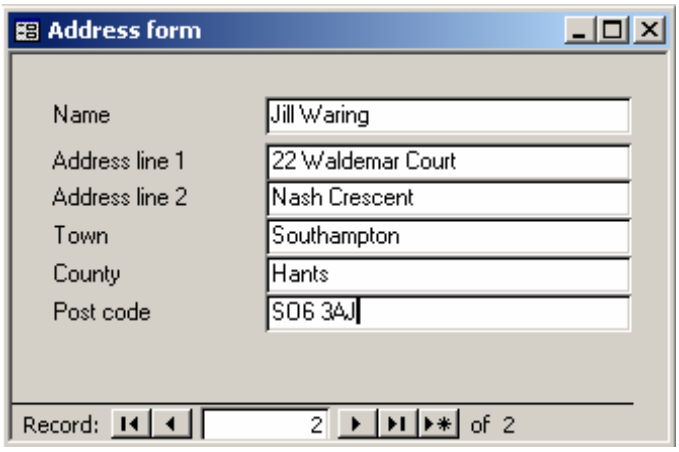
Access Address Clipboarding the Automated Way

How to combine several address field values into a single string and have it placed on the Windows clipboard.

I was recently installing an Access system for a client which maintained personal information on a variety of people. As usual, I had divided the address information between a number of fields, thus:



It is normal practice to split up address data like this, because:

(a) the user can filter records more precisely, by town, for instance
(b) queries can target individual parts of the address
(c) different address layouts, like the two below, are easier to generate in reports:

Address layout 1:	**Address layout 2:**
10 North Lane, Rayleigh, Essex, SS6 8LF	10 North Lane Rayleigh Essex SS6 8LF

For all these reasons, the system I was installing had segregated address fields. But one user who had been using the system for a few days asked me if I could spare her a minute. She explained that she often had to prepare contracts in Word, and needed to take an address from the system and insert it into a contract clause. We talked about setting up a template and passing fields over automatically, but there were so many different contracts (many of which did not yet exist) that this would have been impractical.

What she actually wanted was to be able to copy the whole address to the clipboard so she could then paste it wherever she wanted. Because the address fields were

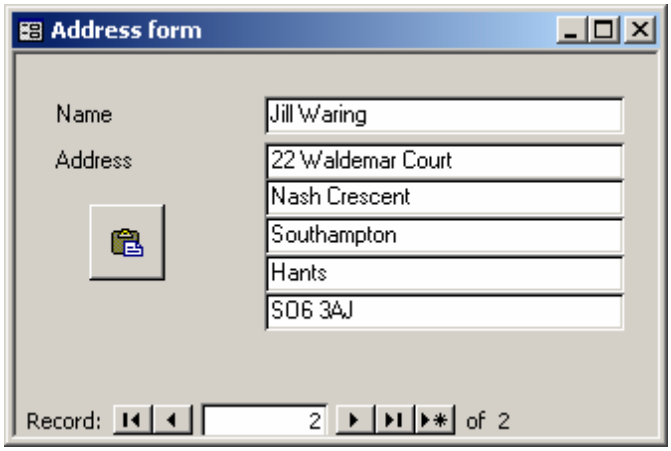
segregated, right now she was having to do 5 different copy-switch-paste operations to reconstruct the address in Word from the individual fields. “Couldn’t you just put a button there,” she said, pointing at the address form, “that I could just click, and it would put the whole address on the clipboard?” Of course I could!

I knew that from Access 2000 on, there was a simple VBA command which would let me do this:

```
DoCmd.RunCommand acCmdCopy
```

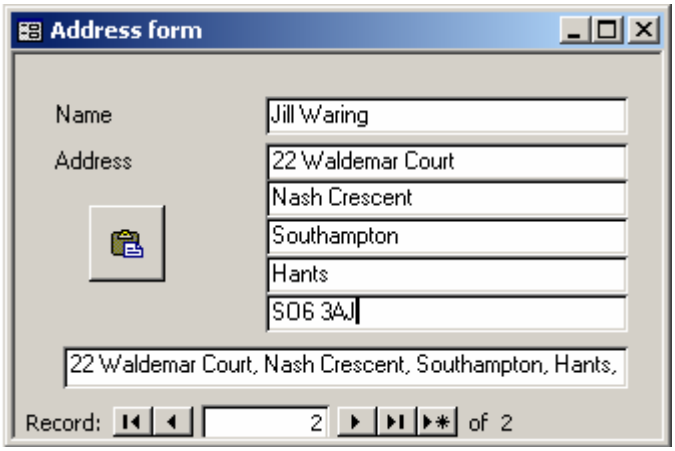
This line of code, linked to a button click event, would be at the core of my solution. But first, I had to do some preparation.

The address form pictured above was the form where the address got entered by the user. A different form was used to lookup addresses already on the system, and it had plenty of space to put in a little button:

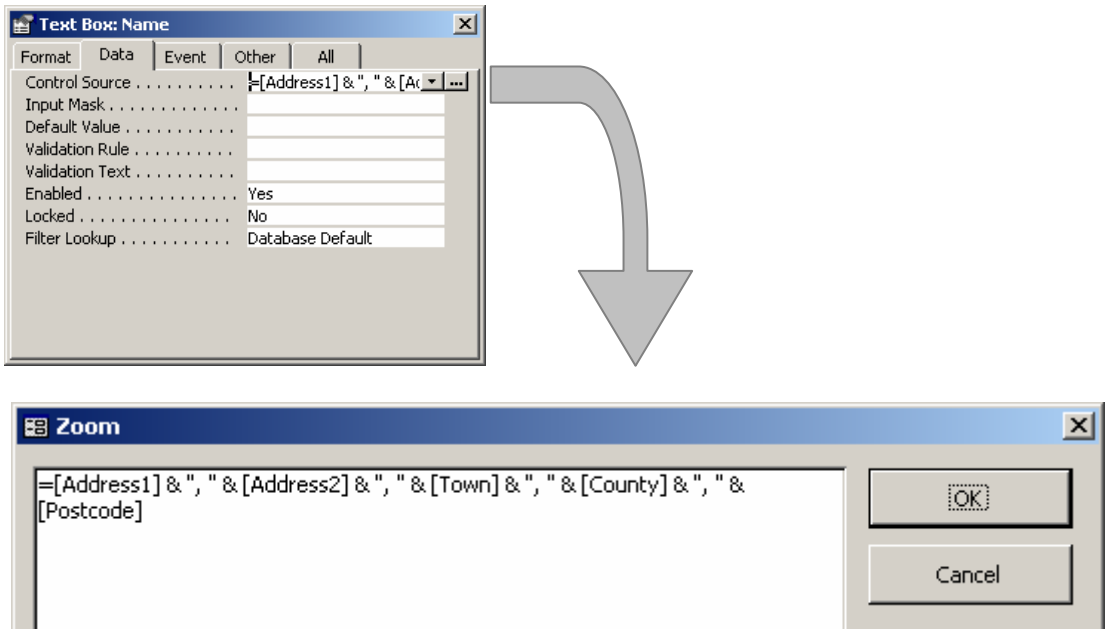


I gave this button control the Name property of ***btnClipboard***.

The acCmdCopy works by placing the value of the control which has focus onto the clipboard. My form, or course, has a total of 5 different controls that together make up the address, so my next task was to create a new control which lifted all the fields into a single text box. I set its Name property as ***txtFullAddress***.



The Control Source for this text box is detailed below:



Note that the 5 fields are all simply concatenated with a comma in between each, as in "Address layout 1" at the start of this article. That is the format in which the address needed to be inserted into a contract, but I could have concatenated them equally as easily with Chr(13) functions to insert return characters and give the equivalent of Address layout 2.

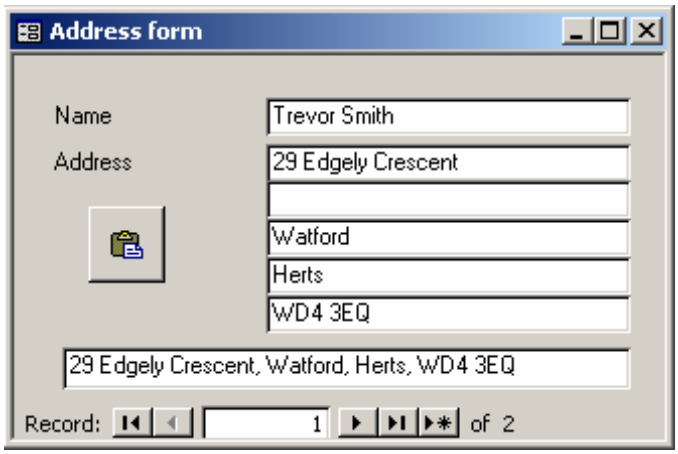
The box was working well, but look what happened when I moved to a different record:

Note that because one of the fields was empty, I got a double comma in my new text box. It is very common in a database, of course, for one or more of the address fields simply not to be needed, so I was facing an issue I would have to fix.

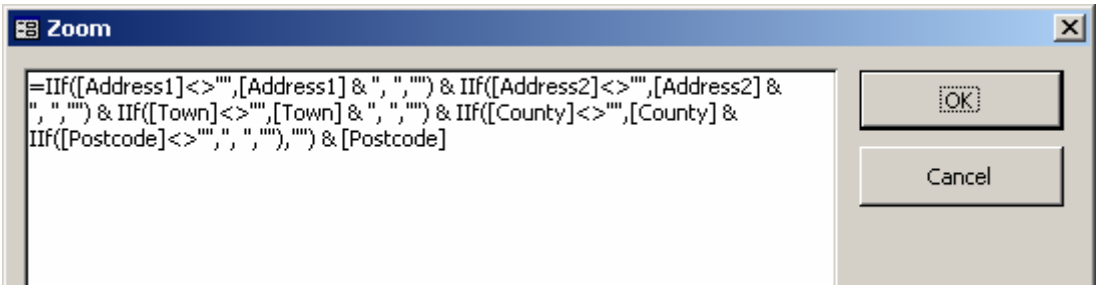
I chose to rewrite the control source a little to check for empty fields, and suppress the concatenation and comma if any were found, like this:

Note that I did not need the Iif construction for the Postcode field. Since this came last, there would be no trailing comma to suppress in any case

Now, the box was behaving itself:

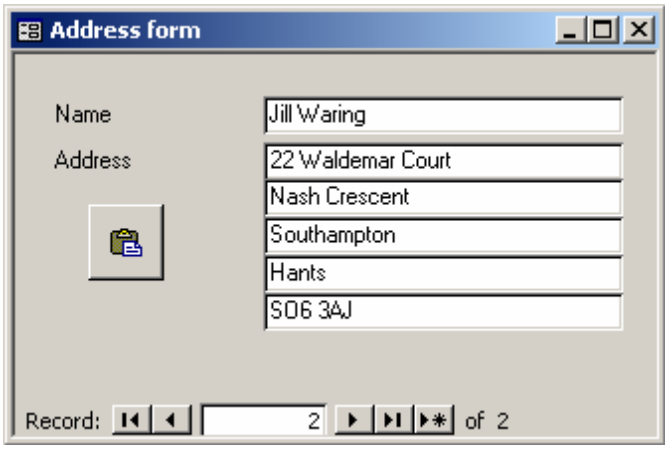


While playing a little more, I found that I hadn't quite eliminated the comma problem. If the postcode were omitted, there would still be a trailing comma. So I tightened up the control source clause like this:



Now, when it inserts the County, it checks that there is a Postcode; only if there is does it add the final comma.

My last step on the form was then to hide this new field by settings its Visible property to false. You can see the result overleaf.



I was now ready to add the code to the click event of my button. Because of the requirements of the acCmdCopy, I'd have to add a few extra lines. Firstly, remember that this command places the value of the control *which has focus* onto the clipboard, so I would have to give the new control focus before executing the acCmdCopy command. This gave me an additional problem, because an invisible control cannot have the focus!

So what I realised I had to do was to:

1. make the control visible
2. give it focus
3. copy its value onto the clipboard
4. move the focus back to my button
5. make the field invisible again

Here is what the resulting code looks like:

```
Private Sub btnClipboard_Click
     With Me.txtFullAddress
          .Visible = True
          .SetFocus
          DoCmd.RunCommand acCmdCopy
          Me.btnClipboard.SetFocus
          .Visible = False
     End With
End Sub
```

I thought about turning off the screen updating while the code ran, so that the hidden field didn't flash, but in testing the code ran so quickly that it didn't seem worth the bother.

I installed the update and let the user play with it. She was delighted, as was I. I now have a standard routine that I'll be using in virtually every database I design.