Document Type:	Tech Brief
Document ID:	TB010_JM
Last Modified Date:	6/7/2017 10:07 AM

Challenge:

How to place a Pop-Up screen where the button is, esp. on dual screen monitors.

Here is a method to accomplish this, extend as necessary:

Introduction: Pop-up screens are normally (almost always) opened with a *Click_Event*. Even though it is still possible to place the Pop-up within the *OpenPicture* subroutine, it gives no context **where** the *Click_Event* occurred. This is more troublesome when there are multiple monitors involved as it may not be immediately apparent that the Pop-up did open as the operator may be focused on the other screen.

Method: Determine the position of the object that is going to open the new picture and use the ",Top, Left" parameters of the **OpenPicture** function to position the new picture. This is a bit of an art and won't ever be exact, but we can get pretty close.

Two key points to remember: First the coordinates for the object on the screen and the picture to be opened, are the upper left hand corner point. Second the overall coordinates for **Workspace** are also the upper left hand corner and that point is 0,0 in both percentage and pixels. The lower right hand corner is the resolution of the screen.

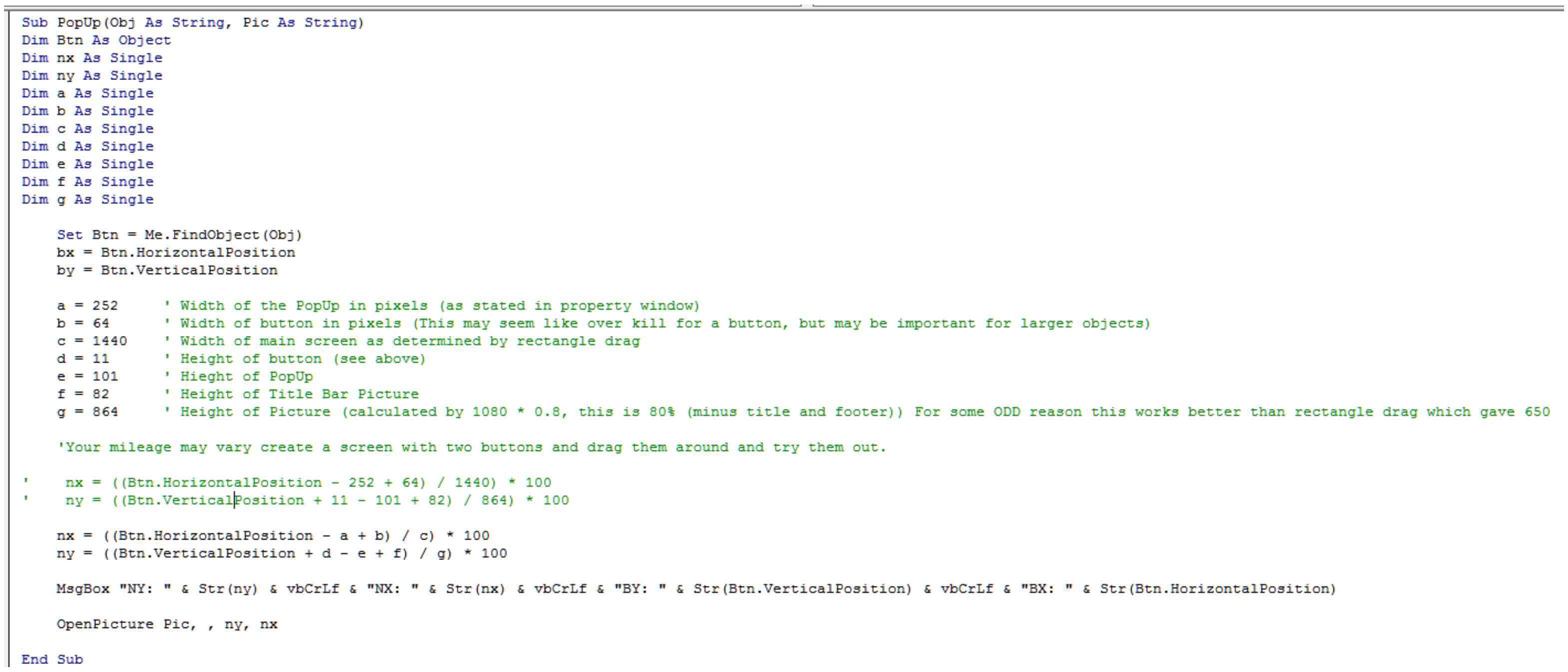
Note: Even though you can do calculations based on the actual screen resolution, I advise against this for getting best results. Create a small rectangle and drag it around and take note of the screen pixels from the "**Property Window**". Don't forget if you have a title bar etc. you may have to add those coordinates.

Step 1) Create a routine to open the picture. Depending on your Application (and taste) you may want to create a routine for each picture, or the other option is to pass more parameters or sacrifice accuracy. In my Sample I created a routine with picture scope, you can easily globalize this

Step 2) Get the position of the calling object with the **FindObject** Method.

Step 3) Use predetermined calculated numbers, re-cast them as picture percantages.

Step 4) Call the **OpenPicture** Method with to newly calculated position.



```
Sub PopUp(Obj As String, Pic As String)
Dim Btn As Object
Dim nx As Single
Dim ny As Single
Dim a As Single
Dim b As Single
Dim c As Single
Dim d As Single
Dim e As Single
Dim f As Single
Dim g As Single

    Set Btn = Me.FindObject(Obj)
    bx = Btn.HorizontalPosition
    by = Btn.VerticalPosition

    a = 252     ' Width of the PopUp in pixels (as stated in property window)
    b = 64      ' Width of button in pixels (This may seem like over kill for a button, but may be important for larger objects)
    c = 1440    ' Width of main screen as determined by rectangle drag
    d = 11      ' Height of button (see above)
    e = 101     ' Hieght of PopUp
    f = 82      ' Height of Title Bar Picture
    g = 864     ' Height of Picture (calculated by 1080 * 0.8, this is 80% (minus title and footer)) For some ODD reason this works better than rectangle drag which gave 650

    'Your mileage may vary create a screen with two buttons and drag them around and try them out.

'    nx = ((Btn.HorizontalPosition - 252 + 64) / 1440) * 100
'    ny = ((Btn.VerticalPosition + 11 - 101 + 82) / 864) * 100

    nx = ((Btn.HorizontalPosition - a + b) / c) * 100
    ny = ((Btn.VerticalPosition + d - e + f) / g) * 100

    MsgBox "NY: " & Str(ny) & vbCrLf & "NX: " & Str(nx) & vbCrLf & "BY: " & Str(Btn.VerticalPosition) & vbCrLf & "BX: " & Str(Btn.HorizontalPosition)

    OpenPicture Pic, , ny, nx

End Sub
```

Original Author: John McCue on 5/26/2017
Key Words: #popup #screen #dual #monitors #dualmonitors #workspace #picture #hmi #hmiscreens #HMI #screens
Products: iFIX
Version History: Original version