

## Move Forms & Controls Example

This example application was developed in response to various forum questions including:

<http://www.accessforums.net/showthread.php?t=75572> (ironfelix717)

<https://www.access-programmers.co.uk/forums/showthread.php?t=302641> (Iana)

---

Moving (and resizing) objects to precise locations on the screen is very easy using the **Move** method expression **Move(Left, Top, Width, Height)**

For example, **Forms!Form1.Move 100,200,500,350**

This moves the top left of Form1 to x-y co-ordinates 100,200 and changes the width and height to 500 & 350 (where all values are in **twips** – see later)

Similarly, controls can easily be moved on a form.

For example, this code moves Box9 so it is located immediately below textbox Text2:

**Me.Box9.Left = Me.Text2.Left**

**Me.Box9.Top = Me.Text2.Top + Me.Text2.Height**

The example application does both of the above but also demonstrates some much more complex processes.

For example, it shows how:

- A popup form such as a customised zoom box can be moved to a precise position over another form irrespective of form settings
- A listbox record can be 'selected' without clicking on the listbox. This is done by accurately detecting the record underneath the mouse cursor based on the height of each row in the listbox. This means the record can be used e.g. to open a filtered form / view an image without actually selecting the listbox record!
- The x-y coordinates of an object on a form can be determined and the object nudged by a specified amount in any direction

Each of these has been successfully tested using a variety of situations:

- Navigation bar - maximised/minimised/removed
- Ribbon - maximised/minimised/removed
- Application window – maximised / restored
- Different screen sizes and resolution
- Enlarging the screen display setting from the default 100% to 125%

Different form conditions have been tested including:

- Border style – none / thin / sizable / dialog
- Scrollbars – none / horizontal only / vertical only / both
- Navigation button bar – visible / hidden
- Record selectors – visible / hidden
- Different fonts – font name / point size & style (bold / italic / underline)

NOTE:

It is NOT possible to hide the application window for this example.

Doing so would require the use of popup forms which isn't possible here due to the simultaneous use of 2 forms in example forms 1-4

There are many complications that need to be managed for this to work well:

### 1. Units of measurement – twips, pixels and points

The position of objects in the Access window is determined in **twips (one twentieth of imperial point size)** where 1440 twips = 1 inch or 567 twips = 1 cm

The x-y coordinates of the top left of the application window are 0, 0

However, the position of the mouse cursor is measured in **pixels** with reference to the overall screen:

- 1 pixel (px) = 15 twips so 96px = 1440 twips = 1 inch

Font size is measured in **points (pt)** where 1 point = 20 twips

A 72 **point** font = 1 inch = 1440 twips = 96px ;

12pt = 16px = 240 twips = 1/6 inch, 3 points = 60 twips = 4 px etc

Complications arise with font sizes that are not factors of 72. For example:

- 11 pt font = 220 twips = 14.667 px but as you cannot have a part pixel that actually requires 15 px
- 10 pt font = 200 twips = 13.33px so this takes up 14px which is a significant difference

For further info, see <https://websemantics.uk/tools/convert-pixel-point-em-rem-percent/>

This makes pt => px conversions difficult to do precisely and can lead to errors on the screen

There are three main ways of doing this conversion:

- Use conversion values as above with arbitrary corrections to align objects as well as possible  
This may be adequate for a selected font name & size but is very likely to be inaccurate if either / both of these are altered.
- Use values calculated by a direct points/pixels to twips conversion based on code such as the **ConvertToTwipsYFromPoint** function based on the widely used **GetSystemMetrics API**.  
This manages the inexact conversion between 72 points and 96 pixels per inch by building in a 'jump' every 3 points

Point size = 1 ; Twips = 15
Point size = 2 ; Twips = 30
Point size = 3 ; Twips = 60
Point size = 4 ; Twips = 75
Point size = 5 ; Twips = 90
Point size = 6 ; Twips = 120
Point size = 7 ; Twips = 135
Point size = 8 ; Twips = 150
Point size = 9 ; Twips = 180
Point size = 10 ; Twips = 195
Point size = 11 ; Twips = 210
Point size = 12 ; Twips = 240
Point size = 13 ; Twips = 255
Point size = 14 ; Twips = 270
Point size = 15 ; Twips = 300
Point size = 16 ; Twips = 315
Point size = 17 ; Twips = 330
Point size = 18 ; Twips = 360
Point size = 19 ; Twips = 375
Point size = 20 ; Twips = 390
Point size = 21 ; Twips = 420
Point size = 22 ; Twips = 435
Point size = 23 ; Twips = 450
Point size = 24 ; Twips = 480

This approach can work reasonably well for some standard fonts between about 10pt & 14pt.

However, it gets increasingly inaccurate for much larger/smaller fonts. It also makes no allowance for certain fonts e.g. **Comic Sans** being taller than the normal value for that point size.

The image below shows a capital A in 13 different standard Windows fonts - all are 72 points



**Stephen Lebens** uses an enhanced version of this code in several example applications such as

- <http://www.lebens.com/SelectRow.htm>
- <http://www.lebens.com/textwidth-height.htm>

The code I originally developed was partly based on the second example above.

I am always amazed by Stephen's ability to do things that us mere mortals would never achieve alone.

Even so, the results using his code are not perfect for all situations

- A MUCH better method makes use of the little known VBA **WizHook** function. This actually measures the height and width of a character string based on the font name, font style normal, italic etc. This approach should ALWAYS work no matter what the situation. My tests confirmed that to be so.

The issue with **WizHook** is that it is a hidden function which has been available for over 20 years but is not documented by Microsoft. In theory it could be removed in a future release, but as it is used in some built-in wizards, I believe that to be highly unlikely

However, there is very little information about this function online apart from:

- <http://www.mvp-access.es/juanmafán/wizhook/wizhook.htm> (in Spanish)
- [ftp://developpez.com/cafeine/access/access\\_wizhook.pdf](ftp://developpez.com/cafeine/access/access_wizhook.pdf)

See the end of this article for additional information about hidden VBA functions such as **WizHook**

## 2. Form components

In order to locate an object **precisely on / over** a specific control on a different form, we need to know the size and position of each component of a form – not all items will be present depending on form settings. The application calculates ALL of these items for use as required

**Screen & Form Information** Close

<b>Screen Dimensions (twips) :</b>		<b>Screen Resolution (pixels) :</b>	
Width:	(SW) <input type="text" value="25200"/>	Horizontal:	(HR) <input type="text" value="1680"/>
Height:	(SH) <input type="text" value="15750"/>	Vertical:	(VR) <input type="text" value="1050"/>

---

<b>Form Name:</b>	Form2		
<b>Form Position:</b>	<b>Form Dimensions Key :</b>		
Top:	(FT) <input type="text" value="1800"/>		
Left:	(FL) <input type="text" value="855"/>		
<b>Form Dimensions:</b>			
Total Height:	(TH) <input type="text" value="5025"/>		
Total Width:	(TW) <input type="text" value="8475"/>		
Inside Height:	(IH) <input type="text" value="4305"/>		
Inside Width:	(IW) <input type="text" value="8130"/>		
Header Height:	(HH) <input type="text" value="1095"/>		
Detail Height:	(DH) <input type="text" value="405"/>		
Footer Height:	(FH) <input type="text" value="390"/>		
Title Bar Height:	(TBH) <input type="text" value="390"/>		
Vertical Border Width: (2 of)	(VBW) <input type="text" value="45"/>		
Horizontal Border Height:	(HBH) <input type="text" value="45"/>		
Navigation Button Bar Height:	(NBH) <input type="text" value="285"/>		
Record Selector Width:	(RSW) <input type="text" value="255"/>		
Vertical Scroll Bar Width:	(VSW) <input type="text" value="255"/>		
Horizontal Scroll Bar Height:	(HSH) <input type="text" value="0"/>		

Horizontal scroll bar shares space with navigation button bar (if visible)

Version 7.3 20/02/2019
Screen dimensions & resolution updated every 10 seconds
Mendip Data Systems 2005-2019

### 3. Using the example application

The example application includes an **Images folder**

For the purposes of this example, this needs to be a subfolder of the example app

The size of the navigation pane, ribbon and application window can all be controlled from the startup form.

**Move Forms and Controls** Form Info Quit

There are 6 forms available in this example:

Forms 1 - 3 show how a zoom box can be used to view the entire contents of a standard textbox when it is too large to fit in the available space. Each form has 2 textboxes. Double click either textbox. The zoom box is opened / moved directly below the textbox. It should align whether or not record selectors / navigation buttons are used. Similarly the border style should have no effect on the alignment.

Form 4 & 5 use listboxes to display images or open / move another popup form. The actions can be controlled using mouse move or a mouse click

Form 6 just shows how coordinates can be updated as a control is moved on a form or as the form is moved around the screen

Form 1 - Single Form 4 - Listbox Contacts

Form 2 - Continuous Form 5 - Listbox Images

Form 3 - Continuous Subform Form 6 - Coordinate Test

Options: Navigation Pane Ribbon Application Window

Hide  Hide  Hide (DISABLED)

Minimise  Minimise  Restore

Maximise  Maximise  Maximise

NOTE: The application window can not be hidden for this demo due to use of multiple forms

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

There are 6 test forms available:

#### Forms 1 - 3 (Single / Continuous / Continuous Subform)

Each of these is designed show how a zoom box can be used to view the entire contents of a standard textbox when it is too large to fit in the available space.

Each form has 2 textboxes and several different options that can be applied

**Double click** either textbox. The **zoom box** is opened / moved directly below the textbox.

**Single Form** Form Info Close

Double click either textbox.  
Form2 will open/move directly below that textbox  
The orange box will move directly below the other textbox

Options: Border Style Scrollbars

None  None

Thin  Horizontal only

Sizable  Vertical Only

Dialog  Both

Text0 hello world

Text2 Goodbye

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

**Single Form** Form Info Close

Double click either textbox.  
Form2 will open/move directly below that textbox  
The orange box will move directly below the other textbox

Options: Border Style Scrollbars

None  None

Thin  Horizontal only

Sizable  Vertical Only

Dialog  Both

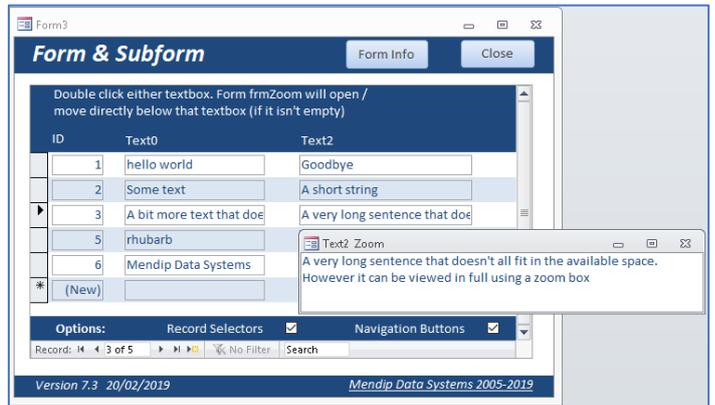
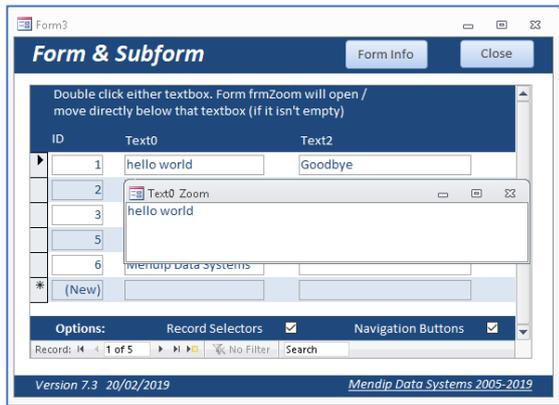
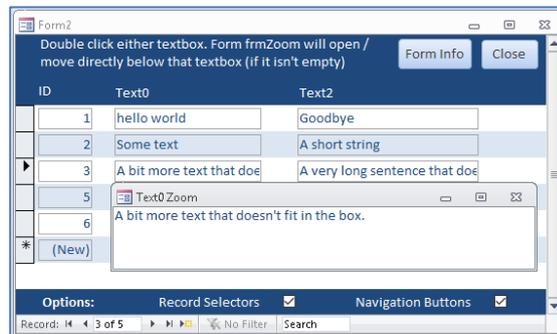
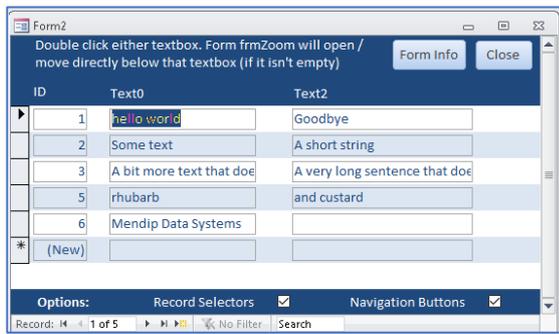
Text0 A bit more text that doe

Text2 A very long sentence that

Text0 Zoom  
A bit more text that doesn't fit in the box.

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

Record: 1 of 5 No Filter Search



The zoom box should align closely whether or not record selectors / navigation buttons / scrollbars are used. Similarly, the border style should have no effect on the alignment.

Typical code (this is taken from Form2)

```
Private Sub Text0_DbClick(Cancel As Integer)
```

```
    'close zoom form if open
    If CurrentProject.AllForms("frmZoom").IsLoaded = True Then DoCmd.Close acForm, "frmZoom"
```

```
    strText = Nz(Me.Text0, "")
    strCaption = "Text0 "
```

```
    'check record containing text has been clicked
    If Nz(Me.RecNum, "") <> "" And strText <> "" Then
        intLeft = Me.WindowLeft + Me.Text0.Left + intRecSelWidth
```

```
    'check if subform
    If IsSubform Then
```

```
        intBase = intTitleBarHeight + intHeightHeader + Me.WindowTop + Me.Text0.Top + Me.Text0.Height _
            + (Me.RecNum - 1) * Me.Detail.Height - intNavBarHeight - intBorderHeight
```

```
    Else
```

```
        intBase = intTitleBarHeight + intHeightHeader + Me.WindowTop + Me.Text0.Top + Me.Text0.Height _
            + (Me.RecNum - 1) * Me.Detail.Height
```

```
    End If
```

```
    'open / move the form
```

```
    DoCmd.OpenForm "frmZoom"
    Forms!frmZoom.Move intLeft, intBase
```

```
End If
```

```
End Sub
```

## Forms 4 & 5 - Listboxes

The listbox selection opens / moves another popup form or displays an image for the selected/highlighted record. The actions can be controlled using mouse move or a mouse click

As stated earlier in this article, **records are NOT selected when moving the mouse over the listbox**

Instead, code is used to determine the listbox position based on the calculated height of each listbox row and the mouse cursor position. The listbox row height depends on several things including font name, point size and the 1 pixel (15 twips) space left between each row for legibility.

To complicate matters, the first row is 45 twips (3 pixels) taller than all following rows. That happens whether or not column headers are displayed!

That information is used to **highlight** the record under the cursor so the data in the listbox record can be 'read' just as if it had been **selected by clicking**

Options – mouse move code can be enabled / disabled

Listbox options – column headers on/off ; change font name and font size, normal or italic style

**Listbox Popup** [Form Info] [Close]

Move the mouse or click any record in the listbox. A popup form opens in the same place to the right showing the full record

Options: Show Column headers?   
 Enable Mouse Move   
 Move Popup Form With Selection

LastName	FirstName
Bailey	Bill
Barnes	Fred
Black	Roger
Brown	Ron
Browning	Elaine
<b>Cooper</b>	<b>Angela</b>
Duncan	Roger
Jones	Alan
Ellen	
Redmond	Nadia
Riddington	Colin
Rogers	Shaun
Sanders	Tom
Spalding	Stephen
Wardley	Alyson

**Contacts**

ContactID:   
 Company: Mendip Heights Community School  
 LastName: Cooper  
 FirstName: Angela  
 EmailAddress: acooper@mendipheights.co.uk  
 JobTitle: Network Manager  
 BusinessPhone: 01904 564321  
 HomePhone:   
 MobilePhone:   
 FaxNumber:   
 Address: 123 High Street  
 City: Wells  
 County: Somerset  
 PostCode: BA5 2AA  
 Country:   
 WebPage:

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

**Listbox Images** [Form Info] [Close]

Move the mouse over the listbox (with or without the mouse held down). As you do so, the image is displayed together with the file name  
 NOTE: It is not necessary to click the listbox item

ID	Name	Type
1	Connect	gif
2	Exclamation	gif
3	Fail	gif
4	FilterOff	png
5	FilterOn	png
6	Full_Screen	png
7	Information	gif
8	Lock	png
9	Logo	gif
<b>10</b>	<b>Question</b>	<b>gif</b>
11	Success	gif
12	Unlock	png

Options: Show Column headers?  Font Name: Calibri  
 Font Size: 11 Italic? No

Image: Question.gif

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

**Listbox Images** [Form Info] [Close]

Move the mouse over the listbox (with or without the mouse held down). As you do so, the image is displayed together with the file name  
 NOTE: It is not necessary to click the listbox item

1	Connect	gif
2	Exclamation	gif
3	Fail	gif
4	FilterOff	png
5	FilterOn	png
6	Full_Screen	png
7	Information	gif
8	Lock	png
9	Logo	gif
10	Question	gif
<b>11</b>	<b>Success</b>	<b>gif</b>
12	Unlock	png

Options: Show Column headers?  Font Name: Trebuchet  
 Font Size: 14 Italic? Yes

Image: Success.gif

Version 7.3 20/02/2019 Mendip Data Systems 2005-2019

In both forms, the height of each listbox row is calculated using code like this:

```
Private Function GetListboxRowHeight()  
    'Wizhook converts row height perfectly  
    WizHook.Key = 51488399  
    Dim lx As Long, ly As Long 'width & height of character string  
    Dim LBRH As Long 'LBRH = listbox row height in twips  
  
    With Me.lstContacts 'listbox name  
        If WizHook.TwipsFromFont(.FontName, .FontSize, .FontWeight, .FontItalic, .FontUnderline, 0, _  
            "ABCghj", 0, lx, ly) = True Then  
            LBRH = ly + 15 'font height +15 twips (1px space between rows)  
        End If  
    End With  
  
End Function
```

This is used in the **listbox mouse move** code to determine the 'current' record for highlighting / 'reading'.  
For example:

```
Private Sub lstImages_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)  
    'get record in listbox by moving over record  
    'LBRH = listbox row height - calculated in GetListboxRowHeight procedure  
  
    If LBRH = 0 Then Exit Sub  
    Screen.MousePointer = 1 'arrow- helps prevent flicker  
  
    intColumnHeads = Me.lstImages.ColumnHeads 'are column headers visible?  
    '-1 if true, 0 if false so subtracting it adds 1 if headers shown  
    'deduct 45 to allow for larger top row  
    lstPos = ((y - 45) \ LBRH) + 1 + intColumnHeads  
  
    If lstPos <> OldlstPos Then 'cursor has moved  
        LC = Me.lstImages.ListCount + intColumnHeads '-1 to allow for header  
        Me.lstImages.Selected(lstPos - 1 - intColumnHeads) = True 'highlight on mouse move  
  
        'get image path  
        strPath = Nz(DLookup("FileName", "tblImages", "ID=" & lstPos), "") & "." & _  
            Nz(DLookup("FileType", "tblImages", "ID=" & lstPos), "")  
  
        If strPath <> "." Then  
            Me.lblFileName.Caption = strPath  
            strPath = CurrentProject.Path & "\Images\" & strPath  
            Me.Image1.Picture = strPath  
        End If  
  
        'show image if in list  
        If lstPos > 0 And lstPos <= LC Then  
            Me.lblImage.Visible = True  
            Me.Image1.Visible = True  
            Me.lblFileName.Visible = True  
        Else  
            Me.lblImage.Visible = False  
            Me.Image1.Visible = False  
            Me.lblFileName.Visible = False  
  
        'deselect all records if move beyond end of list  
        For N = 1 To Me.lstImages.ListCount  
            Me.lstImages.Selected(N) = False  
        Next
```

```

End If
OldIstPos = IstPos
End If

```

```

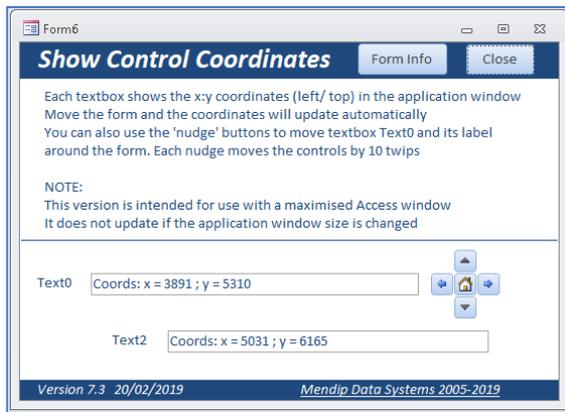
Screen.MousePointer = 0 'reset to default
End Sub

```

### Form 6 - coordinate display

This shows how coordinates can be updated as a control is moved on a form (using 'nudge' buttons) or as the form is moved around the screen

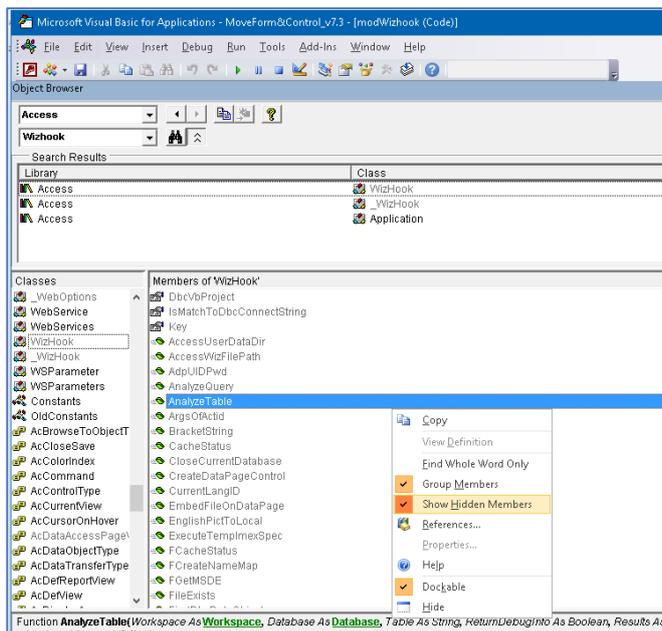
Each 'nudge' moves both textbox and label by 10 *twips* in the direction of the arrow. The 'home' button restores the original position



#### 4. Using WizHook and other hidden VBA functions

Access contains many built-in **hidden functions** - most are undocumented and its always possible that they could be removed when new versions are added. Over the years, I have used a number of these hidden members including **LoadFromText**, **SaveAsText**, **Wizhook** and others. All those I've tried are definitely functional and indeed useful.

To view **hidden functions** in the VBE right click in the object browser and select **Show Hidden Members**  
NOTE: You can use these functions even when they are hidden



The screenshot shows some of the members of the hidden **Wizhook** function.  
There are many more as you will see if you check for yourself  
I've tried several of them and found some very useful features.  
However, I've no idea what a few of them do ....YET! Give me enough time though ....!

**Wizhook** has one additional feature that is AFAIK unique.  
You MUST supply the **Wizhook** key in your code or it won't work

Code:

```
WizHook.Key = 51488399
```

Many thanks to **skrol** for alerting me to that key value a couple of years ago.  
Skrol is the author of the excellent free Access add-in, **V-Tools** <http://www.skrol29.com/us/vtools.php>  
The V-Tools deep search feature is particularly useful

#### 5. **Current and future developments**

I hope this example application will prove useful to others

I have been successfully using the **movable zoom box form** for a few years with several of my applications  
I also have a few ideas for deploying the **listbox mouse move highlight & 'select'** code in real world applications of my own.

I would be very interested in **user feedback** about how this code can be applied in other Access applications

#### 6. **Acknowledgements:**

I am extremely grateful for the valuable assistance provided by the following:

- **Stephens Lebens** – various conversion functions partly based on the **GetSystemMetrics API**
- **AccessForums.net** member **Ajax (Chris Arnold)** for various improvements to prevent **screen flicker** with the **mouse move** code used in the listbox example forms 4 & 5
- **AccessForums.net** member **daolix** for alerting me to the use of the **WizHook** function for this example
- **Skrol** for providing the **Wizhook** key needed to use the function