Do-It-Yourself Automated Mass Readings

How to transfer analytical balance readings directly into an Excel worksheet

By Rick Collard

You probably know that your lab's balance has a communication interface that can transmit data to a computer. Reading the balance's data directly into the Excel™ spreadsheet program, manufactured by Microsoft (Redmond, Wash.), eliminates manual data entry and associated errors.

But you may have avoided interfacing the instrument because of the extra software required and the perceived complexities. Some balance vendors provide software tools to retrieve the data, and third parties also sell software for this purpose. However, on its own, Microsoft Excel offers a simple solution. A single button click or keystroke retrieves the balance's weighing results and places it directly into an Excel cell using only Excel and a free communications control.

When Microsoft made Visual Basic for Applications (VBA) the standard development environment in its Office 97 product line, Excel 97 gained significant developer features, resulting in a powerful general purpose programming platform. Since then, Mountain States Consulting LLC (Jackson, Wyo.) has used Excel extensively to help users of its laboratory information management system, MSC-LIMS, integrate their existing Excel reports, instrument data, and data entry spreadsheets with their LIMS software.

What You Will Need

This article provides step-by-step instructions to create a new Excel workbook with the infrastructure required to read weighing data directly from your balance. To follow the example, you will need Excel 97 or a newer version. You will also need the free XMComm communication control (see download and installation instructions below) and WinZip or other software capable of extracting ZIP archives.

You also need a balance with a bi-directional serial (RS-232C) communication interface and the proper cable to connect the instrument to your workstation. You will need the balance's technical documentation, identifying its communication commands and returned data formats. Our example works with both A&D Weighing HR Series and Mettler Toledo AE Series balances. Only minor adaptations should be necessary for other instruments, and we will point out where those changes may be necessary.

Finally, you will need to use a small amount of VBA code to make it all work. Do not let the concept of VBA programming deter you. Just as you have tackled your own home improvement projects without hiring a general contractor, you can follow along and learn enough VBA to get the job done without hiring a programmer.

Download and Install XMComm

To handle many of the details of serial communications, we will use an ActiveX control that encapsulates all of the required functionality. An ActiveX control, or control for short, is a reusable software component that is used by simply adding it to a suitable container — an Excel workbook in our example.

We will use the XMComm ActiveX control, developed by Richard Grier, author of *Visual Basic Programmer's Guide to Serial Communications*. XMComm is a wrapper for Microsoft's Communication (MSComm) control. That is, it provides most or all of the capabilities of MSComm plus some additions.

Because only basic serial communications are needed to create an interface to our balance, either XMComm or MSComm are suitable. Our example uses XMComm, which can be installed for free on any workstation. Alternatively, if you have a design time license for MSComm, which requires an installed Microsoft developer platform such as Visual Basic or Visual Studio, you can use it in place of XMComm with only minor syntax changes in the VBA code.

To download XMComm, visit Grier's XMComm information page at http://ourworld.compuserve.com/homepages/richard_grier/xmcomm.htm. Click on the link "Download XMCommCRC" at the top of the page and, when prompted, save file XMCommCRC.zip (2.2 MB) to your computer. To install XMComm, first double-click on the file XMCommCRC.zip, which should open the file and display its contents in WinZip. Now double-click on the file Setup.exe in the WinZip window and follow the installation instructions accepting all default options.

Balance Serial Communication Basics

Every computer comes with one or more serial ports labeled COM1, COM2, etc. A serial port can be used both to transmit and receive data. Our Excel example will use the serial port to send a command to the balance requesting weighing results and will then receive the result transmitted by the balance. We must understand the balance's command syntax and returned data formats to create our solution. Figure 1 (p. 3) shows the command and data formats for the two balance types in our example.

To request weighing data, we will use the balance's "Send Immediate" command. That is, our workbook will open the serial port and send the character "S" then "I" followed by carriage return and line-feed characters. When we receive the balance's results from the serial port, we then use the first two characters of the data received to detect a stable, unstable, or invalid result. When we receive a stable result, the result value will be extracted from characters 4 through 12 of the data received. Note that the data sent and received is all text, so our workbook must convert the text-based representation of a number into a floating-point value.

A&D Weighing HR Series

Command	*Command Format*
Send data immediately	S I CR LF

Data Type	*Data Format*
Stable result	S T , + 0 0 0 . 0 0 0 0 g CR LF
Unstable result	U S , - 0 9 8 . 3 2 1 0 g CR LF
Overload error	O L , + 9 9 9 9 9 9 9 E + 1 g CR LF

Mettler Toledo AE Series

Command	*Command Format*
Send data immediately	S I CR LF

Data Type	*Data Format*
Stable result	S + 0 0 0 . 0 0 0 0 g CR LF
Unstable result	S D + 0 0 0 . 0 0 0 0 g CR LF
Invalid result	S I + 0 0 0 . 0 0 0 0 g CR LF

Figure 1. Balance Command and Data Formats

Create an Example Workbook

With XMComm installed and the balance connected to the workstation's serial port, we can build an example workbook to demonstrate the interface. Later, you can copy your existing worksheets into a copy of the example workbook to add the necessary infrastructure to your own solutions.

To begin, open Excel with a new blank workbook. Right-click on the "Sheet1" name, select "Rename" from the pop-up menu and change the name to "Results." Rename "Sheet2" to "Settings." Right-click "Sheet3" and select "Delete" to remove the sheet. Save the new workbook naming it "BalanceInterface.xls" or something equally descriptive.

Switch to the "Settings" sheet and add labels in Column A for COM Port, Baud Rate, Parity, Data Bits, Stop Bits, and Terminator. Using your balance's technical documentation, find the required serial communication parameters and enter the appropriate values in the adjacent Column B cells (see Figure 2, p. 4). These values will be used to configure the workstation's serial port to match the balance's communication parameters.

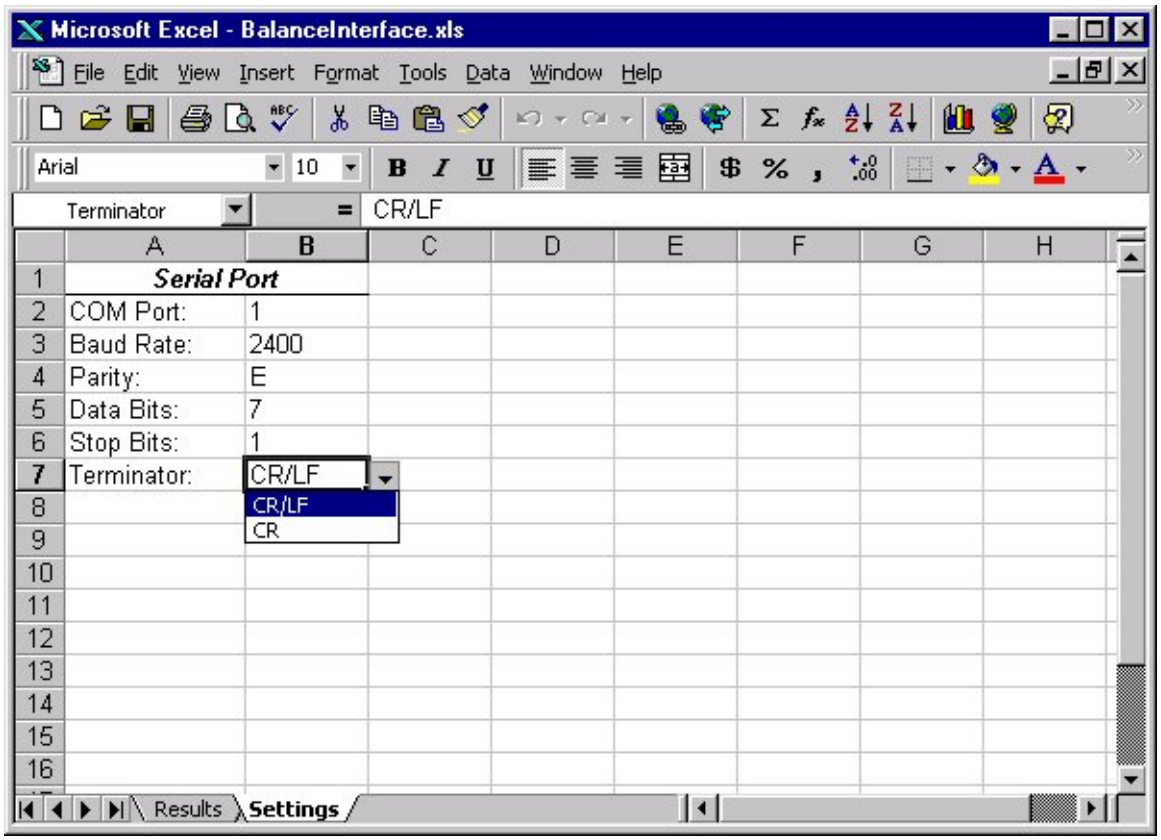

Figure 2. Example of Workbook's Settings Sheet

To enable our VBA code to refer to the Setting sheet's values by name, we will use Excel's named ranges to name the cells for better maintenance and readability. To create the names, select cells A2 through B7, use Insert → Name → Create, and then enable only the "Left Column" option and click "OK." Now click in one of the Column B value cells and you will notice Excel has named the cells COM_Port, Baud_Rate, Parity, Data_Bits, Stop_Bits, and Terminator. Our VBA code is unaffected if these cells are later moved, since the code references the cells by name. Make sure your worksheet matches these names.

Add the XMComm Control to a UserForm

Some ActiveX controls can be dropped directly on an Excel worksheet; however, this technique does not work with all controls and all versions of Excel. To accommodate all versions of Excel since 97, we will add our XMComm control to a UserForm object. In Excel, a UserForm object is most often used to create a custom window or dialog box, but our application does not need a dialog box and, thus, our UserForm will never be displayed. Its sole purpose is to serve as a container to hold the XMComm control and associated VBA code.

To create the UserForm, use Tools → Macro → Visual Basic Editor to open the VBA editor. Use View → Project Explorer to display the project explorer pane if it is not

already visible. Use Insert → UserForm to add a new empty UserForm object named "UserForm1" to our VBA project. We must use the control "Toolbox" to add the XMComm control to UserForm1. Use View → Toolbox to display the Toolbox if it is not already visible. Now add the XMComm control to the Toolbox by using Tools → Additional Controls, checking the option for "XMComCRC.XMCommCRC" and clicking the "OK" button. Click the new "XMCommCRC" control in the Toolbox, then click in UserForm1 to add the control to the form (see Figure 3).

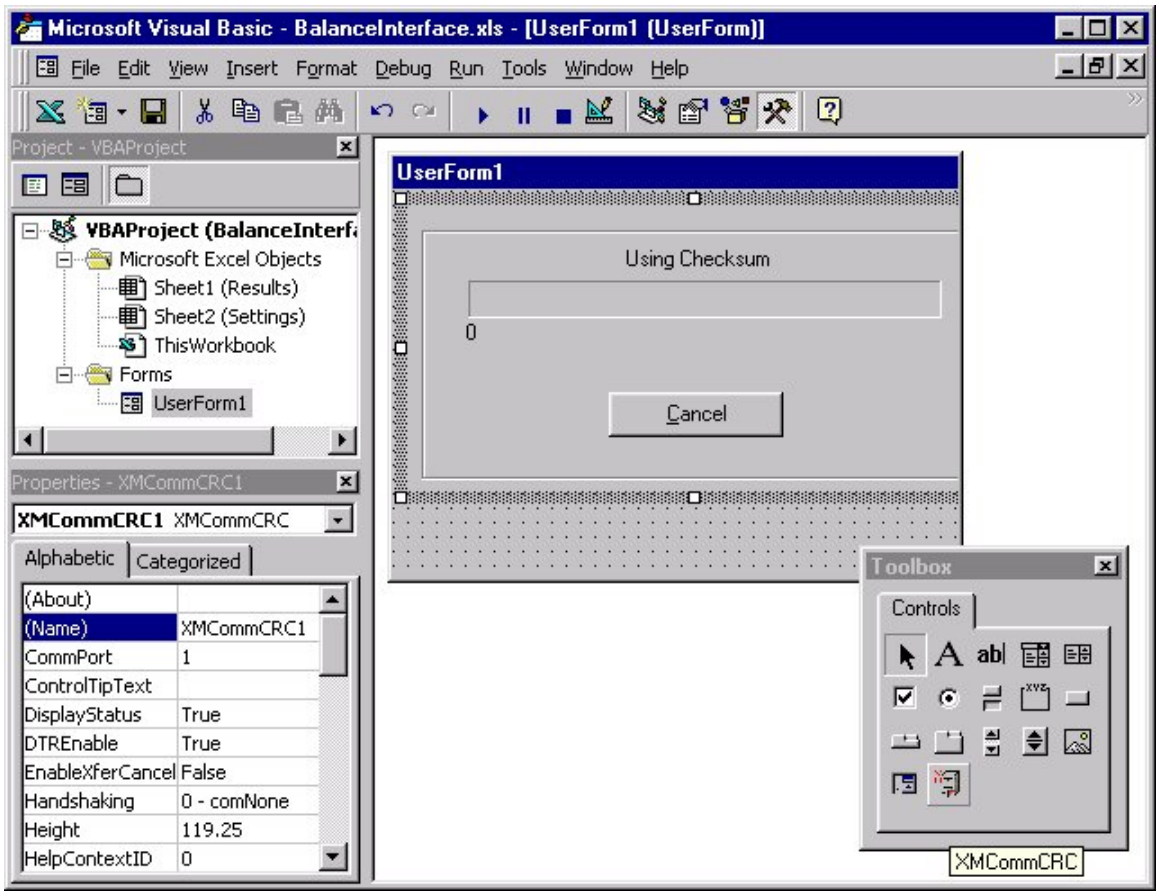

Figure 3. Adding the XMComm Control to our UserForm

Add the UserForm's VBA Code

Now we need to add VBA code that will request the balance's data and receive the results. Right-click within the XMComm control just added to UserForm1 and select "View Code" from the pop-up menu. Excel will automatically add an empty event procedure for the control's OnComm event (see Figure 4, p. 6). The OnComm event handles serial-port communication. That is, when data is received by the workstation's serial port, the OnComm event is triggered so that it can process the data. We will use this event to retrieve the balance's weighing results. Complete the OnComm event procedure by entering the VBA code in Listing 1 (p. 7). Enter the code exactly as it

appears between the "`Private Sub XMCommCRC1_OnComm()`" and "`End Sub`" lines in Listing 1, using tabs to indent for readability.

The OnComm event procedure can handle many different types of serial-port communication events, but here we are only using it to process a data-received event. The code in Listing 1 sends the received data to a buffer until it receives the instrument's termination character, which is either a carriage return or a combination carriage return and line feed. Once the termination character is detected, the code looks at the first two characters of the data to determine whether it has received a stable, unstable, or invalid result. Stable results are extracted from the data buffer, converted to a floating-point value, and added to the active Excel cell. Otherwise, a message indicates an unstable or invalid result was received.

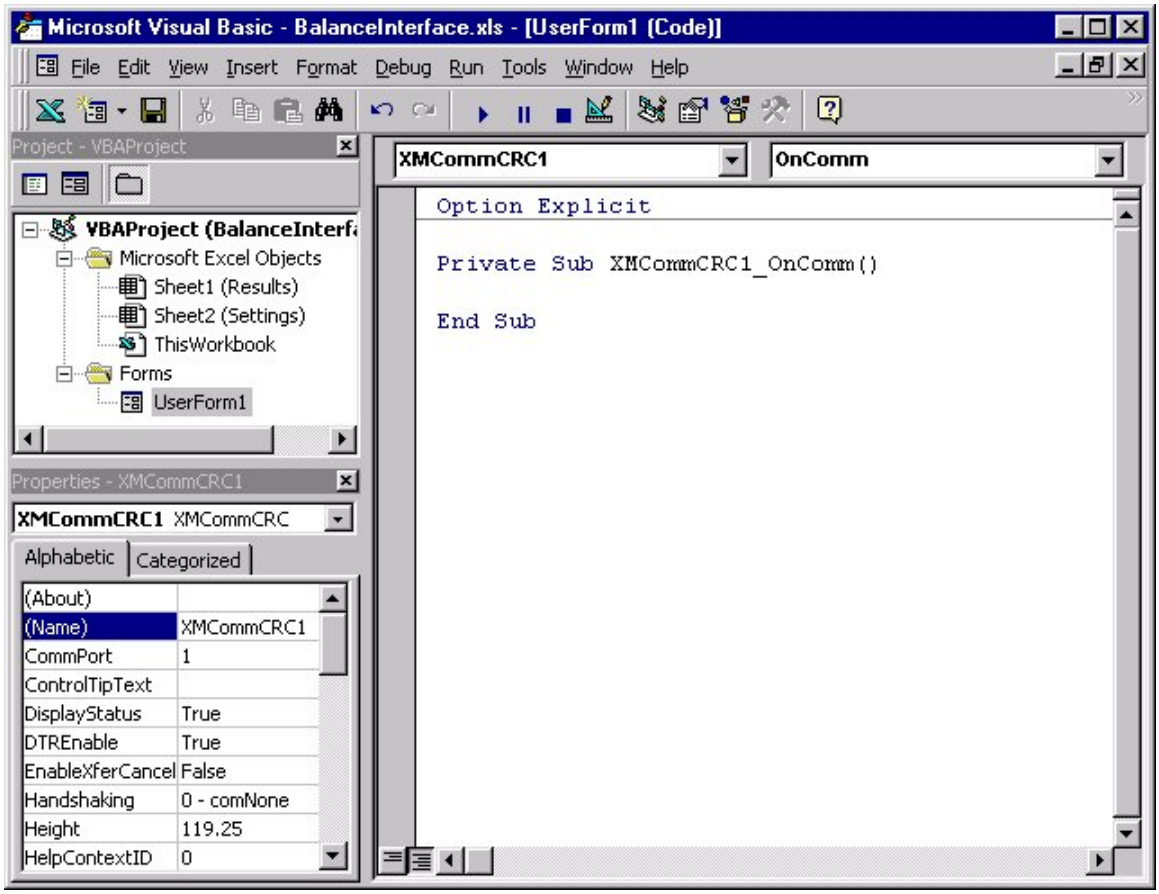

Figure 4. Adding the UserForm's VBA Code

After you enter the code for the OnComm event, proceed to an empty line below the "`End Sub`" line and add all of the code in Listing 2 (p. 8). Where the OnComm event procedure responds to data received from the serial port, the RequestBalanceData procedure initiates the process by sending the balance a command requesting the weighing data. After entering the VBA code in listings 1 and 2 in the UserForm1 code window, use Debug → Compile VBAProject to check for errors. Any syntax errors will be highlighted in the code window. Compare any errors with the code in the listings to

find and correct the problem. When no errors are detected, use File → Close and Return to Microsoft Excel to close the VBA editor.

```
Private Sub XMCommCRC1_OnComm()

    Static sInput As String
    Dim sTerminator As String
    Dim Buffer As Variant

    ' Branch according to the CommEvent property.
    Select Case XMCommCRC1.CommEvent
        Case XMCOMM_EV_RECEIVE
            Buffer = XMCommCRC1.InputData  ' Use Input property for MSComm
            sInput = sInput & Buffer
            If Worksheets("Settings").Range("Terminator") = "CR/LF" Then
                sTerminator = vbCrLf
            Else
                sTerminator = vbCr
            End If
            If Right$(sInput, Len(sTerminator)) = sTerminator Then
                XMCommCRC1.PortOpen = False
                sInput = Left$(sInput, Len(sInput) - Len(sTerminator))
                Select Case Left$(sInput, 2)
                    Case "ST", "S "
                        ActiveCell.Value = CDbl(Mid$(sInput, 4, 9))
                        ActiveCell.Activate
                    Case "US", "SD"
                        MsgBox "The balance is unstable."
                    Case "OL", "SI"
                        MsgBox "The balance is showing an error value."
                End Select
                sInput = ""
            End If
    End Select

End Sub
```

Listing 1

Note that the VBA code in listings 1 and 2 has been created to handle the command and data formats for the balances listed in Figure 1. You may need to alter these procedures to accommodate the command and data formats for your specific instrument. For example, in the code in Listing 1, the following statement detects a stable result for either balance type listed in Figure 1:

```
Case "ST", "S "
```

You will need to change this line if your balance uses a different format. Similar changes may also be required for the lines detecting unstable and invalid results. Also, consider the following statement in Listing 1:

```
ActiveCell.Value = CDbl(Mid$(sInput, 4, 9))
```

This statement converts the nine text characters beginning at Position 4 in the result received (see the Stable result data format in Figure 1) to a numeric value and stores the value in the worksheet's active cell. You will need to alter this statement to accommodate

your balance's data format. The following two statements in Listing 2 transmit the "Send Immediate" command to the balance:

```
XMCommCRC1.Output = "SI" & vbCrLf
XMCommCRC1.Output = "SI" & vbCr
```

Alter these lines as necessary with your balance's appropriate command format. Also note that we have deliberately omitted all error handling for simplicity. After you have successfully tested the example, you may want to consider adding proper VBA error handling for a more robust application (see the On Error statement in Microsoft Visual Basic Help for more information).

```
Public Sub RequestBalanceData()

    With Worksheets("Settings")
        ' Configure and open the COM port
        If Not XMCommCRC1.PortOpen Then
            XMCommCRC1.RThreshold = 1
            XMCommCRC1.RTSEnable = True
            XMCommCRC1.CommPort = .Range("COM_Port")
            XMCommCRC1.Settings = .Range("Baud_Rate") & "," & _
                                  .Range("Parity") & "," & _
                                  .Range("Data_Bits") & "," & _
                                  .Range("Stop_Bits")
            XMCommCRC1.PortOpen = True
        End If

        ' Send balance's "SI" (Send Immediate) command
        ' to request weighing data immediately
        If .Range("Terminator") = "CR/LF" Then
            XMCommCRC1.Output = "SI" & vbCrLf
        Else
            XMCommCRC1.Output = "SI" & vbCr
        End If
    End With

End Sub
```

Listing 2

Completing Our Example

The final step is to add a command button, which users click to retrieve and add the balance's weighing result to the current worksheet cell. Switch to our example's Results sheet then use View → Toolbars → Control Toolbox to display the Control Toolbox if it is not already visible. To add a button to the worksheet, click the "Command Button" option on the Control Toolbox, then click anywhere on the worksheet to add the new button. Now right-click the button and choose "Properties" to display the property window. Set the Accelerator property to "B" and the Caption property to "Get Balance Data," then use the grippers to resize the button as necessary (see Figure 5, p. 9). The Accelerator property sets the button's accelerator key and underlines the key in the button's caption. Users can either click the button with the mouse or use "Alt+B" to select the button. Close the Properties window.

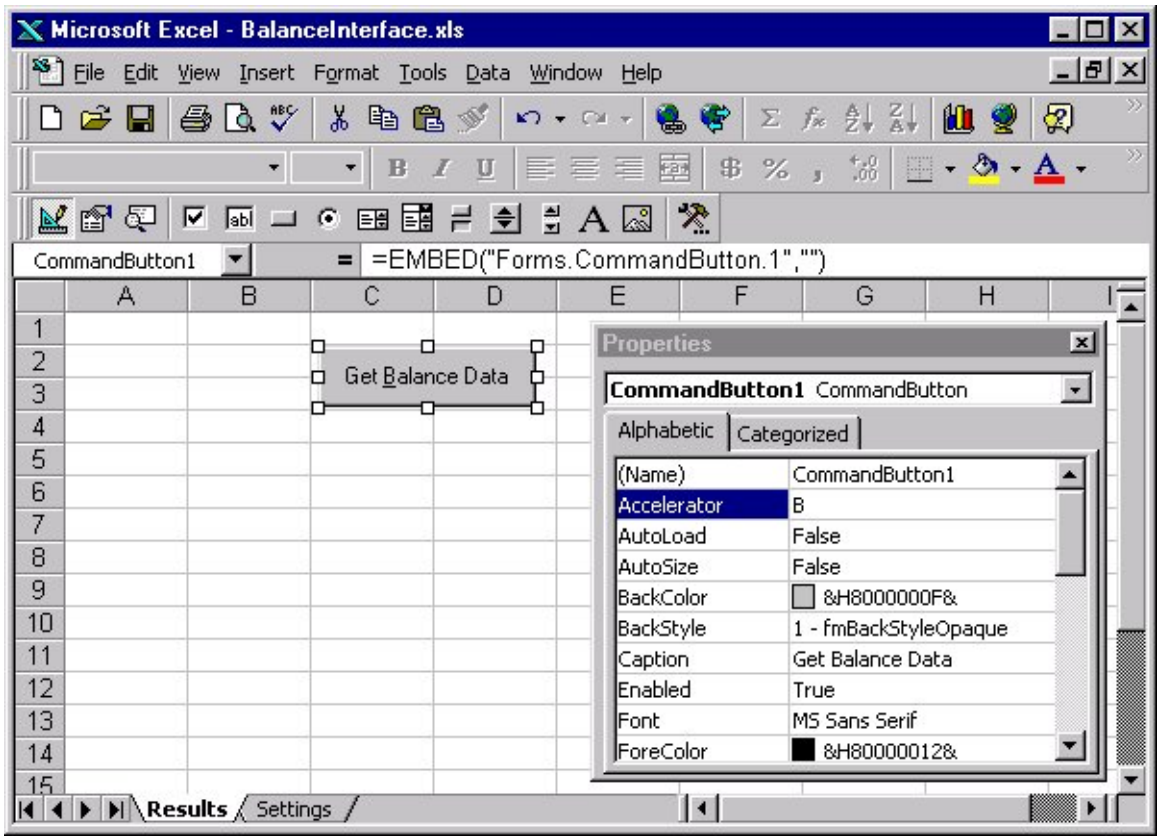

Figure 5. Adding a Command Button

Finally, we need to add a VBA event procedure for the command button's "Click event." Right-click the button and choose "View Code" from the pop-up menu. Excel will open the VBA editor and insert an empty Click event procedure. Complete the event procedure by entering the single line of code shown below between the `Private Sub` and `End Sub` statements as shown:

```
Private Sub CommandButton1_Click()
    UserForm1.RequestBalanceData
End Sub
```

Use File → Close and Return to Microsoft Excel to close the VBA editor. In the Control Toolbox, locate and click "Exit Design Mode," then save your workbook and you are finished. To test the example, simply select any cell on the Results worksheet and click the "Get Balance Data" button or use "Alt+B," and your balance's current weighing result will appear in the cell.

Let's review what occurs to make it all work. When the "Get Balance Data" button is clicked, the button's click event is triggered. The single line of VBA code we added to the event procedure above invokes the RequestBalanceData procedure we added to our UserForm1 (Listing 2). RequestBalanceData opens the serial port specified in the COM_Port named range on our Settings sheet then configures the port's settings by

retrieving these values using the named ranges on the Settings sheet. RequestBalanceData concludes by transmitting the "Send Immediate" command to the instrument. The balance responds to the command and sends the weighing result to the serial port. The OnComm event procedure we added to the communication control (see Listing 1) accepts the data and, following receipt of the termination character, extracts the weighing result, converts it to a floating-point value, and inserts it in the worksheet's active cell.

Putting It To Work

Some software provided by balance vendors and third parties may require the user to initiate the transfer of the balance's data either from the balance's keypad or from the middleware software. Our solution, with fewer components, is simpler and requires only a single keystroke or mouse click to insert the data in an Excel cell.

You can easily add our example's balance interface to your existing workbooks. Copy an existing worksheet into a copy of our example workbook, then delete the example's Results sheet, and add a new Get Balance Data button and its single line of VBA code.

Using these techniques, you can quickly integrate your lab's balances with Excel without purchasing any additional software, use a familiar spreadsheet tool, and eliminate costly error-prone manual data entry.

Rick Collard, *software engineer, is the founder of Mountain States Consulting LLC (Jackson, Wyoming), www.msc-lims.com. Reach him at 307-733-1442 or info@msc-lims.com.*

This article originally appeared in the June/July 2004 issue of *Water Environment Laboratory Solutions* published by the Water Environment Federation® (Alexandria, Virginia), Hwww.wef.orgH.