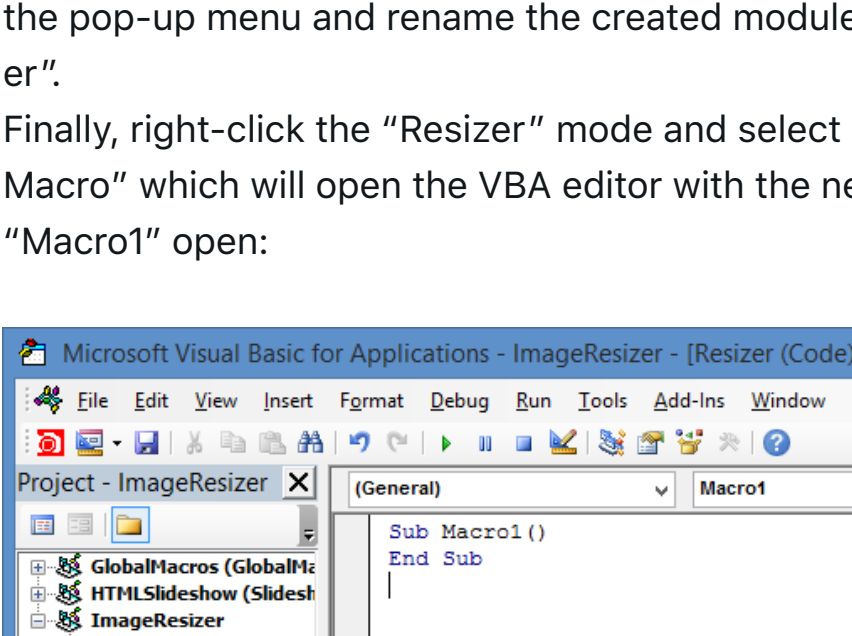
Creating VBA Macros with User Interface in CorelDRAW and Corel DESIGNER

This tutorial demonstrates how to create macros for CorelDRAW Graphics Suite (CDGS) X7 and CorelDRAW Technical Suite (CDTS) X7 using Visual Basic for Applications (VBA) development environment. This can be done in any of the CDGS/CDTS applications such as CorelDRAW, Corel DESIGNER or Corel PHOTO-PAINT.

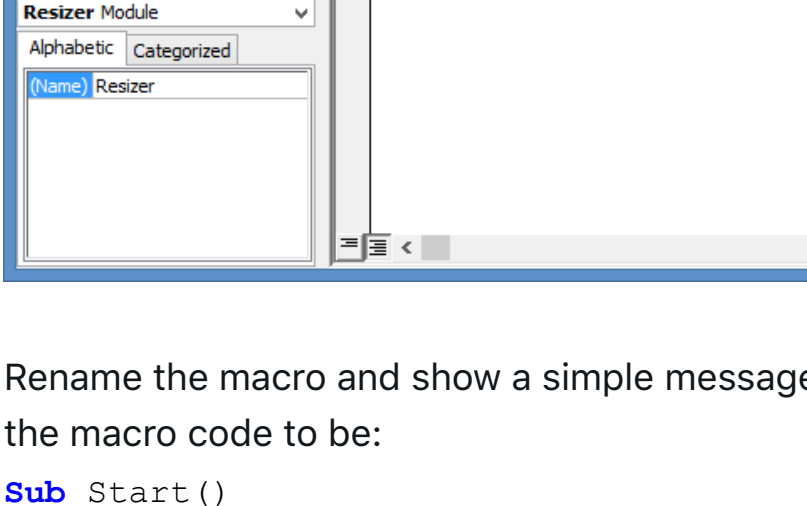
This article walks the user through the steps of creating a simple Image Resizer macro which takes images from a specified folder, opens them in Corel PHOTO-PAINT, resizes them to specific dimensions and saves them to a different folder. Even though the example described below is designed for Corel PHOTO-PAINT X7, the same principles can be easily applied to CorelDRAW and Corel DESIGNER.
The macro project will contain a simple macro that shows a dialog box which allows the user to enter the location of images to process, required dimensions and the output folder.

Creating New Macro Project

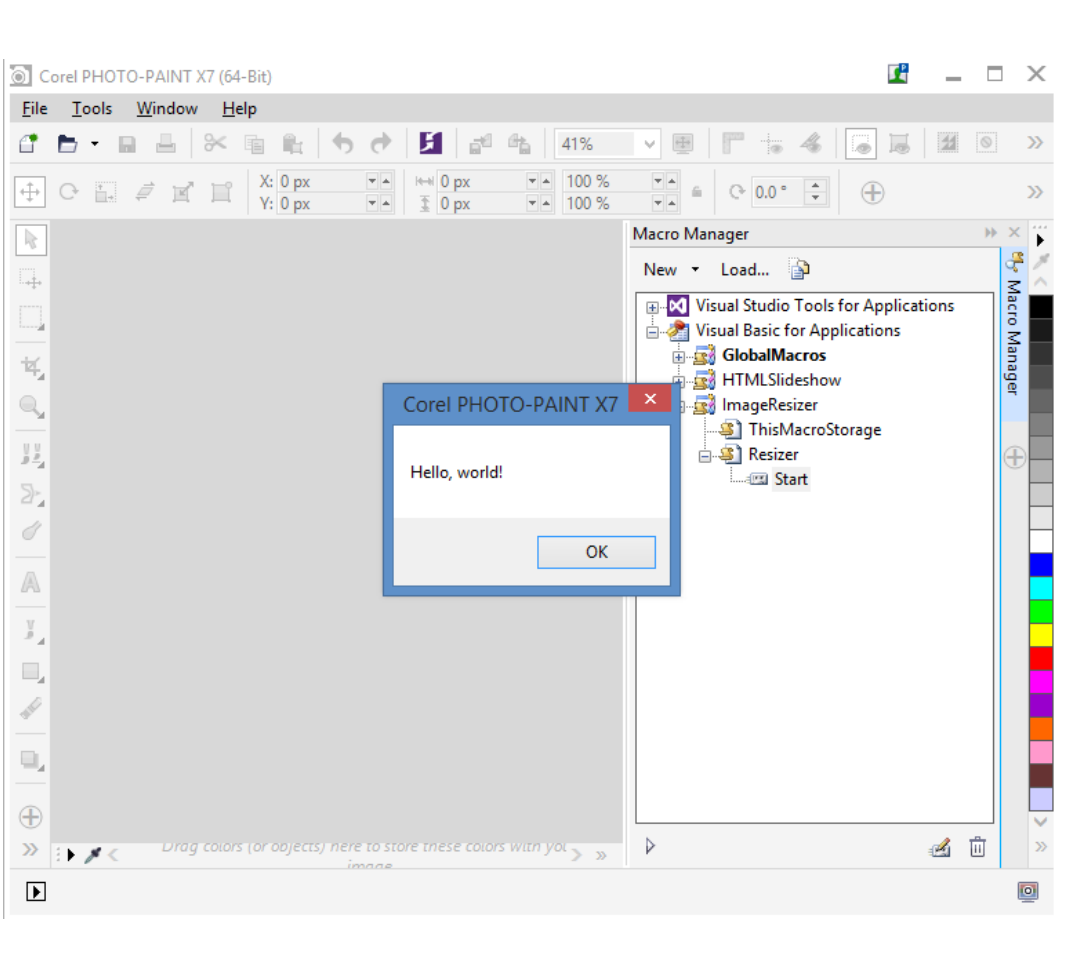
First create a new macro project for Image Resizer macro. For this, open the Macro Manager docker (Alt-Shitf-F11), select “Visual Basic for Applications” in the list, right-click and select “New Macro Project...”



When the “Save As” dialog box appears, enter “ImageResizer” in the file name. This will create a new macro project called ImageResizer.gms and it will be loaded into the Macro Manager docker now:



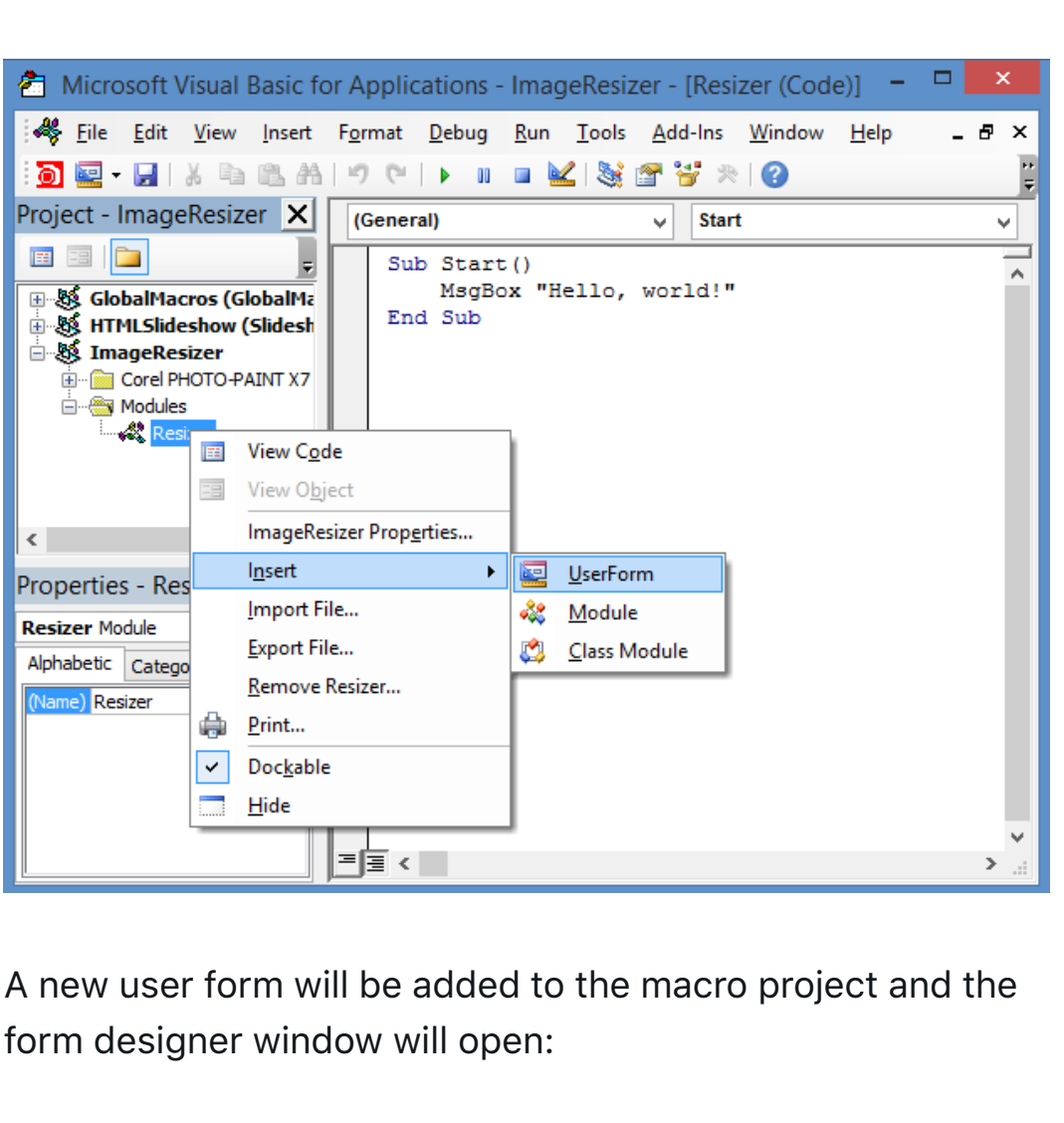
Right-click “ImageResizer” and select “New Module” from the pop-up menu and rename the created module to “Resizer”.
Finally, right-click the “Resizer” mode and select “New Macro” which will open the VBA editor with the new “Macro1” open:



Rename the macro and show a simple message by editing the macro code to be:

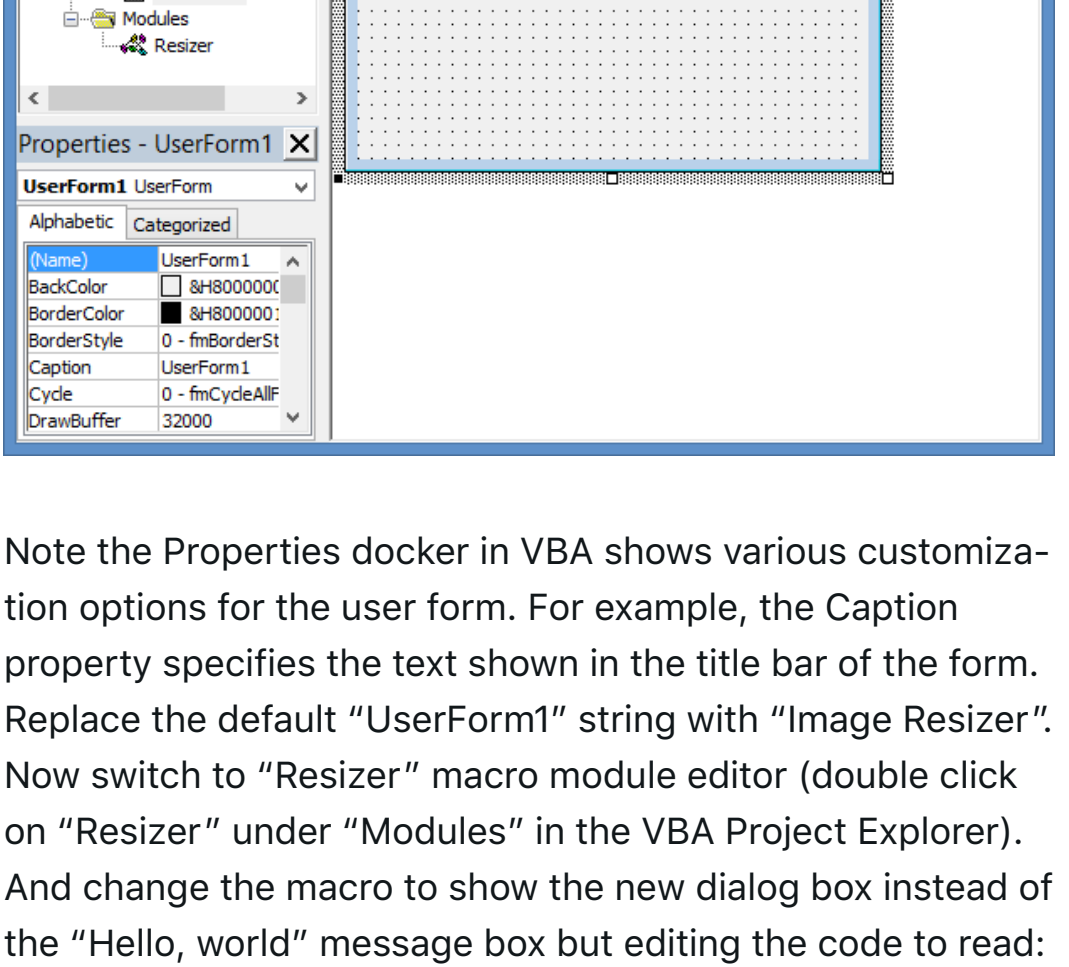
```
Sub Start()
  MsgBox "Hello, world!"
End Sub
```

Now the basic macro project is set up. You can try running this macro by double-clicking the “Start” macro under ImageResizer > Resizer in Macro Manager docker:

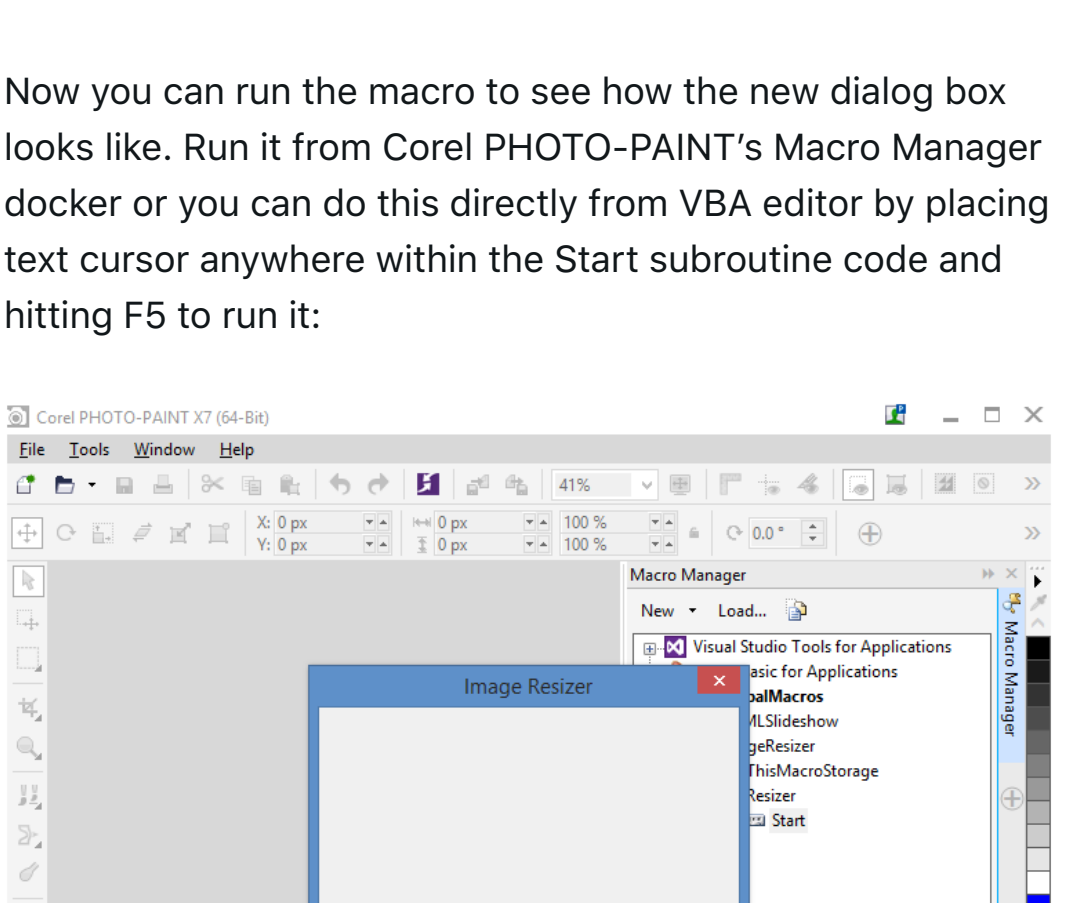


Creating Macro Dialog Box

Now that the macro project is set up, it’s time to add a user interface to it. Switch back to VBA editor, select the “Resizer” module in VBA’s Project Explorer docker (Ctrl-R), right-click it and select Insert > UserForm from the pop-up menu:



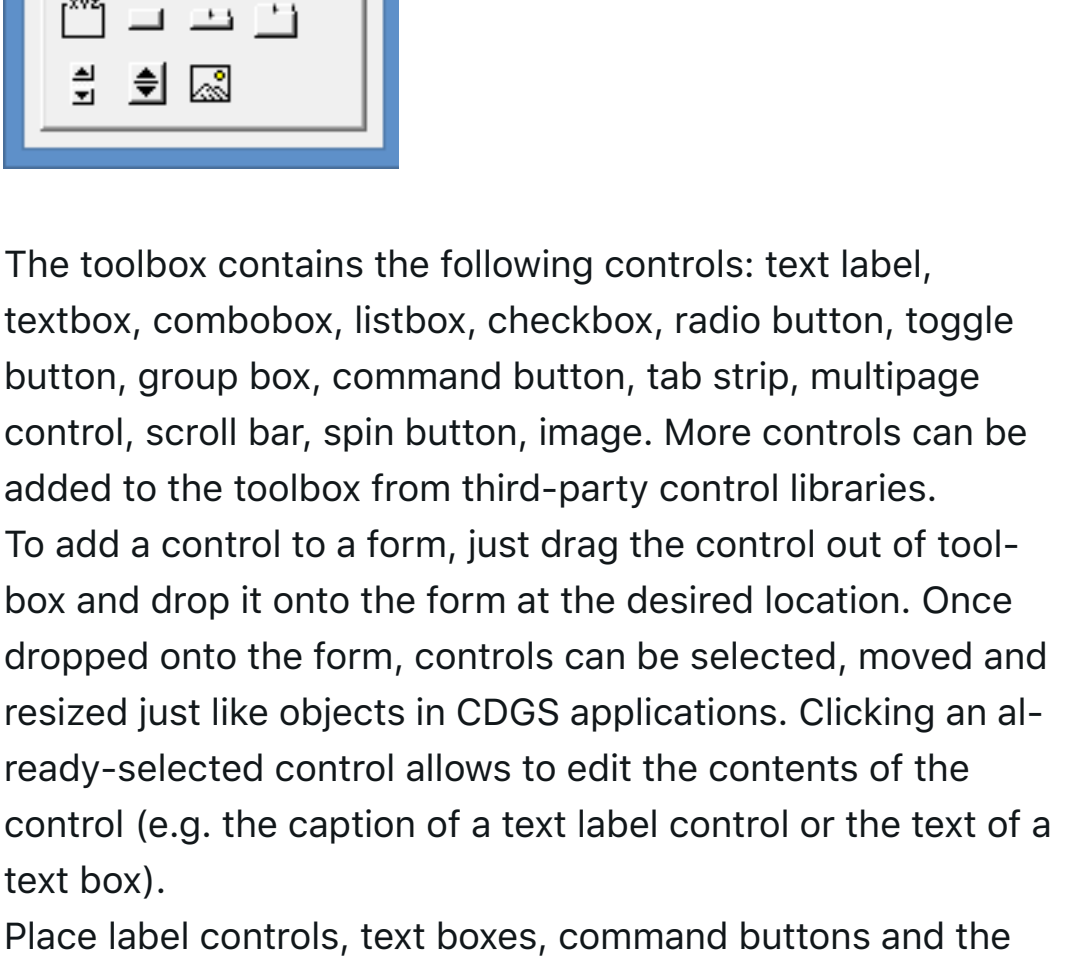
A new user form will be added to the macro project and the form designer window will open:



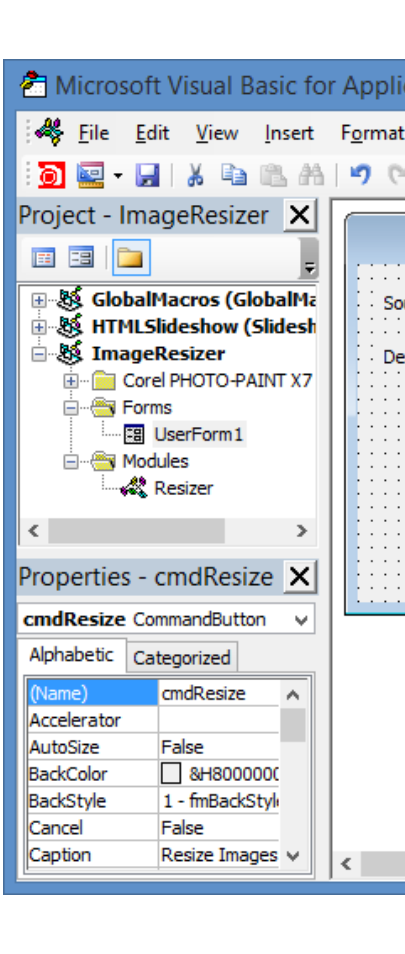
Note the Properties docker in VBA shows various customization options for the user form. For example, the Caption property specifies the text shown in the title bar of the form. Replace the default “UserForm1” string with “Image Resizer”. Now switch to “Resizer” macro module editor (double click on “Resizer” under “Modules” in the VBA Project Explorer). And change the macro to show the new dialog box instead of the “Hello, world” message box but editing the code to read:

```
Sub Start()
  UserForm1.Show
End Sub
```

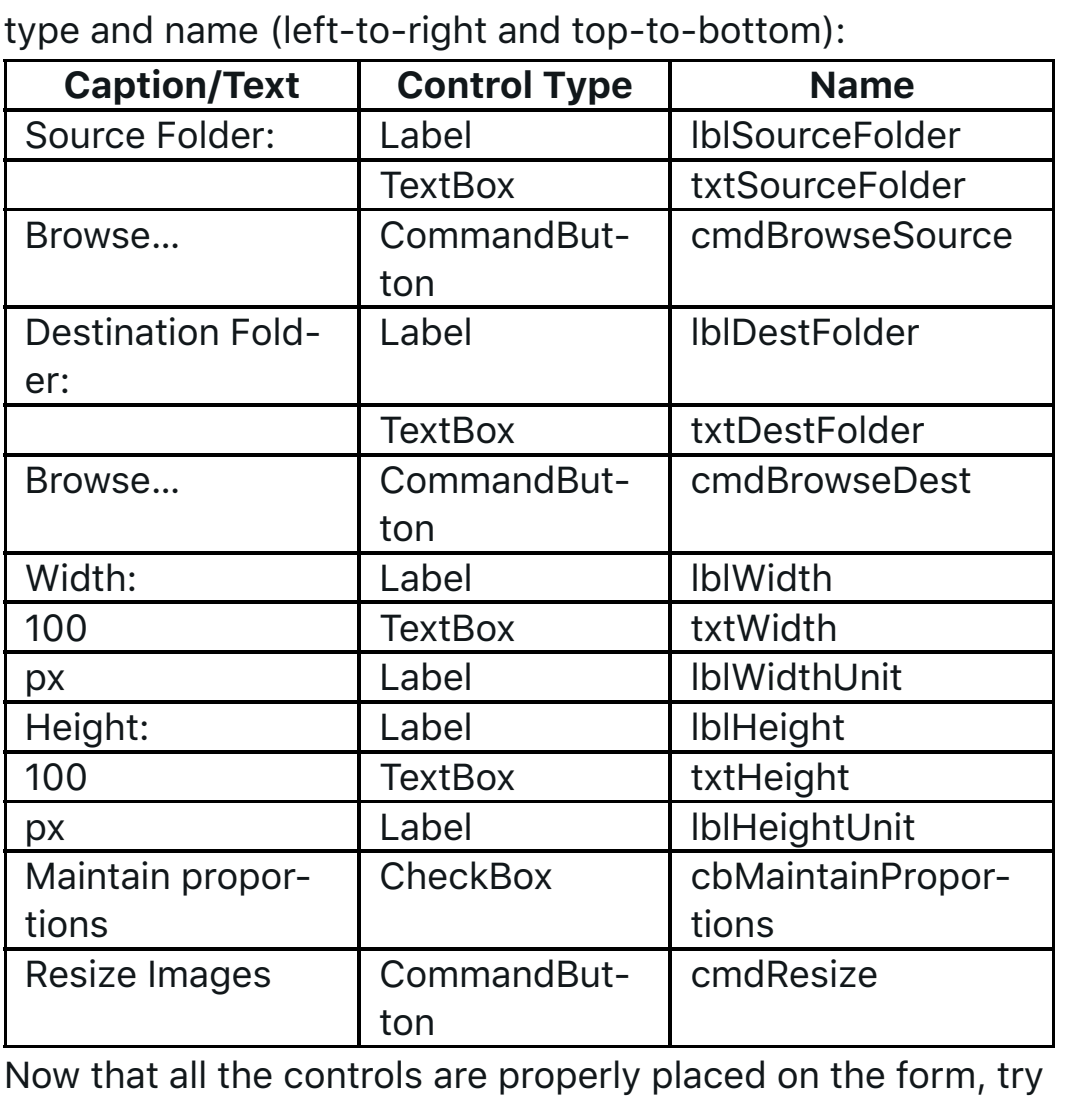
Now you can run the macro to see how the new dialog box looks like. Run it from Corel PHOTO-PAINT’s Macro Manager docker or you can do this directly from VBA editor by placing text cursor anywhere within the Start subroutine code and hitting F5 to run it:



Now that a dialog box can be shown by Image Resizer macro, it’s time to add some controls to it. Toolbox in VBA form editor contains all the user controls currently available:



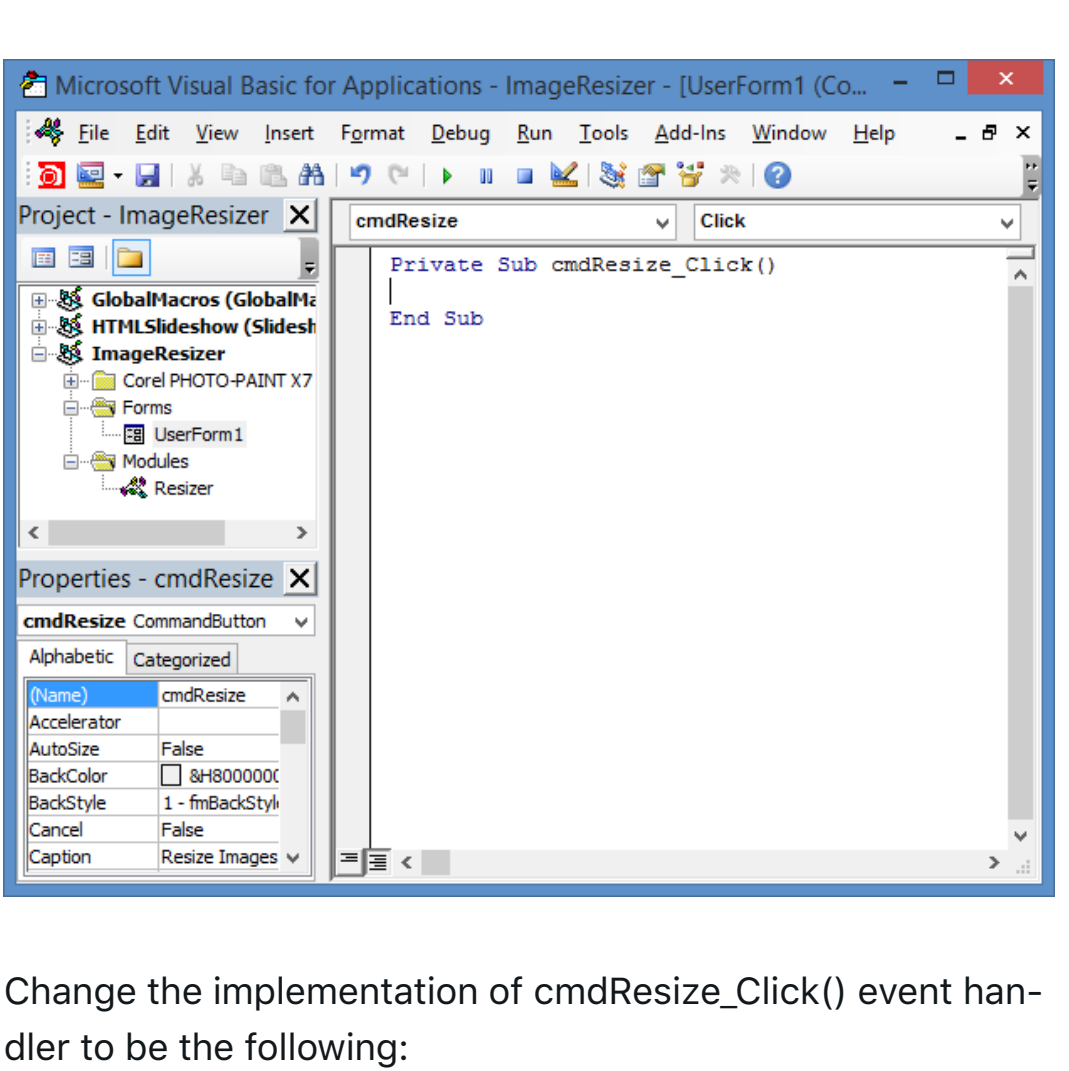
The toolbox contains the following controls: text label, textbox, combobox, listbox, checkbox, radio button, toggle button, group box, command button, tab strip, multipage control, scroll bar, spin button, image. More controls can be added to the toolbox from third-party control libraries.
To add a control to a form, just drag the control out of toolbox and drop it onto the form at the desired location. Once dropped onto the form, controls can be selected, moved and resized just like objects in CDGS applications. Clicking an already-selected control allows to edit the contents of the control (e.g. the caption of a text label control or the text of a text box).
Place label controls, text boxes, command buttons and the checkbox as shown in the following screenshot:



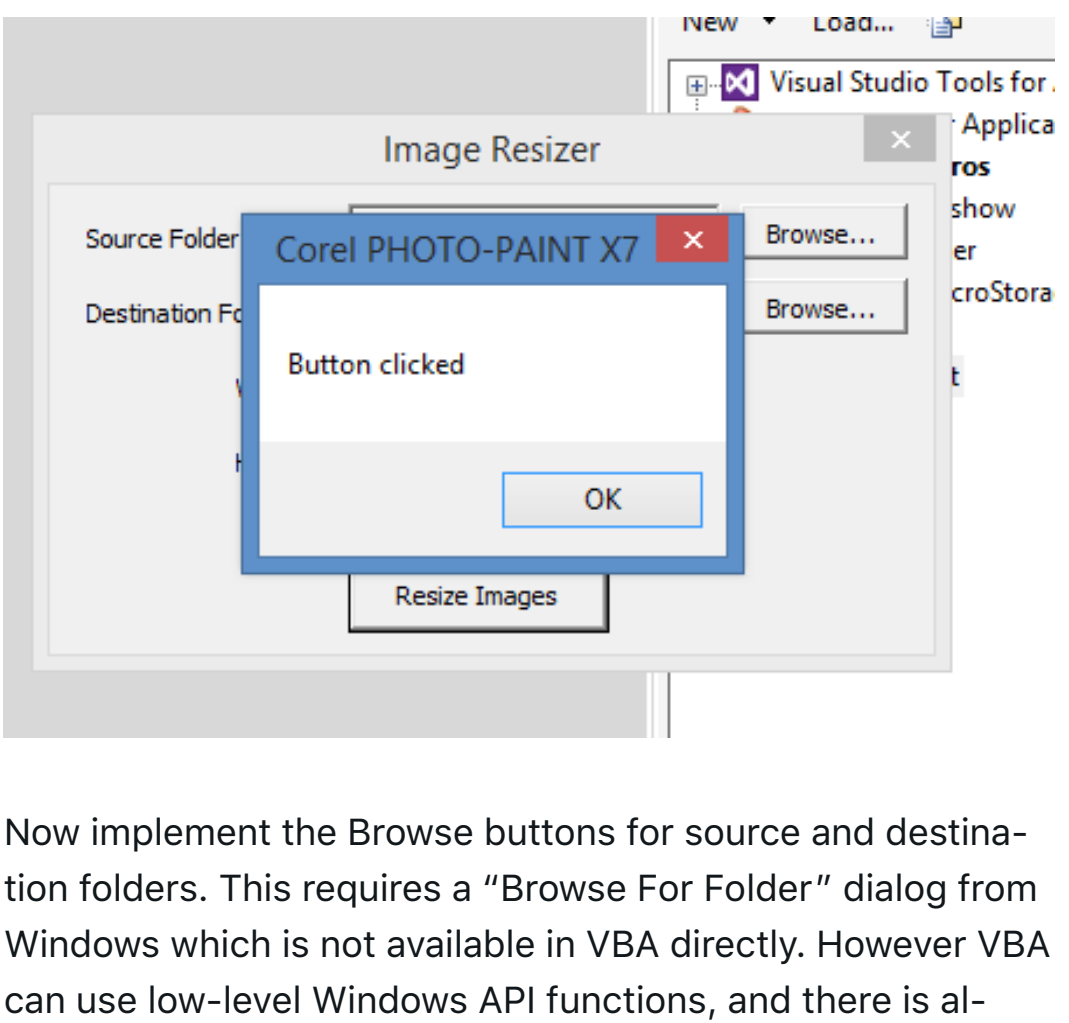
When a control is selected, you can also give it a more descriptive name to identify the control in the macro code, because the automatic generated control names such as TextBox1 and Label2 are not very descriptive. The control name can be specified in the “(Name)” property in the Properties docker.
The following controls are used in the form, including their type and name (left-to-right and top-to-bottom):

Caption/Text	Control Type	Name
Source Folder:	Label	lblSourceFolder
	TextBox	txtSourceFolder
Browse...	CommandButton	cmdBrowseSource
Destination Folder:	Label	lblDestFolder
	TextBox	txtDestFolder
Browse...	CommandButton	cmdBrowseDest
Width:	Label	lblWidth
100	TextBox	txtWidth
px	Label	lblWidthUnit
Height:	Label	lblHeight
100	TextBox	txtHeight
px	Label	lblHeightUnit
Maintain proportions	CheckBox	cbMaintainProportions
Resize Images	CommandButton	cmdResize

Now that all the controls are properly placed on the form, try running the macro again to see how the real form will look like when run from Corel PHOTO-PAINT’s Macro Manager docker:



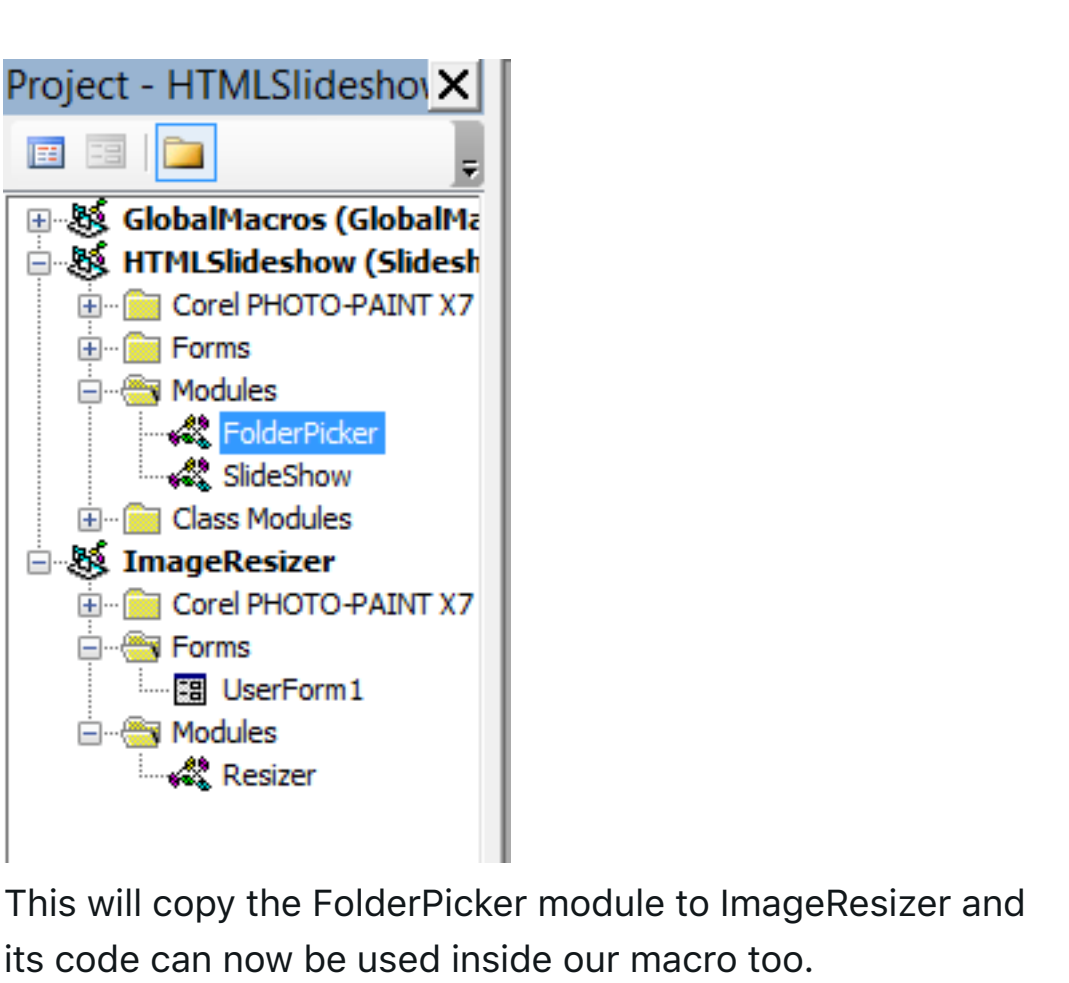
However you will notice that clicking any buttons has no effect because there is no code written to specify what should happen when a button is clicked.
This can be done simply by double-clicking a command button in VBA form designer. For example, double-click the “Resize Images” button. This will open the code window for the user form and add an event handler for mouse click for cmdResize control:



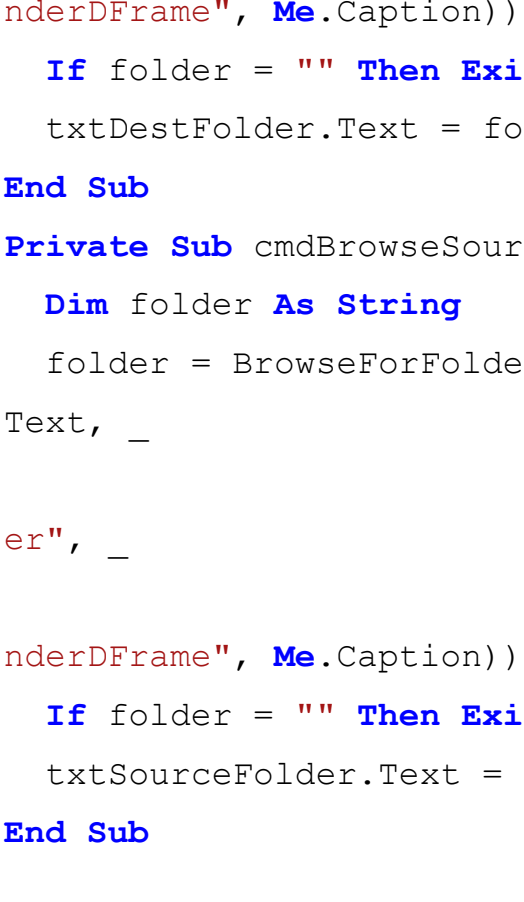
Change the implementation of cmdResize_Click() event handler to be the following:

```
Private Sub cmdResize_Click()
  MsgBox "Button clicked"
End Sub
```

This adds a simple code to show a message box when the button is clicked:



Now implement the Browse buttons for source and destination folders. This requires a “Browse For Folder” dialog from Windows which is not available in VBA directly. However VBA can use low-level Windows API functions, and there is already code available for this which is part of Corel PHOTO-PAINT’s HTMLSlideshow macro. You can just copy that code directly from that macro into your own. Expand the HTMLSlideshow macro project in VBA’s Project Explorer and locate the “FolderPicker” module. Select and drag it into the ImageResizer project:



This will copy the FolderPicker module to ImageResizer and its code can now be used inside our macro too.
Add the two button click event handlers for both Browse... buttons and provide their implementations as follows:

```
Private Sub cmdBrowseDest_Click()
  Dim folder As String
  folder = BrowseForFolderDlg(txtDestFolder.Text, _
                              "Select Destination Folder", _
                              GetWindowHandle("ThunderDFrame", Me.Caption))
  If folder = "" Then Exit Sub
  txtDestFolder.Text = folder
End Sub
Private Sub cmdBrowseSource_Click()
  Dim folder As String
  folder = BrowseForFolderDlg(txtSourceFolder.Text, _
                              "Select Source Folder", _
                              GetWindowHandle("ThunderDFrame", Me.Caption))
  If folder = "" Then Exit Sub
  txtSourceFolder.Text = folder
End Sub
```

Here, both event handlers are pretty much identical, except that they use different text controls (txtSourceFolder and txtDestFolder) to obtain the current folder name entered in those controls, call BrowseForFolderDlg() function from FolderPicker module and the new folder path returned is set back into the respective text box.

Adding Image Processing Functionality

Now we are ready to implement the main image processing function to do all the image manipulation. Change the cmdResize_Click() function to be the following:

```
Private Sub cmdResize_Click()
  Dim sourceFolder As String, destFolder As String
  Dim width As Long, height As Long
  Dim maintainAspect As Boolean
  sourceFolder = txtSourceFolder.Text
  If (CorelScriptTools.FileAttr(sourceFolder) And vbDirectory) = 0 Then
    MsgBox "Invalid source folder " & sourceFolder, vbCritical
    Exit Sub
  End If
  If Right(sourceFolder, 1) <> "\" Then sourceFolder = sourceFolder & "\"
  destFolder = txtDestFolder.Text
  If (CorelScriptTools.FileAttr(destFolder) And vbDirectory) = 0 Then
    MsgBox "Invalid destination folder " & destFolder, vbCritical
    Exit Sub
  End If
  If Right(destFolder, 1) <> "\" Then destFolder = destFolder & "\"
  If Not IsNumeric(txtWidth.Text) Then
    MsgBox "Invalid width", vbCritical
    Exit Sub
  End If
  width = CLng(txtWidth.Text)
  If Not IsNumeric(txtHeight.Text) Then
    MsgBox "Invalid height", vbCritical
    Exit Sub
  End If
  height = CLng(txtHeight.Text)
  maintainAspect = cbMaintainProportions.Value
  ResizeImages sourceFolder, destFolder, width, height, maintainAspect
  Unload Me
End Sub
```

This function takes the input from source and destination folder text boxes and validates that they contain valid folder names. Then it converts the required width and height of the image from the content of the text boxes (strings) to integer numbers which will be used when resizing images.
Finally, it calls the ResizeImages function to do the actual image conversion. After all the images are processed inside ResizeImages, the dialog is closed by invoking the Unload method.

ResizeImages iterates over all the files inside a folder identified by sourceFolder, opens each one of them in Corel PHOTO-PAINT, resize each image and saves them as JPEGs into the output folder specified by destFolder.
When maintainAspect variable is set to True, there is additional logic to preserve the image aspect ratio.

```
Sub ResizeImages(ByVal sourceFolder As String, ByVal destFolder As String, _
                 ByVal width As Long, ByVal height As Long, _
                 ByVal maintainAspect As Boolean)
  Dim f As String
  Dim doc As Document, w As Long, h As Long
  f = Dir(sourceFolder & "*.*")
  While f <> ""
    doc = OpenDocument(sourceFolder & f)
    w = width
    h = height
    If maintainAspect Then
      If w * doc.SizeHeight < h * doc.SizeWidth Then
        h = h * doc.SizeHeight / doc.SizeWidth
      Else
        w = w * doc.SizeWidth / doc.SizeHeight
      End If
    End If
    doc.Resample w, h
    doc.SaveAs(destFolder & f, cdrJPEG).Finish
    doc.Close
    f = Dir()
  Wend
End Sub
```

Now we have a fully functional macro that shows a custom dialog, accepts user input and processes a bunch of images using Corel PHOTO-PAINT. Feel free and experiment with the source code and build more complex solutions to the problems you may encounter in your day-to-day work.

Download source: MacroUI.zip

Share History