS</i> is a handle to a view. <i>pMaxRow</i> is the destination of the maximum row value.
<b>Remarks</b>	Rows beyond the last row are not displayed but can be used to hold data and formulas.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetMaxRow</a> function and <a href="#">MaxRow</a> property
<b>Example</b>	<pre>sserror = SSGetMaxRow(Sheet1.SS, maxrow)</pre>

## SSGetMinCol

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the first column that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSGetMinCol</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pMinCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetMinCol</b> (HSS <i>hSS</i> , LPRC <i>pMinCol</i> )  <i>hSS</i> is a handle to a view. <i>pMinCol</i> is the destination of the minimum column value.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetMinCol</a> function and <a href="#">MinCol</a> property
<b>Example</b>	<code>sserror = SSGetMinCol(Sheet1.SS, mincol)</code>

## SSGetMinRow

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the first row that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSGetMinRow</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pMinRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetMinRow</b> (HSS <i>hSS</i> , LPRC <i>pMinRow</i> )  <i>hSS</i> is a handle to a view. <i>pMinRow</i> is the destination of the minimum row value.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetMinRow</a> function and <a href="#">MinRow</a> property
<b>Example</b>	<code>sserror = SSGetMinRow(Sheet1.SS, minrow)</code>

## SSGetNumber

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the numeric value of the active cell.
<b>Syntax (VB)</b>	<b>SSGetNumber</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pNumber</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetNumber</b> (HSS <i>hSS</i> , double FAR * <i>pNumber</i> )  <i>hSS</i> is a handle to a view. <i>pNumber</i> is the destination of the cell value.
<b>Remarks</b>	If the active cell contains a formula, the numeric result of the formula is returned. If the cell contains text, an attempt is made to convert the text to a number. If the text cannot be converted, 0 is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetNumberRC</a> and <a href="#">SSSetNumber</a> functions and <a href="#">Number</a> property
<b>Example</b>	<pre>sserror = SSGetNumber(Sheet1.SS, thenumber)</pre>

## SSGetNumberRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the numeric value of the specified cell.
<b>Syntax (VB)</b>	<b>SSGetNumberRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, <i>pNumber</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetNumberRC (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , double FAR * <i>pNumber</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which to return the numeric value.  <i>pNumber</i> is the destination of the cell value.
<b>Remarks</b>	If the specified cell contains a formula, the numeric result of the formula is returned. If the cell contains text, an attempt is made to convert the text to a number. If the text cannot be converted, 0 is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetNumber</a> , <a href="#">SSSetNumberRC</a> functions and <a href="#">Number</a> property
<b>Example</b>	<code>sserror = SSGetNumberRC(Sheet1.SS, 1, 1, thenumber)</code>

## SSGetPrintArea

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the current print area.
<b>Syntax (VB)</b>	<b>SSGetPrintArea</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintArea</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer of the print area formula.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetPrintArea</b> returns a string containing a formula for the Print_Area user defined name. The formula can contain one or more ranges (e.g., A1:C3, A11:C13).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintArea</a> function and <a href="#">PrintArea</a> property
<b>Example</b>	<pre>sserror = SSGetPrintArea(Sheet1.SS, areabuf\$, bufsize)</pre>



## SSGetPrintBottomMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the bottom page margin used during printing.
<b>Syntax (VB)</b>	<b>SSGetPrintBottomMargin</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintBottomMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetPrintBottomMargin</b> (HSS <i>hSS</i> , double FAR <i>*pPrintBottomMargin</i> )  <i>hSS</i> is a handle to a view. <i>pPrintBottomMargin</i> is the destination of the bottom margin value.
<b>Remarks</b>	Page margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintBottomMargin</a> function and <a href="#">PrintBottomMargin</a> property
<b>Example</b>	<code>sserror = SSGetPrintBottomMargin(Sheet1.SS, bottommargin)</code>

## SSGetPrintColHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the print column heading flag.
<b>Syntax (VB)</b>	<b>SSGetPrintColHeading%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintColHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintColHeading</b> (HSS <i>hSS</i> , BOOL FAR <i>*pPrintColHeading</i> )  <i>hSS</i> is a handle to a view. <i>pPrintColHeading</i> is the destination of the print column heading flag.
<b>Remarks</b>	If the print column heading flag is True, column headings are enabled and printed at the top of the worksheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintColHeading</a> function and <a href="#">PrintColHeading</a> property
<b>Example</b>	<code>sserror = SSGetPrintColHeading(Sheet1.SS, colheading)</code>

## SSGetPrintFooter

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the current page footer.
<b>Syntax (VB)</b>	<b>SSGetPrintFooter</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintFooter</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer of the footer string.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	The page footer is printed at the bottom of each page.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintFooter</a> function and <a href="#">PrintFooter</a> property
<b>Example</b>	<pre>sserror = SSGetPrintFooter(Sheet1.SS, footbuf\$, bufsize)</pre>

## SSGetPrintGridLines

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the print grid lines flag.
<b>Syntax (VB)</b>	<b>SSGetPrintGridLines</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintGridLines</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetPrintGridLines</b> (HSS <i>hSS</i> , BOOL FAR <i>*pPrintGridLines</i> )  <i>hSS</i> is a handle to a view. <i>pPrintGridLines</i> is the destination of the print grid lines flag.
<b>Remarks</b>	If the print grid lines flag is True, grid lines are printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintGridLines</a> function and <a href="#">PrintGridLines</a> property
<b>Example</b>	<code>sserror = SSGetPrintGridLines(Sheet1.SS, gridlines)</code>

## SSGetPrintHCenter

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the horizontal center flag.
<b>Syntax (VB)</b>	<b>SSGetPrintHCenter%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintHCenter</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetPrintHCenter</b> (HSS <i>hSS</i> , BOOL FAR <i>*pPrintHCenter</i> )  <i>hSS</i> is a handle to a view. <i>pPrintHCenter</i> is the destination of the horizontal center flag.
<b>Remarks</b>	If the horizontal center flag is True, the worksheet is centered horizontally on the page when printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintHCenter</a> function and <a href="#">PrintHCenter</a> property
<b>Example</b>	<pre>sserror = SSGetPrintHCenter(Sheet1.SS, hcenter)</pre>

## SSGetPrintHeader

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the page header.
<b>Syntax (VB)</b>	<b>SSGetPrintHeader</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetPrintHeader (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer of the page header.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i>  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	The page header is printed at the top of each page.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintHeader</a> function and <a href="#">PrintHeader</a> property
<b>Example</b>	<pre>sserror = SSGetPrintHeader(Sheet1.SS, headerbuf\$, bufsize)</pre>

## SSGetPrintLeftMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the left page margin used during printing.
<b>Syntax (VB)</b>	<b>SSGetPrintLeftMargin%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintLeftMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetPrintLeftMargin (HSS <i>hSS</i> , double FAR <i>*pPrintLeftMargin</i> )  <i>hSS</i> is a handle to a view. <i>pPrintLeftMargin</i> is the destination of the left margin value.
<b>Remarks</b>	Page margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintLeftMargin</a> function and <a href="#">PrintLeftMargin</a> property
<b>Example</b>	<code>sserror = SSGetPrintLeftMargin(Sheet1.SS, leftmargin)</code>

## SSGetPrintLeftToRight

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the left to right flag.
<b>Syntax (VB)</b>	<b>SSGetPrintLeftToRight</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintLeftToRight</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetPrintLeftToRight (HSS <i>hSS</i> , BOOL FAR <i>*pPrintLeftToRight</i> )  <i>hSS</i> is a handle to a view. <i>pPrintLeftToRight</i> is the destination of the left to right flag.
<b>Remarks</b>	If the left to right flag is True, pages in a worksheet are printed left to right before printing top to bottom.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintLeftToRight</a> function and <a href="#">PrintLeftToRight</a> property
<b>Example</b>	<code>sserror = SSGetPrintLeftToRight(Sheet1.SS, lefttoright)</code>



## SSGetPrintNoColor

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the print no color flag.
<b>Syntax (VB)</b>	<b>SSGetPrintNoColor</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pNoColor</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetPrintNoColor</b> (HSS <i>hSS</i> , BOOL FAR * <i>pNoColor</i> )  <i>hSS</i> is a handle to a view. <i>pNoColor</i> is the destination of the print no color flag.
<b>Remarks</b>	Color formats are translated by the printer driver and printed as patterns. This translation sometimes makes text unreadable. If the print no color flag is True, all worksheet colors are converted to black and white, and all patterns are removed. A cleaner output is produced.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintNoColor</a> function and <a href="#">PrintNoColor</a> property
<b>Example</b>	<pre>sserror = SSGetPrintNoColor(Sheet1.SS, nocolor)</pre>

## SSGetPrintRightMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the right page margin used during printing.
<b>Syntax (VB)</b>	<b>SSGetPrintRightMargin%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintRightMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintRightMargin</b> (HSS <i>hSS</i> , double FAR <i>*pPrintRightMargin</i> )  <i>hSS</i> is a handle to a view. <i>pPrintRightMargin</i> is the destination of the right margin value.
<b>Remarks</b>	Page margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintRightMargin</a> function and <a href="#">PrintRightMargin</a> property
<b>Example</b>	<code>sserror = SSGetPrintRightMargin(Sheet1.SS, rightmargin)</code>

## SSGetPrintRowHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the print row heading flag.
<b>Syntax (VB)</b>	<b>SSGetPrintRowHeading%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintRowHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintRowHeading</b> (HSS <i>hSS</i> , BOOL FAR <i>*pPrintRowHeading</i> )  <i>hSS</i> is a handle to a view. <i>pPrintRowHeading</i> is the destination of the print row heading flag.
<b>Remarks</b>	If the print row heading flag is True, row headings are enabled and printed at the left edge of the worksheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintRowHeading</a> function and <a href="#">PrintRowHeading</a> property
<b>Example</b>	<pre>sserror = SSGetPrintRowHeading(Sheet1.SS, rowheading)</pre>

## SSGetPrintTitles

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the print titles.
<b>Syntax (VB)</b>	<b>SSGetPrintTitles</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintTitles</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer of the print titles.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i>  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function. <hr/>
<b>Remarks</b>	<b>SSGetPrintTitles</b> returns a string containing the formula for the Print_Titles user defined name. 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 function returns null (""), no titles are printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintTitles</a> function and <a href="#">PrintTitles</a> property
<b>Example</b>	<pre>sserror = SSGetPrintTitles(Sheet1.SS, titlebuf\$, bufsize)</pre>

## SSGetPrintTopMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the top page margin used during printing.
<b>Syntax (VB)</b>	<b>SSGetPrintTopMargin</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintTopMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetPrintTopMargin</b> (HSS <i>hSS</i> , double FAR <i>*pPrintTopMargin</i> )  <i>hSS</i> is a handle to a view. <i>pPrintTopMargin</i> is the destination of the top margin value.
<b>Remarks</b>	Page margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintTopMargin</a> function and <a href="#">PrintTopMargin</a> property
<b>Example</b>	<code>sserror = SSGetPrintTopMargin(Sheet1.SS, topmargin)</code>

## SSGetPrintVCenter

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the vertical center flag.
<b>Syntax (VB)</b>	<b>SSGetPrintVCenter</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pPrintVCenter</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetPrintVCenter</b> (HSS <i>hSS</i> , BOOL FAR <i>*pPrintVCenter</i> )  <i>hSS</i> is a handle to a view. <i>pPrintVCenter</i> is the destination of the vertical center flag.
<b>Remarks</b>	If the vertical center flag is True, the worksheet is centered vertically on the page when printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintVCenter</a> function and <a href="#">PrintVCenter</a> property
<b>Example</b>	<pre>sserror = SSGetPrintVCenter(Sheet1.SS, vcenter)</pre>

## SSGetRepaint

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the repaint flag.
<b>Syntax (VB)</b>	<b>SSGetRepaint%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pRepaint</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGetRepaint</b> (HSS <i>hSS</i> , BOOL FAR * <i>pRepaint</i> )  <i>hSS</i> is a handle to a view. <i>pRepaint</i> is the destination of the repaint flag.
<b>Remarks</b>	If the repaint flag is True, repainting occurs in the entire window when Windows sends a WM_PAINT message. No repainting occurs when the repaint flag is False.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetRepaint</a> function and <a href="#">Repaint</a> property
<b>Example</b>	<pre>sserror = SSGetRepaint(Sheet1.SS, repaint)</pre>

## SSGetRowHeight

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the height of the specified row.
<b>Syntax (VB)</b>	<b>SSGetRowHeight%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, <i>pHeight</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetRowHeight</b> (HSS <i>hSS</i> , RC <i>nRow</i> , int FAR <i>*pHeight</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> is the row for which to return the height. <i>pHeight</i> is the destination of the row height value.
<b>Remarks</b>	<b>SSGetRowHeight</b> returns the height of the specified row in twips. A twip is 1/1440th of an inch.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetColWidth</a> and <a href="#">SSSetRowHeight</a> functions
<b>Example</b>	<code>sserror = SSGetRowHeight(Sheet1.SS, 1, height)</code>



## SSGetRowMode

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the row mode flag.
<b>Syntax (VB)</b>	<b>SSGetRowMode%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pRowMode</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetRowMode</b> (HSS <i>hSS</i> , BOOL FAR * <i>pRowMode</i> )  <i>hSS</i> is a handle to a view. <i>pRowMode</i> is the destination of the row mode flag.
<b>Remarks</b>	If the row mode flag is True, an entire row is selected when you select a cell. Normal cell selection occurs when the flag is False.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetRowMode</a> function and <a href="#">RowMode</a> property
<b>Example</b>	<pre>sserror = SSGetRowmode(Sheet1.SS, rowmode)</pre>

## SSGetSelection

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the start and end row and column of the specified selection.
<b>Syntax (VB)</b>	<b>SSGetSelection%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nSel</i> %, <i>pRow1</i> %, <i>pCol1</i> %, <i>pRow2</i> %, <i>pCol2</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetSelection</b> (HSS <i>hSS</i> , int <i>nSel</i> , LPRC <i>pRow1</i> , LPRC <i>pCol1</i> , LPRC <i>pRow2</i> , LPRC <i>pCol2</i> )  <i>hSS</i> is a handle to a view.  <i>nSel</i> is the selection number for which to return the row and column.  <i>pRow1</i> , <i>pCol1</i> , <i>pRow2</i> , and <i>pCol2</i> are the returned row and column numbers.
<b>Remarks</b>	<b>SSGetSelection</b> returns the start and end row and column numbers for the specified selection. An index of 0 returns the row and column coordinates of the first selection.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetSelectionCount</a> and <a href="#">SSSetSelection</a> functions and <a href="#">Selection</a> property
<b>Example</b>	<pre>sserror = SSGetSelection(Sheet1.SS, 0, row1, col1, row2, col2)</pre>

## SSGetSelectionCount

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the number of selected ranges.
<b>Syntax (VB)</b>	<b>SSGetSelectionCount</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pCount</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetSelectionCount</b> (HSS <i>hSS</i> , int FAR * <i>pCount</i> )  <i>hSS</i> is a handle to a view. <i>pCount</i> is the destination of the selected ranges count.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddSelection</a> function and <a href="#">Selection</a> property
<b>Example</b>	<code>sserror = SSGetSelectionCount(Sheet1.SS, selcount)</code>

## SSGetSelectionRef

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the current selection as a formula.
<b>Syntax (VB)</b>	<b>SSGetSelectionRef</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetSelectionRef</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is the destination buffer for the current selection formula.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i>  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetSelectionRef</b> returns the current selection as a formula without the leading equal sign (=).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetSelection</a> function and <a href="#">Selection</a> property
<b>Example</b>	<pre>sserror = SSGetSelectionRef(Sheet1.SS, selectionbuf\$, bufsize)</pre>

## SSGetShowColHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show column heading flag.
<b>Syntax (VB)</b>	<b>SSGetShowColHeading%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowColHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowColHeading</b> (HSS <i>hSS</i> , BOOL FAR <i>*pShowColHeading</i> )  <i>hSS</i> is a handle to a view. <i>pShowColHeading</i> is the destination of the show column heading flag.
<b>Remarks</b>	If the show column heading flag is True, column headings are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowColHeading</a> function and <a href="#">ShowColHeading</a> property
<b>Example</b>	<code>sserror = SSGetShowColHeading(Sheet1.SS, showcolheading)</code>

## SSGetShowFormulas

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show formulas flag.
<b>Syntax (VB)</b>	<b>SSGetShowFormulas%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowFormulas</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowFormulas</b> (HSS <i>hSS</i> , BOOL FAR <i>*pShowFormulas</i> )  <i>hSS</i> is a handle to a view. <i>pShowFormulas</i> is the destination of the show formulas flag.
<b>Remarks</b>	If the show formulas flag is True, formula text is displayed in cells instead of the values formulas produce.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowFormulas</a> function
<b>Example</b>	<code>sserror = SSGetShowFormulas(Sheet1.SS, showformulas)</code>

## SSGetShowGridLines

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show grid lines flag.
<b>Syntax (VB)</b>	<b>SSGetShowGridLines%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowGridLines</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowGridLines</b> (HSS <i>hSS</i> , BOOL FAR <i>*pShowGridLines</i> )  <i>hSS</i> is a handle to a view. <i>pShowGridLines</i> is the destination of the show grid lines flag.
<b>Remarks</b>	If the show grid lines flag is True, grid lines are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowGridLines</a> function and <a href="#">ShowGridLines</a> property
<b>Example</b>	<code>sserror = SSGetShowGridLines(Sheet1.SS, gridlines)</code>

## SSGetShowHScrollBar

See also [A-Z Function Call List](#)

**Description** Returns the show horizontal scroll bar flag.

**Syntax (VB)** **SSGetShowHScrollBar**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, *pShowHScrollBar*%)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSGetShowHScrollBar** (HSS *hSS*, int FAR *\*pShowHScrollBar*)

*hSS* is a handle to a view.

*pShowHScrollBar* is the destination of the show horizontal scroll bar flag.

**Remarks** The show horizontal scroll bar flag has three settings. The following table lists the settings for this flag.

Setting	Description
---------	-------------

---

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

If the flag is 0, the horizontal scroll bar is hidden. If the flag is 1, the horizontal scroll bar is displayed. If the flag is 2, the horizontal scroll bar is displayed if the worksheet is wider than the window and the worksheet is active.

**Return Value** Integer

**See Also** [SSGetShowVScrollBar](#) and [SSSetShowHScrollBar](#) functions and [ShowHScrollBar](#) property

**Example** `sserror = SSGetShowHScrollBar(Sheet1.SS, hscrollbars)`



## SSGetShowRowHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show row heading flag.
<b>Syntax (VB)</b>	<b>SSGetShowRowHeading</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowRowHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowRowHeading</b> (HSS <i>hSS</i> , BOOL FAR <i>*pShowRowHeading</i> )  <i>hSS</i> is a handle to a view. <i>pShowRowHeading</i> is the destination of the show row heading flag.
<b>Remarks</b>	If the show row heading flag is True, row headings are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowRowHeading</a> function and <a href="#">ShowRowHeading</a> property
<b>Example</b>	<code>sserror = SSGetShowRowHeading(Sheet1.SS, rowheading)</code>

## SSGetShowSelections

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show selections flag.
<b>Syntax (VB)</b>	<b>SSGetShowSelections%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowSelections</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowSelections</b> (HSS <i>hSS</i> , int FAR <i>*pShowSelections</i> )  <i>hSS</i> is a handle to a view. <i>pShowSelections</i> is the destination of the show selections flag.
<b>Remarks</b>	If the show selections flag is True, selections are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowSelections</a> function and <a href="#">ShowSelections</a> property
<b>Example</b>	<code>sserror = SSGetShowSelections(Sheet1.SS, showselections)</code>

## SSGetShowVScrollBar

See also [A-Z Function Call List](#)

**Description** Returns the show vertical scroll bar flag.

**Syntax (VB)** **SSGetShowVScrollBar%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, *pShowVScrollBar%*)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSGetShowVScrollBar** (HSS *hSS*, int FAR *\*pShowVScrollBar*)

*hSS* is a handle to a view.

*pShowVScrollBar* is the destination of the show vertical scroll bar flag.

**Remarks** The show vertical scroll bar flag has three settings. The following table lists the settings for this flag.

Setting	Description
---------	-------------

---

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

If the flag is 0, the vertical scroll bar is hidden. If the flag is 1, the vertical scroll bar is displayed. If the flag is 2, the vertical scroll bar is displayed if the worksheet is taller than the window and the worksheet is active.

**Return Value** Integer

**See Also** [SSGetShowHScrollBar](#) and [SSSetShowVScrollBar](#) functions and [ShowVScrollBar](#) property

**Example** `sserror = SSGetShowVScrollBar(Sheet1.SS, vscrollbar)`

## SSGetShowZeroValues

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the show zero values flag.
<b>Syntax (VB)</b>	<b>SSGetShowZeroValues</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pShowZeroValues</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetShowZeroValues</b> (HSS <i>hSS</i> , BOOL FAR <i>*pShowZeroValues</i> )  <i>hSS</i> is a handle to a view. <i>pShowZeroValues</i> is the destination of the show zero values flag.
<b>Remarks</b>	If the show zero values flag is True, cells with zero values are displayed. If False, zero value cells are displayed as blanks.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetShowZeroValues</a> function
<b>Example</b>	<code>sserror = SSGetShowZeroValues(Sheet1.SS, zerovalues)</code>

## SSGetSSEdit

See also [A-Z Function Call List](#)

**Description** Returns the handle of the edit bar attached to the view.

**Syntax (VB)** **SSGetSSEdit%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pSSEdit*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSGetSSEdit** (HSS *hSS*, HSEEDIT \**pSSEdit*)

*hSS* is a handle to a view.

*pSSEdit* is the destination of the returned edit bar handle.

**Important** This function should not be called from Visual Basic.

---

**Return Value** Integer

**See Also** [SSSetSSEdit](#) function and [EditName](#) property

## SSGetText

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the text value of the active cell.
<b>Syntax (VB)</b>	<b>SSGetText</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSGetText (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view. <i>pBuf</i> is the destination buffer of the returned text. <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetText</b> returns the text value of the active cell. If the cell contains a formula, the result of the formula is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetTextRC</a> and <a href="#">SSSetText</a> functions and <a href="#">Text</a> property
<b>Example</b>	<pre>sserror = SSGetText(Sheet1.SS, textbuf\$, bufsize)</pre>

## SSGetTextRC

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the text value of the specified cell.
<b>Syntax (VB)</b>	<b>SSGetTextRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetTextRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell from which text is returned.  <i>pBuf</i> is the destination buffer of the returned text.  <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	<b>SSGetTextRC</b> returns the text value of the specified cell. If the cell contains a formula, the text result of the formula is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetText</a> , <a href="#">SSSetTextRC</a> functions and <a href="#">Text</a> property
<b>Example</b>	<pre>sserror = SSGetText(Sheet1.SS, 1, 1, textbuf\$, bufsize)</pre>

## SSGetTitle

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the title of the worksheet.
<b>Syntax (VB)</b>	<b>SSGetTitle</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pBuf</i> \$, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetTitle</b> (HSS <i>hSS</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>hSS</i> is a handle to a view.  <i>pBuf</i> is a string in which the title is returned. A title can be of any length. This string must be of sufficient length to hold the returned title.  <i>nBufSize</i> is the size of the string in which the title is returned. If the title is longer than <i>nBufSize</i> , an error is returned and an empty string is returned in <i>pBuf</i> .  <b>Important</b> Before calling this function, you must initialize a string to Space\$( <i>n</i> ), where <i>n</i> is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	The title of a worksheet can be used in external references to access multiple worksheets.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> and <a href="#">SSSetTitle</a> functions
<b>Example</b>	<pre>sserror = SSGetTitle(Sheet1.SS, titlebuf\$, bufsize)</pre>



## SSGetTopRow

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the top row displayed in the view.
<b>Syntax (VB)</b>	<b>SSGetTopRow</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, <i>pTopRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSGetTopRow</b> (HSS <i>hSS</i> , LPRC <i>pTopRow</i> )  <i>hSS</i> is a handle to a view. <i>pTopRow</i> is the destination of the top row value.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetTopRow</a> function and <a href="#">TopRow</a> property
<b>Example</b>	<code>sserror = SSGetTopRow(Sheet1.SS, toprow)</code>

## SSGetTypeRC

See also [A-Z Function Call List](#)

**Description** Returns the cell type of the specified cell.

**Syntax (VB)** **SSGetTypeRC**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow*%, ByVal *nCol*%, *pType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSGetTypeRC** (HSS *hSS*, RC *nRow*, RC *nCol*, int FAR *\*pType*);

*hSS* is a handle to a view.

*nRow* and *nCol* are the row and column numbers of the cell from which to return the type.

*pType* is the returned cell type.

**Remarks** The following table lists the cell types that can be returned.

<b>Value</b>	<b>Cell Type</b>
-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

**Return Value** Integer

**See Also** [Entry](#), [Formula](#), and [Text](#) properties

**Example** `sserror = SSGetTypeRC(Sheet1.SS, 1, 1, type)`

## SSGotoDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Goto dialog box.
<b>Syntax (VB)</b>	<b>SSGotoDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSGotoDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSGotoDlg</b> displays the Goto dialog box. This dialog box allows you to select the worksheet page to display.
<b>Return Value</b>	Integer
<b>Example</b>	<code>sserror = SSGotoDlg(Sheet1.SS)</code>

## SSInitTable

See also [A-Z Function Call List](#)

**Description**            Initializes a view.

**Syntax (VB)**            **SSInitTable**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)**        SSERROR SSEXPORTAPI **SSInitTable** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks**            SSInitTable initializes the worksheet attached to a view. If there is no worksheet attached to the view, a new worksheet is created. Use this function after calling **SSNew**.

**Return Value**        Integer

**See Also**            [SSDelete](#) and [SSNew](#) functions

## SSInsertRange

See also [A-Z Function Call List](#)

**Description** Inserts cells, rows, or columns in the specified range.

**Syntax (VB)** **SSInsertRange**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow1*%, ByVal *nCol1*%, ByVal *nRow2*%, ByVal *nCol2*%, ByVal *nShiftType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSInsertRange (HSS *hSS*, RC *nRow1*, RC *nCol1*, RC *nRow2*, RC *nCol2*, int *nShiftType*)

*hSS* is a handle to a view.

*nRow1*, *nCol1*, *nRow2*, and *nCol2* specify the range where cells, rows, or columns are inserted. If *nRow1* is -1, all rows are included in the selection; if *nCol1* is -1, all columns are included.

*nShiftType* determines how the insert should occur.

**Remarks** **SSInsertRange** inserts empty cells, rows, or columns from the given range. *nShiftType* specifies how the insert occurs. The following table lists the settings for *nShiftType*. These values are defined in VTSS.H and VTSS.TXT.

Setting	Number	Description
kShiftHorizontal	1	Cells of the specified range are shifted right to make room for the inserted cells.
kShiftVertical	2	Cells of the specified range are shifted down to make room for the inserted cells.
kShiftRows	3	Rows in which the specified range resides are shifted down to make room for the inserted cells.
kShiftCols	4	Columns in which the specified range resides are shifted right to make room for the inserted cells.

**Return Value** Integer

**See Also** [SSDeleteRange](#), [SSEditInsert](#), and [SSEditDelete](#) functions

**Example**  
`sserror = SSInsertRange(Sheet1.SS, 1, 1, 10, 10, kShiftHorizontal)`

## SSMaxCol

See also [A-Z Function Call List](#)

**Description** Returns the maximum number of columns supported by this version of VTSSDLL.DLL.

**Syntax (VB)** **SSMaxCol**% Lib "VTSSDLL.DLL" ()

**Syntax (VC++)** RC SSEXPORTAPI **SSMaxCol** ()

**Return Value** Integer

**See Also** [SSMaxRow](#) and [SSVersion](#) functions

**Example** `maxcol = SSMaxCol ()`

## SSMaxRow

See also [A-Z Function Call List](#)

**Description** Returns the maximum number of rows supported by this version of VTSSDLL.DLL.

**Syntax (VB)** **SSMaxRow**% Lib "VTSSDLL.DLL" ()

**Syntax (VC++)** RC SSEXPORTAPI **SSMaxRow** ()

**Return Value** Integer

**See Also** [SSMaxCol](#) and [SSVersion](#) functions

**Example** `maxrow = SSMaxRow()`

## SSMoveRange

See also [A-Z Function Call List](#)

<b>Description</b>	Moves a range.
<b>Syntax (VB)</b>	<b>SSMoveRange</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, ByVal <i>nRowOffset</i> %, ByVal <i>nColOffset</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSMoveRange (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , RC <i>nRowOffset</i> , RC <i>nColOffset</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> , and <i>nCol2</i> specify the source range. If <i>nRow1</i> is -1, all rows are included in the selection; if <i>nCol1</i> is -1, all columns are included.  <i>nRowOffset</i> and <i>nColOffset</i> specify the offset of the destination range from the source range.
<b>Remarks</b>	When <b>SSMoveRange</b> 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSEditCut</a> and <a href="#">SSEditPaste</a> functions
<b>Example</b>	<code>sserror = SSMoveRange(Sheet1.SS, 1, 1, 10, 10, 15, 15)</code>



## SSNew

See also [A-Z Function Call List](#)

<b>Description</b>	Creates a new worksheet view.
<b>Syntax (VB)</b>	<b>SSNew%</b> Lib "VTSSDLL.DLL" (ByVal <i>hWnd%</i> , <i>phSS&amp;</i> )
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSNew</b> (HWND <i>hWnd</i> , HSS FAR * <i>phSS</i> )  <i>hWnd</i> is the handle of the window used by the view. <i>phSS</i> is the destination of the returned view handle.
<b>Remarks</b>	<p><b>SSNew</b> creates a new worksheet view and returns a handle to it. This worksheet view does not have a worksheet to which it is attached. It must be attached to a worksheet before it can be used.</p> <p>To create a new worksheet and attach it to the view, call <b>SSInitTable</b>. To attach the view to a worksheet that was previously created, call <b>SSAttach</b> or <b>SSAttachToSS</b>. To read a worksheet from an Excel 4.0 file, call <b>SSRead</b>. To read a worksheet embedded in a file, call <b>SSReadIO</b>.</p> <p><b>Important</b> This function should not normally be called from Visual Basic.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> , <a href="#">SSAttachToSS</a> , <a href="#">SSDelete</a> , <a href="#">SSDeleteTable</a> , <a href="#">SSInitTable</a> , <a href="#">SSRead</a> , and <a href="#">SSReadIO</a> functions

## SSNextColPageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the next column where there is a page break.
<b>Syntax (VB)</b>	<b>SSNextColPageBreak</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nCol</i> %, <i>pNextCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSNextColPageBreak</b> (HSS <i>hSS</i> , RC <i>nCol</i> , LPC <i>pNextCol</i> )  <i>hSS</i> is a handle to a view. <i>nCol</i> is the starting column. <i>pNextCol</i> is set to the next column where there is a page break, or zero if there is no page break after <i>nCol</i> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddPageBreak</a> , <a href="#">SSAddRowPageBreak</a> , <a href="#">SSNextRowPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , <a href="#">SSRemovePageBreak</a> , and <a href="#">SSRemoveRowPageBreak</a> functions
<b>Example</b>	<pre>sserror = SSNextColPageBreak(Sheet1.SS, currcol, nextcol)</pre>

## SSNextRowPageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the next row where there is a page break.
<b>Syntax (VB)</b>	<b>SSNextRowPageBreak%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, <i>pNextRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSNextRowPageBreak</b> (HSS <i>hSS</i> , RC <i>nRow</i> , LPRC <i>pNextRow</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> is the starting row. <i>pNextRow</i> is set to the next row where there is a page break, or zero if there is no page break after <i>nRow</i> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddPageBreak</a> , <a href="#">SSAddRowPageBreak</a> , <a href="#">SSNextColPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , <a href="#">SSRemovePageBreak</a> , and <a href="#">SSRemoveRowPageBreak</a> functions
<b>Example</b>	<pre>sserror = SSNextRowPageBreak(sheet1.SS, currow, nextrow)</pre>

## SSOpenFileDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Open File dialog box.
<b>Syntax (VB)</b>	<b>SSOpenFileDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>pTitle</i> %, ByVal <i>hWndParent</i> %, ByVal <i>pBuf</i> %, ByVal <i>nBufSize</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSOpenFileDlg</b> (LPCSTR <i>pTitle</i> , HWND <i>hWndParent</i> , LPSTR <i>pBuf</i> , int <i>nBufSize</i> )  <i>pTitle</i> is the title of the dialog box. Use 0 for the default title. <i>hWndParent</i> is a handle to a parent window. <i>pBuf</i> is the destination buffer for the name of the file to open. <i>nBufSize</i> is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in <i>pBuf</i> <b>Important</b> Before calling this function, you must initialize a string to Space\$(n), where n is the <i>nBufSize</i> value you pass to the function.
<b>Remarks</b>	The Open File dialog box allows you to select a file to open.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSaveFileDlg</a> function

## SSProtectionDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Cell Protection dialog box.
<b>Syntax (VB)</b>	<b>SSProtectionDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSProtectionDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<p>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.</p> <p>After locking cells and hiding formulas, you must enable protection for the worksheet before cell locking and formula hiding is enabled. Protection for a worksheet is enabled using the <b>EnableProtection</b> property or the <b>SSSetEnableProtection</b> function call.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEnableProtection</a> , <a href="#">SSSetEnableProtection</a> , <a href="#">SSSetProtection</a> functions and <a href="#">EnableProtection</a> property
<b>Example</b>	<pre>sserror = SSProtectionDlg(Sheet1.SS)</pre>

## SSRangeToTwips

See also [A-Z Function Call List](#)

**Description** Determines the offset, width, and height of the specified range in twips.

**Syntax (VB)** **SSRangeToTwips**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow1*%, ByVal *nCol1*%, ByVal *nRow2*%, ByVal *nCol2*%, *pX*&, *pY*&, *pCX*&, *pCY*&, *pShown*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSRangeToTwips** (HSS *hSS*, RC *nRow1*, RC *nCol1*, RC *nRow2*, RC *nCol2*, LONG FAR \**pX*, LONG FAR \**pY*, LONG FAR \**pCX*, LONG FAR \**pCY*, int \**pShown*)

*hSS* is a handle to a view.

*nRow1*, *nCol1*, *nRow2*, and *nCol2* specify the range for which to find the offset, width, and height.

*X* is the returned horizontal offset of the range

*Y* is the returned vertical offset of the range.

*pCX* is the width of the range.

*pCY* is the height of the range.

*pShown* indicates whether the specified range is displayed, not displayed, or partially displayed in the worksheet. The following table lists the values returned by *pShown*.

Value	Description
0	Not shown
1	Shown
2	Partially shown

**Remarks** The coordinates returned by this function are measured in twips from the upper left corner of the worksheet control. The height and width of the range are also returned in twips.

Use **SSRangeToTwips** if you want to place a control or object in a worksheet at a specific range location.

**Return Value** Integer

**See Also** [SSTwipsToRC](#) function and [TopLeftChanged](#) event

**Example**  
`sserror = SSRangeToTwips(Sheet1.SS, 2, 2, 4, 2, xoffset, yoffset, xwidth, yheight, shown)`

## SSRead

**Description** Reads a worksheet from disk.

**Syntax (VB)** **SSRead**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pPathName*\$, *pFileType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSRead** (HSS *hSS*, LPCSTR *pPathName*, int FAR *\*pFileType*)

*hSS* is a handle to a view.

*pPathName* is a string containing the name of the file to read. The name can include drive, path, and file name.

*pFileType* returns the type of file that is read. The following table lists the values returned by this parameter.

Setting	Description
---------	-------------

1	Formula One format
---	--------------------

2	Excel 4.0 format
---	------------------

3	Tab-delimited text file
---	-------------------------

The parameter is undefined if **SSRead** returns an error.

**Remarks** **SSRead** initializes a worksheet structure and reads a worksheet from the specified file. If there is not a worksheet attached to the view, a new worksheet is created.

**Return Value** Integer

**See Also** [SSReadIO](#), [SSWrite](#), and [SSWriteIO](#) functions

**Example** `sserror = SSRead(Sheet1.SS, filename$, filetype)`

## SSReadIO

See also [A-Z Function Call List](#)

<b>Description</b>	Reads a worksheet using a user specified read function.
<b>Syntax (VB)</b>	<b>SSReadIO</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>dwUserData</i> &, ByVal <i>ioFunc</i> &, <i>pUserRet</i> &)
<b>Syntax (VC++)</b>	<b>SSERROR SSEXPORTAPI SSReadIO</b> (HSS <i>hSS</i> , DWORD <i>dwUserData</i> , IOFUNC <i>ioFunc</i> , DWORD FAR * <i>pUserRet</i> )  <i>hSS</i> is a handle to a view.  <i>dwUserData</i> is passed to <i>ioFunc</i> each time <i>ioFunc</i> is called.  <i>ioFunc</i> is the function called to read data from the worksheet. It takes the following form:  <pre>typedef DWORD (FAR PASCAL *IOFUNC)(DWORD dwUserData, LPVOID p, UINT nBytes);</pre> <i>pUserRet</i> returns the last value returned by <i>ioFunc</i> . If this pointer is not null, it returns the last value returned by <i>ioFunc</i> . Any non-zero value returned by <i>ioFunc</i> causes reading to fail immediately.
<b>Remarks</b>	<b>SSReadIO</b> is the same as <b>SSRead</b> except that <i>ioFunc</i> is called to read data instead of reading from a specified file. If a worksheet is not attached to the view, a new worksheet is created. If <i>ioFunc</i> returns a non-zero value, the value <i>ioFunc</i> returned is returned by <b>SSReadIO</b> in <i>pUserRet</i> . If the file is successfully read, 0 is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRead</a> , <a href="#">SSWrite</a> , and <a href="#">SSWriteIO</a> functions



## SSRecalc

See also [A-Z Function Call List](#)

<b>Description</b>	Recalculates the worksheet attached to a view.
<b>Syntax (VB)</b>	<b>SSRecalc</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSRecalc</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<b>SSRecalc</b> recalculates all formulas in the worksheet attached to the specified view.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetAutoRecalc</a> function and <a href="#">AutoRecalc</a> property
<b>Example</b>	<pre>sserror = SSRecalc(Sheet1.SS)</pre>

## SSRemoveColPageBreak

See also [A-Z Function Call List](#)

**Description** Removes a vertical page break adjacent to the left edge of the specified column.

**Syntax (VB)** **SSRemoveColPageBreak**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nCol*%)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSRemoveColPageBreak** (HSS *hSS*, RC *nCol*)

*hSS* is a handle to a view.

*nCol* is the column where the page break is removed.

**Return Value** Integer

**See Also** [SSAddColPageBreak](#), [SSAddPageBreak](#), [SSAddRowPageBreak](#), [SSNextColPageBreak](#), [SSNextRowPageBreak](#), [SSRemovePageBreak](#), and [SSRemoveRowPageBreak](#) functions

**Example** `sserror = SSRemoveColPageBreak(Sheet1.SS, 2)`

## SSRemovePageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Removes page breaks adjacent to the active cell.
<b>Syntax (VB)</b>	<b>SSRemovePageBreak</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSRemovePageBreak</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddPageBreak</a> , <a href="#">SSAddRowPageBreak</a> , <a href="#">SSNextColPageBreak</a> , <a href="#">SSNextRowPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , and <a href="#">SSRemoveRowPageBreak</a> functions
<b>Example</b>	<pre>sserror = SSRemovePageBreak(Sheet1.SS)</pre>

## SSRemoveRowPageBreak

See also [A-Z Function Call List](#)

<b>Description</b>	Removes a horizontal page break adjacent to the top edge of the specified row.
<b>Syntax (VB)</b>	<b>SSRemoveRowPageBreak%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSRemoveRowPageBreak</b> (HSS <i>hSS</i> , RC <i>nRow</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> is the row where the page break is removed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAddColPageBreak</a> , <a href="#">SSAddPageBreak</a> , <a href="#">SSAddRowPageBreak</a> , <a href="#">SSNextColPageBreak</a> , <a href="#">SSNextRowPageBreak</a> , <a href="#">SSRemoveColPageBreak</a> , and <a href="#">SSRemovePageBreak</a> functions
<b>Example</b>	<code>sserror = SSRemoveRowPageBreak(Sheet1.SS, 2)</code>

## SSRowHeightDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Row Height dialog box.
<b>Syntax (VB)</b>	<b>SSRowHeightDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSRowHeightDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetRowHeight</a> and <a href="#">SSSetRowHeightAuto</a> functions.
<b>Example</b>	<code>sserror = SSRowHeightDlg(Sheet1.SS)</code>

## SSSaveFileDialog

See also [A-Z Function Call List](#)

**Description** Displays the Save As dialog box.

**Syntax (VB)** **SSSaveFileDialog%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pTitle*\$, ByVal *pBuf*\$, ByVal *nBufSize*%, *pFileType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSaveFileDialog** (HSS *hSS*, LPCSTR *pTitle*, LPSTR *pBuf*, int *nBufSize*, int FAR \**pFileType*)

*hSS* is a handle to a view.

*pTitle* is the title of the dialog box. Use 0 for the default title.

*pBuf* is the destination buffer for the name by which the worksheet is saved.

*nBufSize* is the maximum buffer size. If the returned string is larger than the buffer, an error is returned and an empty string is placed in *pBuf*.

*pFileType* is the file type used when saving the file. The following table lists the settings for this parameter.

Setting	Description
---------	-------------

1	Formula One format
---	--------------------

2	Excel 4.0 format
---	------------------

**Important** Before calling this function, you must initialize a string to Space\$(n), where n is the *nBufSize* value you pass to the function.

**Remarks** The Save As dialog box allows you to save and name a file.

**Return Value** Integer

**See Also** [SSOpenFileDialog](#) function

## SSSaveWindowInfo

See also [A-Z Function Call List](#)

**Description** Saves the window specific information from a view to its worksheet.

**Syntax (VB)** **SSSaveWindowInfo**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSaveWindowInfo** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** Window specific information from the view must be saved to its worksheet if the information is to be saved the next time the worksheet is written to disk.

The following table lists the window information that is saved.

### Saved information

---

AllowArrows	DataRowMode	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

**Return Value** Integer

**See Also** [SSWrite](#) and [SSWriteIO](#) Functions

**Example** `sserror = SSSaveWindowInfo(Sheet1.SS)`

## SSSetActiveCell

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the active cell to the specified row and column.
<b>Syntax (VB)</b>	<b>SSSetActiveCell</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetActiveCell</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> and <i>nCol</i> are the row and column numbers of the new active cell.
<b>Remarks</b>	The active cell is the cell in which data is entered or edited if the user starts typing. When <b>SSSetActiveCell</b> is called, the active cell becomes the cell specified by this function. If this cell is within a selection only the active cell changes.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetActiveCell</a> function and <a href="#">Col</a> and <a href="#">Row</a> properties
<b>Example</b>	<pre>sserror = SSSetActiveCell(Sheet1.SS, 1, 1)</pre>



## SSSetAlignment

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the data alignment for a selection.																																														
<b>Syntax (VB)</b>	<code>SSSetAlignment% Lib "VTSSDLL.DLL" (ByVal <i>hSS</i>&amp;, ByVal <i>nHorizontal</i>%, ByVal <i>bWordWrap</i>%, ByVal <i>nVertical</i>%, ByVal <i>nOrientation</i>%)</code>																																														
<b>Syntax (VC++)</b>	<code>SSERROR SSEXPORTAPI SSSetAlignment (HSS <i>hSS</i>, int <i>nHorizontal</i>, BOOL <i>bWordWrap</i>, int <i>nVertical</i>, int <i>nOrientation</i>)</code>  <i>hSS</i> is a handle to a view. <i>nHorizontal</i> specifies the horizontal alignment. <i>bWordWrap</i> specifies whether word wrap is enabled. <i>nVertical</i> specifies the vertical alignment. <i>nOrientation</i> specifies the text orientation. (Not implemented in this version.)																																														
<b>Remarks</b>	<b>SSSetAlignment</b> sets the alignment and word wrap for data in the current selection. The following table lists the settings and constants for <i>nHorizontal</i> . <table><thead><tr><th>Setting</th><th>Description</th><th>Constants</th></tr></thead><tbody><tr><td>1</td><td>General</td><td>kHAlignGeneral</td></tr><tr><td>2</td><td>Left</td><td>kHAlignLeft</td></tr><tr><td>3</td><td>Center</td><td>kHAlignCenter</td></tr><tr><td>4</td><td>Right</td><td>kHAlignRight</td></tr><tr><td>5</td><td>Fill</td><td>kHAlignFill</td></tr><tr><td>6</td><td>Justify</td><td>kHAlignJustify</td></tr><tr><td>7</td><td>Center across cells</td><td>kHAlignCenterAcrossCells</td></tr></tbody></table> The following table lists the settings for <i>nVertical</i> . <table><thead><tr><th>Setting</th><th>Description</th><th>Constants</th></tr></thead><tbody><tr><td>1</td><td>Top</td><td>kVAlignTop</td></tr><tr><td>2</td><td>Center</td><td>kVAlignCenter</td></tr><tr><td>3</td><td>Bottom</td><td>kVAlignBottom</td></tr></tbody></table> The following table lists the settings for <i>nOrientation</i> . <table><thead><tr><th>Setting</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Horizontal</td></tr><tr><td>1</td><td>Vertical</td></tr><tr><td>2</td><td>Upward</td></tr><tr><td>3</td><td>Downward</td></tr></tbody></table>	Setting	Description	Constants	1	General	kHAlignGeneral	2	Left	kHAlignLeft	3	Center	kHAlignCenter	4	Right	kHAlignRight	5	Fill	kHAlignFill	6	Justify	kHAlignJustify	7	Center across cells	kHAlignCenterAcrossCells	Setting	Description	Constants	1	Top	kVAlignTop	2	Center	kVAlignCenter	3	Bottom	kVAlignBottom	Setting	Description	0	Horizontal	1	Vertical	2	Upward	3	Downward
Setting	Description	Constants																																													
1	General	kHAlignGeneral																																													
2	Left	kHAlignLeft																																													
3	Center	kHAlignCenter																																													
4	Right	kHAlignRight																																													
5	Fill	kHAlignFill																																													
6	Justify	kHAlignJustify																																													
7	Center across cells	kHAlignCenterAcrossCells																																													
Setting	Description	Constants																																													
1	Top	kVAlignTop																																													
2	Center	kVAlignCenter																																													
3	Bottom	kVAlignBottom																																													
Setting	Description																																														
0	Horizontal																																														
1	Vertical																																														
2	Upward																																														
3	Downward																																														
<b>Return Value</b>	Integer																																														
<b>See Also</b>	<a href="#">SSFormatAlignmentDlg</a> function																																														
<b>Example</b>	<code>sserror = SSSetAlignment(Sheet1.SS, 4, False, 3, 0)</code>																																														

## SSSetAllowArrows

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether the arrow keys can reposition the active cell.
<b>Syntax (VB)</b>	<b>SSSetAllowArrows%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowArrows</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetAllowArrows (HSS <i>hSS</i> , BOOL <i>bAllowArrows</i> )  <i>hSS</i> is a handle to a view. <i>bAllowArrows</i> specifies the setting of the allow arrows flag.
<b>Remarks</b>	<b>SSSetAllowArrows</b> sets the allow arrows flag. If the flag is True, the arrow keys on your keyboard can move the active cell in the worksheet. By default, the allow arrows flag is True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowArrows</a> function and <a href="#">AllowArrows</a> property
<b>Example</b>	<pre>sserror = SSSetAllowArrows(Sheet1.SS, False)</pre>

## SSSetAllowDelete

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether the delete key deletes records and clears selections.
<b>Syntax (VB)</b>	<b>SSSetAllowDelete</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowDelete</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetAllowDelete (HSS <i>hSS</i> , BOOL <i>bAllowDelete</i> )  <i>hSS</i> is a handle to a view. <i>bAllowDelete</i> specifies the setting of the allow delete flag.
<b>Remarks</b>	<b>SSSetAllowDelete</b> sets the allow delete flag. If the flag is True and data browsing mode is enabled, 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.  If the flag is False, the delete key does not delete records or clear selections. By default, the allow delete flag is True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowDelete</a> function and <a href="#">AllowDelete</a> property
<b>Example</b>	<pre>sserror = SSSetAllowDelete(Sheet1.SS, False)</pre>

## SSSetAllowEditHeaders

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether row, column, and top left headers can be edited.
<b>Syntax (VB)</b>	<b>SSSetAllowEditHeaders%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowEditHeaders</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAllowEditHeaders</b> (HSS <i>hSS</i> , BOOL <i>bAllowEditHeaders</i> )  <i>hSS</i> is a handle to a view.  <i>bAllowEditHeaders</i> specifies the setting of the edit headers flag.
<b>Remarks</b>	<p>If the edit headers flag is True, the names displayed in row, column, and top left headers can be edited by double clicking the header to be edited. The Header Name dialog box is displayed, allowing you to enter a new header name.</p> <p>If the flag is False, editing of headers is not allowed; a <b>DbClick</b> event is passed when a header is double clicked.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowEditHeaders</a> function, <a href="#">AllowEditHeaders</a> property, and <a href="#">DbClick</a> event.
<b>Example</b>	<pre>sserror = SSSetAllowEditHeaders(Sheet1.SS, False)</pre>

## SSSetAllowFillRange

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether filling by dragging a range is allowed.
<b>Syntax (VB)</b>	<b>SSSetAllowFillRange</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowFillRange</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAllowFillRange</b> (HSS <i>hSS</i> , BOOL <i>bAllowFillRange</i> )  <i>hSS</i> is a handle to a view. <i>bAllowFillRange</i> specifies the setting of the fill range flag.
<b>Remarks</b>	<b>SSSetAllowFillRange</b> sets the fill range flag. If the flag is True, filling ranges by dragging a selection is allowed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowFillRange</a> function and <a href="#">AllowFillRange</a> property
<b>Example</b>	<pre>sserror = SSSetAllowFillRange(Sheet1.SS, True)</pre>

## SSSetAllowFormulas

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether the user is allowed to enter formulas.
<b>Syntax (VB)</b>	<b>SSSetAllowFormulas</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowFormulas</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetAllowFormulas (HSS <i>hSS</i> , BOOL <i>bAllowFormulas</i> )  <i>hSS</i> is a handle to a view.  <i>bAllowFormulas</i> specifies the setting of the user formula flag.
<b>Remarks</b>	<b>SSSetAllowFormulas</b> sets the user formula flag. If the flag is True, formulas can be added by the user at run-time.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowFormulas</a> function and <a href="#">AllowFormulas</a> property
<b>Example</b>	<code>sserror = SSSetAllowFormulas(Sheet1.SS, True)</code>

## SSSetAllowInCellEditing

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether in-cell editing is allowed.
<b>Syntax (VB)</b>	<b>SSSetAllowInCellEditing%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowInCellEditing</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAllowInCellEditing</b> (HSS <i>hSS</i> , BOOL <i>bAllowInCellEditing</i> )  <i>hSS</i> is a handle to a view. <i>bAllowInCellEditing</i> specifies the setting of the in-cell editing flag.
<b>Remarks</b>	<b>SSSetAllowInCellEditing</b> sets the in-cell editing flag. If the flag is True, data can be entered and edited directly in the cell without using an edit bar.  If a double click event is defined for a control, the blinking cursor does not appear when you invoke edit mode for in-cell editing.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowInCellEditing</a> function and <a href="#">AllowInCellEditing</a> property
<b>Example</b>	<pre>sserror = SSSetAllowInCellEditing(Sheet1.SS, True)</pre>

## SSSetAllowMoveRange

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether moving ranges by dragging is allowed.
<b>Syntax (VB)</b>	<b>SSSetAllowMoveRange</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowMoveRange</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAllowMoveRange</b> (HSS <i>hSS</i> , BOOL <i>bAllowMoveRange</i> )  <i>hSS</i> is a handle to a view. <i>bAllowMoveRange</i> specifies the setting of the move range flag.
<b>Remarks</b>	<b>SSSetAllowMoveRange</b> sets the move range flag. If the flag is True, moving ranges by dragging a cell is allowed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowMoveRange</a> function and <a href="#">AllowMoveRange</a> property
<b>Example</b>	<pre>sserror = SSSetAllowMoveRange(Sheet1.SS, True)</pre>



## SSSetAllowResize

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether resizing rows and columns by dragging is allowed.
<b>Syntax (VB)</b>	<b>SSSetAllowResize</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowResize</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetAllowResize</b> (HSS <i>hSS</i> , BOOL <i>bAllowResize</i> )  <i>hSS</i> is a handle to a view. <i>bAllowResize</i> specifies the setting of the resize flag.
<b>Remarks</b>	<b>SSSetAllowResize</b> sets the resize flag. If the flag is True, the size of rows and columns can be set by dragging row and column heading borders.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowResize</a> function and <a href="#">AllowResize</a> property
<b>Example</b>	<code>sserror = SSSetAllowResize(Sheet1.SS, True)</code>

## SSSetAllowSelections

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether selecting ranges is allowed.
<b>Syntax (VB)</b>	<b>SSSetAllowSelections%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAllowSelections</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAllowSelections</b> (HSS <i>hSS</i> , BOOL <i>bAllowSelections</i> )  <i>hSS</i> is a handle to a view. <i>bAllowSelections</i> specifies the setting of the select range flag.
<b>Remarks</b>	<b>SSSetAllowSelections</b> sets the select range flag. If the flag is True, ranges can be selected with the keyboard and by clicking and dragging with the mouse.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAllowSelections</a> function and <a href="#">AllowSelections</a> property
<b>Example</b>	<pre>sserror = SSSetAllowSelections(Sheet1.SS, True)</pre>

## SSSetAllowTabs

See also [A-Z Function Call List](#)

**Description** Specifies whether the tab key can reposition the active cell in a selected range.

**Syntax (VB)** **SSSetAllowTabs**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *bAllowTabs*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetAllowTabs** (HSS *hSS*, BOOL *bAllowTabs*)

*hSS* is a handle to a view.

*bAllowTabs* specifies the setting of the allow tabs flag.

**Remarks** **SSSetAllowTabs** sets the allow tabs flag. If the flag is True, the tab key can 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.

**Return Value** Integer

**See Also** [SSGetAllowTabs](#) function and [AllowTabs](#) property

**Example** `sserror = SSSetAllowTabs(Sheet1.SS, False)`

## SSSetName

See also [A-Z Function Call List](#)

**Description** Specifies the application name that is displayed in the title bar of error dialog boxes.

**Syntax (VB)** **SSSetName%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pAppName*\$)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSSetName**(HSS *hSS*, LPCSTR *pAppName*)

*hSS* is a handle to a view.

*pAppName* specifies the application name to be displayed.

**Remarks** Only the name displayed in the title bar of error dialog boxes is affected by this function. Other dialog boxes display functional names (e.g., Alignment, Custom Format, Font)

**Return Value** Integer

**Example** `sserror = SSSetName(Sheet1.SS, "Application Name")`

## SSSetAutoRecalc

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether automatic recalculation is enabled.
<b>Syntax (VB)</b>	<b>SSSetAutoRecalc</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bAutoRecalc</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetAutoRecalc</b> (HSS <i>hSS</i> , BOOL <i>bAutoRecalc</i> )  <i>hSS</i> is a handle to a view. <i>bAutoRecalc</i> specifies the setting of the automatic recalc flag.
<b>Remarks</b>	<p><b>SSSetAutoRecalc</b> sets the automatic recalc flag. If the flag is True, the worksheet is recalculated if needed. Thereafter, any change to the worksheet causes all formulas to be recalculated.</p> <p>You may notice that the worksheet is not recalculated immediately after each change you make from your program. To force the worksheet to be recalculated immediately, call <b>SSSetAutoRecalc</b> with <i>bAutoRecalc</i> set to True. The worksheet is recalculated immediately, if needed.</p> <p>Calling <b>SSGetText</b>, <b>SSGetTextRC</b>, <b>SSGetNumber</b>, <b>SSGetNumberRC</b>, <b>SSGetFormattedText</b>, <b>SSGetFormattedTextRC</b>, <b>SSGetTypeRC</b> and <b>SSGetLogicalRC</b> also causes the worksheet to be recalculated immediately, if needed.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetAutoRecalc</a> , <a href="#">SSRecalc</a> , and <a href="#">SSUpdate</a> functions and <a href="#">AutoRecalc</a> property
<b>Example</b>	<pre>sserror = SSSetAutoRecalc(Sheet1.SS, True)</pre>

## SSSetBackColor

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the background color of the worksheet.
<b>Syntax (VB)</b>	<b>SSSetBackColor</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>crBackColor</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetBackColor</b> (HSS <i>hSS</i> , COLORREF <i>crBackColor</i> )  <i>hSS</i> is a handle to a view.  <i>crBackColor</i> is an RGB background color. This value can be one of the following:



**Normal RGB Colors.** These colors are specified using the color palette, or by using the RGB or QBColor functions.



**System default colors.** System color constants are specified in the Visual Basic CONSTANT.TXT file.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

<b>Remarks</b>	All cells within the view are set to the background color except those with patterns.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetBackColor</a> function and <a href="#">BackColor</a> property
<b>Example</b>	<pre>sserror = SSSetBackColor(Sheet1.SS, 2)</pre>

## SSSetBorder

See also [A-Z Function Call List](#)

**Description** Specifies the border for all selected cells.

**Syntax (VB)** **SSSetBorder**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nOutline*%, ByVal *nLeft*%, ByVal *nRight*%, ByVal *nTop*%, ByVal *nBottom*%, ByVal *nShade*%, ByVal *crOutline*&, ByVal *crLeft*&, ByVal *crRight*&, ByVal *crTop*&, ByVal *crBottom*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetBorder** (HSS *hSS*, int *nOutline*, int *nLeft*, int *nRight*, int *nTop*, int *nBottom*, int *nShade*, COLORREF *crOutline*, COLORREF *crLeft*, COLORREF *crRight*, COLORREF *crTop*, COLORREF *crBottom*)

*hSS* is a handle to a view.

*nOutline* specifies 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*, and *nBottom* specify the border type for the sides of cells in the selected range.

*nShade* specifies the border shading; it must correspond to the built-in shades (not implemented in this version).

*crOutline* specifies the color of the outline border. This is an RGB color. It is translated into one of the 16 colors in the color palette.

*crLeft*, *crRight*, *crTop*, and *crBottom* specify the colors of the cell border sides. This is an RGB color. It is translated into one of the 16 colors in the color palette.

**Remarks** **SSSetBorder** determines the border for all selected cells. The following table lists the border and outline settings.

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

	Q1	Q2	Q3	Q4
<b>Northern</b>	<b>29,434.39</b>	<b>25,553.89</b>	<b>26,014.06</b>	<b>25,498.30</b>
Allen	4,563.89	8,934.10	7,674.37	9,979.86
Jackson	5,373.33	5,337.85	3,181.70	1,573.39
Simson	9,596.95	3,218.38	4,848.34	4,011.16
Thomas	5,956.68	1,285.58	6,017.39	5,554.49
White	3,943.53	6,777.99	4,292.26	4,379.39
<b>Southeast</b>	<b>14,531.42</b>	<b>32,001.74</b>	<b>23,280.86</b>	<b>31,416.64</b>
Brooks	4,362.58	9,344.59	531.27	8,659.98
Carter	1,898.27	4,525.52	8,991.83	8,290.25
Hayes	4,232.79	345.66	5,717.63	4,255.74
Murphy	531.53	9,457.86	6,263.57	6,245.85
Robbins	3,506.24	8,328.11	1,776.56	3,964.82
<b>Pacific</b>	<b>18,954.05</b>	<b>31,857.46</b>	<b>36,033.48</b>	<b>27,517.22</b>
Dunn	6,707.33	8,505.52	4,919.86	3,654.67
Fisher	108.22	2,299.30	8,111.16	4,035.09
Johnson	8,908.24	4,319.61	4,269.58	7,731.57
Newton	859.31	9,640.45	8,792.17	4,575.33
Vaughn	2,370.95	7,092.58	9,940.72	7,520.56
<b>Totals</b>	<b>62,919.86</b>	<b>89,413.09</b>	<b>85,328.40</b>	<b>84,432.16</b>

*This range has a double line border placed along the top and bottom of the range. The first three ranges of data in this worksheet each have a thick outline border.*

**Return Value**

Integer

**See Also**

[SSFormatBorderDlg](#) function

**Example**

```
thin = 1
thick = 5
shade = 0
acolor = RGB(255, 0, 255)

sserror = SSSetBorder(form1.Sheet1.SS, thick, thin, thin, thin,
thin, shade, acolor, acolor, acolor, acolor, acolor)
```



## SSSetColText

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the name for a column.
<b>Syntax (VB)</b>	<b>SSSetColText</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nCol</i> %, ByVal <i>pColText</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetColText</b> (HSS <i>hSS</i> , RC <i>nCol</i> , LPCSTR <i>pColText</i> )  <i>hSS</i> is a handle to a view. <i>nCol</i> is the column to be named. <i>pColText</i> is the new column name.
<b>Remarks</b>	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 new 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetRowText</a> and <a href="#">SSSetTopLeftText</a> functions
<b>Example</b>	<pre>sserror = SSSetColText(Sheet1.SS, 1, "Sales")</pre>

## SSSetColWidth

See also [A-Z Function Call List](#)

<b>Description</b>	Determines the width of the specified columns.
<b>Syntax (VB)</b>	<b>SSSetColWidth%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nCol1</i> %, ByVal <i>nCol2</i> %, ByVal <i>nWidth</i> %, ByVal <i>bDefColWidth</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetColWidth</b> (HSS <i>hSS</i> , RC <i>nCol1</i> , RC <i>nCol2</i> , int <i>nWidth</i> , BOOL <i>bDefColWidth</i> )  <i>hSS</i> is a handle to a view. <i>nCol1</i> specifies the starting column to change. <i>nCol2</i> specifies the ending column to change. <i>nWidth</i> is the new column width. <i>bDefColWidth</i> specifies whether the default column width is changed. True specifies that the default width is set to <i>nWidth</i> , 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.
<b>Remarks</b>	<b>SSSetColWidth</b> specifies the width of the specified columns. The width is specified by <i>nWidth</i> in units equal to 1/256th of an average character's width in the default font.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSColWidthDlg</a> , <a href="#">SSGetColWidth</a> , and <a href="#">SSSetColWidthAuto</a> functions
<b>Example</b>	<pre>sserror = SSSetColWidth(Sheet1.SS, 1, 10, (5*256), False)</pre>

## SSSetColWidthAuto

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the widths of the specified columns automatically.
<b>Syntax (VB)</b>	<b>SSSetColWidthAuto%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, ByVal <i>bSetDefaults</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetColWidthAuto</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , BOOL <i>bSetDefaults</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> , and <i>nCol2</i> specify the range containing the columns for which to set the width.  <i>bSetDefaults</i> determines when the specified columns are resized. If True, all specified columns are adjusted automatically. If False, only columns in the specified column range that need to be larger than their current size are adjusted.
<b>Remarks</b>	<b>SSSetColWidthAuto</b> 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSColWidthDlg</a> , <a href="#">SSGetColWidth</a> , and <a href="#">SSSetColWidth</a> functions
<b>Example</b>	<pre>sserror = SSSetColWidthAuto(Sheet1.SS, 1, 1, 5, 6, False)</pre>

## SSSetDefinedName

See also [A-Z Function Call List](#)

<b>Description</b>	Defines or changes a user-defined name.
<b>Syntax (VB)</b>	<b>SSSetDefinedName</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pName</i> \$, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetDefinedName</b> (HSS <i>hSS</i> , LPCSTR <i>pName</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view.  <i>pName</i> is the user defined name.  <i>pFormula</i> is the formula that describes the item to which <i>pName</i> refers (e.g., "A1:C3"). The formula should not contain a leading equal sign (=).
<b>Remarks</b>	<b>SSSetDefinedName</b> allows a user-defined name to be defined or changed. A name can refer to a cell, a group of cells, a value, or a formula.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSDeleteDefinedName</a> , <a href="#">SSDefinedNameDlg</a> , and <a href="#">SSGetDefinedName</a> functions
<b>Example</b>	<pre>sserror = SSSetDefinedName(Sheet1.SS, "Gross_Sales", "\$C\$1:\$C\$20")</pre>

## SSSetDefWindowProc

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the default window procedure for a worksheet view.
<b>Syntax (VB)</b>	<b>SSSetDefWindowProc%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pWindowProc</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetDefWindowProc</b> (HSS <i>hSS</i> , WNDPROC <i>pWindowProc</i> )  <i>hSS</i> is a handle to a view. <i>pWindowProc</i> is a standard Windows procedure parameter.
<b>Remarks</b>	If a message is passed to <b>SSCallWindowProc</b> and it is not needed by the worksheet view, it is passed to the default window procedure.  <b>Important</b> This function should not be called from Visual Basic.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCallWindowProc</a> function

## SSSetDoSetCursor

See also [A-Z Function Call List](#)

<b>Description</b>	Determines how the cursor is set.
<b>Syntax (VB)</b>	<b>SSSetDoSetCursor%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bDoSetCursor</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetDoSetCursor</b> (HSS <i>hSS</i> , BOOL <i>bDoSetCursor</i> )  <i>hSS</i> is a handle to a view.  <i>bDoSetCursor</i> specifies the setting of the set cursor flag.
<b>Remarks</b>	<b>SSSetDoSetCursor</b> sets the set cursor flag. If the flag is True, the spreadsheet sets the cursor normally. If the flag is False, the spreadsheet never sets the cursor and passes WM_SETCURSOR messages to the default window procedure.  The set cursor flag is not saved to disk. The default set cursor setting for a new view is always True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">MousePointer</a> property
<b>Example</b>	<pre>sserror = SSSetDoSetCursor(Sheet1.SS, True)</pre>

## SSSetEnableProtection

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether protection is enabled for a worksheet.
<b>Syntax (VB)</b>	<b>SSSetEnableProtection</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bEnableProtection</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetEnableProtection</b> (HSS <i>hSS</i> , BOOL <i>bEnableProtection</i> )  <i>hSS</i> is a handle to a view. <i>bEnableProtection</i> specifies the setting of the enable protection flag.
<b>Remarks</b>	If the enable protection flag is True, protection is activated for cells that have been locked and formulas that have been hidden in the worksheet. Cells can be marked as locked and formulas marked as hidden using the <b>SSSetProtection</b> and <b>SSProtectionDlg</b> function calls.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEnableProtection</a> , <a href="#">SSProtectionDlg</a> , and <a href="#">SSSetProtection</a> functions and <a href="#">EnableProtection</a> property
<b>Example</b>	<pre>sserror = SSSetEnableProtection(Sheet1.SS, True)</pre>

## SSSetEnterMovesDown

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether protection is enabled for a worksheet.
<b>Syntax (VB)</b>	<b>SSSetEnterMovesDown%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bEnterMovesDown</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetEnterMovesDown</b> (HSS <i>hSS</i> , BOOL <i>bEnterMovesDown</i> )  <i>hSS</i> is a handle to a view. <i>bEnterMovesDown</i> specifies the setting of the enter moves down flag.
<b>Remarks</b>	If the enter moves down flag is True, the enter key moves the active cell down to the next row, even if no range is selected. If False, the enter key does not advance the active cell.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEnterMovesDown</a> function
<b>Example</b>	<pre>sserror = SSSetEnterMovesDown(Sheet1.SS, False)</pre>



## SSSetEntry

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the value of the active cell based on the entry format.
<b>Syntax (VB)</b>	<b>SSSetEntry</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pEntry</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetEntry</b> (HSS <i>hSS</i> , LPCSTR <i>pEntry</i> )  <i>hSS</i> is a handle to a view. <i>pEntry</i> is a string containing the value to put in the cell.
<b>Remarks</b>	SSSetEntry 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetEntryRC</a> and <a href="#">SSGetEntry</a> functions and <a href="#">Entry</a> and <a href="#">Text</a> properties
<b>Example</b>	<pre>sserror = SSSetEntry(Sheet1.SS, "10%")</pre>

## SSSetEntryRC

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the value of the specified cell based on the entry format.
<b>Syntax (VB)</b>	<b>SSSetEntryRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pEntry</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetEntryRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPCSTR <i>pEntry</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell in which a value is entered.  <i>pEntry</i> is a string containing the value to put in the cell.
<b>Remarks</b>	<b>SSSetEntryRC</b> 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetEntry</a> and <a href="#">SSGetEntryRC</a> functions and <a href="#">Entry</a> and <a href="#">Text</a> properties
<b>Example</b>	<pre>sserror = SSSetEntryRC(Sheet1.SS, 1, 1, "10%")</pre>

## SSSetExtraColor

See also [A-Z Function Call List](#)

- Description** Specifies the color of the view area outside the worksheet.
- Syntax (VB)** **SSSetExtraColor**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *crExtraColor*&)
- Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetExtraColor** (HSS *hSS*, COLORREF *crExtraColor*)
- hSS* is a handle to a view.
- crExtraColor* is an RGB color. This value can be one of the following:



**Normal RGB Colors.** These colors are specified using the color palette, or by using the RGB or QBColor functions.



**System default colors.** System color constants are specified in the Visual Basic CONSTANT.TXT file.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF).

- Remarks** **SSSetExtraColor** sets the extra color. The extra color fills the space in the view window not covered by the worksheet. This space occurs when the worksheet is smaller than the view window.
- Return Value** Integer
- See Also** [SSGetExtraColor](#) function and [ExtraColor](#), [MaxCol](#), and [MaxRow](#) properties
- Example** `sserror = SSSetExtraColor(Sheet1.SS, extracolor)`

## SSSetFireEvent

See also [A-Z Function Call List](#)

**Description** Determines if an event can be fired.

**Syntax (VB)** **SSSetFireEvent**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nEvent*%, ByVal *bFireIt* %)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetFireEvent** (HSS *hSS*, UINT *nEvent*, BOOL *bFireIt*)

*hSS* is a handle to a view.

*nEvent* specifies the event on which to operate.

*bFireIt* determines if the event can be fired.

**Remarks** **SSSetFireEvent** sets an event flag to determine if the given event can be fired. If *bFireIt* is True, the event is enabled and can be fired; False disables the event. If you are calling these functions from C/C++ and not using the Visual Basic VBX, *bFireIt* is False by default. The Visual Basic VBX sets this value to True when a new view is created.

The following table lists the events defined in VTSS.TXT and VTSS.H that can be affected by this function call.

<b>Event</b>	<b>Description</b>
SSM_SELCHANGE	Selection changes
SSM_STARTEDIT	Edit mode is entered
SSM_ENDEDIT	Edit mode is exited
SSM_STARTRECALC	Recalc is started
SSM_ENDRECALC	Recalc ends
SSM_CLICK	Click
SSM_DBLCLICK	Double Click
SSM_CANCELEDIT	Edit mode is canceled

**Return Value** Integer

**See Also** [SSGetFireEvent](#) function

**Example** `sserror = SSSetFireEvent(Sheet1.SS, SSM_SelChange, True)`

## SSSetFixedCols

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the number of fixed columns.
<b>Syntax (VB)</b>	<b>SSSetFixedCols</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nCol1</i> %, ByVal <i>nCols</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetFixedCols</b> (HSS <i>hSS</i> , RC <i>nCol1</i> , RC <i>nCols</i> )  <i>hSS</i> is a handle to a view. <i>nCol1</i> is the starting column to fix. <i>nCols</i> is the number of columns to fix.
<b>Remarks</b>	Fixed columns do not scroll and are fixed at the left edge of the worksheet window.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFixedCols</a> function and <a href="#">FixedCol</a> and <a href="#">FixedCols</a> properties
<b>Example</b>	<pre>sserror = SSSetFixedCols(Sheet1.SS, 1, 2)</pre>

## SSSetFixedRows

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the number of fixed rows.
<b>Syntax (VB)</b>	<b>SSSetFixedRows%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nRows</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetFixedRows</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nRows</i> )  <i>hSS</i> is a handle to a view. <i>nRow1</i> is the starting row to fix. <i>nRows</i> is the number of rows to fix.
<b>Remarks</b>	Fixed rows do not scroll and are fixed at the top edge of the worksheet window.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFixedRows</a> function and <a href="#">FixedRow</a> and <a href="#">FixedRows</a> properties
<b>Example</b>	<pre>sserror = SSSetFixedRows(Sheet1.SS, 1, 2)</pre>

## SSSetFont

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the font information for all selected cells.
<b>Syntax (VB)</b>	<b>SSSetFont%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pName</i> \$, ByVal <i>nSize</i> %, ByVal <i>bBold</i> %, ByVal <i>bItalic</i> %, ByVal <i>bUnderline</i> %, ByVal <i>bStrikeout</i> %, ByVal <i>crColor</i> &, ByVal <i>bOutline</i> %, ByVal <i>bShadow</i> %)
<b>Syntax (VC++)</b>	<code>SSERROR SSEXPORTAPI <b>SSSetFont</b> (HSS <i>hSS</i>, LPCSTR <i>pName</i>, int <i>nSize</i>, BOOL <i>bBold</i>, BOOL <i>bItalic</i>, BOOL <i>bUnderline</i>, BOOL <i>bStrikeout</i>, COLORREF <i>crColor</i>, BOOL <i>bOutline</i>, BOOL <i>bShadow</i>)</code>  <i>hSS</i> is a handle to a view. <i>pName</i> is the font name. <i>nSize</i> is the font size in points or twips. <i>bBold</i> , <i>bItalic</i> , <i>bUnderline</i> , and <i>bStrikeout</i> specify the bold, italic, strikeout, and underline attributes for the font. <i>crColor</i> specifies the font color. <i>crColor</i> is an RGB color and is translated into one of the 16 colors in the color palette. <i>bOutline</i> and <i>bShadow</i> specify the outline and shadow attributes for the font. These attributes are not supported in this version of Formula One.
<b>Remarks</b>	When specifying point sizes, use positive numbers for <i>nSize</i> when specifying point sizes in points; use negative numbers when specifying point sizes in twips. <b>SSSetFont</b> uses the absolute value of the number you specify.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFormatFontDlg</a> function
<b>Example</b>	<code>sserror = SSSetFont(Sheet1.SS, "Arial", 14, False, False, False, False, False, acolor, False, False)</code>

## SSSetFormula

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the formula of the active cell.
<b>Syntax (VB)</b>	<b>SSSetFormula</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetFormula</b> (HSS <i>hSS</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view. <i>pFormula</i> is a string containing the formula to put in the cell.
<b>Remarks</b>	<b>SSSetFormula</b> places a formula in the active cell. The formula should not have a leading equal sign (=).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFormula</a> , <a href="#">SSSetFormulaRC</a> function and <a href="#">Formula</a> property
<b>Example</b>	<pre>sserror = SSSetFormula(Sheet1.SS, "A1+B1")</pre>



## SSSetFormulaRC

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the formula of the specified cell.
<b>Syntax (VB)</b>	<b>SSSetFormulaRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetFormulaRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell in which the formula is placed.  <i>pFormula</i> is a string containing the formula to put in the cell.
<b>Remarks</b>	<b>SSSetFormulaRC</b> places a formula in the specified cell. The formula should not have a leading equal sign (=).
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetFormulaRC</a> , <a href="#">SSSetFormula</a> function and <a href="#">Formula</a> property
<b>Example</b>	<pre>sserror = SSSetFormulaRC(Sheet1.SS, 3, 1, "A1+A2")</pre>

## SSSetHdrHeight

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the height of the column headers.
<b>Syntax (VB)</b>	<b>SSSetHdrHeight%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nHeight</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetHdrHeight</b> (HSS <i>hSS</i> , int <i>nHeight</i> )  <i>hSS</i> is a handle to a view. <i>nHeight</i> is the height of the column headers.
<b>Remarks</b>	<b>SSSetHdrHeight</b> specifies the height of the column headers. The height is specified by <i>nHeight</i> in twips. A twip is 1/1440 of an inch.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetRowHeight</a> and <a href="#">SSSetHdrWidth</a> functions
<b>Example</b>	<code>sserror = SSSetHdrHeight(Sheet1.SS, 1440)</code>

## SSSetHdrSelection

See also [A-Z Function Call List](#)

<b>Description</b>	Selects the column headings, row headings, and cell at the intersection of the column and row headings.
<b>Syntax (VB)</b>	<b>SSSetHdrSelection%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bTopLeftHdr</i> %, ByVal <i>bRowHdr</i> %, ByVal <i>bColHdr</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetHdrSelection</b> (HSS <i>hSS</i> , BOOL <i>bTopLeftHdr</i> , BOOL <i>bRowHdr</i> , BOOL <i>bColHdr</i> )  <i>hSS</i> is a handle to a view. <i>bTopLeftHdr</i> specifies the setting of the top left header selection flag. <i>bRowHdr</i> specifies the setting of the row header selection flag. <i>bColHdr</i> specifies the setting of the column header selection flag.
<b>Remarks</b>	<b>SSSetHdrSelection</b> sets the header selection flags. The flags allow you to select the row headings, column headings, and the cell at the intersection of the row and column headings. If a flag is True, the corresponding heading is selected. If False, the heading is not selected.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetHdrSelection</a> function
<b>Example</b>	<pre>sserror = SSSetHdrSelection(Sheet1.SS, True, False, True)</pre>

## SSSetHdrWidth

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the width of the row headers.
<b>Syntax (VB)</b>	<b>SSSetHdrWidth%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nWidth</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetHdrWidth</b> (HSS <i>hSS</i> , int <i>nWidth</i> )  <i>hSS</i> is a handle to a view. <i>nWidth</i> is the width of the row headers.
<b>Remarks</b>	<b>SSSetHdrWidth</b> specifies the width of the row headers. The width is specified by <i>nWidth</i> in units equal to 1/256th of an average character's width in the default font.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetColWidth</a> and <a href="#">SSSetHdrHeight</a> functions
<b>Example</b>	<code>sserror = SSSetHdrWidth(Sheet1.SS, (5*256))</code>

## SSSetIteration

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the iteration information.
<b>Syntax (VB)</b>	<b>SSSetIteration%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bliteration</i> %, ByVal <i>nMaxIterations</i> %, ByVal <i>nMaxChange</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetIteration</b> (HSS <i>hSS</i> , BOOL <i>bliteration</i> , int <i>nMaxIterations</i> , double <i>nMaxChange</i> )  <i>hSS</i> is a handle to a view. <i>bliteration</i> specifies whether iteration is enabled. <i>nMaxIterations</i> specifies the maximum number of iterations to perform. <i>nMaxChange</i> specifies the maximum change between iterations.
<b>Remarks</b>	<b>SSSetIteration</b> sets the iteration flag, maximum number of iterations, and the maximum change value. Iteration is used to solve circular references. The program continues to calculate until it iterates the number of times specified by <i>nMaxIterations</i> or until all cells change by less than the amount specified by <i>nMaxChange</i> .
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetIteration</a> function
<b>Example</b>	<pre>sserror = SSSetIteration(Sheet1.SS, True, 100, 0.01)</pre>

## SSSetLeftCol

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the leftmost column in the view.
<b>Syntax (VB)</b>	<b>SSSetLeftCol</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nLeftCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetLeftCol</b> (HSS <i>hSS</i> , RC <i>nLeftCol</i> )  <i>hSS</i> is a handle to a view. <i>nLeftCol</i> is the leftmost column in the view.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetLeftCol</a> function and <a href="#">LeftCol</a> property
<b>Example</b>	<code>sserror = SSSetLeftCol(Sheet1.SS, 1)</code>

## SSSetLogicalRC

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the logical value of the specified cell.
<b>Syntax (VB)</b>	<b>SSSetLogicalRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>blsTrue</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetLogicalRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , BOOL <i>blsTrue</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell in which to place the logical value.  <i>blsTrue</i> is the logical value to put in the cell.
<b>Remarks</b>	<b>SSSetLogicalRC</b> sets the logical value (True or False) of the specified cell. If the cell contains a formula, the formula is deleted when the logical value is placed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetLogicalRC</a> function
<b>Example</b>	<code>sserror = SSSetLogicalRC(Sheet1.SS, 1, 2, True)</code>

## SSSetMaxCol

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the last column that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSSetMaxCol</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMaxCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetMaxCol</b> (HSS <i>hSS</i> , RC <i>nMaxCol</i> )  <i>hSS</i> is a handle to a view. <i>nMaxCol</i> specifies the number of the last column.
<b>Remarks</b>	Columns beyond the last displayable column are not displayed but can hold data and formulas.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetMaxCol</a> function and <a href="#">MaxCol</a> property
<b>Example</b>	<code>sserror = SSSetMaxCol(Sheet1.SS, 5)</code>



## SSSetMaxRow

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the last row that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSSetMaxRow%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMaxRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetMaxRow</b> (HSS <i>hSS</i> , RC <i>nMaxRow</i> )  <i>hSS</i> is a handle to a view. <i>nMaxRow</i> specifies the number of the last row.
<b>Remarks</b>	Rows beyond the last displayable row are not displayed but can hold data and formulas.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetMaxRow</a> function and <a href="#">MaxRow</a> property
<b>Example</b>	<code>sserror = SSSetMaxRow(Sheet1.SS, 1000)</code>

## SSSetMinCol

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the first column that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSSetMinCol</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMinCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetMinCol</b> (HSS <i>hSS</i> , RC <i>nMinCol</i> )  <i>hSS</i> is a handle to a view. <i>nMinCol</i> specifies the number of the first column displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetMinCol</a> function and <a href="#">MinCol</a> property
<b>Example</b>	<code>sserror = SSSetMinCol(Sheet1.SS, 3)</code>

## SSSetMinRow

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the first row that can be displayed in a view.
<b>Syntax (VB)</b>	<b>SSSetMinRow</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMinRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetMinRow</b> (HSS <i>hSS</i> , RC <i>nMinRow</i> )  <i>hSS</i> is a handle to a view. <i>nMinRow</i> specifies the number of the first row displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetMinRow</a> function and <a href="#">MinRow</a> property
<b>Example</b>	<code>sserror = SSSetMinRow(Sheet1.SS, 5)</code>

## SSetNumber

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the numeric value of the active cell.
<b>Syntax (VB)</b>	<b>SSetNumber</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nNumber</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSetNumber</b> (HSS <i>hSS</i> , double <i>nNumber</i> )  <i>hSS</i> is a handle to a view. <i>nNumber</i> is the number to put in the cell.
<b>Remarks</b>	If the active cell contains a formula, the formula is deleted when the numeric value is placed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetNumber</a> , <a href="#">SSGetNumberRC</a> , and <a href="#">SSSetNumberRC</a> functions and <a href="#">Number</a> property
<b>Example</b>	<pre>sserror = SSetNumber(Sheet1.SS, 1234.567)</pre>

## SSetNumberFormat

See also [A-Z Function Call List](#)

**Description** Specifies the number format for all selected cells.

**Syntax (VB)** **SSetNumberFormat**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pNumberFormat*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSetNumberFormat** (HSS *hSS*, LPCSTR *pNumberFormat*)

*hSS* is a handle to a view.

*pNumberFormat* is a string that specifies the format for the cells.

**Remarks** *pNumberFormat* is a format string that specifies how numbers in the selected range are displayed. The following table lists the symbols that can be used in the 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 0s 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.
[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 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
mmm 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

**Return Value** Integer

**See Also** [SSFormatNumberDlg](#) function

**Example** `sserror = SSSetNumberFormat(Sheet1.SS, "#,##0")`



## SSSetNumberRC

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the numeric value of the specified cell.
<b>Syntax (VB)</b>	<b>SSSetNumberRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>nNumber</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetNumberRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , double <i>nNumber</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell in which to place the number.  <i>nNumber</i> is the number to place in the cell.
<b>Remarks</b>	If the cell contains a formula, the formula is deleted when the number is placed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetNumber</a> , <a href="#">SSGetNumberRC</a> , and <a href="#">SSSetNumber</a> functions
<b>Example</b>	<code>sserror = SSSetNumberRC(Sheet1.SS, 1, 2, 1234.567)</code>

## SSSetPattern

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the pattern for the selected cells.
<b>Syntax (VB)</b>	<b>SSSetPattern%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nPattern</i> %, ByVal <i>crFG</i> &, ByVal <i>crBG</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPattern</b> (HSS <i>hSS</i> , int <i>nPattern</i> , COLORREF <i>crFG</i> , COLORREF <i>crBG</i> )  <i>hSS</i> is a handle to a view.  <i>nPattern</i> specifies the cell pattern. <i>nPattern</i> can range from 0 to 18 and represents one of the 18 patterns, as shown in the following illustration; 0 represents no pattern.  <i>crFG</i> and <i>crBG</i> specify the foreground and background colors for the pattern.
<b>Remarks</b>	<b>SSSetPattern</b> sets the pattern, foreground color, and background color for all selected cells.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSFormatPatternDlg</a> function
<b>Example</b>	<pre>sserror = SSSetPattern(Sheet1.SS, 2, 128, 0)</pre>

## SSSetPrintArea

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the print area.
<b>Syntax (VB)</b>	<b>SSSetPrintArea</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintArea</b> (HSS <i>hSS</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view. <i>pFormula</i> is a formula describing the print area.
<b>Remarks</b>	<b>SSSetPrintArea</b> sets the "Print_Area" user-defined name to the formula pointed to by <i>pFormula</i> . This name defines the worksheet ranges to be printed. It can contain one or more ranges (e.g. A1:C3,A11:C13). If "Print_Area" is Null (""), the active portion of the worksheet is printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintArea</a> function and <a href="#">PrintArea</a> property
<b>Example</b>	<pre>sserror = SSSetPrintArea(Sheet1.SS, "A1:D20, F1:J20")</pre>

## SSSetPrintAreaFromSelection

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the print range to the currently selected ranges.
<b>Syntax (VB)</b>	<b>SSSetPrintAreaFromSelection%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintAreaFromSelection</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetPrintArea</a> and <a href="#">SSGetPrintArea</a> functions and <a href="#">PrintArea</a> property
<b>Example</b>	<pre>sserror = SSSetPrintAreaFromSelection(Sheet1.SS)</pre>

## SSSetPrintBottomMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the bottom page margin used during printing.
<b>Syntax (VB)</b>	<b>SSSetPrintBottomMargin%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetPrintBottomMargin</b> (HSS <i>hSS</i> , double <i>nMargin</i> )  <i>hSS</i> is a handle to a view. <i>nMargin</i> specifies the bottom margin in inches.
<b>Remarks</b>	Margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintBottomMargin</a> function and <a href="#">PrintBottomMargin</a> property
<b>Example</b>	<code>sserror = SSSetPrintBottomMargin(Sheet1.SS, 2.25)</code>

## SSSetPrintColHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether column headings are printed.
<b>Syntax (VB)</b>	<b>SSSetPrintColHeading%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bColHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetPrintColHeading (HSS <i>hSS</i> , BOOL <i>bColHeading</i> )  <i>hSS</i> is a handle to a view. <i>bColHeading</i> specifies the setting of the print column heading flag.
<b>Remarks</b>	If the print column heading flag is True, column headings are enabled and printed at the top of the worksheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintColHeading</a> function and <a href="#">PrintColHeading</a> property
<b>Example</b>	<code>sserror = SSSetPrintColHeading(Sheet1.SS, True)</code>

## SSSetPrintFooter

See also [A-Z Function Call List](#)

**Description** Specifies the footer to print at the bottom of each page.

**Syntax (VB)** **SSSetPrintFooter**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pPrintFooter*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI SSSetPrintFooter (HSS *hSS*, LPCSTR *pPrintFooter*)

*hSS* is a handle to a view.

*pPrintFooter* is a string specifying the footer.

**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
&D	Prints the current date
&T	Prints the current time
&F	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

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

**Return Value** Integer

**See Also** [SSGetPrintFooter](#) function and [PrintFooter](#) property

**Example** `sserror = SSSetPrintFooter(Sheet1.SS, "&L Page: &P")`

## SSSetPrintGridLines

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether grid lines are printed.
<b>Syntax (VB)</b>	<b>SSSetPrintGridLines</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bGridLines</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetPrintGridLines</b> (HSS <i>hSS</i> , BOOL <i>bGridLines</i> )  <i>hSS</i> is a handle to a view. <i>bGridLines</i> specifies the setting of the print grid lines flag.
<b>Remarks</b>	If the print grid lines flag is True, grid lines are printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintGridLines</a> function and <a href="#">PrintGridLines</a> property
<b>Example</b>	<pre>sserror = SSSetPrintGridLines(Sheet1.SS, True)</pre>



## SSSetPrintHCenter

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether a worksheet is horizontally centered when printed.
<b>Syntax (VB)</b>	<b>SSSetPrintHCenter</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bHCenter</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintHCenter</b> (HSS <i>hSS</i> , BOOL <i>bHCenter</i> )  <i>hSS</i> is a handle to a view. <i>bHCenter</i> specifies the setting of the horizontal center flag.
<b>Remarks</b>	If the horizontal center flag is True, the worksheet is centered between the left and right margins when printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintHCenter</a> function and <a href="#">PrintHCenter</a> property
<b>Example</b>	<pre>sserror = SSSetPrintHCenter(Sheet1.SS, True)</pre>

## SSSetPrintHeader

See also [A-Z Function Call List](#)

**Description** Specifies the header to print at the top of each page.

**Syntax (VB)** **SSSetPrintHeader**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pPrintHeader*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetPrintHeader** (HSS *hSS*, LPCSTR *pPrintHeader*)

*hSS* is a handle to a view.

*pPrintHeader* is a string specifying the header.

**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
&D	Prints the current date
&T	Prints the current time
&F	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

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

**Return Value** Integer

**See Also** [SSGetPrintHeader](#) function and [PrintHeader](#) property

**Example**  

```
sserror = SSSetPrintHeader(Sheet1.SS, "&C October Sales Report")
```

## SSSetPrintLeftMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the left page margin used during printing.
<b>Syntax (VB)</b>	<b>SSSetPrintLeftMargin</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintLeftMargin</b> (HSS <i>hSS</i> , double <i>nMargin</i> )  <i>hSS</i> is a handle to a view. <i>nMargin</i> specifies the left margin in inches.
<b>Remarks</b>	Margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintLeftMargin</a> function and <a href="#">PrintLeftMargin</a> property
<b>Example</b>	<pre>sserror = SSSetPrintLeftMargin(Sheet1.SS, 1.5)</pre>

## SSSetPrintLeftToRight

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the order in which pages are printed.
<b>Syntax (VB)</b>	<b>SSSetPrintLeftToRight%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bLeftToRight</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI SSSetPrintLeftToRight (HSS <i>hSS</i> , BOOL <i>bLeftToRight</i> )  <i>hSS</i> is a handle to a view. <i>bLeftToRight</i> specifies the setting of the print left to right flag.
<b>Remarks</b>	If the print left to right flag is True, pages in a worksheet are printed left to right before printing top to bottom.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintLeftToRight</a> function and <a href="#">PrintLeftToRight</a> property
<b>Example</b>	<pre>sserror = SSSetPrintLeftToRight(Sheet1.SS, True)</pre>

## SSSetPrintNoColor

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether the worksheet is printed in color.
<b>Syntax (VB)</b>	<b>SSSetPrintNoColor</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bNoColor</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetPrintNoColor (HSS <i>hSS</i> , BOOL <i>bNoColor</i> )  <i>hSS</i> is a handle to a view. <i>bNoColor</i> specifies the setting of the print no color flag.
<b>Remarks</b>	Color formats are translated by the printer driver and printed as patterns. This translation sometimes makes text unreadable. If the print no color flag is True, all worksheet colors are converted to black and white, and all patterns are removed. A cleaner output is produced.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintNoColor</a> function and <a href="#">PrintNoColor</a> property
<b>Example</b>	<pre>sserror = SSSetPrintNoColor(Sheet1.SS, True)</pre>

## SSSetPrintRightMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the right page margin used during printing.
<b>Syntax (VB)</b>	<b>SSSetPrintRightMargin</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetPrintRightMargin</b> (HSS <i>hSS</i> , double <i>nMargin</i> )  <i>hSS</i> is a handle to a view. <i>nMargin</i> specifies the right margin in inches.
<b>Remarks</b>	Margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintRightMargin</a> function and <a href="#">PrintRightMargin</a> property
<b>Example</b>	<pre>sserror = SSSetPrintRightMargin(Sheet1.SS, 1.5)</pre>

## SSSetPrintRowHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether row headings are printed.
<b>Syntax (VB)</b>	<b>SSSetPrintRowHeading</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bRowHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintRowHeading</b> (HSS <i>hSS</i> , BOOL <i>bRowHeading</i> )  <i>hSS</i> is a handle to a view. <i>bRowHeading</i> specifies the setting of the print row heading flag.
<b>Remarks</b>	If the print row heading flag is True, row headings are enabled and printed at the left edge of the worksheet.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintRowHeading</a> function and <a href="#">PrintRowHeading</a> property
<b>Example</b>	<pre>sserror = SSSetPrintRowHeading(Sheet1.SS, True)</pre>

## SSSetPrintTitles

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies titles to be printed at the top or left of each page.
<b>Syntax (VB)</b>	<b>SSSetPrintTitles</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintTitles</b> (HSS <i>hSS</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view. <i>pFormula</i> is a string containing a formula to which the Print_Titles name is set.
<b>Remarks</b>	<b>SSSetPrintTitles</b> sets the user defined name Print_Titles to the formula specified by <i>pFormula</i> . 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). 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.  <b>Important</b> When setting print titles, you must specify entire rows and columns.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintTitles</a> and <a href="#">SSSetPrintTitlesFromSelection</a> functions and <a href="#">PrintTitles</a> property
<b>Example</b>	<pre>sserror = SSSetPrintTitles(Sheet1.SS, "A1:IV2, A1:A16384")</pre>



## SSSetPrintTitlesFromSelection

See also [A-Z Function Call List](#)

**Description** Specifies the current selection as print titles to be printed at the top or left of each page.

**Syntax (VB)** **SSSetPrintTitlesFromSelection**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetPrintTitlesFromSelection** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** **SSSetPrintTitlesFromSelection** sets the "Print\_Titles" user-defined name to a formula referring to the current selection. The entire rows and columns to be print titles must be selected.

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.

**Return Value** Integer

**See Also** [SSGetPrintTitles](#) and [SSSetPrintTitles](#) functions and [PrintTitles](#) property

**Example** `sserror = SSSetPrintTitlesFromSelection(Sheet1.SS)`

## SSSetPrintTopMargin

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the top page margin used during printing.
<b>Syntax (VB)</b>	<b>SSSetPrintTopMargin</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nMargin</i> #)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetPrintTopMargin</b> (HSS <i>hSS</i> , double <i>nMargin</i> )  <i>hSS</i> is a handle to a view. <i>nMargin</i> specifies the top page margin in inches.
<b>Remarks</b>	Margins can range from 0 to 48 inches.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintTopMargin</a> function and <a href="#">PrintTopMargin</a> property
<b>Example</b>	<pre>sserror = SSSetPrintTopMargin(Sheet1.SS, 1.5)</pre>

## SSSetPrintVCenter

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether a worksheet is vertically centered when printed.
<b>Syntax (VB)</b>	<b>SSSetPrintVCenter%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bVCenter</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetPrintVCenter</b> (HSS <i>hSS</i> , BOOL <i>bVCenter</i> )  <i>hSS</i> is a handle to a view. <i>bVCenter</i> specifies the setting of the vertical center flag.
<b>Remarks</b>	If the vertical center flag is True, the worksheet is centered between the top and bottom margins when printed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetPrintVCenter</a> function and <a href="#">PrintVCenter</a> property
<b>Example</b>	<pre>sserror = SSSetPrintVCenter(Sheet1.SS, True)</pre>

## SSSetProtection

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the protection of all currently selected cells.
<b>Syntax (VB)</b>	<b>SSSetProtection%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bLocked</i> %, ByVal <i>bHidden</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetProtection (HSS <i>hSS</i> , BOOL <i>bLocked</i> , BOOL <i>bHidden</i> )  <i>hSS</i> is a handle to a view. <i>bLocked</i> specifies the setting of the locked cell flag. <i>bHidden</i> specifies setting of the hide formulas flag.
<b>Remarks</b>	If the locked cell flag is True, all selected cells are locked. If the hide formulas flag is True, all formulas are hidden (formula results are not hidden).  After locking cells and hiding formulas, you must enable protection for the worksheet before cell locking and formula hiding is enabled. Protection for a worksheet is enabled using the <b>EnableProtection</b> property or the <b>SSSetEnableProtection</b> function call.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetEnableProtection</a> , <a href="#">SSProtectionDlg</a> , and <a href="#">SSSetEnableProtection</a> functions and <a href="#">EnableProtection</a> property
<b>Example</b>	<pre>sserror = SSSetProtection(Sheet1.SS, True, False)</pre>

## SSSetRepaint

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the repaint status for a view.
<b>Syntax (VB)</b>	<b>SSSetRepaint%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bRepaint</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSetRepaint</b> (HSS <i>hSS</i> , BOOL <i>bRepaint</i> )  <i>hSS</i> is a handle to a view. <i>bRepaint</i> specifies the setting of the repaint flag.
<b>Remarks</b>	If the repaint flag is True, repainting occurs in the entire window when Windows sends a WM_PAINT message. No repainting occurs when the repaint flag is False.  The repaint flag is not saved to disk. The default repaint setting for a new view is always True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetRepaint</a> function and <a href="#">Repaint</a> property
<b>Example</b>	<pre>sserror = SSSetRepaint(Sheet1.SS, True)</pre>

## SSSetRowHeight

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the height for the specified rows.
<b>Syntax (VB)</b>	<b>SSSetRowHeight%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nHeight</i> %, ByVal <i>bDefRowHeight</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetRowHeight</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nRow2</i> , int <i>nHeight</i> , BOOL <i>bDefRowHeight</i> )  <i>hSS</i> is a handle to a view. <i>nRow1</i> specifies the starting row to change. <i>nRow2</i> specifies the ending row to change. <i>nHeight</i> is the new row height. <i>bDefRowHeight</i> specifies whether the default row height is changed. True specifies that the default height is set to <i>nHeight</i> , 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.
<b>Remarks</b>	<b>SSSetRowHeight</b> specifies the height of the specified rows. The height is specified by <i>nHeight</i> in twips. A twip is 1/1440 of an inch.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRowHeightDlg</a> , <a href="#">SSGetRowHeight</a> , and <a href="#">SSSetRowHeightAuto</a> functions
<b>Example</b>	<pre>sserror = SSSetRowHeight(Sheet1.SS, 1, 10, 1440, False)</pre>

## SSSetRowHeightAuto

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the height of the specified rows automatically.
<b>Syntax (VB)</b>	<b>SSSetRowHeightAuto</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, ByVal <i>bSetDefaults</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetRowHeightAuto</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , BOOL <i>bSetDefaults</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> , and <i>nCol2</i> specify the range containing the rows for which to set the height.  <i>bSetDefaults</i> 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.
<b>Remarks</b>	<b>SSSetRowHeightAuto</b> 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRowHeightDlg</a> , <a href="#">SSGetRowHeight</a> , and <a href="#">SSSetRowHeight</a> functions
<b>See Also</b>	<b>SSRowHeightDlg</b> , <b>SSGetRowHeight</b> , and <b>SSSetRowHeight</b> functions
<b>Example</b>	<pre>sserror = SSSetRowHeightAuto(Sheet1.SS, 1, 1, 8, 4, True)</pre>

## SSSetRowMode

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the row mode status for a view.
<b>Syntax (VB)</b>	<b>SSSetRowMode%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bRowMode</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetRowMode</b> (HSS <i>hSS</i> , BOOL <i>bRowMode</i> )  <i>hSS</i> is a handle to a view. <i>bRowMode</i> specifies the setting of the row mode flag.
<b>Remarks</b>	If the row mode flag is True, an entire row is selected when you select a cell. Normal cell selection occurs when the flag is False. The default row mode flag is True.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetRowMode</a> function and <a href="#">RowMode</a> property
<b>Example</b>	<code>sserror = SSSetRowMode(Sheet1.SS, True)</code>



## SSSetRowText

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies the name for a row.
<b>Syntax (VB)</b>	<b>SSSetRowText</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>pRowText</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetRowText</b> (HSS <i>hSS</i> , RC <i>nRow</i> , LPCSTR <i>pRowText</i> )  <i>hSS</i> is a handle to a view. <i>nRow</i> is the row to be named. <i>pRowText</i> is the new row name.
<b>Remarks</b>	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 new 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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetColText</a> and <a href="#">SSSetTopLeftText</a> functions
<b>Example</b>	<pre>sserror = SSSetRowText(Sheet1.SS, 1, "Region 1")</pre>

## SSetSelection

See also [A-Z Function Call List](#)

**Description** Selects the specified range and moves the active cell to the top left cell in the range.

**Syntax (VB)** **SSetSelection**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nRow1*%, ByVal *nCol1*%, ByVal *nRow2*%, ByVal *nCol2*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSetSelection** (HSS *hSS*, RC *nRow1*, RC *nCol1*, RC *nRow2*, RC *nCol2*)

*hSS* is a handle to a view.

*nRow1*, *nRow2*, *nCol1*, and *nCol2* specify the range. If *nRow1* is -1, all rows are included in the selection; if *nCol1* is -1, all columns are included.

**Return Value** Integer

**See Also** [SSGetSelection](#) and [SSetSelectionRef](#) functions and [Selection](#) property

**Example** `sserror = SSetSelection(Sheet1.SS, 1, 1, 4, 4)`

## SSetSelectionRef

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the current selection from a formula.
<b>Syntax (VB)</b>	<b>SSetSelectionRef</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pFormula</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSetSelectionRef</b> (HSS <i>hSS</i> , LPCSTR <i>pFormula</i> )  <i>hSS</i> is a handle to a view. <i>pFormula</i> is a formula specifying one or more ranges.
<b>Remarks</b>	<b>SSetSelectionRef</b> sets the current selection from a formula that returns one or more ranges. For example "A1:C2,D4" selects two ranges. The first range encompasses column 1 through 3 and rows 1 and 2. The second range contains a single cell at the intersection of column 4 and row 4.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetSelectionRef</a> and <a href="#">SSSetSelection</a> functions and <a href="#">Selection</a> property
<b>Example</b>	<pre>sserror = SSetSelectionRef(sheet1.SS, "A1:C2,A4:D7")</pre>

## SSetShowColHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether column headings are displayed.
<b>Syntax (VB)</b>	<b>SSetShowColHeading</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bColHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSetShowColHeading</b> (HSS <i>hSS</i> , BOOL <i>bColHeading</i> )  <i>hSS</i> is a handle to a view. <i>bColHeading</i> specifies the setting of the show column headings flag.
<b>Remarks</b>	If the show column headings flag is True, column headings are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetShowColHeading</a> function and <a href="#">ShowColHeading</a> property
<b>Example</b>	<code>sserror = SSetShowColHeading(Sheet1.SS, True)</code>

## SSetShowFormulas

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether formulas are displayed in place of cell values.
<b>Syntax (VB)</b>	<b>SSetShowFormulas%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bFormulas</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSetShowFormulas</b> (HSS <i>hSS</i> , BOOL <i>bFormulas</i> )  <i>hSS</i> is a handle to a view. <i>bFormulas</i> specifies the setting of the show formulas flag.
<b>Remarks</b>	If the show formulas flag is True, formula text is displayed in cells instead of the values formulas produce.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetShowFormulas</a> function
<b>Example</b>	<pre>sserror = SSetShowFormulas(Sheet1.SS, True)</pre>

## SSetShowGridLines

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether grid lines are displayed.
<b>Syntax (VB)</b>	<b>SSetShowGridLines%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bGridLines</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSetShowGridLines</b> (HSS <i>hSS</i> , BOOL <i>bGridLines</i> ) <i>hSS</i> is a handle to a view. <i>bGridLines</i> specifies the setting of the show grid lines flag.
<b>Remarks</b>	If the show grid lines flag is True, grid lines are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetShowGridLines</a> function and <a href="#">ShowGridLines</a> property
<b>Example</b>	<code>sserror = SSetShowGridLines(Sheet1.SS, True)</code>

## SSetShowHScrollBar

See also [A-Z Function Call List](#)

**Description** Determines how the horizontal scroll bar is displayed.

**Syntax (VB)** **SSetShowHScrollBar%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nShowHScrollBar%*)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSetShowHScrollBar** (HSS *hSS*, int *nShowHScrollBar*)

*hSS* is a handle to a view.

*nShowHScrollBar* specifies the setting for the show horizontal scroll bar flag.

**Remarks** The following table lists the settings for the show horizontal scroll bar flag.

Setting	Description
---------	-------------

---

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

When the flag is 0, the horizontal scroll bar is hidden; when the flag is 1, the horizontal scroll bar is displayed. Setting the flag to 2 causes the horizontal scroll bar to display if the worksheet is active and it is larger than the window.

**Return Value** Integer

**See Also** [SSGetShowHScrollBar](#) function and [ShowHScrollBar](#) property

**Example** `sserror = SSetShowHScrollBar(Sheet1.SS, 2)`

## SSSetShowRowHeading

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether row headings are displayed.
<b>Syntax (VB)</b>	<b>SSSetShowRowHeading</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bRowHeading</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetShowRowHeading</b> (HSS <i>hSS</i> , BOOL <i>bRowHeading</i> )  <i>hSS</i> is a handle to a view. <i>bRowHeading</i> specifies the setting for the show row headings flag.
<b>Remarks</b>	If the show row headings flag is True, row headings are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetShowRowHeading</a> function and <a href="#">ShowRowHeading</a> property
<b>Example</b>	<code>sserror = SSSetShowRowHeading(Sheet1.SS, True)</code>



## SSSetShowSelections

See also [A-Z Function Call List](#)

**Description** Specifies whether selections are displayed.

**Syntax (VB)** **SSSetShowSelections**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nSelections*%)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSSetShowSelections** (HSS *hSS*, int *nSelections*)

*hSS* is a handle to a view.

*nSelections* specifies the setting for the show selections flag.

**Remarks** The following table lists the settings for the show selections flag.

Setting	Description
---------	-------------

---

0	Do not display selections
---	---------------------------

1	Display all selections
---	------------------------

2	Display selections in this control only
---	---

When the flag is 0, the display of selections is disabled; when the flag is 1, all selections are displayed at all times. When the flag is 2, selections are displayed in the Formula One/VB control only when the control is active.

**Return Value** Integer

**See Also** [SSGetShowSelections](#) function and [ShowSelections](#) property

**Example** `sserror = SSSetShowSelections(Sheet1.SS, 1)`

## SSetShowVScrollBar

See also [A-Z Function Call List](#)

**Description** Determines how the vertical scroll bar is displayed.

**Syntax (VB)** **SSetShowVScrollBar%** Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *nShowVScrollBar%*)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSetShowVScrollBar** (HSS *hSS*, int *nShowVScrollBar*)

*hSS* is a handle to a view.

*nShowVScrollBar* specifies the setting for the show vertical scroll bar flag.

**Remarks** The following table lists the settings for the show vertical scroll bar flag.

Setting	Description
---------	-------------

---

0	Off
---	-----

1	On
---	----

2	Automatic
---	-----------

When the flag is 0, the vertical scroll bar is hidden; when the flag is 1, the vertical scroll bar is displayed. Setting the flag to 2 causes the vertical scroll bar to display if the worksheet is active and it is larger than the window.

**Return Value** Integer

**See Also** [SSGetShowVScrollBar](#) function and [ShowVScrollBar](#) property

**Example** `sserror = SSetShowVScrollBar(Sheet1.SS, 2)`

## SSetShowZeroValues

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies whether zero value cells are displayed.
<b>Syntax (VB)</b>	<b>SSetShowZeroValues</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bZeroValues</i> %)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI SSetShowZeroValues (HSS <i>hSS</i> , BOOL <i>bZeroValues</i> )  <i>hSS</i> is a handle to a view. <i>bZeroValues</i> specifies the setting for the show zero values flag.
<b>Remarks</b>	If the show zero values flag is True, zero values are displayed.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetShowZeroValues</a> function
<b>Example</b>	<code>sserror = SSetShowZeroValues(Sheet1.SS, True)</code>

## SSetSSEdit

See also [A-Z Function Call List](#)

**Description** Attaches a view to the specified edit bar.

**Syntax (VB)** **SSetSSEdit**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *hSSEdit*&)

**Syntax (VC++)** SSERROR SEXPORTAPI **SSetSSEdit** (HSS *hSS*, HSSEDIT *hSSEdit*)

*hSS* is a handle to a view.

*hSSEdit* is a handle to an edit bar.

**Important** This function should not be called from Visual Basic.

---

**Return Value** Integer

**See Also** [SSGetSSEdit](#) function and [EditName](#) property

## SSSetText

See also [A-Z Function Call List](#)

<b>Description</b>	Places text in the active cell.
<b>Syntax (VB)</b>	<b>SSSetText</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pText</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetText</b> (HSS <i>hSS</i> , LPCSTR <i>pText</i> )  <i>hSS</i> is a handle to a view. <i>pText</i> is the text to put in the cell.
<b>Remarks</b>	If the active cell contains a formula, the formula is deleted when the text is placed in the cell.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetText</a> and <a href="#">SSSetTextRC</a> functions and <a href="#">Text</a> property
<b>Example</b>	<code>sserror = SSSetText(Sheet1.SS, "Some Text")</code>

## SSSetTextRC

See also [A-Z Function Call List](#)

<b>Description</b>	Places text in the specified cell.
<b>Syntax (VB)</b>	<b>SSSetTextRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow</i> %, ByVal <i>nCol</i> %, ByVal <i>pText</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetTextRC</b> (HSS <i>hSS</i> , RC <i>nRow</i> , RC <i>nCol</i> , LPCSTR <i>pText</i> )  <i>hSS</i> is a handle to a view.  <i>nRow</i> and <i>nCol</i> are the row and column numbers of the cell in which the text is placed.  <i>pText</i> is a string containing the text to put in the cell.
<b>Remarks</b>	If the specified cell contains a formula, the formula is deleted when the text is placed in the cell.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSetText</a> , <a href="#">SSGetTextRC</a> Functions and <a href="#">Text</a> Property
<b>Example</b>	<pre>sserror = SSSetTextRC(Sheet1.SS, 1, 1, "Some Text")</pre>

## SSSetTitle

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the title of the worksheet.
<b>Syntax (VB)</b>	<b>SSSetTitle</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pTitle</i> \$)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSetTitle</b> (HSS <i>hSS</i> , LPCSTR <i>pTitle</i> )  <i>hSS</i> is a handle to a view. <i>pTitle</i> is a string containing the new title.
<b>Remarks</b>	The worksheet title can be used in external references to access multiple worksheets.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetTitle</a> function
<b>Example</b>	<pre>sserror = SSSSetTitle(Sheet1.SS, "Table1")</pre>

## SSSetTopLeftText

See also [A-Z Function Call List](#)

**Description** Specifies the text displayed at the intersection of the row and column headings in the top left corner of the spreadsheet.

**Syntax (VB)** **SSSetTopLeftText**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pTopLeftText*\$)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSSetTopLeftText** (HSS *hSS*, LPCSTR *pTopLeftText*)

*hSS* is a handle to a view.

*pTopLeftText* is the text for the top left corner.

**Remarks** The text placed in the top left corner of the spreadsheet is used for display purposes only.

**Return Value** Integer

**See Also** [SSSetColText](#) and [SSSetRowText](#) functions

**Example** `sserror = SSSetTopLeftText(Sheet1.SS, "Current Sales")`



## SSSetTopRow

See also [A-Z Function Call List](#)

<b>Description</b>	Sets the top row displayed in the view.
<b>Syntax (VB)</b>	<b>SSSetTopRow%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nTopRow</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI SSSetTopRow (HSS <i>hSS</i> , RC <i>nTopRow</i> )  <i>hSS</i> is a handle to a view. <i>nTopRow</i> is the row number of the top row.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSGetTopRow</a> function and <a href="#">TopRow</a> property
<b>Example</b>	<code>sserror = SSSetTopRow(Sheet1.SS, 10)</code>

## SSShowActiveCell

See also [A-Z Function Call List](#)

**Description** Positions the view to show the active cell if it is not currently displayed in the window.

**Syntax (VB)** **SSShowActiveCell**% Lib "VTSSDLL.DLL" (ByVal *hSS*&)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSShowActiveCell** (HSS *hSS*)

*hSS* is a handle to a view.

**Remarks** If the active cell is not displayed in the view window the worksheet is repositioned so the active cell is visible. The worksheet is repositioned by scrolling the worksheet until the cell becomes visible.

**Return Value** Integer

**See Also** [SSGetActiveCell](#) and [SSSetActiveCell](#) functions and [Row](#) and [Col](#) properties

**Example** `sserror = SSShowActiveCell(Sheet1.SS)`

## SSSort

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies a range of data to be sorted and the keys by which to sort the data.
<b>Syntax (VB)</b>	<b>SSSort%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, ByVal <i>bSortByRows</i> %, <i>pkeys</i> %, ByVal <i>nKeys</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSort</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , BOOL <i>bSortByRows</i> , int FAR <i>*pkeys</i> , int <i>nKeys</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> , and <i>nCol2</i> specify the range of data to be sorted.  <i>bSortByRows</i> specifies how data is sorted. If True, data is sorted by rows; if False, data is sorted by columns.  <i>pKeys</i> is an array of integers specifying 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.  <i>nKeys</i> is the number of sort keys specified in <i>pKeys</i> .
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSort3</a> and <a href="#">SSSortDlg</a> functions
<b>Example</b>	<code>sserror = SSSort(Sheet1.SS, 1, 1, 10, 6, True, sortkeys, 4)</code>

## SSSort3

See also [A-Z Function Call List](#)

<b>Description</b>	Specifies a range of data to be sorted and as many as three keys by which to sort the data. For Visual Basic programmers, this function call provides an easier method for sorting data than <b>SSSort</b> .
<b>Syntax (VB)</b>	<b>SSSort3</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, ByVal <i>bSortByRows</i> %, ByVal <i>nKey1</i> %, ByVal <i>nKey2</i> %, ByVal <i>nKey3</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSort3</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , BOOL <i>bSortByRows</i> , int <i>nKey1</i> , int <i>nKey2</i> , int <i>nKey3</i> )  <i>hSS</i> is a handle to a view.  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> , and <i>nCol2</i> specify the range of data to be sorted.  <i>bSortByRows</i> specifies how data is sorted. If True, data is sorted by rows; if False, data is sorted by columns.  <i>nKey1</i> , <i>nKey2</i> , and <i>nKey3</i> 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. <i>nKey1</i> is the primary key, <i>nKey2</i> is the secondary key, and <i>nKey3</i> is the last sort key.
<b>Remarks</b>	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 <i>nKey1</i> .  To sort on one key, <i>nKey2</i> must be zero. To sort on two keys, <i>nKey3</i> must be zero.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSort</a> and <a href="#">SSSortDlg</a> functions
<b>Example</b>	<code>sserror = SSSort3(Sheet1.SS, 1, 1, 10, 6, True, 2, -4, 5)</code>

## SSSortDlg

See also [A-Z Function Call List](#)

<b>Description</b>	Displays the Sort dialog box.
<b>Syntax (VB)</b>	<b>SSSortDlg</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSSortDlg</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	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.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSSort</a> and <a href="#">SSSort3</a> functions
<b>Example</b>	<pre>sserror = SSSortDlg(Sheet1.SS)</pre>

## SSStartEdit

See also [A-Z Function Call List](#)

<b>Description</b>	Begins edit mode for the active cell.
<b>Syntax (VB)</b>	<b>SSStartEdit</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>bClear</i> %, ByVal <i>bInCellEditFocus</i> %, ByVal <i>bArrowsExitEditMode</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSStartEdit</b> (HSS <i>hSS</i> , BOOL <i>bClear</i> , BOOL <i>bInCellEditFocus</i> , BOOL <i>bArrowsExitEditMode</i> )  <i>hSS</i> is a handle to a view. <i>bClear</i> sets the clear edit bar flag. <i>bInCellEditFocus</i> sets the in cell edit flag. <i>bArrowsExitEditMode</i> sets the arrows exit edit mode flag. <b>SSStartEdit</b> starts edit mode for the active cell and allows you to specify how the cell is edited. If <i>bClear</i> is True, the edit bar is cleared as edit mode commences. If <i>bInCellEditFocus</i> is True, editing focus is given to in-cell editing; if False, editing focus is given to the edit bar. If <i>bArrowsExitEditMode</i> is True, edit mode is exited if you press an arrow key on the keyboard.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSCancelEdit</a> and <a href="#">SSEndEdit</a> functions
<b>Example</b>	<pre>sserror = SSStartEdit(Sheet1.SS, False, True, True)</pre>

## SSSwapTables

See also [A-Z Function Call List](#)

<b>Description</b>	Exchanges the worksheets attached to two views.
<b>Syntax (VB)</b>	<b>SSSwapTables</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS1</i> &, ByVal <i>hSS2</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSSwaptables</b> (HSS <i>hSS1</i> , HSS <i>hSS2</i> )  <i>hSS1</i> and <i>hSS2</i> are handles to views.
<b>Remarks</b>	When the worksheets are exchanged, the view information is not swapped - only the worksheets are swapped. The following properties are not swapped when <b>SSSwapTables</b> is called: <b>EditName</b> , <b>AllowAppLaunch</b> , <b>Tablename</b> , all Data properties, all Do properties, and <b>FileName</b> . If you want to swap both worksheets and view information between two views, then you must swap these properties using Visual Basic code.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSAttach</a> and <a href="#">SSAttachToSS</a> functions
<b>Example</b>	<pre>sserror = SSSwapTables(Sheet1.SS, Sheet2.SS)</pre>

## SSTransactCommit

See also [A-Z Function Call List](#)

<b>Description</b>	Commits changes made during a transaction.
<b>Syntax (VB)</b>	<b>SSTransactCommit</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SEXPORTAPI <b>SSTransactCommit</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	<p>Transactions perform 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 <b>SSTransactRollback</b>. If all operations succeed, <b>SSTransactCommit</b> should be called to make the changes permanent and release resources associated with the transaction.</p> <p>Every <b>SSTransactStart</b> call must have a matching <b>SSTransactCommit</b> or <b>SSTransactRollback</b> call.</p>
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSTransactRollback</a> and <a href="#">SSTransactStart</a> functions
<b>Example</b>	<pre>sserror = SSTransactCommit(Sheet1.SS)</pre>



## SSTransactRollback

See also [A-Z Function Call List](#)

<b>Description</b>	Undoes all changes made to a table since the last SSTransactStart function was called.
<b>Syntax (VB)</b>	<b>SSTransactRollback%</b> Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSTransactRollback</b> (HSS <i>hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	Transactions perform 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 <b>SSTransactRollback</b> . If all operations succeed, <b>SSTransactCommit</b> should be called to make the changes permanent and release resources associated with the transaction.  Every <b>SSTransactStart</b> call must have a matching <b>SSTransactCommit</b> or <b>SSTransactRollback</b> call.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSTransactCommit</a> and <a href="#">SSTransactStart</a> functions
<b>Example</b>	<pre>sserror = SSTransactRollback(Sheet1.SS)</pre>

## SSTransactStart

See also [A-Z Function Call List](#)

<b>Description</b>	Starts a transaction.
<b>Syntax (VB)</b>	<b>SSTransactStart</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSTransactStart</b> ( <i>HSS hSS</i> )  <i>hSS</i> is a handle to a view.
<b>Remarks</b>	Transactions perform 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 <b>SSTransactRollback</b> . If all operations succeed, <b>SSTransactCommit</b> should be called to make the changes permanent and release resources associated with the transaction.  Every <b>SSTransactStart</b> call must have a matching <b>SSTransactCommit</b> or <b>SSTransactRollback</b> call.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSTransactCommit</a> and <a href="#">SSTransactRollback</a> functions
<b>Example</b>	<pre>sserror = SSTransactStart(Sheet1.SS)</pre>

## SSTwipsToRC

See also [A-Z Function Call List](#)

<b>Description</b>	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.
<b>Syntax (VB)</b>	<b>SSTwipsToRC</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>pX</i> &, ByVal <i>pY</i> &, <i>pRow</i> %, <i>pCol</i> %)
<b>Syntax (VC++)</b>	SSERROR SSEXPORTAPI <b>SSTwipsToRC</b> (HSS <i>hSS</i> , LONG <i>pX</i> LONG <i>pY</i> , LPRC <i>pRow</i> , LPRC <i>pCol</i> )  <i>hSS</i> is a handle to a view. <i>pX</i> is the horizontal coordinate of the point to convert. <i>pY</i> is the vertical coordinate of the point to convert. <i>pRow</i> is the number of the row returned. <i>pCol</i> is the number of the column returned.
<b>Remarks</b>	The coordinates specified by this function are measured in twips from the upper left corner of the worksheet control.  <b>SSTwipsToRC</b> can determine the row and column that corresponds to a point returned by the <b>DragDrop</b> and <b>DragOver</b> events. <b>SSTwipsToRC</b> returns 0 if the referenced point is located in a row or column heading.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRangeToTwips</a> function and <a href="#">DragDrop</a> and <a href="#">DragOver</a> events
<b>Example</b>	<code>sserror = SSTwipsToRC(Sheet1.SS, 50, 100, therow, thecol)</code>

## SSUpdate

See also [A-Z Function Call List](#)

<b>Description</b>	Updates all worksheets.
<b>Syntax (VB)</b>	Declare Sub <b>SSUpdate</b> Lib "VTSSDLL.DLL" ()
<b>Syntax (VC++)</b>	Void SSEXPORTAPI <b>SSUpdate</b> ()
<b>Remarks</b>	<b>SSUpdate</b> updates everything that might be delayed. This includes recalculating any worksheets with the <b>AutoRecalc</b> property set to True, updating the scroll bar position, and sending the SSM_MODIFIED message if needed.
<b>Return Value</b>	Nothing
<b>See Also</b>	<a href="#">SSCheckRecalc</a> and <a href="#">SSSetAutoRecalc</a> functions and <a href="#">AutoRecalc</a> property
<b>Example</b>	Call SSUpdate()

## SSVBXCopyCellsFromDoubleArray

See also [A-Z Function Call List](#)

<b>Description</b>	Copies an array of numbers to a range.
<b>Syntax (VB)</b>	<b>SSVBXCopyCellsFromDoubleArray</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>Row1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, <i>hArray</i> () As Double)
<b>Syntax (VC++)</b>	SSERROR SSEXPORT API <b>SSVBXCopyCellsFromDoubleArray</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , HAD <i>hArray</i> )  <i>hSS</i> is a handle to a view  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> and <i>nCol2</i> are the row and column numbers of the range to which the array is copied.  <i>hArray</i> is a two-dimensional array from which numbers are copied.
<b>Remarks</b>	The size of the range and the size of the array must match. The first dimension of the array must match the number of rows in the range; the second dimension of the array must match the number of columns in the range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSVBXCopyCellsToDoubleArray</a> function call

## SSVBXCopyCellsToDoubleArray

See also [A-Z Function Call List](#)

<b>Description</b>	Copies a range of numbers to an array.
<b>Syntax (VB)</b>	<b>SSVBXCopyCellsToDoubleArray</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>nRow1</i> %, ByVal <i>nCol1</i> %, ByVal <i>nRow2</i> %, ByVal <i>nCol2</i> %, <i>hArray</i> () As Double)
<b>Syntax (VC++)</b>	SSERROR SSEXPORT API <b>SSVBXCopyCellsToDoubleArray</b> (HSS <i>hSS</i> , RC <i>nRow1</i> , RC <i>nCol1</i> , RC <i>nRow2</i> , RC <i>nCol2</i> , HAD <i>hArray</i> )  <i>hSS</i> is a handle to a view  <i>nRow1</i> , <i>nCol1</i> , <i>nRow2</i> and <i>nCol2</i> are the row and column numbers of the range from which numbers are copied.  <i>hArray</i> is a two-dimensional array to which the range of numbers is copied.
<b>Remarks</b>	The size of the range and the size of the array must match. The first dimension of the array must match the number of rows in the range; the second dimension of the array must match the number of columns in the range.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSVBXCopyCellsFromDoubleArray</a> function call

## SSVersion

See also [A-Z Function Call List](#)

<b>Description</b>	Returns the version number of the Formula One control.
<b>Syntax (VB)</b>	<b>SSVersion</b> % Lib "VTSSDLL.DLL" ()
<b>Syntax (VC++)</b>	WORD SSEXPORTAPI <b>SSVersion</b> ()
<b>Remarks</b>	<b>SSVersion</b> returns the version number of the Formula One control. The major version number is stored in the high byte. Minor version numbers are stored in the low byte. For example, 0x0100 is version 1.0, while 0x0301 is version 3.01.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSMaxCol</a> and <a href="#">SSMaxRow</a> functions
<b>Example</b>	<pre>Dim ver% ver = SSVersion() MsgBox "Version: " &amp; (ver / 256) &amp; "." &amp; (ver And &amp;HFF)</pre>

## SSWrite

See also [A-Z Function Call List](#)

**Description** Saves the worksheet to the specified file.

**Syntax (VB)** **SSWrite**% Lib "VTSSDLL.DLL" (ByVal *hSS*&, ByVal *pPathName*\$, ByVal *nSaveType*%)

**Syntax (VC++)** SSERROR SSEXPORTAPI **SSWrite** (HSS *hSS*, LPCSTR *pPathName*, int *nSaveType*)

*hSS* is a handle to a view.

*pPathName* is a string containing the name of the file to write. The name can include drive, path, and filename.

*nSaveType* is the file type used when writing the file. The following table lists the settings for this parameter.

Setting	Description
1	Formula One format
2	Excel 4.0 format

**Return Value** Integer

**See Also** [SSRead](#), [SSReadIO](#), [SSSaveWindowInfo](#), and [SSWriteIO](#) functions

**See Also** [SSRead](#), [SSReadIO](#), [SSSaveWindowInfo](#), and [SSWriteIO](#) functions

**Example** `sserror = SSWrite(Sheet1.SS, filename, 1)`



## SSWriteIO

See also [A-Z Function Call List](#)

<b>Description</b>	Writes a worksheet using the specified write function.
<b>Syntax (VB)</b>	<b>SSWriteIO</b> % Lib "VTSSDLL.DLL" (ByVal <i>hSS</i> &, ByVal <i>dwUserData</i> &, ByVal <i>ioFunc</i> &, <i>pUserRet</i> &)
<b>Syntax (VC++)</b>	<b>SSERROR SSEXPORTAPI SSWriteIO</b> ( <i>HSS hSS</i> , <i>DWORD dwUserData</i> , <i>IOFUNC ioFunc</i> , <i>DWORD FAR *pUserRet</i> )  <i>hSS</i> is a handle to a view. <i>dwUserData</i> is passed to <i>ioFunc</i> each time <i>ioFunc</i> is called. <i>ioFunc</i> is the function called to write data from the worksheet. It has following form: <pre>typedef DWORD (FAR PASCAL *IOFUNC) (DWORD dwUserData, LPVOID p, UINT nBytes);</pre> <i>pUserRet</i> returns the last value returned by <i>ioFunc</i> . If this pointer is not null, it returns the last value returned by <i>ioFunc</i> . Any non-zero value returned by <i>ioFunc</i> causes writing to fail immediately.
<b>Remarks</b>	<b>SSWriteIO</b> is the same as <b>SSWrite</b> except that <i>ioFunc</i> is called to write data instead of writing to a specified file.  If <i>ioFunc</i> returns non-zero, the value is returned by <b>SSWriteIO</b> . If the file is successfully written, 0 is returned.
<b>Return Value</b>	Integer
<b>See Also</b>	<a href="#">SSRead</a> , <a href="#">SSReadIO</a> , <a href="#">SSSaveWindowInfo</a> , and <a href="#">SSWrite</a> functions
<b>See Also</b>	<b>SSRead</b> , <b>SSReadIO</b> , <b>SSSaveWindowInfo</b> , and <b>SSWrite</b> functions

