Access Forms Masterclass 2
Custom Record Counter

Published: 1 August 2013
Author: Martin Green
Screenshots: Access 2010, Windows 7
For Access Versions: 2007, 2010, 2013

Why Add a Custom Record Counter?

A new Access form comes complete with a set of built-in navigation buttons and a record counter, located in the lower-left corner of the form (*Fig. 1*).

Fig. 1 A form's built-in record counter and navigation buttons.

Whilst these are useful tools, on modern high-resolution screens they appear quite small and many users find them hard to see and operate. The solution is to replace them with ones you can build yourself, to your own design and to suit your users' needs.

The first Masterclass in this series showed you how to build a set of Custom Navigation Buttons. If, having built them you want to remove the built-in ones you will also lose the built-in record counter. This Masterclass will show you how to add a replacement Record Counter of your own design (*Fig. 2*).

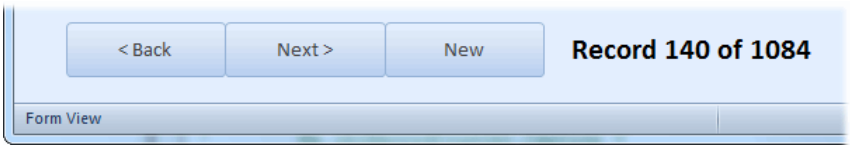

Fig. 2 Custom Record Counter and Navigation Buttons.

In this Masterclass you will learn how to add a Label Control to a form and write the necessary VBA code to turn it into a custom Record Counter which you can position and format to suit your own requirements.

Step 1: Add a Label to the Form

Draw the Label

When you want to display some text on a form you have a couple of methods to choose from. You can use an unbound *Text Box* control (one that is not bound to a field in the form's underlying recordset) then use some code to change its *Value* property, or you can use a *Label* control and use some code to change its *Caption* property. I normally prefer the latter because, from the user's point of view, it is effectively inert. They can't select it or otherwise interfere with it. If you use a Text Box the user might find their way to it and try to enter something in it. Even if you lock a Text Box and exclude it from the Tab Order the user can still click on it. Whilst that isn't a problem it might confuse them and you don't want a confused user!

You can place the label on any part of the form. In this example I am going to locate the Record Counter next to the Custom Navigation Buttons I created in the first Masterclass.

To add a Label control to your form, put the form into **Design View** and click the **Label** tool in the **Controls** section of the **Design** tab of the Ribbon (*Fig. 3*).

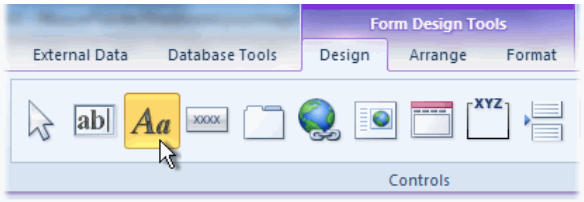

Fig. 3 Choose the Label tool from the Ribbon.

The mouse pointer changes to show that it is ready to add a Label. Click on the form approximately where you want the Record Counter to appear (*Fig. 4*).

Fig. 4 Click on the form to add a new Label control.

It is important that you don't click anywhere else or press [Enter] at this point because Access will immediately delete the empty label. To prevent the label from disappearing you must add some text to it. Anything will do but you might as well enter something sensible so, immediately after adding the label to the form, type *Record Counter* then press **[Enter]**.

Your label now appears with the text you typed and is highlighted to indicate that it is selected. If you click anywhere else is becomes de-selected. If necessary, simply click on it again to select it.

You will see that in the upper left corner of the selected label there is a small green triangle and adjacent to the label is a small warning icon. This means that Access thinks there might be a problem with the control you just added. If you point at the warning icon Access tells you that: *This is a new label and is not associated with a control* (*Fig. 5*).

Fig. 5 Access warns that the new label is not associated with a control.

We know that this isn't a problem. The label is there just to display some information and not to provide a label for another control so, to remove the warning, click the down-arrow next to the warning icon and choose **Ignore Error** from the menu.

Rename the Label

Access will have automatically named the new label but we need to give it a meaningful name before we refer to it in our code. In **Design View** select the Label control. If the Property Sheet is not already open display it by right-clicking on the label and choosing **Properties** from the context menu. Go to the **Other** tab of the **Property Sheet** and change the **Name** property to *lblRecordCounter* (*Fig. 6*).

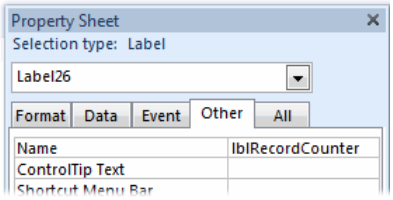

Fig. 6 Give the label a meaningful name.

Format and Position the Label

Switch the form into **Form View** and take a look at the label. You can decide how large you want the text to appear and where exactly to position it. In this example the default text appears quite small and the default light grey colour is not sufficiently prominent for my users (*Fig. 7*) so I am going to increase the font size and change its colour to something more visible.

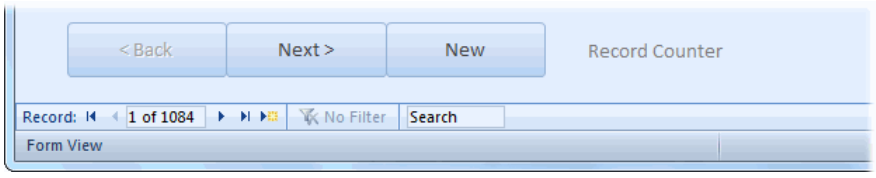

Fig. 7 You may decide to enlarge the text and change its colour.

Return to **Design View**, click on the label to select it, and use the tools in the **Font** section of the **Format** tab of the Ribbon to change the colour and size of the text to your preference. I have chosen **Black**, **Bold** and **16 point** text size.

Although the appearance of the text changes, the size of the label does not. To make the label the correct size for the newly formatted text, double-click the dot in its lower-right corner (*Fig. 8*).

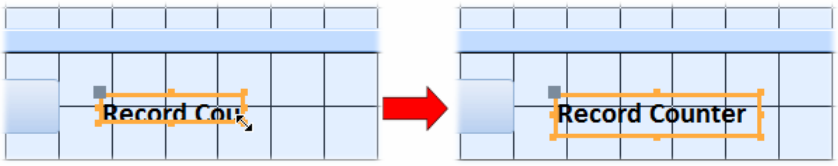

Fig. 8 Resize the label to fit the text.

This snaps the label to the correct height but it is probably not wide enough for the text that it will display. I intend to display text that reads something like "Record 140 of 1084". You can move and resize the label by dragging the dots that are located along its edges when it is selected, but for accuracy I prefer to move and resize controls using the keyboard as follows.

With the label selected, use the **[Arrow]** keys to move the label to the required position. Then, hold down **[Shift]** and click the **[Right Arrow]** key to increase its width by the required amount (*Fig. 9*). Since the label has no visible border (unless you have chosen to add one) it is quite safe to make it wider than necessary to ensure that it will be able to accommodate any text that might be created.

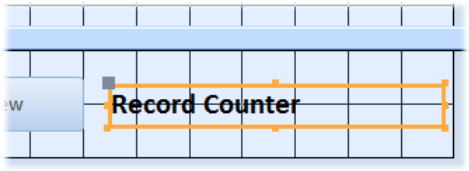

Fig. 9 Widen and position the label as required.

Check the label's finished appearance by switching to **Form View** (*Fig. 10*).

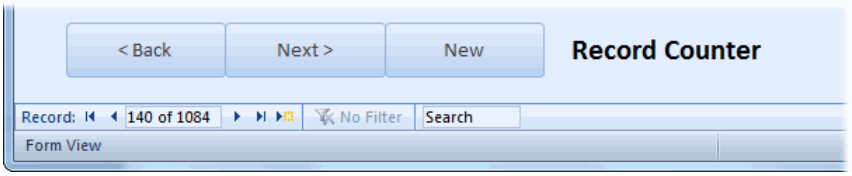

Fig. 10 The formatted label seen in Form View.

Step 2: Write VBA Code to Power the Record Counter

We are going to create an *Event Procedure* that will change the label's caption each time a different record is displayed. An event procedure is a VBA macro that runs by itself whenever a specified event happens. The appropriate one in this case is the form's *Current* event. This event fires when the form opens and when the user moves from record to record, or asks for a new record.

To create the event procedure, in **Design View** select the form itself by choosing **Form** from the drop-down list at the top of the Property Sheet. Choose the **Event** tab, then click in the text box next to **On Current** then click the **Build** button (**[...]**)*. In the **Choose Builder** dialog select **Code Builder** (*Fig. 11*) and click **OK**.

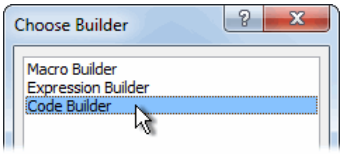

Fig. 11 The Choose Builder dialog.

**NOTE: If a Form_Current event procedure already exists (such as the one you might have created in Masterclass 1) then clicking the build button takes you directly to it in the Visual Basic Editor.*

If one does not already exist, Access creates a new, empty *Form_Current* event procedure and places the cursor between the *Private Sub…* and *End Sub* statements ready for you to add your code (*Listing 1*).

Listing 1:

```
Private Sub Form_Current()

End Sub
```

Our code will make use of some of the form's properties to create a string of text that is then applied to the *Caption* property of the label each time a different record is displayed.

The form's *CurrentRecord* property has a value which represents the index number of the currently selected record in the form's recordset. The first record has an index of *1*, the second record *2* and so on. The form's *Recordset* property has a *RecordCount* property which represents the total number of saved records in the form's recordset.

I want the label to read *Record x of y* where *x* is the index number of the current record and *y* is the total number of records, for example: *Record 140 of 1084*.

Enter the code as follows:

1. Place the cursor between the lines beginning *Private Sub…* and *End Sub* and press **[Tab]** to indent your typing by one tab-space.
2. Type the text exactly as shown below (*Listing 2*).

Listing 2:

```
    Me.lblrecordCounter.Caption = _
        "Record " & Me.CurrentRecord & " of " & Me.Recordset.RecordCount
```

NOTE: Immediately after the equals sign (=) I have added a space followed by the underscore character. This allows me to break the code statement on to a second line. I am only doing this because it makes the long line of code easier to read and can be omitted if you prefer to write the code statement as a single line.

The completed event procedure looks like this (*Listing 3*) (in addition to any existing code you might have there):

Listing 3:

```
Private Sub Form_Current()
    Me.lblrecordCounter.Caption = _
        "Record " & Me.CurrentRecord & " of " & Me.Recordset.RecordCount
End Sub
```

Before testing your code, first check your typing then open the Visual Basic Editor's **Debug** menu and choose **Compile...** (the name of your database is shown). Compiling the code checks for any errors you might have missed. If everything is OK (the Visual Basic Editor will tell you if it finds a problem) click the **Save** button. This is an important step because, if you test code without saving your database first and an error causes Access to crash, you might lose some of your work.

Switch the form into **Form View** and use the navigation buttons to move through the recordset. In this example (*Fig. 12*) I have retained the built in navigation buttons to allow me to check that the code is working correctly, but I will remove them later.

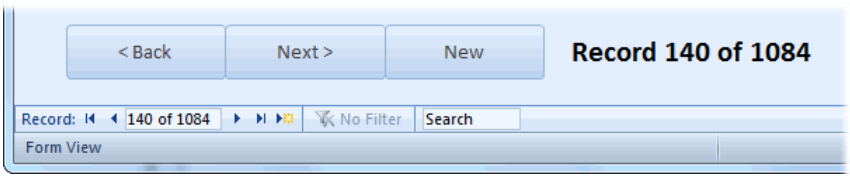

Fig. 12 The finished Record Counter.

Step 3: Additional Code Refinements

An anomaly occurs when the form is moved to a new and as yet unsaved record. On a new record the *CurrentRecord* property has a value of 1 greater than the value of the *RecordCount* property, even though the new record has not yet been saved to the recordset (*Fig. 13*). In this example the current record is shown as *1085* although the recordset contains only *1084* saved records.

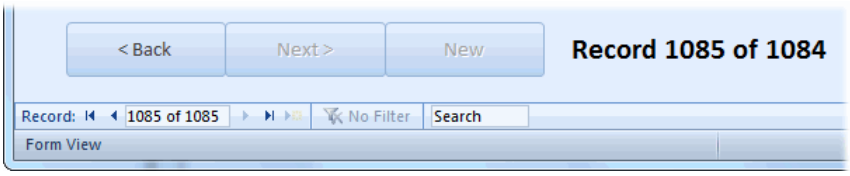

Fig. 13 On a new record the numbers are shown incorrectly.

This happens too in the built-in counter but Access resolves this by treating the new, unsaved record as if it were already part of the recordset and increases the record count by 1. So, in this example the built-in counter reads *1085 of 1085*.

If this anomaly bothers you can resolve it in a number of ways. You could have the counter behave the same way as the built-in counter, alternatively you could have the counter display something else, such as the text *New Record*. Both methods require the code to check whether or not the form is displaying a new record. This can be done with an *If Statement* examining the value of the form's *NewRecord* property.

If you want the label to display the same values for the record index number and the total record count when a new, unsaved record is displayed, edit the code as follows (*Listing 4*):

Listing 4:

```
Private Sub Form_Current()
    If Me.NewRecord Then
        Me.lblrecordCounter.Caption = _
         "Record " & Me.CurrentRecord & " of " & Me.Recordset.RecordCount + 1
    Else
        Me.lblrecordCounter.Caption = _
         "Record " & Me.CurrentRecord & " of " & Me.Recordset.RecordCount
    End If
End Sub
```

This changes the label's behaviour so that the record index number and total record count are the same (*Fig. 14*).

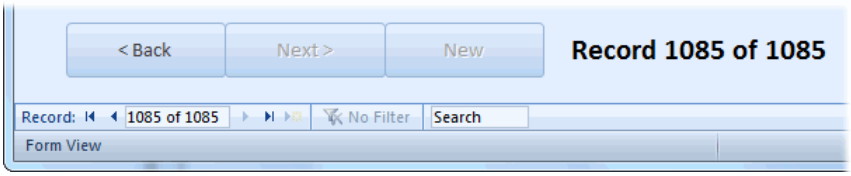

Fig. 14 Both the record index number and total record count are the same.

If you want the label to display some text when a new, unsaved record is displayed, edit the code as follows (*Listing 5*):

Listing 5:

```
Private Sub Form_Current()
    If Me.NewRecord Then
        Me.lblrecordCounter.Caption = "New Record"
    Else
        Me.lblrecordCounter.Caption = _
         "Record " & Me.CurrentRecord & " of " & Me.Recordset.RecordCount
    End If
End Sub
```

This changes the label's behaviour so that the text *New Record* is displayed (*Fig. 15*).

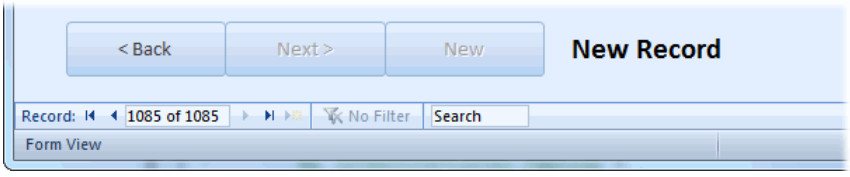

Fig. 15 Text is displayed when the form shows a New record.

Step 4: Additional Design Refinements

If you have added your own Custom Navigation Buttons (see Masterclass 1), now that you also have your own Custom Record Counter you might like to remove the built-in one.

Remove the Built-In Navigation Buttons

This is a simple process. In **Design View**, open the Property Sheet and select **Form** from the drop-down list at the top. Then, on the **Format** tab change the **Navigation Buttons** property to *No*. This causes the navigation bar to be hidden on that particular form (*Fig. 16*).

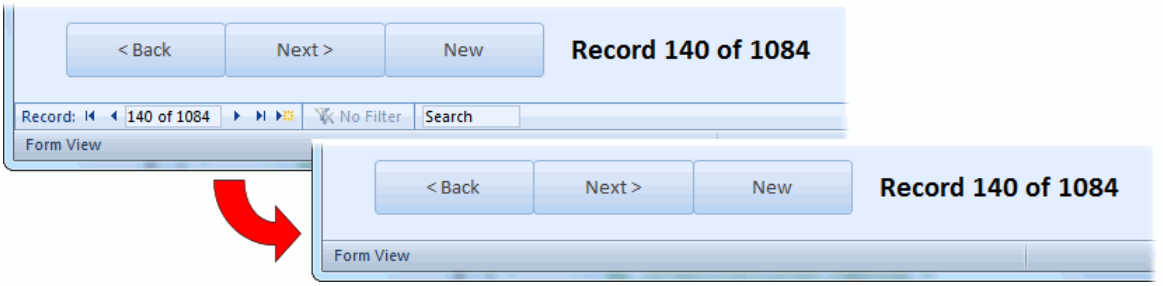

Fig. 16 Hiding the form's built-in navigation buttons.

Summary

In this Masterclass you learned how to add a Label Control to an Access form, how to size and position it accurately, and how to change its appearance. You then added VBA code to change the label's caption so that it behaved as a Record Counter showing both the index number of the current record and the total number of records in the form's recordset.

All the techniques described here have been tested successfully in Microsoft Access versions 2007, 2010 and 2013.