Let us start with Sample App and Import Required Components into Your Application to Get Started:

1. Download Sample application as available on website link https://erpmakers.com/MS%20Access%20Form%20Resizer.php .

2. From the sample application, import following tables into your application:
   a. *TBL_FE_FormResizer_ControlReferences*
   b. *TBL_FE_FormResizer_ControlTypes*
   c. *TBL_FE_FormResizer_HashTables*
   d. TBL_FE_FormResizer_DatasheetViewFormSetting
   e. TBL_FE_FormResizer_MasterSettings

   Tables marked with red color text are system tables and should not be modified to keep Form-Resizer working without any trouble.

   When it comes to changing the Fonts of Datasheet view forms, sometimes, MS-Access acts unpredictably. To deal with the issue, Form Resizer stores Font Size of all Datasheet View forms in Table TBL_FE_FormResizer_DatasheetViewFormSetting. Usually, this table is also going to act as a read only table. If you want to change font-size of any Datasheet View Form, simply look for desired form-name in the table and change the Font-Size. Form Resizer will start showing the Form accordingly.

   Table TBL_FE_FormResizer_MasterSettings is the place where all settings related to your setup will be stored. We will discuss about this table in detail at later stage.

3. From the sample application, import module named “mdlMSAFormResizer_SysReferences_Register”. This module has code to auto determine your Application Type (ACCDB vs MDB) and Platform Bit-Size (64-Bit vs 32 Bit). It also has function named FR_AddSystemReference, which is used to auto register system-components based on your Application Type and Platform Bit-Size. This function is used either in AutoExec Macro or Open Event of Startup-Form.

4. In the Sample App, there is a macro named AutoExec_Sample and a form named frmWelcome. You just need to use one component out of these two. If you are using older version of Access (2000/2003), import the macro named “AutoExec_Sample” from the sample application and use this. If you are using newer version of Microsoft Access, copy the code available in Open-Event of form named “frmWelcome” and use that code in your own startup form. In Sample Application, form named “frmWelcome” has been setup as startup form and it triggers before AutoExec Macro.

   Both AutoExec_Sample macro and frmWelcome form serves the same purpose and have same lines of code. Just use the one that suits you better.
   a. Your application may already have macro named “AutoExec. In such case, just manually add code available in the AutoExec_Sample macro to your own macro, else after importing the macre, change the name of AutoExec_Sample macro to AutoExec.

b. If you are using newer version of Access and do not want to use AutoExec Macro and have setup a startup form instead, just manually add code available in open event of form frmWelcome to your own startup form.
c. First line of code in the macro or form is responsible to auto register required system-components with your application and thus makes all form-resizing related functions available for your application. This also helps in auto deployment of Form-Resizer to multiple machines without any manual/special effort because desired components automatically get registered silently.
   The function called for this purpose is named as FR_AddSystemReference().
d. When required system-components are missing on a machine, which is an obvious situation when you are going to run the Sample App (or your own application) for the very first time on your machine/location, component registration code will execute. This execution of code take place only once.
e. Before our code gets any opportunity to register the required components, MS-Access looks for missing references and fires an Informative Message. Click Ok to move next and our code will start its process. Just open the application again and everything will run fine second time.
f. Second Line of code in macro/form is to open frmEmployee. In Sample App, frmEmployee is the main/only form in the application. You should change this code to open your own main form as desired.

5. In the sample application, there is a form named frmEmployee, which is the main form of Sample-Application to display some information. In frmEmployee Form's Open-Event, there are four lines of code. Copy these four lines of code in your main form's open event. In your application, you need not to import any form/table related to Form frmEmployee.
   a. There may be a possibility that older version of MS-Access Form-Resizer's components may be existing on your machine. First line of code is a call to function FR_CheckSystemComponentsPathAndVersion which will automatically update the components if latest version is not found.
   b. There are several Static Variables available in MS Access Form Resizer those hold the Master-Settings and information related to screen resolution etc. Function called in the second line of code is responsible to set/reset all such Static Variable. Function name is FR_ResetValuesToResizeFormsBasedOnResolution(). You should call this function during application startup. You should also call this function whenever you make any change to the Master-Settings.
   c. Function called in third line of code is responsible to register your machine with us. The function called is FR_LoadMachineInfo("RegisteredEmail@mail.com", "Password"). Placement of this function in startup form helps you in automated deployment of your application on different machines and makes sure that every-machine gets auto registered.
      If you have purchased an App-Based License (Instead of Machine Based License), please use function named FR_LoadAppInfo("RegisteredEmail@mail.com", "Password").
   d. Fourth line of code is FR_ReSizeFormAllSubFormBasedOnSystemResolution(Me). This is the main function responsible to resize the form itself (along with its sub-forms). In our case main form's name is frmEmployee but we are passing just Me because Me represents the Form itself. You will be using this function all across your application.

Registering the License:-

There are following functions available for the said purpose.

1. *FR_LoadMachineInfo (UserName As String, Password As String) As Boolean*
   This function is used to register your machine. This can be safely (without any speed issue) called in Application startup and will run only if your machine is not already registered.

2. *FR_LoadMachineInfo_Once(UserName As String, Password As String) As Boolean*
   This function is used to register your machine. This should not be called in Application startup because it will try to register the machine when called.

3. *FR_UnloadMachineInfo(UserName As String, Password As String) As Boolean*
   Purpose of this function is to deregister (previously registered) machine from the associated license. This function is used when you want to migrate your license to some other machine. If you have lost access to your machine and want to deregister the license info from such machine, just let us know.

4. *FR_LoadAppInfo(UserName As String, Password As String) As Boolean*
   In case you have purchased App-Based License for distribution of your application to your larger audience, use this function. Purpose of this function is to register your application. This can be safely (without any speed issue) called in Application startup and will run only if your application is not already registered. App-Based Licensing is liked to your Application Title. If you want to change the Application Title linked with your License, just inform us.

5. *FR_LoadAppInfo_Once(UserName As String, Password As String) As Boolean*
   Purpose of this function is same as above with a difference that this should not be called in Application startup because it will try to register the application when called.

6. *FR_ChangeUserPassword(UserName As String, CurrentPassword As String, NewPassword As String) As Boolean*
   An account is composed of a username (email-address) and a password. Password is issued from our end. You can use this function to change your password. This function can also be used if your password has been stolen somehow.

What's inside the Master-Settings Table:-

There is a table named TBL_FE_FormResizer_MasterSettings which you must have imported from Sample-Application. In this table, you can store your settings.

If you already have a table/form which is dedicated to manage your own application's settings, then you can also keep Form-Resizer related settings in your own table too. In such case, you need to write an event which can post the updated values into TBL_FE_FormResizer_MasterSettings table.

Values stored in this table are also stored in Static Variables for faster access to the desired values. If you make any change to any value in this table, please make sure to call the function responsible to reset such static values.

Function responsible to reset all static values as per new setting is named as *FR_ResetValuesToResizeFormsBasedOnResolution(Optional ByVal blnShowMsg As Boolean = False)* . You can also call this function in AutoExec Macro or your Startup Form. This function also refreshes many other static values those are not part of Master-Settings Table (like Screen-Resolution, DPI, etc).

Following are the fields available in this table.

1. *FormResizeMethod*
   Usually, users prefer to increase/decrease the Form's Width and Height both as per screen resolution. But some users prefer to utilize larger screen's height to show more records/content (instead of larger controls). Sometimes, you may have to turn-off the resizing feature to have a look at the real picture of the forms, without opening them in design view.

   There are three resizing methods available. Mentioned below is the values and their meaning:-

Value	Meaning
0	Do not resize the form.
1	Just resize the Width of Controls. Do not change height.
2	Increase both Width and Height of all controls.

   When you plan to use Value-1, make sure to add Stretch-Down anchoring to the container-control in which you want to show more records. If any control is available at the bottom of container-control (inside which you want to show more records/content), make sure to add Bottom-Left anchoring to such control. Note that Anchoring feature is not available in older versions of access. MS-Access Form Resizer handle anchoring well when applied and also runs fine on MS-Access versions where anchoring feature is not available.

2. *ScreenSizeSource*
   There may be a need to resize your application's screen to fit on Physical-Screen or Access-Application.

   In case of "Physical-Screen" option, form-controls resize as per screen resolution but their size does not change when Access-Application Window's Size changes (Not Maximized State).

   In case of option "Access Application", form-controls resize to fit within MS-Access-Application's Inside-Windows during all states of MS-Access Application (whether maximized or not).

   Both options have their own benefits. Please choose as per your requirement. It is suggested to use "Physical Screen" option for large applications and use "Access Application" for light/mid weight applications.

3. *FormSizeSource*

   There are two options available "StandardDimension" and "Form_OwnDimension".

   When option "StandardDimension" is selected, all forms (whether Small/Large) are going to increase/decrease by same proportion. This helps in giving professional look to your forms because user is going to find uniformity in fort/control size increase/decrease across application.

When option "Form_OwnDimension" is selected, all forms (whether Small/Large) are going to resize to fit well in available screen area. This option is bypassed in case of Popups and Datasheet View Forms for a reason as explained below. If you want to open a full-size form as popup, in the form's tag, put text "Main Form".

This point is quite important for you to understand how the size calculation works. MS Access Form Resizer compares Top-Level Form's Dimension with Screen/Application's Dimensions and calculates the factor by which all components (Including Sub-Forms) should be scaled up/down. This calculation brings up two values i.e. Width-Factor and Height-Factor. MS Access Form Resizer uses Width-factor as it is calculated and scales up/down the width of all controls. For Height Scaling, MS Access Form Resizer uses least of the two calculated factors (Width vs Height) and resizes height of all controls accordingly. In this case, there may be some space left at the bottom of the screen which can be increased with help of anchoring. Anchoring is also used by users who prefer to only increase the width of controls but utilizes Extra-Hight to display more records.

So, for above said calculation, we need Form's Dimensions. Interestingly, fact is that we cannot simply use all Forms' Dimensions as it is, because we would like font on all forms (whether small or large) to be increased/decreased by almost same proportion. For example, we do not want Popups to enlarge to the extent that they fit on whole screen (to look as large as the main forms are going to look like) but want Popups' dimensions to increase in a proportion by which main forms are going to scale up/down.

4. *StandardFormWidthInTwips*
   Usually, most of the forms in an application are designed with same Width and Height (Except Popups). This field is the place where you can keep Width Value that is used by most of the forms in your application.

   The functions those are responsible for resizing your forms, take Form-Width as optional argument. If Form Width is not provided to these functions, they read the value from this field.

   If all/few forms in your applications are of varying size, you need not to worry. You can simply pass the Form-Width as argument to the resizing functions. But it is suggested that you depend more on the standard value in this field (instead of passing individual form's dimension to function) so that all your forms (whether Small or Large) resize to the same scale.

   You cannot keep this value blank as it also helps in rendering the Popups and Datasheet View Forms to the scale by which Standard-Forms should be resized.

   Form Width in the Field must be stored in Twips. If your form's dimensions are in Inches, simply multiply the Inches Value with 1440 and store the resultant value in this field. If your form's dimensions are in centimeters, simply multiply the centimeters value with 567 (566.9291 to be precise) and store the resultant value in this field. Or, simply test Me.Width on a form as this variable itself returns value in Twips.

5. *StandardFormHeightInTwips*
   Just similar to above field, this field is the place where you keep Height Value (Including Header/Footer) used by most of the forms in your application.

If all/few forms in your applications are of varying size, you need not to worry. You can simply pass the Form-Height (Including Header/Footer if they are visible) as argument to the resizing functions.

6. *DefaultTabPageBorderHeightInTwips*
End purpose of this field is to help resizer in calculating Tab-Fixed-Height (A property of Tab-Control).

In MS Access, you can keep Tab's Fixed Height (even Width) to zero which means MS Access should automatically set the Tab-Fixed-Height as per the font-size used for Tab-Control. When Tab's Fixed-Height is Zero, Form Resizer also set that Tab's Fixed Height to Zero (because anything multiplied with Zero is Zero). When Form Resizer has set the Tab's Fixed-Height to zero and changed the Font as per resolution, outcome may have slight variance from the desired result.

To generate proper appearance for Tab-Controls, where Tab's Fixed Height has been set to zero, set a Tab's Fixed Height (other than zero) to such a value that you get same appearance as you get after setting the value to zero. Note this value and enter in below mentioned formula to get the desired value for this table-field.
(Tab's Total Height – Tab Page's Height – Tab's Fixed Height)

7. *NavigationPaneState*
Navigation Pane is the area from where users access their form/reports/tables as per their need. Navigation page can be made Hidden, Folded, and can also be kept Unfolded for quick-access to the desired Objects.

This Table Field is a Drop-Down field, and you can choose the setting as desired by you. Options available are Folded, Unfolded and Hidden.

Navigation-Pane is not available in older versions of Microsoft Access. For such versions, keep the NavigationPaneState Field's value to Hidden.

This setting ceases to play any role when value in field ScreenSizeSource has been set to "Access Application" because in such case Forms are always going to be resized to fit within the area left inside the MS-Access Application Screen.

8. *DPIAwareness*
In Microsoft Windows, there is a feature available using which one can show the text larger by 25% or 50%. This setting is also called as DPI-Setting. When user changes the text size setting, applications designed in MS Access fail to render properly. In MS Access Form Resizer, feature has been added to resize the form as per DPI setting.

This table-field is a checkbox and you can turn it ON/OFF. When you change DPI-Setting in Windows, application needs to restart to take effect as per new setting.

When you turn-on this table-field, MS Access Form Resizer will try to fit all your form controls within the screen area. Since height/width of Title-Bar, Menu-Bar, Toolbar, and Task-Bar has already been increased by Windows and that cannot be reduced by our code, area available to show you controls (Form's Inside Area) actually gets reduced. So, when MS-Access Form Resizer tries to fit controls as per screen, the form's (including its controls) width and height gets reduced.

If you want to look at large sized controls as per the chosen DPI setting in Microsoft Windows, keep this Table-Field turned off. In such case, it is always advised to add both horizontal and vertical scrollbars to your forms so that excessively increased form area by windows remains accessible to the users.

Instead of increasing the Text-Size in Microsoft Windows (using DPI Setting), it is always advised to choose lower resolution for better viewing of the forms. Such option helps in having bigger fonts while keeping all components of screen in proportion.

This setting ceases to play any role when value in field ScreenSizeSource has been set to "Access Application" because in such case Forms are going to be resized to fit within the area left inside the MS-Access Application Screen.

9. *ErrorHandlingMethod*
This field is a drop-down field, and you can choose the option as desired. The options available are "Show Errors On Screen", and "Write Errors to Log File".

In development mode, choose option "Show Errors On Screen" to have quick access to the underlying issues.

For end users, you can turn on option "Write Errors to Log File". When this option is turned on, MS Access Form Resizer will create a text file named "MS Access Form Resizer Error Log.txt" which gets stored in a folder named "Resources" available in the Current-Project Path.

This option is applicable to main functions responsible for resizing the forms. If any error fires in secondary functions which is beyond the expectations of the developers, popup message will appear on screen.

10. *KeepBlank_HeightOfNonScreenComponentsInTwips*
This field should be kept blank in most of the situations. Keep this field Blank because MS Access Form Resizer can calculate Form Height Dynamically. Value entered in this field (if not left blank) supersedes the dynamic calculations performed by MS Access Form Resizer.

This fields come to use only in case of older version of MS Access like 2000/2003. In these versions, user can add any number of toolbars in the header section which can reduce the screen-height available for displaying form. If you are adding/reducing number of toolbars or your screen has extra height available at the bottom, just enter a value in this field calculated as per below mentioned formula.

(Height Values of TitleBar+MenuBar+ToolBar+TabbedPage+StatusBar+TaskBar).

In newer versions of Microsoft Access, there is a feature called Ribbon. You need not to use this field if you keep your Ribbon open because MS Access Form Resizer automatically handles the Ribbon height as per its visibility.

This setting ceases to play any role when value in field ScreenSizeSource has been set to "Access Application" because in such case Forms are going to be resized to fit within the area left inside the MS-Access Application Screen.

Let's learn the Form-Resizing Subroutines: -

These subroutines are the ones you are going to use in your code which will resize the forms as per resolution.

You can call these Subroutines in your form's open event or load vent or resize event. Preferable, call them at the end of the Open event so that if you are changing Width/Height/Left/Top of any control in the Open Event, form will be resized properly while taking your own code into effect.

Following is the list of functions available for the said purpose.

1. *Public Sub FR_ReSizeFormAllSubFormBasedOnSystemResolution (frm As Form, Optional ByVal byteFormResizeMethod As Variant, Optional ByVal dblTopLevelFormDesignTimeWidthInTwips As Double, Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Double, Optional ByVal blnResizeFormAsPerInsideWidthHeight As Boolean, Optional ByVal blnOpenPopUpFormAsFullScreen As Boolean = False, Optional ByVal strSkipControlsToResizeBasedOnResolution As String = "")*

   This function will resize all controls available on your form. It also resizes controls available on all subforms of all types (Single View Form, Continuous Form, and Datasheet Form). This function is the main function and will be used for resizing almost all your forms.

   First argument of this function named frm is of type Form Object. You should pass Me or Form_FormName as value for this variable.

   All arguments, except first one, are optional arguments.

   Second argument *byteFormResizeMethod* represents the form resize method. The possible values are 0/1/2. In case value is not provided to this argument, Resize-Method-Setting is derived from the table responsible for holding master settings. You should pass value to this function only if you want to resize a specific form always to a particular resizing method, else should be kept blank.

   Third and Fourth arguments dblTopLevelFormDesignTimeWidthInTwips and dblTopLevelFormDesignTimeHeightInTwips are also optional. Their values are derived from table responsible to hold master settings. If your form size is different from the standard size (what is defined in master settings table) and you want such form to be resized by a different proportion, use these variables for the said purpose by passing form's width and height to get desired rendering.

   Argument *blnResizeFormAsPerInsideWidthHeight* is mostly come into use when you have a secondary monitor (Dual Monitor) attached to your machine and you want to keep one form open on your secondary monitor. Secondary monitor may be serving the purpose of Customer-Facing Screen or an additional screen where you want to display some other information. If you open a simple-popup, you can move that Popup to secondary monitor. Usually, such popup is displayed as Full Screen Form. On secondary monitor, screen-resolution may be different from your main monitor. To deal with this issue, you can set your form's inside height/width in your code and then make a call to this function with this argument as True, form resizer will resize all the controls on the form as per the form's inside width/height set in last step (before making call to this function). This argument does not apply in case of Datasheet-View-Form and you may need to convert your form to Continuous-Form if you want to show such form on secondary monitor.

Argument blnOpenPopUpFormAsFullScreen is self-explanatory. Pass True if you want to open popups as Full Screen Form.

If you want to handle a special resizing/repositioning situation for some specific controls on your form through your own customized code, you will prefer MS Access Form Resizer to not to resize/reposition such controls. For this purpose, argument strSkipControlsToResizeBasedOnResolution comes into picture. You can pass the list of such controls as a comma separated list. List of controls must start with comma and end with comma like ",control1,control2,control3,"

Though, there is a dedicated function available to resize Datasheet Forms, but this function can also be safely used for resizing Datasheet Forms.

2. *Public Sub FR_ReSizeFormBasedOnSystemResolution(frm As Form, Optional ByVal byteFormResizeMethod As Variant, Optional ByVal dblTopLevelFormDesignTimeWidthInTwips As Variant, Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Variant, Optional ByVal blnResizeFormAsPerInsideWidthHeight As Boolean, Optional ByVal blnOpenPopUpFormAsFullScreen As Boolean = False, Optional ByVal strSkipControlsToResizeBasedOnResolution As String = "")*

   This function works just same as function *FR_ReSizeFormAllSubFormBasedOnSystemResolution* with exception that this function will not resize the subforms.

   Though, there is a dedicated function available to resize Datasheet Form, but this function can also be safely used for resizing Datasheet Form.

3. *Public Sub FR_ReSizeDatasheetFormAllSubFormBasedOnSystemResolution(frm As Form, Optional ByVal byteFormResizeMethod As Variant, Optional ByVal strSkipControlsToResizeBasedOnResolution As String = "")*

   This function works just same as function *FR_ReSizeFormAllSubFormBasedOnSystemResolution* except one point. This function will work only for Datasheet Forms.

4. *Public Sub FR_ReSizeDatasheetFormBasedOnSystemResolution(frm As Form, Optional ByVal byteFormResizeMethod As Variant, Optional ByVal strSkipControlsToResizeBasedOnResolution As String = "")*

   This function works just same as function *FR_ReSizeDatasheetFormAllSubFormBasedOnSystemResolution* with exception that this function will not resize the subforms.

5. *Public Sub FR_ResetValuesToResizeFormsBasedOnResolution (Optional ByVal blnShowMsg As Boolean = False)*

This function resets all the values stored in static variables those are used in form-resizing related operations.

6. *Public Sub FR_FormResizeMethod (Optional ByVal blnReRun As Boolean = False) As Byte*

   Returns the value of parameter *FormResizeMethod as stored in table TBL_FE_FormResizer_MasterSettings.*

   This function holds the value of this parameter as a static variable. After making any change to this value in table, you need to pass the argument value blnReRun as True so that this function starts providing you the latest value. It is always better to call function FR_ResetValuesToResizeFormsBasedOnResolution because this function resets value stored in all such static variable.

   These functions are not generally of any use to end programmer but still provided to help you in dealing special situations.

   More such functions are available to retrieve values stores in table TBL_FE_FormResizer_MasterSettings. Names of such functions are as below.
   FR_ScreenSizeSource()
   FR_FormSizeSource()
   FR_StandardFormWidthInTwips()
   FR_StandardFormHeightInTwips()
   FR_DefaultTabPageBorderHeightInTwips()
   FR_NavigationPaneState()
   FR_DPIAwareness()
   FR_ErrorHandlingMethod()
   FR_KeepBlank_HeightOfNonScreenComponentsInTwips()

7. *Public Sub FR_GetFormFactors(frm As Form, Optional ByRef dblResolutionFactor_Width_Actual As Double, Optional ByRef dblResolutionFactor_Height_Lower As Double, Optional ByRef dblResolutionFactor_Height_Actual As Double, Optional ByVal ReturnOneWhenParentNotVisible As Boolean = False)*

   If you want to change Left/Top/Width/Height of your controls on form in some special way and need to know the multiplying-factor that has been applied by Form-Resizer and would like to use that factor in your calculation, then this function is of use.

   You have to pass Form Object (usually Me), and two variables by reference. This function will set value in these two variables and then after you can use them.

   When it comes to Height-Factor, system always compare Width Factor with Height Factor and uses the smaller-one of the two as Height-Factor. Parameter dblResolutionFactor_Height_Actual can be used if you want to know actual Height-Factor. But this will not be of much use.

   Sometimes, programmer may need the function to return One as Factor Value when Parent-Form is still not visible. In such case pass True for argument ReturnOneWhenParentNotVisible.

8. *Public Function FR_GetResolutionFactor_Width_Actual(Optional ByVal dblTopLevelFormDesignTimeWidthInTwips As Double, Optional ByVal blnReRun As Boolean = False) As Single*

   Returns the factor value calculated after dividing main screen width with form width. This function is provided only to deal special situations.

9. *Public Function FR_GetResolutionFactor_Height_InsideScreen_Actual(Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Double, Optional ByVal blnReRun As Boolean = False) As Single*

   Returns the factor value calculated after dividing inside screen height with form height. This function is provided only to deal special situations.

10. *Public Function FR_GetResolutionFactor_Height_InsideScreen_LowerValue(Optional ByVal dblTopLevelFormDesignTimeWidthInTwips As Double, Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Double, Optional ByVal blnReRun As Boolean = False) As Single*

    Returns minimum of the internal-screen-height-factor vs width-factor. This function is provided only to deal special situations.

11. *Public Function FR_GetResolutionFactor_Height_FullScreen_Actual(Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Double, Optional ByVal blnReRun As Boolean = False) As Single*

    Returns the factor value calculated after dividing main screen height with form height. This function is provided only to deal special situations.

12. *Public Function FR_GetResolutionFactor_Height_FullScreen_LowerValue(Optional ByVal dblTopLevelFormDesignTimeWidthInTwips As Double, Optional ByVal dblTopLevelFormDesignTimeHeightInTwips As Double, Optional ByVal blnReRun As Boolean = False) As Single*

    Returns minimum of the main-screen-height-factor vs width-factor. This function is provided only to deal special situations.

Miscellaneous Points: -

1. *Public Function FR_Version() As String*
   This function provides the current version number of MS Access Form Resizer.

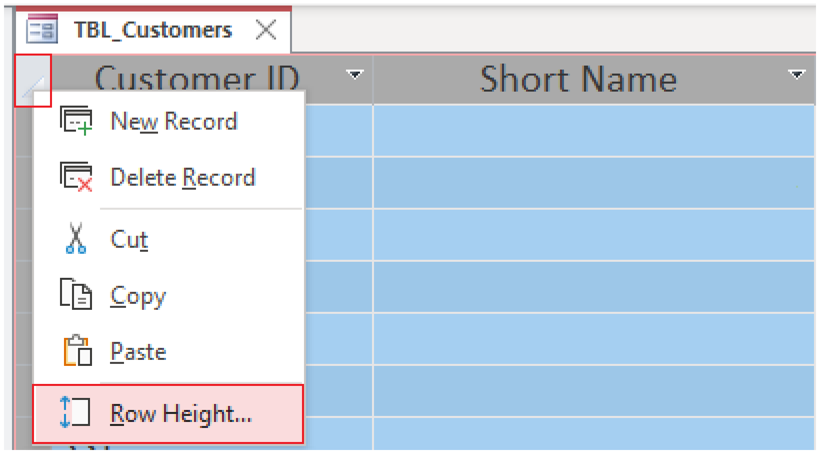
2. *What is special about Popup Form*
   Form has a property named Popup. You need to turn-on this property of form to get your popup forms resized properly because Form-Resizer treat a form as Popup only when Popup property of form is Turned ON. Programmers can open a small sized form as Pop-up by opening them in Dialog-Mode, but form resizer will not treat it as Pop-up and will resize it as a normal form.

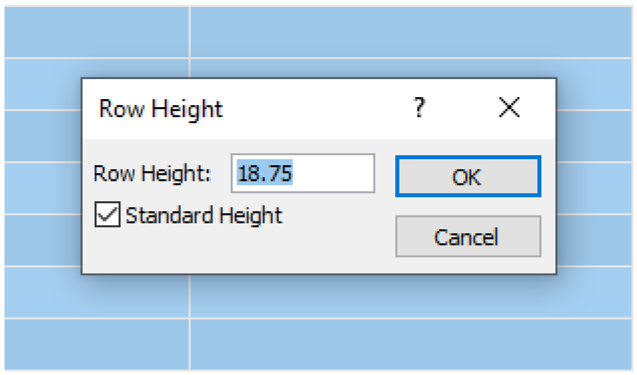
3. *How to open Popup as Full-Sized Forms*
   MS Access Form Resizer treats Popups in a special way. It uses StandardFormWidth and StandardFormHeight to calculate the factor by which Popup should resize and they do not re-resize during its open state. If you want to open your popup as a Main Form (Full-Sized on whole screen area), add text "Main Form" in the tag-property of the Form. This way, Popup-Form will Open in Popup-State and will also behave like a Full-Size form.

4. *Datasheet Forms are not resizing at all*

   If your controls are not visible in design time, resize function may fail to increase/decrease the control size. Please try changing control's visibility in design time.

   If your control width is increasing/decreasing but font is not increasing/decreasing, it means that Standard Height Option is turned off for your Datasheet View. You need to open your form in datasheet view, then right click in the left top corner, then choose Row Height Option, then then Turn-On Standard Height Option. Please refer to the images provided below.





Row Height: 18.75

☑ Standard Height

5. *Datasheet Forms are showing some erratic behavior*
   When it comes to changing the Datasheet view forms, sometimes, MS-Access sometimes acts unpredictably. It tries to save the changes one has made during runtime but fails at some points.

   It is generally found that if there is single visible datasheet sub-form available on screen, MS Access usually able to save the changes. Whereas when there are two datasheet sub-forms being visible on screen, it usually fails to save the runtime changes.

   In Form-Resizer, we have tried to deal with this erratic behavior of MS Access.
   a. There is a Table named TBL_FE_FormResizer_DatasheetViewFormSetting. Whenever Form Resizer tries to resize a Datasheet View Form, it adds a record for that Form in this table.
   b. Table TBL_FE_FormResizer_DatasheetViewFormSetting stores Form Name, Font Size read at First Time (Design Time), and the Resolution based Multiplier applier on Width of All Controls.
   c. At the time of resize event, chances are there that MS Access has saved the new Font Size. So, whenever Form is resized again, Form Resizer uses the Font Size Stored in Table (Stored at First Time) and applies the multiplying factor based on resolution on this value to calculate the desired Font Size.
   d. At the time of resize event, chances are there that MS Access has saved the new Column Width for all controls. So, whenever Form is resized again, MS Access Form Resizer calculates the new Factor based on Screen-Resolution but does not apply this new Factor to calculate new Column Widths. Instead, it stores the new Factor in Table (to be used next time). On Column Width, it applies the factor derived after dividing new factor by old factor.
   e. At the time of resize event, since chances are there that MS Access could not save the new Column Width for all controls. To deal with this issue, you need not to do anything with respect to font size. But regarding calculation of Column Widths, you are required to add code to deal with the situation. There are two code options available as mentioned below. You can use any one option as desired (Option ii is suggested method).
      i. In Table TBL_FE_FormResizer_DatasheetViewFormSetting, there is a field named PreviouslyApplied_ResolutionFactor_Width. You need to set this value to 1 before calling the Form-Resizer related Functions.
      ii. There is a Public TempVar Variable available with name FR_dblPreviouslyApplied_ResolutionFactorOnDataSheetForm. You need to set its value to 1 before calling resize function. You also need to set its value to Zero or Null after call to the resize function. Sample Code as below:-
          TempVars! FR_dblPreviouslyApplied_ResolutionFactorOnDataSheetForm = 1

   f. If there is a sub-form on two different main forms and sub-form's behavior with respect to saving runtime changes is different on different main-forms, then you have two options available as mentioned below (Option ii is suggested method)
      i. Consider two copies of sub-form
      ii. Consider setting sub-form's control-widths through code so that datasheet controls always have same width before call to resize method and set TempVar to 1. Code example as below :-

```
TempVars! FR_dblPreviouslyApplied_ResolutionFactorOnDataSheetForm = 1

Me.Field2.ColumnWidth = 1600
Me.Field3.ColumnWidth = 1600
Me. Field4.ColumnWidth = 900
Me.Field5.ColumnWidth = 1600

Call FR_ReSizeDatasheetFormBasedOnSystemResolution(Me)
```

g. Usually, table TBL_FE_FormResizer_DatasheetViewFormSetting is also going to act as a read only table. If you want to change font-size of any Datasheet View Form, simply look for desired form-name in the table and change the Font-Size. Form Resizer will start showing the font accordingly.

6. *What is the best way to deal with erratic behavior of Datasheet Forms?*

   Follow the steps (a and b both) as mentioned below: -

   a. Set TempVar to 1 once in Open Event of First Form of the application. Code example as below :-

   ```
   TempVars! FR_dblPreviouslyApplied_ResolutionFactorOnDataSheetForm = 1
   ```

   b. Always set datasheet-form's control-widths through code so that datasheet controls always have same width before call to resize method. Code example as below :-

   ```
   Me.Field2.ColumnWidth = 1600
   Me.Field3.ColumnWidth = 1600
   Me. Field4.ColumnWidth = 900
   Me.Field5.ColumnWidth = 1600
   Call FR_ReSizeDatasheetFormBasedOnSystemResolution(Me)
   ```

7. *Using MS Access Form Resizer, Image Controls can be resized in two ways.*
   a. By default, proportion of the Image-Control is maintained while performing the resizing/reposition of images. This means that Height and Width both will increase/decrease by equal percentage. Left/Top is also changed while respecting the proportion.
   b. If you want to use an Image Control as a Background Image whose Width and Height should increase/decrease as the form's Width/Heights changes, you can use Tag Property available for Image-Control Field. In the Tag, simply put text "Background Image" (Without Double Quotes). This tag is an indication for MS Access Form Resizer to treat the image differently. To get the desired results, you may also need to set Properties for Image Control as mentioned below.
      Picture Tiling = Yes
      Size Mode = Stretch
      Picture Alignment = Center

8. *Handling Stacked/Tabular Layout*
   If you are using Stacked/Tabular Layout feature for designing your forms, which works well in case of Simple-Form-Designs, MS Access Form Resizer respects the layout-rules and resizes the form accordingly.

9. *Using Horizontal/Vertical Anchoring with MS Access Form Resizer*
   Horizontal/Vertical Anchoring feature helps in resizing/reposition the controls to greater extent. This is quite useful for auto-resizing the forms (deigned with sample layouts) as per screen resolution. But to auto-resize/auto-reposition the complex screens, one must look for professional tool like MS Access Form Resizer. The good news is that by using Horizontal/Vertical Anchoring Feature in combination with MS Access Form Resizer, you can accomplish great results.

10. *Never use Auto-Resize Property of Form along with Stretch-Down or Stretch-Across*

    When using Form Resizer, you should turn off Auto-Resize Property of Form if you are using inbuilt anchoring of MS Access. This case applies only when you try to use Stretch-Down or Stretch-Across. In such case, form may show unexpected results/errors.

11. *How to convert the design of an application to one resolution of my choice?*
    a. If you are intending to use your application on just one resolution (screen size) but unfortunately your application does not fit well on your screen,
    b. And if you are also not intending to change your application's view on the fly by MS Access Form Resizer to make application suitable for all kind of resolutions,
    c. And if you want your application's design to be changed to one specific screen-size and then want to save the changed side in design mode,

    Then you can simply open all your forms in design mode and run the MS Access Form Resizer to resize your forms to the intended resolution and then just save all your forms. You are good to go.

12. *Does MS Access Form Resizer keep information in Tags?*
    a. We store minimal information in Form's Tag. If you want to keep some form related information on the fly, please keep a separate (hidden) control in form's header/footer.
    b. We understand that in case of controls, you may need to store your own information. We also understand that keeping a hidden control in form's header/footer to store your control's related information is not easy to manage. Hence, we don't modify Tag-Property of your controls.
    c. For any momentary processing, if any information is kept in the controls' tag, that always get restored back to the original value. This restoration of value always takes place just before function exits its block of statements. So, you never experience any change in your controls' Tag-Property.

13. *Why don't my forms keep-resizing when resizer functions are called in Form's Resize Event.?*

    Most of the resizers available in market demonstrates their tool using videos or demo-apps and demonstrates how form/controls resize as user resizes the Form by dragging their border left/right. They use Form's Resize Event to demonstrate the effect. But this feature has many shortcomings or may show some ill effects too, if your application is a large application.

    After lot many implementations of Form-Resizer, we found that users do not want font/screen to keep on getting resized at each and every change in screen size. In general practice, users do not keep on increasing/decreasing the screen/application size, just to test how form resizer work. Users want a simple/robust facility to auto-fit their forms on different screen sizes. Users want to keep on using the application without any flickering or disturbances because of resizer. They usually keep screen at one size of their preference and then expect the screen to open again and again with same size/dimension. Whenever a change in font/size is needed, they want that change to happen on some button-click or keyboard shortcut.

MS Access Form Resizer is a most practical tool that can be used in applications of all sizes. We believe in delivering robust solutions (not fancy solution). You should give a button on your dashboard/main-menu or a shortcut-key to the users, using which, they can reset the form/controls' dimensions as per new screen area. This is just like hitting the Ctrl+ or Ctrl- keys on your keyboard while using Google-Chrome to increase/decrease font size.