Layout Assistant Help

The intent of this tool is to allow one to group controls on a form and move groups out of the way during the design phase, and then easily return them to their original positions when done. This is handy for complex forms with many controls. I have several forms that have many layers of controls that I need to hide and unhide on occasion. Different groups of controls are shown to the user depending on the circumstances.

My application was begun in late 1991, before we had tab controls so; I had to use the enabled and visible properties often to control the interface.

For example, the "Home Page" (form "Main Switchboard") has been designed with a clean appearance and I don't really want to clutter it with tabs (although I may have to eventually for some of the things I want to do with it). It's extremely complex, with over 300 controls and has three modes:

1. **First time the application is run**: we show a few controls to record the name of the organization and facility type, then allow the user to finish and get prompted for an internet unlock code (license key).

2. **Normal Home Page**: shows controls for general user information and some decorative controls for visual organization

3. **Control Panel Mode**: hides main layer (normal home page visible) and reveals a number of user-adjustable variables.

My problem as a developer was that the sheer clutter of the form made it nearly impossible to add new controls and/or work with existing controls. My first solution was to do some custom functions that allowed me to hard-code all controls (and their positions) I wanted to move out of the way and then move them back. But this approach proved almost as impossible to maintain as the forms in question.

It occurred to me that I could use the strengths of Access to solve the development issue I'd created for myself, and, after a good 20 hours of work, the results seem useful.

To start, you'll need to open the form in design view, then run:
Function RecordControlPositions(FormName As String)

This function will capture and record the form in table tblObjectProperties and call function CaptureFormProperties to make a record of each control on the form in table tblControlPositions.

Once the records are created, one can optionally, use the "Open Form" button to open a form in design view (only pre-recorded forms will show in the list).

For the first pass, I recommend following this simple procedure:

0. Make a copy of the form you are working with in case something goes awry and work with the copy until you get the hang of it.

1. Open the form frmLayoutAssistant, select the form needing “assistance” from the list, select the section from the list, select group 1 (all controls are initially placed in group 1).

2. Manually drag the top layer (normally visible) of controls out of the way.

3. Make a new group by changing the number of one of the controls in the subform list to another number.

4. Refresh the subform and the “Section” combo box list to show new group in the “Group” list. The easy way to refresh is to simply make a selection in the section list.

5. Pick the new group from the Group list and verify it shows in the subform.

6. Select the controls you wish to place in the new group and then click the Update Selection button. This will record the controls’ current position and change their group number to the one you’ve selected.

7. Use the Expand/Collapse button in conjunction with the group list selector to confirm operation is as you intend. Selecting group 1 and then clicking the Expand toggle should throw all group members down and out of the way. Clicking the collapse toggle should bring the selected group back to their original positions.

Other functionality:

1. If you move controls around during the design phase, you can use the Update Control Properties button to record their new positions. Be careful to update the correct group since it only updates the selected group. No harm in just updating all the groups, one-by-one (especially if you are unsure). I suffer from CRS (can’t remember shit) so I am often unsure.

2. If you add new controls you can click the new controls to group button to make of record of their positions and add them to the current group (remember the group button).

3. The Delete Obsolete Controls button will remove the records for any controls you’ve deleted or renamed.

Limitations:

1. The Expand button attempts to preserve some semblance of the original control layout below the form’s original bottom. There is a limit to form height, so if you start with a form over (I think) 11.5 inches (total of header, footer and detail), it may bomb due to lack of available space. There is also a width limitation, so we can’t go to the right. I generally design to a form width of 13.125 inches and a total height of 6.5 inches which corresponds to no scaling needed on a 1366 x 768 monitor (app

maximized). I use a module called shrinker-stretcher sold by Peter's Software for form scaling in different size monitors. (Note, I hide the status bar to maximize screen real estate and the title bar (which I cannot hide) for navigation assistance and messaging).

2. Testing: I haven't worked with this extensively, so there may be bugs. This is a beta and may remain a beta forever. Beta stands for "**B**ugs **E**xist **T**o **A**nnoy".

3. Option Groups: Are a pain since moving a toggle or button or anything in them stretches the frame. If anyone has the time to come up with a solution, I'd appreciate it. I simply don't try to move them in practice. I'd guess the solution would involve recording the frames original height and width properties and then restoring them after the move(s). I just don't have any time left currently to devote to this issue.

Contents of Package

Forms

frmHelp
frmProposalDashboard
frmLayoutAssistant
frmLayoutAssistantSub
frmProposalDashboard (example 1)
Main Switchboard (example 2)

Tables

tblObjectProperties
tblControlPositions
helpTbl

Queries

qryfrmLayoutAssistant00
qryRecordControlPositions
qryRecordControlPositions00 thru qryRecordControlPositions14 (15 each)

Reports

Help

Modules

Utilities

Functions In Module Utilities

1. Public Sub Restart(Optional Compact As Boolean = False) this is not part of the form layout subsystem but I've found it handy for force a restart for users working in my app who, due to custom ribbons, do not have the standard Access commands. Otherwise, since I distribute an accde file, they would have to completely close and reopen. It was only needed recently, only because I've implemented Excel automation for data import and export that tends to bloat the front end.

2. Function RecordControlPositions(FormName As String) 'records control positions into table tblControlPositions after making a record of the form in tblObjectProperties by calling:

3. Function CaptureFormProperties(FormName As String)

4. Function RecordNewControlPositions(FormName As String, Group As Integer) 'Adds new controls to an existing layout and designate their group.

5. Function ExpandContract(Expand As Boolean, FormName As String, ControlGroup As Integer, Section As Integer) 'moves controls in a selected group out of the way, and back to their original places.

6. Function RevertForm(FormName As String, SectionNumber As Integer) 'cleans up after collapsing by attempting to return the form to its design dimensions (all groups must be collapsed)

7. Function UpdateControlProperties(FormName As String, ControlGroup As Integer) 'updates positions of controls that have been moved (selected group only).

8. Function UpdateSelection(FormName As String, ControlGroup As Integer) 'update of positional data for selected controls and changes their group to that selected. Note, there controls must first be added to the layout. Thanks Marshall Barton, wherever you are, I did not know about the control.inselection property until I noticed it in one of your posting.

9. Function ConvertSectionToString(SecNum As Integer) As String 'used in a query to return form section indices in English.

10. Function SetGroupVisibilities(ShowBottom As Boolean, FormName As String, ControlGroup As Integer, Section As Integer). Not for design view: I found that once the controls are grouped and in a table, it's easy enough to set their visible property to hide and show whole groups. Yes, there are other ways such as using the tag property, but once you've got a nice little tool like this, maybe you'll find it easier. There are so many ways to skin cats in Access, I wonder how any cats are left with their fur intact.

11. Function UpdateCurrentControlProperties(FormName As String, ControlName As String) 'used on subform button on continuous record to update an individual control's positional data.

12. Function RemoveRenamedOrDeletedControls(FormName As String) 'used to remove the record of any control no longer existing. Handy with you rename controls as well as when controls are deleted.

13. Function ActiveControlDropdown() 'used on frmLayoutAssistant to automatically drop combo box lists down when gotfocus and/or mousedown events occur.

14. Function CallHelp() 'used in help button's on_click event to open the help form and filter its contents for relevancy to the current form (poor-man's context sensitive help). I suppose could be applied to the active control as well if called from a control.

Comments and a call for help:

1. Yes, I know I overdo typing of the tempvars. The reason is that I've found them temperamental in queries and when capturing the data in form controls. Since sometimes they need the typing and sometimes not, it's just easier for me to always type them.

2. Yes, I know when I use a string in a function call, I don't have to type it – just a bad habit brought on by addiction to tempvars.

3. I spent a couple of weeks developing a work-around to the issue of not being able to append an attachment to a table. The code works nicely, but is hard coded to one particular application (synchronizing inspection records and attached pictures). I solved it by looping through the recordset of the "from" table and then looping through the attachment field and using the savetofile and loadtofile methods to "move" the attachment.

I think the function could be generalized to take just the names of the "from" table, the "to" table, and the attachment field and this would make it much more useful. I suspect the solution would be first take one of the tables (assuming they are identical in structure), recording the names of fields (and counting the number of fields) and then looping through the record set using variables for the rst!field statements where "field" is variable. If anyone wants to help with that, we could post it. I know a lot of people are looking for a solution to that problem.