Predicting trends in techcomm

Learn how documentation could look and feel in the future.

Communicator

The Institute of Scientific and Technical Communicators
Summer 2018

Understand the advantages of working with Open Standards

Better documentation? Work / life balance improved

Discover tips for researching effectively

Information 4.0: content contextualisation and output

Creating user forms: part 3

Coding an interface for your VBA code. By Mike Mee.

Introduction

In this third part in the series, I will show you, warts and all, what goes on behind the About form in my Word Toolbox.

There are some functions that will cover new areas and I will do my best to point them out and/or add them into future articles in this series.

I will go through the form, the variables defined, the form initialisation code and how it detects what an end user does with the form once it is displayed.

Using images

Before I steam ahead describing my About form (Figure 4), I will cover what I do to create the icons that are used.

To fill out some space in this article, I could have lied through my teeth and say that I hand-drew each one using multi-layered images within Photoshop, but actually I just used a Google search and then imported the images into Paint.NET. They were resized down to smaller dimensions (32x32 pixels) and saved out as GIF files – limiting them to have only 256 colours present. The icons are then inserted onto the VBA form.

The reason for stripping out all of the detail is that you don't want your form to be over-bloated with huge, colourful icons that do exactly the same job as the small boring ones!

However, there is one oddity that comes with using images within VBA forms to be aware of. The images on your form can be in any of the more well-known file formats – with the exception of PNG format.

Naming convention

As with the previous articles, each type of control has a three-letter prefix that you can use to help you remember which control each variable is referring to when you interpret the actions. As we're only using images in this article, there is only one to remember.

Table 1. Controls, icons and naming conventions

Control	Icon	Leszynski
Images		`img`

Inserting images

Adding images to your own forms is easy. Drag the image icon from the VBA Toolbox onto your form (Figure 1). By default, it assumes that the image will be larger than an icon, but it will resize automatically once you include your image. The variable name associated with this image will be `Image1` - or whatever the next number in the sequence is, if you have added multiple images.

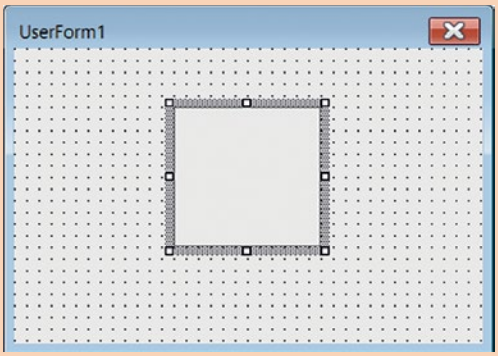

Figure 1. Form example with blank image inserted

To change the image to one of your own, you have to alter the parameters for this image using the standard Parameters window (Figure 2).

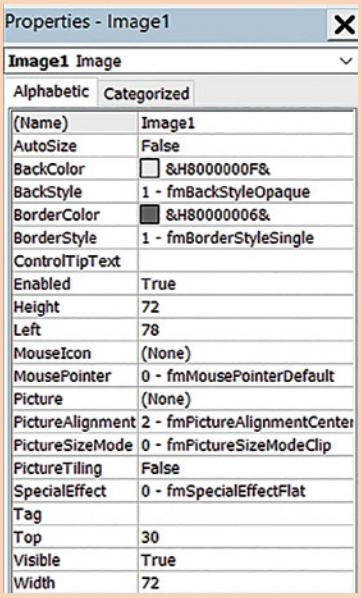

Figure 2. The inserted image's VBA properties

You can see that the parameter Picture is currently set to `(None)`. Click on this parameter and the standard three dots appear. This is to notify you that you can browse for a filename. Click on the three dots and select your image's file. The image will be loaded into the form.

You will need to resize the box that appears around the image by altering the `Height` and `Width` parameters. You can also switch off the border line that surrounds the image via the `BorderStyle` parameter.

In Figure 3, you can see that I have inserted the Fix Language icon that was used within Word Toolbox.

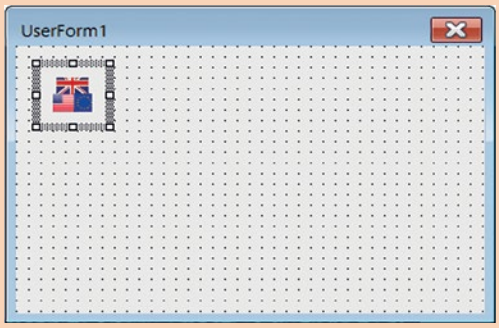

Figure 3. Form example with new image

It has been resized it to 32x32 pixels and I removed the border line. The image's variable name has been set to `imgFixLang`.

You can use the `AutoFit` parameter to resize the image to the best available layout according to the size of your own inserted image. But this function will not remove the border line – that would be too easy.

The rest of this article covers the breakdown of how one of my own forms works.

The About form

Now we are going to wade into the coding behind my toolbox's About form. I will dissect the form before delving into the VBA code behind it.

The About form is displayed in Figure 4.

Figure 4. Form example with frames
Source: Mike Mee's Word Toolbox v2.2

There is a logo (using a bitmap, exported into a GIF to save on the memory requirements) and a tagline (stored as a plain text field) at the top of the form.

The first frame contains a few text and label fields that display the author, version number of the toolbox and its release date.

Underneath, in the second and third frames, are five other image buttons (all icon-sized at 32 x 32 pixels) which are all clickable. The first icon, the PDF one, opens up the manual, the next three will visit various webpages and finally the fifth one displays my email address.

All of these text items and functions are probably seen as "standard" in almost any application that has an About form. Others may display a registered name / serial number, but as my toolbox is always free, I haven't included anything like that.

The VBA code behind the form

Unlike most of my previous articles, this one will be fairly code-heavy – even though there is not much code.

There are only 33 lines of code (and 25 blank lines / comments) in the full VBA form's source code, so my apologies for the amount of Courier-type based text that will appear from here onwards.

Form variables

First of all, the code defines a few variables: both are strings, but one is a Constant (variables and constants are discussed in *Communicator*, Autumn 2015).

```
Private strStartUpPath As String
PrivateConst strPDFLocation As String = "\
Mike's Document Toolbox Manual.pdf"
```

The VBA constant holds the filename of my manual in PDF format. This file is included within the toolbox's distribution .ZIP on my website. The PDF file should be copied into the same folder as the .DOTM file.

Form initialisation

The initialisation routine for the form is very simple. It is just displaying a box which contains some values.

```
Private Sub UserForm_Initialize()
strStartUpPath = Application.StartupPath
lblVersion.Caption = strVersionNumber
lblReleaseDate = "Released on: "
&strReleaseDate
End Sub
```

The `Application.StartupPath` is a built-in VBA constant that contains the folder path where the Toolbox was loaded from. I use the value contained to define where the manual *should* be located.

Both the `strVersionNumber` and `strReleaseDate` variables are defined as VBA constants elsewhere in the source code as they are also used on other forms.

That is all the initialisation performs. Once the form is displayed, it waits (and waits) for some user interaction with the five image buttons along the bottom.

Variable names

The five image buttons are named as per Table 2 below. I have kept them fairly descriptive as you can see.

Table 2. Image variable names

Image	Variable name
PDF	imgPDFManual
Website	imgHomepage
Trello Board	imgTrello
LinkedIn	imgLinkedIn
Email	imgEmail

Launch the PDF manual

This is the first option that the user can click on. Assuming that the PDF file has been copied to the correct location, the app that has been registered with Windows to handle any PDF files is launched. The file name for the manual is sent to it. Hey presto, a user manual at the click of a button.

The code below is split into two separate subroutines: the first launches the PDF file (if found) or, if it is missing, it displays an error message and returns the user back to the form.

Note: I have used the VBA constant vbLf to indicate that a string is to be separated by a linefeed.

This is just one of many constants that are included in VBA. I have briefly covered this before (*Communicator*, Summer 2016), but just in case you have forgotten, the full list can be found at: https://msdn.microsoft.com/en-us/vba/language-reference-vba/articles/miscellaneous-constants.

```
' Launch the PDF Manual
Private Sub imgPDFManual_Click()
On Error GoTo PDF_Manual_Error_Handler

If Dir(strStartUpPath & strPDFLocation) <>
vbNullString Then
    ActiveDocument.FollowHyperlink
Address:=strStartUpPath & strPDFLocation
Else
MsgBox "The manual was not found!" & vbLf &
vbLf & strStartUpPath & vbLf & vbLf & "Copy
the PDF file into the above location and try
again."
End If

Exit Sub

' Sort out any errors here
PDF_Manual_Error_Handler:
ErrorDisplay ("An error has occurred whilst
trying to load the manual. Please ensure
that the PDF file is in the same location
(" & strStartUpPath & ") as the Toolbox and
try again.")
Resume Next

End Sub
```

The VBA function that is used (the ActiveDocument.FollowHyperlink) is very flexible. This function is used by all five of the image buttons in order to launch: the PDF file, send the user to a certain web page, or launch the local email client and create a blank email with my address filled in.

You can use the same function within your own VBA code to similar effect.

Open the website, Trello or LinkedIn

These three routines are duplicates of each other, with the exception of the website address that is launched in the user's web browser.

```
' Website
Private Sub imgHomepage_Click()
ActiveDocument.FollowHyperlink
"http://www.mikestoolbox.co.uk/"
End Sub

' Trello Board
Private Sub imgTrello_Click()
ActiveDocument.FollowHyperlink "https://
trello.com/b/2YHdhJcl/mike-s-toolbox"
End Sub

' LinkedIn
Private Sub imgLinkedIn_Click()
ActiveDocument.FollowHyperlink "https://
uk.linkedin.com/in/mikemee"
End Sub
```

There are no error checks performed, nor are they really required, as the websites are known to be working. In your own code, you could generate the final URL via some form of string-calculation so that the user ended up on a particular item within a web-based shop, for example.

Send an email

This final function is probably the most basic of the five in that it just displays a message box with my Gmail address inside it.

```
' Email (Message box only)
Private Sub imgEmail_Click()
MsgBox "You can contact me via: mugukmail@
gmail.com" & vbLf & vbLf & "Although you are
better off using the Trello board to raise
any bugs and/or feature requests."
End Sub
```

I really must get around to changing it so that it points at the official email address.

And that is it. A full VBA form, albeit a simple one, broken down at source-code level so you can see how it all works.

Roll your own

As this article comes to a close, here is my challenge. Create a form that uses image

buttons, along with some of the other VBA form functionality that has been previously discussed.

Once you are happy with the form, upload the code (or email it directly to me) and show me what you have come up with.

If you have any problems with any particular functionality in your new form, it would be easier to create a new thread on the ISTC forum so that others can see what you have been doing.

Next article

My new ZX Spectrum Next that I mentioned I was waiting for in the last issue has been delayed due to a manufacturer of the casing pulling out at the last minute.

C'est la vie! These things can sometimes happen when you are a backer (or "an investor") of a Kickstarter project.

It will turn up at some point and, depending on how busy I am working away from home due to my new job, I hope to be able to find some time to mess around with it. The time will definitely include some time spent rekindling my misspent youth where I first learned my BASIC programming skills: all the way back in 1983 when my dad bought the family's ZX Spectrum from Laskys. C

Reference s and further reading

Mee M (2015) 'Variables and screen output/input' *Communicator*, Autumn 2015: 44-47

Mee M (2016) 'Creating VBA loops: part 2' *Communicator*, Summer 2016: 52-55

Mee M (2017) 'Optimising your VBA code' *Communicator*, Spring 2017: 37-39

Mee M (2017) 'Using the Quick Access Toolbar' *Communicator*, Summer 2017: 52-53

Mee M (2017) 'Creating user forms: part 1' *Communicator*, Winter 2017: 44-46

Mee M (2018) 'Creating user forms: part 2' *Communicator*, Spring 2018: 34-37

Microsoft 'Declaring Variables' https://msdn.microsoft.com/en-us/VBA/Language-Reference-VBA/articles/declaring-variables (accessed April 2018)

Microsoft 'Miscellaneous Constants' https://msdn.microsoft.com/en-us/vba/language-reference-vba/articles/miscellaneous-constants (accessed April 2018)

Mike Mee MISTC
is a contract technical author.
E: mugukmail@gmail.com
T: @Mug_UK
W: www.mikestoolbox.co.uk
– my toolkit for Word 2007-2018