Tools
Add In
Visual Basic for Applications

by Chris Barlow

TAKE A VOYAGE THROUGH THE RANGE OBJECT

USE THIS POWERFUL OBJECT TO LET USERS EXTEND A RANGE WITH ITS FORMULAS INTACT AND PRINT THE RESULTS.

Now that VBA has been on desktops for a while, more of you are experimenting with VBA apps—and having trouble learning the Excel object model. The questions I receive on e-mail indicate that one of the common stumbling blocks is working with range objects. The range object is one of the most powerful parts of the Excel object model. You can use methods of the range object to copy, print, and create subsets of another range.

Because the object model was developed from the spreadsheet paradigm, however, the Visual Basic programmer often cannot figure out how to perform functions that should be simple. For example, a VB programmer might expect the range object to have an AddItem method that dynamically extends a range with its formulas intact. But there is no such method. In this column I'll show you how to write procedures that work with range objects to let a user select information from one range and add it to another range, make some calculations, and print the results.

As I write this column in mid-February, many of you are fighting the cold and snow. I'm enjoying one of the advantages of living and working in Sarasota, Florida. I'm sitting on my boat, using my laptop to make some notes for this column, and watching the Loran monitor my course up Sarasota Bay (see Figure 1). A Loran is a neat navigational tool that monitors three different broadcast stations and displays my boat's current latitude and longitude every few seconds. Even an inexpensive Loran like mine can save the latitude and longitude of multiple "waypoints" and calculate the course and distance from one waypoint to another.

The Loran's waypoint capability would be very helpful in planning a voyage from my mooring to some distant location: I could select successive waypoints and the Loran would tell me the course and distance as well as the total elapsed time between each one.

The problem is that I usually plan the voyage at home, and the Loran only works on my boat. What I really need is a standalone application that lets me choose successive waypoints and compile the course, distance, and elapsed time for any voyage, starting at my mooring and proceeding from waypoint to waypoint. The idea sounded like an Excel VBA application just waiting to be developed, which I did. I call it Voyage.

Voyage creates a worksheet containing all the waypoints that make up a desired voyage. I assign buttons to some VBA procedures that start a new voyage, pop up a dialog to select the next waypoint, and print out the voyage.

A voyage always starts with two waypoints that lead from my mooring to the main channel marker #10 (see Figure 2). The first four columns for each waypoint make up the Waypoint range, which is stored on the Waypoint worksheet you can access by clicking on the Waypoint Sheet Tab. (I copy data from my Loran and keep it in the Waypoint worksheet as a backup copy in case my Loran battery fails.) You need to calculate the Voyage worksheet's next seven columns from formulas.

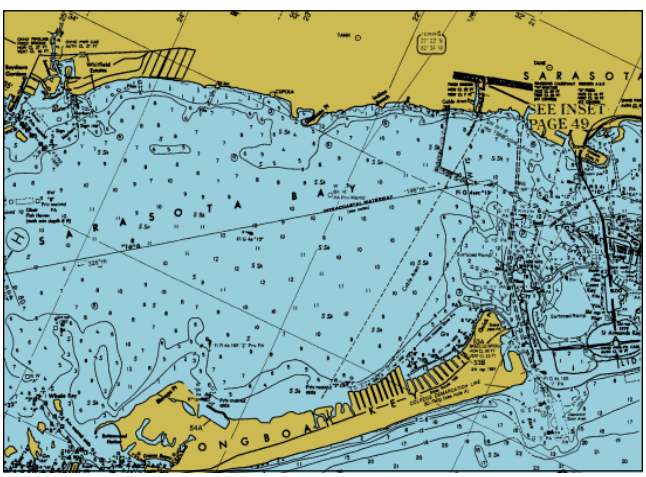

FIGURE 1 ***Sailing Up the Bay.*** *On my boat I chart my course up Sarasota Bay with a Loran, a neat navigational tool that displays my boat's current latitude and longitude every few seconds. But the Loran only works on the boat, and I usually plan my trips at home. So I used Excel's range object to write an application that lets me choose successive "waypoints" and compile the course, distance, and elapsed time for any voyage.*

REMEMBER HIGH-SCHOOL MATH?

But how do you calculate course and distance given latitude and longitude? Think back to your high-school geometry class (never thought you would use that stuff again, huh?). Latitude and longitude are simply the x,y coordinates of a position. If you

Chris Barlow learned Basic from Professors Kemeny and Kurtz at Dartmouth College. He uses Visual Basic to develop manufacturing decision support applications for SunOpTech Group, where he coauthored with Ken Henderson an application that was a finalist in the first Windows Open Contest. Reach Chris at SunOpTech Group, 1500 West University Parkway, Sarasota, Florida, 34243; by telephone at 813-351-9183; by fax at 813-355-4497; on CompuServe at 76446,1370; and on the Internet at ChrisB@SunOpTech.com.

know the distance between two points along the x-axis (east to west) and along the y-axis (north to south) then you can use the Pythagorean theorem (remember: x-squared plus y-squared = z-squared?) to calculate the distance between the points.

But first you have to convert the coordinates from degrees to nautical miles (NM). Latitude distance is easy because one degree of latitude is always equal to 60 nautical miles. But the distance of one degree of longitude varies with your location and must be measured on a nautical chart. For example, here in Sarasota, one degree of longitude equals 52.8 nautical miles.

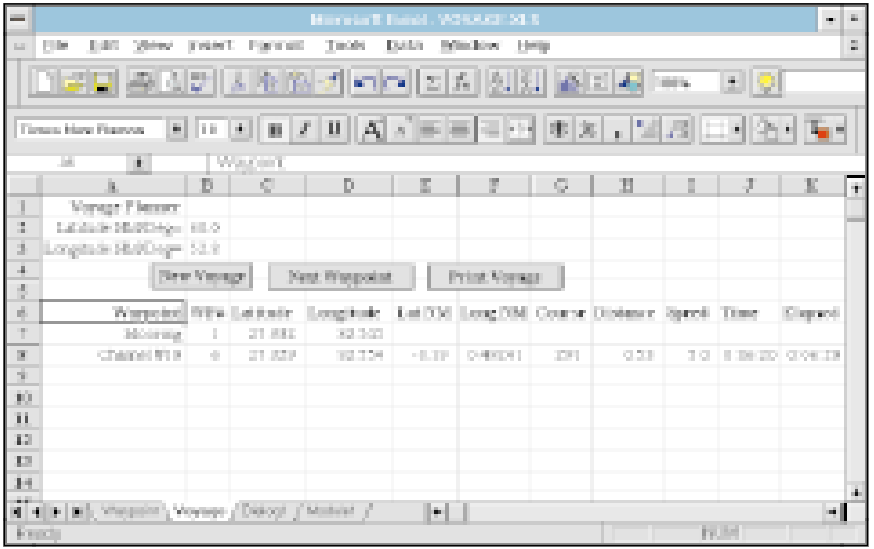

FIGURE 2 ***Voyage Worksheet.*** *Given the constants in cells B2 and B3 and the waypoint information in the first four columns of each row, you can calculate the values for the next seven columns. You can then copy these formulas to subsequent rows.*

Put these constants into cells B2 and B3 of the Voyage worksheet and use them in the formulas in columns E and F. Once you've calculated the east-west distance and the north-south distance, you can calculate the point-to-point distance in column H with the formula SQRT(E8^2+F8^2). Now that you know the distance, it's easy to fill in the formulas for time and elapsed time given a certain speed in knots (nautical miles per hour).

The only column without a formula yet is the Course. All navigational courses are defined as the number of degrees, ranging from zero to 359, from North on the vertical axis. To calculate the course you must apply a little trigonometry. You might remember that, given the length of two sides of a triangle, you can calculate the tangent of the angle by dividing one by the other. Visual Basic has built-in functions for the tangent and arctangent of angles. Let's write a VBA procedure called TCourse that returns the true course given a latitude distance and a longitude distance.

The first step is to define our function, dimension a variable to hold the course, and use the arctangent function (ATN) built into VB to calculate the angle whose

tangent is the first distance divided by the second distance:

```
Function TCourse(LatNM, LongNM) As _
   Single
'returns true course from lat. and
'long. distances
Dim CalcAngle As Single
CalcAngle = RD(Atn(LatNM / LongNM))
```

Now you know the angle, but you're not done yet. The calculated angle is based on the horizontal axis, not the vertical axis by which you determine the course value. You need to use an If statement to convert this calculated angle to a course by adding or subtracting from 90 or 270, depending on the signs of the distances.

Easy-to-document, understandable code is one of the big advantages of using VBA with Excel. You could have used Excel functions alone to write this formula, but with Visual Basic you can write an expanded If statement that is much easier to follow:

```
If LatNM < 0 And LongNM < 0 Then
   TCourse = 90 + CalcAngle
ElseIf LatNM < 0 And LongNM >= 0 Then
   TCourse = 270 - CalcAngle
ElseIf LatNM >= 0 And LongNM < 0 Then
   TCourse = 90 - CalcAngle
Else
   TCourse = 270 + CalcAngle
End If
End Function
```

Now that you have all the formulas for the first two waypoints, you can copy them for each successive waypoint. But how do you select these waypoints and add them to the Voyage range?

CHOOSING A WAYPOINT

The first step is to create a dialog sheet that allows the user to select a waypoint from the Waypoint range. Dialog sheets are convenient ways to display data from a range and allow the user to make a selection. Although not nearly as powerful as the forms in Visual Basic 3.0, dialog sheets are easy to create and require very little code.

To insert a blank dialog sheet into your workbook, select Dialog from the Macro menu item under the Insert menu. Select a list box from the Forms toolbar and draw it on the dialog box. Then select Control Properties from the toolbar to pop up the Format Object dialog box. You can also pop up this box by right-clicking on the new list box and selecting Format Object.

Enter the Waypoint range in the input range field on the Control tab. This fills the list box with the list of available waypoints. When the user clicks on an item in the list, Windows will set the ListIndex property accordingly. Now click on the OK button and rename it to "SelectWP" by typing the new name in the upper-left corner of the dialog sheet. Then select Edit Code from the toolbar to create the code.

First define the variable SelRow to hold the selected row from the list box. Then assign SelRow equal to the ListIndex property of the first list box on the active dialog:

```
Sub SelectWP_Click()
'OK Button
'find info on next waypoint and add
'to voyage list
Dim SelRow As Integer
   SelRow = _
      ActiveDialog.ListBoxes(1)._
      ListIndex
```

Now that you know the row number of the waypoint selected, you can use the Copy method of the range object to copy it to a new destination. I'll break

CONTINUED ON PAGE 108.

HOME

CONTINUED FROM PAGE 104.

this up into small pieces so you can understand how to extend this range dynamically.

First, use the SelRow variable to specify the source range to copy, then specify the destination range:

```
Range("Waypoints").Rows(SelRow)._
   Copy {destination range}
```

THE RANGE OBJECT IS ONE OF THE MOST POWERFUL PARTS OF THE EXCEL OBJECT MODEL.

The complete Voyage range will continue to expand, so you can determine its current dimensions by referring to the Voyage range and using the CurrentRegion method to get the complete Voyage range bounded by blank rows and columns—Range("Voyage").CurrentRegion.

Because you're going to use this often for copying and printing, define a function, CurVoyage, that returns a range object of the current complete Voyage range. Note that because you're returning an object, you must use the Set statement:

```
Function CurVoyage() As Range
Set CurVoyage = Range("Voyage")._
   CurrentRegion
End Function
```

Now you can use CurVoyage() anywhere you need the complete Voyage range. In this case the destination range is really the row below the current complete Voyage range. You can use the End method on this range with the parameter xlDown to get to the last row containing voyage data, and the Offset method to move down one more row:

```
CurVoyage().End(xlDown).Offset(1, 0)
```

It's that easy to add a new row to a range from another range. Now that you've added the selected waypoint's data into a new row within the first four columns of the Voyage worksheet, you're ready to copy the formulas to this new row from the prior row. Use the same Copy method of the range object.

Select the new Voyage row you've just added by using the CurVoyage function with the End method:

```
CurVoyage().End(xlDown).Select
```

Then use the Offset method to identify the source as columns 5–11 in the prior row and the destination range as the new row:

```
   Range(Selection.Offset(-1, 4), _
      Selection.Offset(-1, 10)).Copy _
      Selection.Offset(0, 4)
End Sub
```

Your SelectWP procedure is complete. You can test this dialog by clicking on the Run Dialog button on the toolbar: you should see the new waypoint row appear on the Voyage sheet. But this isn't how you want the user to operate, so place three buttons on the Voyage worksheet. One button clears the Voyage range to start again, one displays the dialog to select another waypoint, and one prints the current voyage.

ADD SOME BUTTONS

When adding buttons to a worksheet, I find it best to write the procedures first, then add the buttons and assign them to the procedures.

The NewVoyage procedure clears all but the first three rows of the Voyage range. It uses the CurVoyage function to identify the complete voyage region offset by three rows, followed by the ClearContents method of the range object:

```
Sub NewVoyage()
'resets for new voyage leaving
'mooring and channel mark
CurVoyage().Offset(3, 0).ClearContents
End Sub
```

The NextWP procedure simply shows the dialog sheet you created so the user can select the next waypoint:

```
Sub NextWP()
dialogsheets(1).Show
End Sub
```

The PrintVoyage procedure uses the PrintOut method of the range object to print the complete current voyage region:

```
Sub PrintVoyage()
CurVoyage().PrintOut
End Sub
```

Now that you've written the procedures, add the buttons. The easiest way to add buttons to the worksheet is to display the Drawing toolbar using the Toolbars menu item under the View menu. Select the button object from toolbar and draw the button where you want it on the sheet. Then right-click on the button, select Assign Macro, and choose the proper procedure. That's all there is to it. Try creating a voyage like the one in Figure 3.

Understanding how to manipulate range objects is one of the hurdles you need to leap to really move ahead with VBA. Mastering this skill will open the floodgates to a sea of new procedures you can develop. The code discussed in this column, along with the Voyage workbook file, is also available on Volume 1, Number 3 of the VB-CD Quarterly and in the Magazine Library (#3) in the *Visual Basic Programmer's Journal* Forum (GO VBPJ) on CompuServe. ■

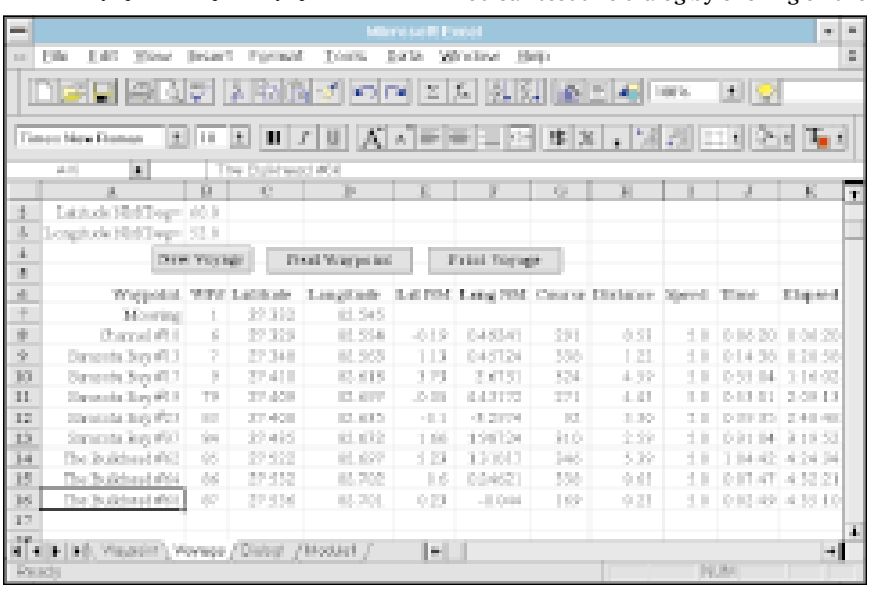

FIGURE 3 ***Complete Voyage.*** *This worksheet contains information about all the waypoints that make up your voyage. The formulas for columns 5 through 11 (E–K) have been copied to each new row. If the user changes the speed for one or more waypoints, Excel will recalculate the Time and Elapsed columns.*