Creating and Using a Fixture Unit Versus Flow Calculator

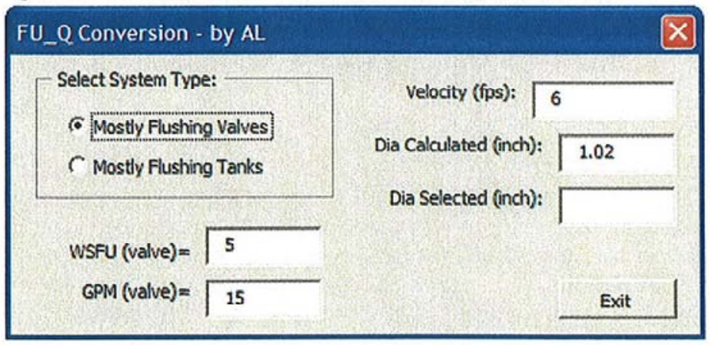
Anjian Lu, CPD

The Hunter's Curve Method is the basis for sizing water supply systems. However, calculating a flow (Q) based on a water supply fixture unit (WSFU) to size numerous pipes in a plan or a riser diagram can be tedious. The WSFU→Q conversion calculator discussed in this column, created using Microsoft Excel's Visual Basic Application (VBA) feature, can resolve this problem.

The WSFU→Q Calculator

The calculator (see Figure 1) offers two system type options: mostly flushing valves and mostly flushing tanks. When the user inputs the WSFU in the first text box on the left, the flow in gallons per minute shows in the text box below. The first text box on the right side of the calculator shows the default velocity, and the second box outputs

Figure 1. The FU→Q Converter



FU_Q Conversion - by AL

Select System Type:
[✓ Mostly Flushing Valves]
Mostly Flushing Tanks

WSFU (valve)= 5
GPM (valve)= 15

Velocity (fps): 6
Dia Calculated (inch): 1.02
Dia Selected (inch):

the calculated diameter based on the preset velocity of six feet per second. In the third box the user can select a diameter based on actual conditions. Clicking the exit button terminates the calculation, as does clicking the red X in the upper right corner.

To use the calculator, first you need to create a database of the WSFUs and flows from the Hunter's Curve sizing guide. To do so, we are going to use Excel's range and name scheme and then assign data to the variables.

Create a workbook (named Topic 2 in this example) and save it in a directory. The workbook will contain three worksheets by default. In sheet 3, type the WSFU and flow data as shown in Figure 2. Highlight range D9:E50 and name it rgFUv, and highlight range B5: C50 and name it rgFUt. To name the range, you need only to type the name in the Name Box in the upper left corner, right above the column header row (A, B, C, ...).

Creating the User Form and Controls

In Excel, the form shown in Figure 1 is called the UserForm and the frames, boxes, buttons, and labels are called controls.

When you click the Visual Basic Editor icon on the Visual Basic toolbar, the screen converts to Microsoft Visual Basic. Figure 3 shows the icon location on the toolbar. (Note: I am using Excel 2003 in this article. Icons and screens may look slightly different in other versions.) Highlight VBAProject (Topic 2) in the Project Explorer window on the left side of the screen. (If the window did not appear automatically, click View in the main menu and select Project Explorer from the pull-down menu.) Then select UserForm from the Insert menu. A form called UserForm1 and a toolbox should appear on the screen (see Figure 4). If a toolbox does not show, click the Control Toolbox icon on the Standard toolbar.

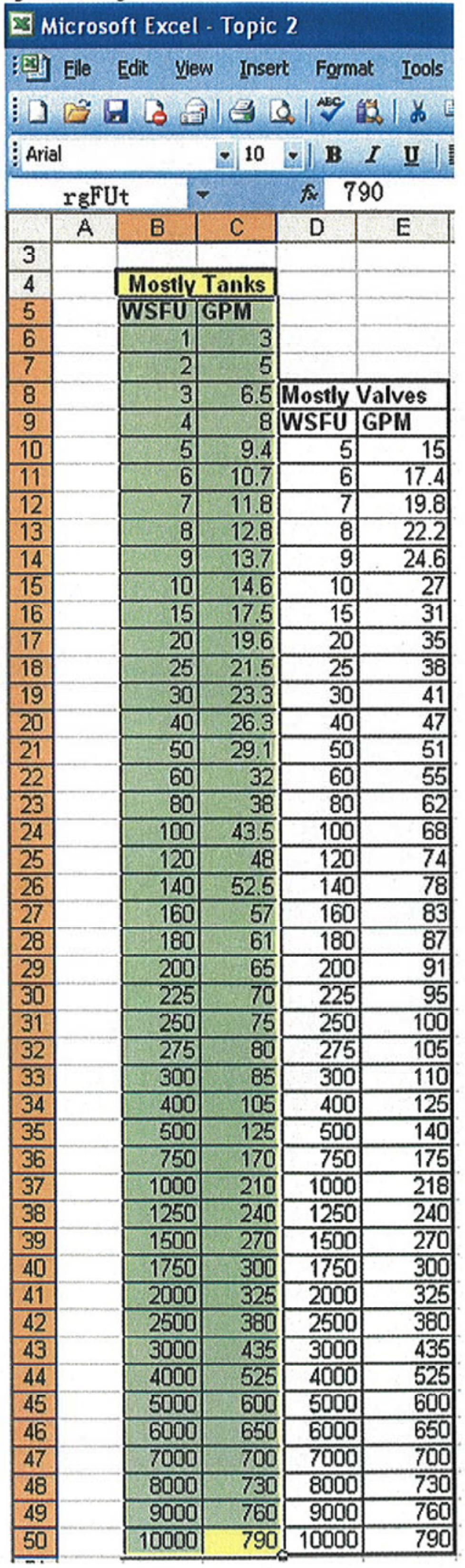
Click on Userform1 and then select Properties window from the View pull-down menu. The Properties window of Userform1 will show on the screen (see Figure 5). Rename UserForm1 as frm-FUQconvert and the caption as FU_Q Conversion. From the Toolbox menu, drag and drop controls on the form. Rename them as listed in Table 1. The UserForm now looks like that shown in Figure 6.

Writing Codes for the Controls

Now you need to write codes for the controls to make them work.

frmFUQconvert. Double clicking on the form leads you to the code

Figure 2. Ranges and Names



rgFUt fx 790

	A	B	C	D	E
3					
4		Mostly Tanks			
5		WSFU	GPM		
6		1	3		
7		2	5		
8		3	6.5	Mostly Valves	
9		4	8	WSFU	GPM
10		5	9.4	5	15
11		6	10.7	6	17.4
12		7	11.8	7	19.8
13		8	12.8	8	22.2
14		9	13.7	9	24.6
15		10	14.6	10	27
16		15	17.5	15	31
17		20	19.6	20	35
18		25	21.5	25	38
19		30	23.3	30	41
20		40	26.3	40	47
21		50	29.1	50	51
22		60	32	60	55
23		80	38	80	62
24		100	43.5	100	68
25		120	48	120	74
26		140	52.5	140	78
27		160	57	160	83
28		180	61	180	87
29		200	65	200	91
30		225	70	225	95
31		250	75	250	100
32		275	80	275	105
33		300	85	300	110
34		400	105	400	125
35		500	125	500	140
36		750	170	750	175
37		1000	210	1000	218
38		1250	240	1250	240
39		1500	270	1500	270
40		1750	300	1750	300
41		2000	325	2000	325
42		2500	380	2500	380
43		3000	435	3000	435
44		4000	525	4000	525
45		5000	600	5000	600
46		6000	650	6000	650
47		7000	700	7000	700
48		8000	730	8000	730
49		9000	760	9000	760
50		10000	790	10000	790

Table 1. The Controls

Control	Name	Caption	Value
UserForm	frmFUQconvert	FU→Q Conversion	
Frame	fraSelectSys	Select System Type	
OptionButton	optValve	Mostly Flushing Valves	True
OptionButton	optTank	Mostly Flushing Tanks	False
Label	lblInput	(Blank)	
Label	lblResult	(Blank)	
Label	lblVelocity	Velocity (fps)	
Label	lblDiameter	Dia Calculated (inch)	
Label	lblDiaSelect	Dia Selected (inch)	
TextBox	txtInput		
TextBox	txtResult		
TextBox	txtVelocity		6
TextBox	txtDiameter		
TextBox	txtDiaSelected		
CommandButton	cmdExit	Exit	

section and a subprocedure with only two lines. First, write the following sentences above the first line.

```
Option Explicit
Const pi = 3.1415926535

Private Sub UserForm_
Initialize()

End Sub
```

Then enter codes above the last line, so the whole paragraph reads like this:

```
Option Explicit
Const pi = 3.1415926535

Private Sub UserForm_
Initialize()
  optValve.Value = True
  optTank.Value = False
  lblResult.Caption = “GPM=
"
  lblInput.Caption = “WSFU
(valve)= "
  txtResult.Locked = True
  txtInput.Text = 5
End Sub
```

Only one Option Explicit line should be listed at the beginning, or you will get an error message when running the program. The second sentence defines a constant pi for calculating the circular pipe sectional area. The fourth line tells Excel that the mostly flushing valves option will be our default choice. Line seven shows the label caption for the mostly flushing valves choice. Line eight locks the txtResult output box. Line nine is for setting the WSFU, which you can change as required.

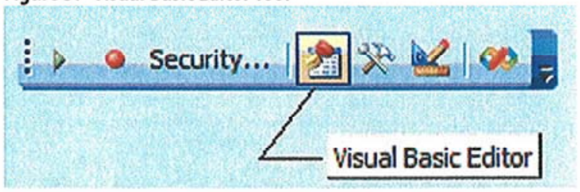
optValve and optTank. Double click the optValve option control and write one sentence, GetGPM, between the two lines automatically created. (Note: There are other ways to bring up the code page. You can select Code from the View pull-down menu. You also can select View Code from the Project Explore window, which you bring up by selecting Project Explore from View on the main menu.)

```
Private Sub
optValve_Click()
  GetGPM
End Sub
```

Do the same for optTank.

```
Private Sub optTank_Click()
  GetGPM
End Sub
```

Figure 3. Visual Basic Editor Tool

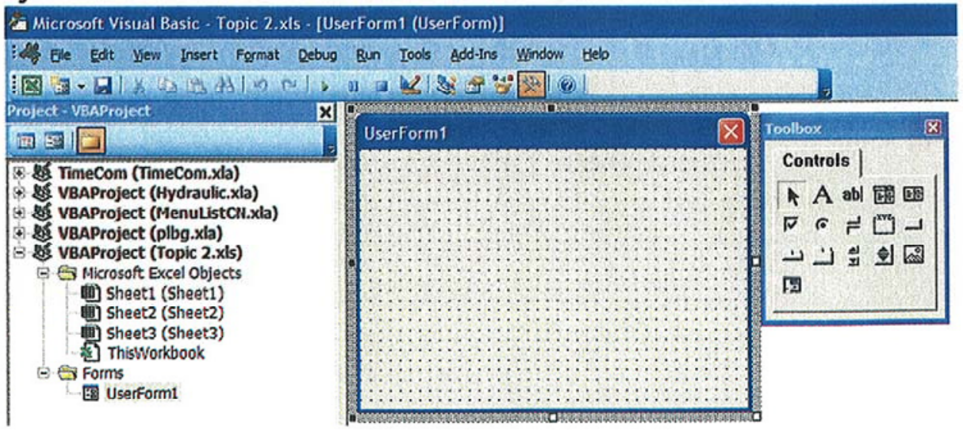


GetGPM is a subprocedure that calculates the flow based on the inputted WSFU. It will be shared by other controls. (Creating the GetGPM subprocedure is discussed later.)

txtInput. Similarly, double click the txtInput text box. Two lines are created automatically. Add GetGPM between them.

```
Private Sub txtInput_
Change()
  GetGPM
End Sub
```

Figure 4. Visual Basic Editor Screen



txtResult and txtVelocity. Double click on the txtResult and txtVelocity text boxes and enter CalcDiameter as follows.

```
Private Sub txtResult_
Change()
  CalcDiameter
End Sub

Private Sub txtVelocity_
Change()
  CalcDiameter
End Sub
```

Notice that the first sentence of the last three subprocedures ends with Change() instead of Click(). This tells Excel VBA to respond to any inputted change.

txtDiaSelected. Double click the txtDiaSelected text box and write the following codes between the two lines created automatically.

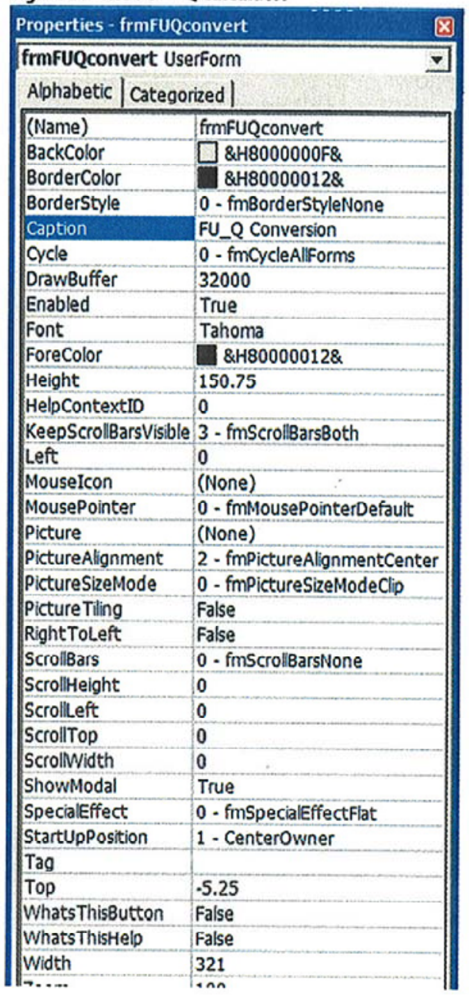
```
Private Sub txtDiaSelected_
Change()
  Dim Q As Double
  Dim V As Double
  Dim D As Double

  On Error GoTo ErrorTrap
  D = txtDiaSelected.Text
  D = D / 12
  Q = txtResult.Text
  Q = Q * 0.002228
  V = (4 * Q / (pi * D ^ 2))
  txtVelocity.Text =
WorksheetFunction.RoundUp(V,
2)
  Exit Sub
ErrorTrap:

End Sub
```

In this subprocedure we define variables for holding the diameter (D) and flow (Q) (shown in the output txtResult.Text text box), as well as the velocity (V).

Figure 5. The FU→Q Calculator



Properties - frmFUQconvert

frmFUQconvert UserForm

Alphabetic | Categorized

(Name)	frmFUQconvert
BackColor	□ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	FU_Q Conversion
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	150.75
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth
Left	0
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PictureAlignment	2 - fmPictureAlignmentCenter
PictureSizeMode	0 - fmPictureSizeModeClip
PictureTiling	False
RightToLeft	False
ScrollBars	0 - fmScrollBarsNone
ScrollHeight	0
ScrollLeft	0
ScrollTop	0
ScrollWidth	0
ShowModal	True
SpecialEffect	0 - fmSpecialEffectFlat
StartUpPosition	1 - CenterOwner
Tag	
Top	-5.25
WhatsThisButton	False
WhatsThisHelp	False
Width	321

Line eight converts the inputted diameter from inches to feet. The calculated velocity to two decimal points then shows in the text box.

The sentence On Error GoTo ErrorTrap is very important because it will trap any input error in the txtDiaSelected text box to avoid a data type mismatch, which could cause the program to crash. Be sure to add the sentence Exit Sub before ErrorTrap:.

cmdExit. You can write a command to end the program. It is simplest to write the word End.

```
Private Sub cmdExit_Click()
  End
End Sub
```

Writing Common Procedures

As mentioned above, GetGPM is a common subprocedure. Let's add the following:

```
Private Sub GetGPM()
  Dim dGPM As Double

  txtVelocity.Text = 6
  txtDiaSelected.Text = ""
  If optValve.Value = True
Then
    lblInput.Caption = "WSFU
(valve)= "
    lblResult.Caption = "GPM
(valve)= "

    On Error GoTo ErrorTrap
    dGPM = QvFU(txtInput.
Text)
    txtResult.Text =
WorksheetFunction.
Round(dGPM, 1)
  Else
    lblInput.Caption = "WSFU
(tank)= "
    lblResult.Caption = "GPM
(tank)= "

    On Error GoTo ErrorTrap
    dGPM = QtFU(txtInput.
Text)
    txtResult.Text =
WorksheetFunction.
Round(dGPM, 1)
  End If
  Exit Sub

ErrorTrap:
  txtInput.Text = 5
  MsgBox "Please enter a
number", vbCritical
End Sub
```

In this subprocedure you define the variable dGPM for holding the flow calculated by the functions QvFU() and QtFU(). Line three defines the velocity at six fps, which you can change based on the selected diameter. Line four clears the diameter selected in the previous operation. The If – Else – End If condition procedure identifies the choice you selected. It will change the labels for input and output text boxes accordingly and calculate flow in gpm based on either the QvGPM or QtGPM function. Lines eight and 14 are for catching type mismatch errors. If an error occurs, the error trap will set the inputted WSFU at five, and the warning "Please enter a number" will appear.

To suggest a diameter based on a six-fps flow velocity, you write the following subprocedure:

```
Private Sub CalcDiameter()
  Dim Q As Double
  Dim V As Double
  Dim D As Double

  On Error GoTo ErrorTrap
  V = txtVelocity.Text
  Q = txtResult.Text
  Q = Q * 0.002228
  D = (4 * Q / (pi * V)) ^
0.5 * 12
  txtDiameter.Text =
WorksheetFunction.RoundUp(D,
2)

  Exit Sub
ErrorTrap:

End Sub
```

Writing Functions

Functions written by users are called User Defined Functions (UDFs). The difference between a subprocedure and a function is that the latter returns a value. For the program to return the flow in gpm when you input a WSFU, you need to write a UDF for the mostly flushing valves water supply system as follows.

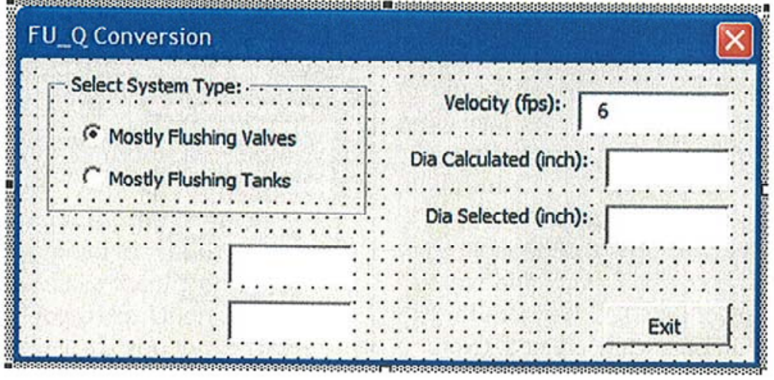
```
Function QvFU(SFUvalve As
Single) As Single

  Dim vaSFUv As Variant
  Dim I As Integer

  vaSFUv = ThisWorkbook.
Sheets(3).Range("rgFUv")

  For I = LBound(vaSFUv) + 2
To UBound(vaSFUv)
```

Figure 6. The FU→Q Calculator



Select System Type: [✓ Mostly Flushing Valves] Mostly Flushing Tanks

Velocity (fps): 6

```
    If SFUvalve = 10000 Then
QVFU = 790
    If SFUvalve < 5 And
SFUvalve <> 0 Then QVFU = 15
    If (vaSFUv(I, 1) >
SFUvalve And vaSFUv(I - 1,
1) <= SFUvalve) Then
       QVFU = vaSFUv(I
- 1, 2) + (vaSFUv(I, 2) -
vaSFUv(I - 1, 2)) * _
       (SFUvalve - vaSFUv(I
- 1, 1)) / (vaSFUv(I, 1) -
vaSFUv(I - 1, 1))
       Exit Function
    End If
  Next I
End Function
```

The data type of the argument of SFUvalve in function QvFU() is defined as single for accepting the WSFU passed. In the procedure are two variables: vaSFUv and I. The former is a variant for holding the data in range rgFU on sheet 3. vaSFUv is a two-dimensional data type. The first dimension holds WSFUs and the second holds flows, including their headers. Line four assigns Range("rgFUv") to vaSFU. The following For … Next loop checks the inputted WSFU and tries to match the data in vaSFU.

The LBound(vaSFUv) value is 1, which corresponds to the head row; since we want to start from the second row, we set the first I at 3 (i.e., I = LBound(vaSFUv) + 2). Line six is for the last WSFU (i.e., if WSFU=10,000; Q=790 GPM). Line seven sets Q=15 gpm when the WSFU is less than five but not zero. The following If — End If condition section locates the inputted WSFU between two numbers in the array. We use interpolation to calculate the flow in gpm. The calculation ends after this, and the value returns to the calling sentence.

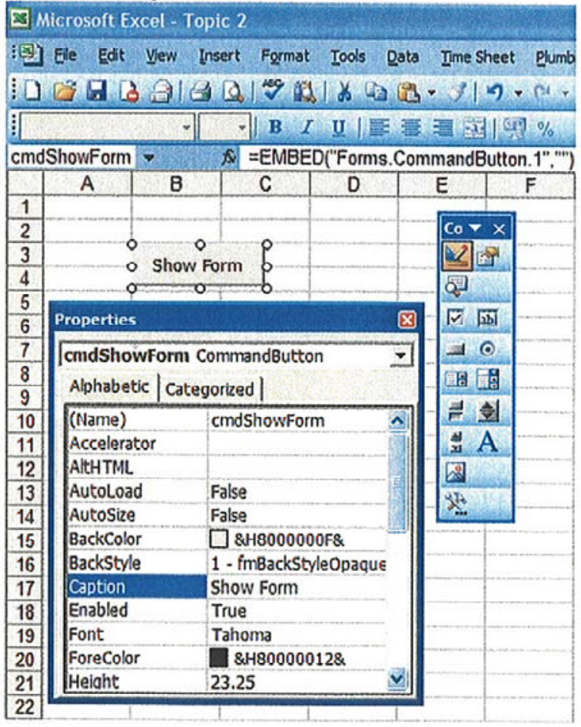
Similarly, for the system with mostly flushing tanks we write the following function:

```
Function
QVFU(SFUvalve As
Single) As Single
  Dim vaSFUv As Variant
  Dim I As Integer
  vaSFUv = ThisWorkbook.
Sheets(3).Range("rgFUv")
  For I = LBound(vaSFUv) + 2
To UBound(vaSFUv)
    If SFUvalve = 10000 Then
QVFU = 790
    If SFUvalve < 5 And
SFUvalve <> 0 Then QVFU = 15
    If (vaSFUv(I, 1) >
SFUvalve And vaSFUv(I - 1,
1) <= SFUvalve) Then
       QVFU = vaSFUv(I
- 1, 2) + (vaSFUv(I, 2) -
vaSFUv(I - 1, 2)) * _
       (SFUvalve - vaSFUv(I
- 1, 1)) / (vaSFUv(I, 1) -
vaSFUv(I - 1, 1))
       Exit Function
    End If
Next I
End Function
```

Writing a Button on the Worksheet

Writing a button on the worksheet is similar to that on the UserForm. Switch the worksheet from sheet 3 to sheet 1. Select the Control Toolbox icon next to the Visual Basic Editor icon (see Figure 3). The Design Mode icon is checked automatically. Select the command button icon in the Control Toolbox and draw a box on the worksheet to place it. Right click the button and select Properties from the pop-up menu. Change the button name from CommandButton1 to cmdShowForm and its caption to Show Form on the properties window (see Figure 7).

Figure 7. Putting Controls on the Worksheet


Microsoft Excel - Topic 2
File Edit View Insert Format Tools Data Time Sheet Plumb
Show Form
Properties
Alphabetic Categorized
Co

cmdShowForm =EMBED("Forms.CommandButton.1","")

cmdShowForm CommandButton

(Name)	cmdShowForm
Accelerator	
AltHTML	
AutoLoad	False
AutoSize	False
BackColor	☐ &H8000000F&
BackStyle	1 - fmBackStyleOpaque
Caption	Show Form
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	23.25

Double click the Show Form button and write the following procedure between the two lines automatically created.

```
Private Sub cmdShowForm_
Click()
  frmFUQconvert.Show
vbModeless
End Sub
```

The keyword vbModeless enables you to work on the worksheet with the User Form shown on the screen. Go back to sheet 1 and click the Exit Design Mode icon on the Visual Basic toolbar.

Be sure to save the workbook. Now when you need to calculate the flow from WSFU, simply open the workbook and click on the Show Form button on sheet 1. The calculator will show up ready for use. ■

Recommended Reading

Green, John, Stephen Bullen and Felipe Martins. *Excel 2000 VBA, Programmer's Reference*. Wrox Press Ltd., 2000.

Green, John, Paul T. Kimmel, Stephen Bullen, Rob Bovey, et al. *Excel 2003 VBA Programmer's Reference*. Wrox Press Ltd., 2004.

Anjian Lu is an engineer with Vanderweil – NCR in Alexandria, Va. To comment on this article, or for more information, contact **articles@psdmagazine.org**.