MICROSOFT ACCESS 97
FOR
DEVELOPERS
ADVANCED MODULE 1
MACROS

NOTES

TABLE OF CONTENTS

NOTES

SECTION 1

CREATING MACROS

NOTES

WORKING WITH MACROS

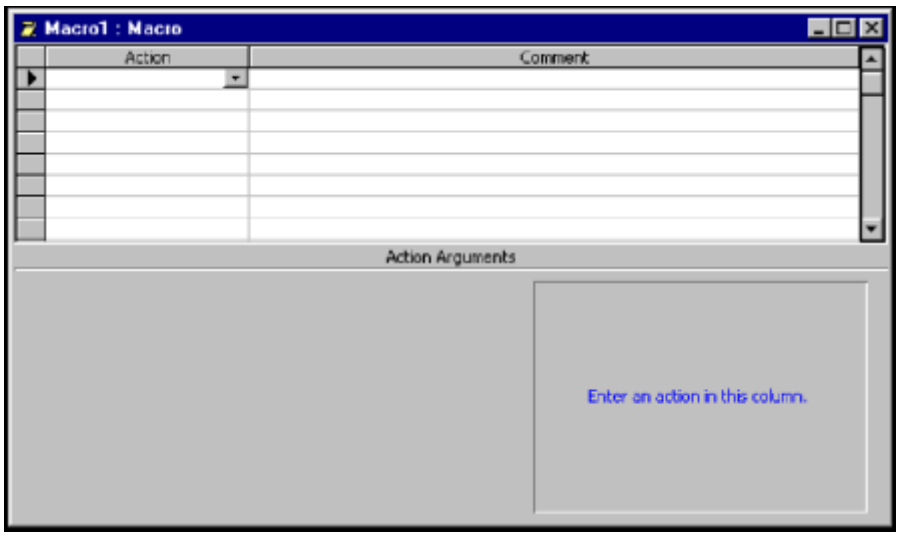
A macro performs a set of commands in sequence. While macros in word processors and spreadsheets are used mainly to duplicate keystrokes or mouse movements, macros in Access often automate an action or a series of actions. Such actions include opening tables, printing forms, finding records, or applying filters. Macros can even be used to add command buttons, create menus and toolbars, and build complete applications.

Macro commands in Access consist of an action and its arguments. The action is the task to be performed, such as opening a form. The arguments determine the specifics for the action, such as which form to open.

OPENING THE MACRO DESIGN WINDOW

The design grid in the **Action** pane of the Macro Design window contains at least two columns: **Action** and **Comment**. In the **Action** column, you enter the actions you want the macro to perform. You can either type the name of the action or select one from the list. In the **Comment** column, you describe the purpose of the action. Adding a comment may not seem important when a macro is first created, but it can prove helpful if you need to edit the macro at a later time.

At the bottom of the window is the **Action Arguments** pane. Most actions require one or more arguments. The arguments detail exactly what function the action performs. The display in this pane varies according to the action selected in the **Action** pane.



The Macro Design window

Notice that since no actions have been selected, the **Action Arguments** pane is blank.

CREATING A MACRO

When you select a field in the **Action** column, a list of actions appears. Most of the actions are self-explanatory and have equivalent menu commands. For instance, the **OpenQuery** action opens a query in **Datasheet** or **Design** view, depending on how the arguments are set.

Other actions can be performed only in macros or in more complex programming modules. The **AddMenu** action, for example, creates a custom menu to appear on a custom menu bar. This action is not available from the menu commands.

A macro can include up to 999 actions. You place each action in a separate row in the design grid in the order in which they are to be performed. For example, a **Maximize** command maximizes the window opened in the step immediately preceding it.

You use the **Comment** column to enter a description of the action to be performed. This field can contain up to 255 characters. While comments are optional, it is beneficial to enter a description of the action. This field is helpful if you want to modify the macro at a later time.

The following table provides a brief description of each action:

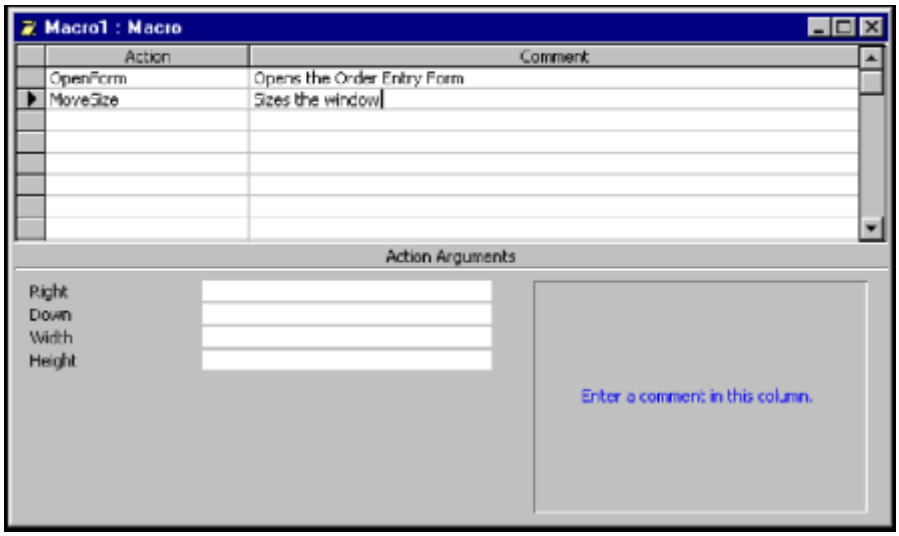
Action	**Description**
AddMenu	Adds a drop-down menu item to a custom menu bar.
ApplyFilter	Applies a filter or query to restrict or sort records.
Beep	Sounds a tone through the computer's speaker.
CancelEvent	Cancels the action that initiated the macro.
Close	Closes a specified window.
CopyObject	Copies the specified object to a different database or the same database under a new name.
DeleteObject	Deletes the specified object.
Echo	Specifies if the screen is to be updated while the macro is running.
FindNext	Locates the next record that meets criteria specified by a find.
FindRecord	Locates the first record following the current record that meets criteria specified in the arguments.
GoToControl	Activates the specified field or control.
GoToPage	Activates the first control on a specified page.
GoToRecord	Makes the specified record the current record.

Action	Description
Hourglass	Changes the mouse pointer to an hourglass while the macro is running.
Maximize	Enlarges the active window to fill the screen.
Minimize	Reduces the active window to a button on the taskbar.
MoveSize	Moves and/or resizes the active window.
MsgBox	Opens a message box containing a warning or informative text.
OpenForm	Opens a form in the selected view.
OpenModule	Opens a module at the selected procedure.
OpenQuery	Opens a query in the selected view.
OpenReport	Opens a report in the selected view.
OpenTable	Opens a table in the selected view.
OutputTo	Outputs data in the specified object to Excel text (.XLS), rich-text (.RTF), or MS-DOS text (.TXT).
PrintOut	Prints the specified datasheet, form, report, or module.
Quit	Quits Access.
Rename	Renames the specified database.
RepaintObject	Completes any pending screen updates for the active or specified object.
Requery	Updates the data in a specified control of an active object or updates itself if no control is specified.
Restore	Restores a maximized or minimized window to its previous size.
RunApp	Runs an application from within Access.
RunCode	Runs a Microsoft Access Basic Function procedure.
RunCommand	Performs a menu command.
RunMacro	Runs a macro.
RunSQL	Runs an action query using the corresponding SQL statement.
Save	Saves the specified object. If an object is not specified, the active object is saved.
SelectObject	Selects an object.
SendKeys	Sends keystrokes directly to Access or to an active Windows application.
SendObject	Includes a specified object in an electronic mail message to be viewed and forwarded.
SetMenuItem	Sets the appearance of a command that appears in a custom menu.
SetValue	Sets the value of a field, control, or property on a form or report.
SetWarnings	Turns system messages on or off.

Action	Description
ShowAllRecords	Removes any applied filter from the applicable active object and displays all records.
ShowToolbar	Displays or hides a toolbar.
StopAllMacros	Stops all currently running macros.
StopMacro	Stops the currently running macro.
TransferDatabase	Imports or exports data between an Access database and another database.
TransferSpreadsheet	Imports or exports data between the active database and a spreadsheet file.
TransferText	Imports or exports text between the active database and a text file.

1. Select the first empty field in the **Action** column.
2. Select the **Action** list.
3. Select the desired action, in this case **OpenForm.**
4. Select the corresponding field in the Comment column next to the action.
5. Type a comment for the action, for example, **Opens the Order Entry Form**
6. Add another action, for example, **MoveSize,** and a comment

Note:	You can use the **[Tab]** key and the **[Shift+Tab]** key combination to navigate between the **Action** and **Comment** columns.



ASSIGNING AN ARGUMENT TO AN ACTION

Most action arguments have a default list of available arguments. For example, the **View** argument box for the **OpenReport** action contains a list with the **Print**, **Design,** and **Print Preview** arguments. For action arguments without a list, you can type the argument into the argument box. You can enter up to 255 characters into the argument box.

Some arguments are required. For example, you must select the **Form Name** argument for the **OpenForm** action. Other arguments, such as **Filter Name**, are not required for the **OpenForm** action. If a required argument is missing, the macro stops when the action containing the missing argument is encountered.

In some cases, a default argument is used. For example, the **View** argument defaults to **Form** for the **OpenForm** action. Other arguments are ignored if they are not selected. For example, the **Filter Name** argument allows you to select a query to apply to the form as a filter. If you do not enter a query name, all the records appear.

Access displays a helpful message explaining the selected argument to the right of the argument boxes. If you press the **[F1]** key while the insertion point is in the lower pane of the Macro Design window, a help window for the argument opens with additional information.

The available macro actions and their associated arguments are listed in the following table. When **(req.)** appears next to an item in the **Arguments** column, either an argument or a selection from a list is required for the macro to run properly:

Action	**Arguments**
AddMenu	Menu Name (req.), Menu Macro Name (req.), Status Bar Text
ApplyFilter	Filter Name, Where Condition (one or both req.)
Beep	None
CancelEvent	None
Close	Object Type, Object Name, Save (req.)
CopyObject	Designation Database, New Name, Source Object Type, Source Object Name
DeleteObject	Object Type, Object Name
Echo	Echo On (req.), Status Bar Text
FindNext	None
FindRecord	Find What (req.), Match (req.), Match Case (req.), Search (req.), Search As Formatted (req.), Only Current Field (req.), Find First (req.)
GoToControl	Control Name (req.)
GoToPage	Page Number (req.), Right, Down (if a Right is specified, a Down must be specified and vice versa)
GoToRecord	Object Type, Object Name, Record (req.), Offset

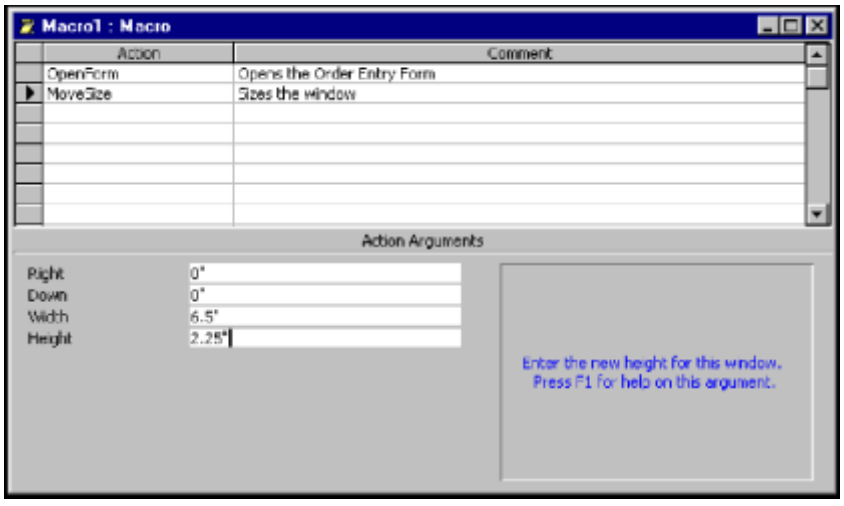
Action	Arguments
Hourglass	Hourglass On (req.)
Maximize	None
Minimize	None
MoveSize	Right, Down, Width, Height
MsgBox	Message, Beep (req.), Type (req.), Title
OpenForm	Form Name (req.), View (req.), Filter Name, Where Condition, Data Mode (req.), Window Mode (req.)
OpenModule	Module Name, Procedure Name
OpenQuery	Query Name (req.), View (req.), Data Mode (req.)
OpenReport	Report Name (req.), View (req.), Filter Name, Where Condition
OpenTable	Table Name (req.), View (req.), Data Mode (req.)
OutputTo	Object Type (req.), Object Name, Output Format, Output File, Auto Start, Template File
PrintOut	Print Range (req.), Page From (req.), Page To (req.), Print Quality (req.), Copies (req.), Collate Copies (req.)
Quit	Options (req.)
Rename	New Name (req.), Object Type, Old Name
RepaintObject	Object Type, Object Name
Requery	Control Name
Restore	None
RunApp	Command Line (req.)
RunCode	Function Name (req.)
RunCommand	Command (req.)
RunMacro	Macro Name (req.), Repeat Count, Repeat Expression
RunSQL	SQL Statement (req.), Use Transaction (req.)
Save	Object Type, Object Name
SelectObject	Object Type (req.), Object Name (req.), In Database Window (req.)
SendKeys	Keystrokes (req.), Wait (req.)
SendObject	Object Type, Object Name, Output Format (req.), To (req.), Cc, Bcc, Subject, Message Text, Edit Message (req.), Template File
SetMenuItem	Menu Index (req.), Command Index (req.), Subcommand Index, Flag (req.)

Action	Arguments
SetValue	Item (req.), Expression (req.)
SetWarnings	Warnings On (req.)
ShowAllRecords	None
ShowToolbar	Toolbar Name (req.), Show (req.)
StopAllMacros	None
StopMacro	None
TransferDatabase	Transfer Type (req.), Database Type (req.), Database Name (req.), Object Type (req.), Source (req.), Destination (req.), Structure Only (req.)
TransferSpreadsheet	Transfer Type (req.), Spreadsheet Type (req.), Table Name (req.), File Name (req.), Has Field Names (req.), Range
TransferText	Transfer Type (req.), Specification Name (req.), Table Name (req.), File Name (req.), Has Field Names (req.), HTML Table Name

You can press the **[F6]** key to toggle between the **Action** pane and the **Action Arguments** pane. You can use the **[Tab]** key and the **[Shift+Tab]** key combination to navigate through the argument fields.

To add an argument to an action:-

1. Select an action, for example, **MoveSiz**e
2. Select the applicable argument in the **Action Arguments** pane, in this case Width is set to 6.5", and Height to 2.25".
3. Type the argument or select the list.
4. Select an argument from the list, if applicable.



Assigning an argument to an action

SAVING A MACRO

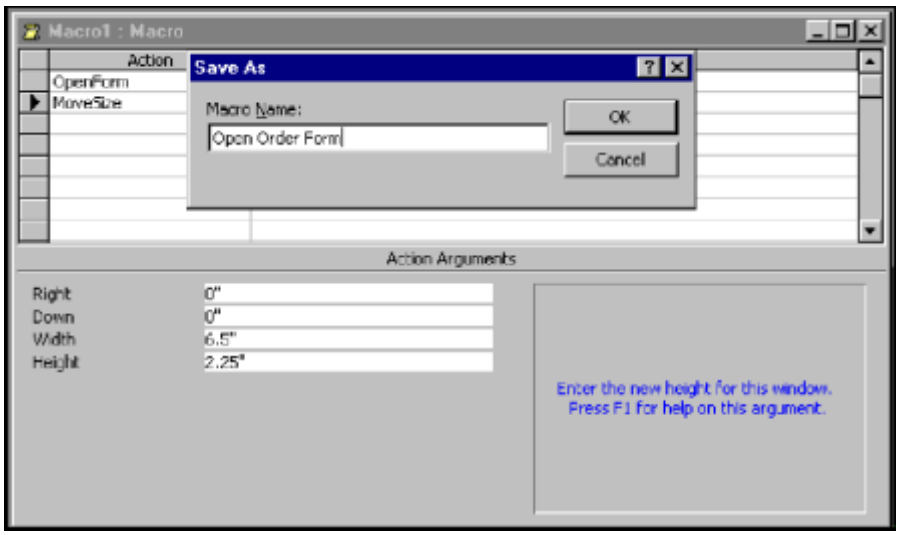
After creating a macro, you must save it before you can test or run it. When you attempt to save a macro or close the Macro window without saving, a Save As dialog box opens with the default name of **Macro#** (numbered consecutively). It is best to use a short name that clearly indicates the function of the macro. The name **Macro1**, for example, does not mean much to other users who may be working with the macro.

The name that you give the macro appears on the **Macros** page in the Database window. All the macros associated with the database being used appear in the Database window.

1. Click the **Save** button  .

   or

2. Select the **Save** command from the **File** menu.
3. Type a name for the macro
4. Select **OK**.



RENAMING A MACRO

You can rename a macro. However, if you rename a macro, it may no longer run, depending on how you initiate the macro. Copying a macro, renaming it, and then editing it is a quick way to create macros that perform similar tasks.

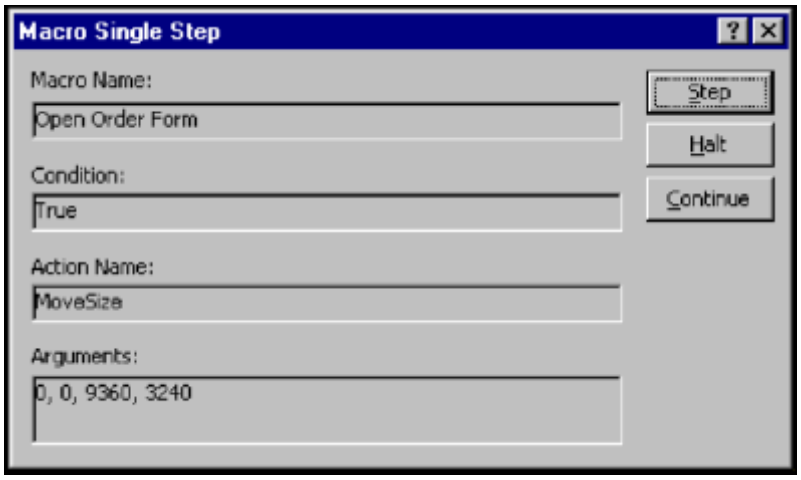
USING SINGLE STEP MODE FOR TESTING

Access runs macros so quickly that you may not be able to see each action as it is performed. If you wish to confirm that the actions are being performed correctly, you can step through the macro. Single step mode is particularly useful when a macro contains numerous actions or arguments, or when a macro does not seem to perform as planned.

When you step through a macro, Access pauses before each action, allowing you to view each step. The Macro Single Step dialog box opens, displaying the macro name, condition, action name, and selected arguments. The Macro Single Step dialog box has three buttons: **Step**, **Halt**, and **Continue**. You select the **Step** button to perform the action in the dialog box. If there are no errors, the next action appears in the dialog box. If you want to stop running the macro in this mode, you select the **Halt** button. The macro stops running and the dialog box closes. When you want to stop stepping through the macro and run the remaining steps, you select the **Continue** button. The remaining macro actions are then completed.

If an error does occur, you can make the changes in the Macro Design window. After you make the changes to a macro, you should retest it.

1. Open the desired macro in the Macro Design window.
2. Click the **Single Step** button  .
3. Click the **Run** button  .
4. Select **Step** as necessary to complete all the macro actions.
5. Select **Halt** to stop running the macro and close the dialog box.
6. Select **Continue** to run the remaining steps. You can run a macro without testing it by selecting the **Run** command from the **Run** menu or clicking the **Run** button on the **Macro Design** toolbar.



Note:	You should only use single step mode when working in the Macro Design window. When you enable single step mode, it remains activated throughout the current Access session. All macros will run in single step mode until you disable it.

RUNNING A MACRO

You can run a macro from the Database window by selecting the **Macros** tab and double-

clicking the desired macro name or by selecting the desired macro name and selecting the **Run** command in the Database window.

Note:	A warning box opens if required arguments are missing, or if the macro cannot run for any reason.

1. Select the **Macros** tab.
2. Select the macro you want to run.
3. Select **Run**.

EDITING AN EXISTING MACRO

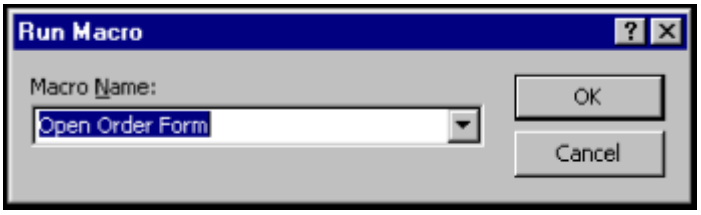
After you have created a macro, you may decide to add or delete existing actions or change actions or action arguments. In the Macro Design window, you can cut, copy, and paste to edit a macro. You can also insert and delete rows. After you make the desired changes, you must save the macro to save the changes. It is a good idea to test the edited macro using single step mode. You must remember to disable single step mode after testing a macro.

1. Select the **Macros** tab.
2. Select the macro you want to edit.
3. Select **Design**.
4. Edit the macro as desired.

RUNNING A MACRO FROM THE TOOLS MENU

You can run a macro using the **Tools** menu. This option allows you to execute a macro from almost anywhere. When the Run Macro dialog box opens, you can either type the name of the macro in the text box or select the macro name from a list.

1. Select the **Macros** tab.
2. Select the **Tools** menu.
3. Point to the **Macro** command.
4. Select Run Macro.
5. Select the **Macro Name** list.



6. Select the macro you want to run.
7. Select **OK**.

EXERCISE ONE

Objectives Practice creating a macro

Preparation Open ***Train11*.**

Instructions

1. Create a new macro.
2. Enter the following actions and comments.

Action	Comment
OpenForm	Opens the Employee Form
MoveSize	Sizes the Employee Form and positions it at the top of the window
OpenForm	Opens the Post Form in Datasheet view
MoveSize	Sizes the Post Form and positions it below the Employee Form

3. Enter the following arguments for each action.

Action	Arguments
OpenForm	Form Name: **Employee Form** Data Mode: **Read Only**
MoveSize	Right: **0** Down: **0** Width: **6** Height: **3**
OpenForm	Form Name: **Post Form** View: **Datasheet** Data Mode: **Read Only**
MoveSize	Right: **0** Down: **3** Width: **6** Height: **2**

4. Save the macro as Open Employee and Post Forms.
5. Enable single step mode and run the macro to test it.
6. Close the Employee and Post forms windows.
7. Edit the macro. In the second step, **MoveSize**, change the **Height** argument to **2**.
8. In the fourth step, **MoveSize**, change the **Down** argument to **2**.
9. Save the macro and disable single step mode.
10. Close the Macro Design window.
11. Run the macro from the Database window.
12. Close the Employee and Post forms windows.
13. Run the macro using the **Tools** menu.
14. Close the Employee and Post forms Window.

EXERCISE TWO

Objectives Practice writing and running macros

Preparation Open Database ***Train11***

1. Create a macro to firstly open the form Employees and then using the action RunCommand , find the appropriate argument to go to a new record. Save these two actions as a macro named "Enter new ". Test the macro.
2. Create a macro that opens the table **Employee** and then opens the **Post table**. Save the macro as "**Opens two tables**". Test the macro.
3. Amend the "**Opens two tables"** macro to Minimize each table after opening them. Test the macro.
4. Create a macro that opens the **Employe**e Table, and then uses the action **FindRecord** to find a record where the arguments of :- Find What ="Little" and Only Current Field is set to No. Save this as a macro named "**Search for Mr Little**". Run the macro. What number record does it return?
5. Create a macro to open the report named **Post list** in Print Preview mode and using the action **OutputTo**, output this report to a rich text format file on the C: , named mine.doc. Save this macro as **"Export file**". Run the macro.

SECTION TWO

USING MACROS

NOTES

USING MACROS

USING PROPERTIES

Properties allow you to specify the appearance and behaviour of objects in a database. Objects include tables, queries, forms, and reports, as well as controls within reports or forms.

Property sheets display the properties of a selected object. They have several tabs including **Format**, **Data**, **Event**, and **Other** that list the properties by group. The groups are the same for every property, but the items in the group change depending on the type of object selected. The **All** Tab displays all the properties in a single list.

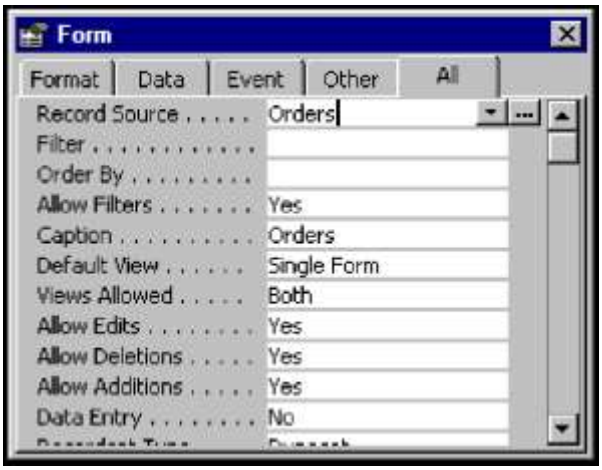
The **Format** properties allow you to control the appearance of an object, such as colour, font, size, and borders. These properties change automatically when you make changes to an object.

The **Data** properties allow you to specify the source of the data and control items, such as default values.

The **Event** properties allow you to control when an action occurs.

The **Other** properties contain items that do not fit into the other three categories, such as the name of a control when used in a macro or text that appears in the status bar.

Below is an example of the Properties for a Form, based on **Order** records.



A property sheet

ASSIGNING A MACRO TO A CONTROL

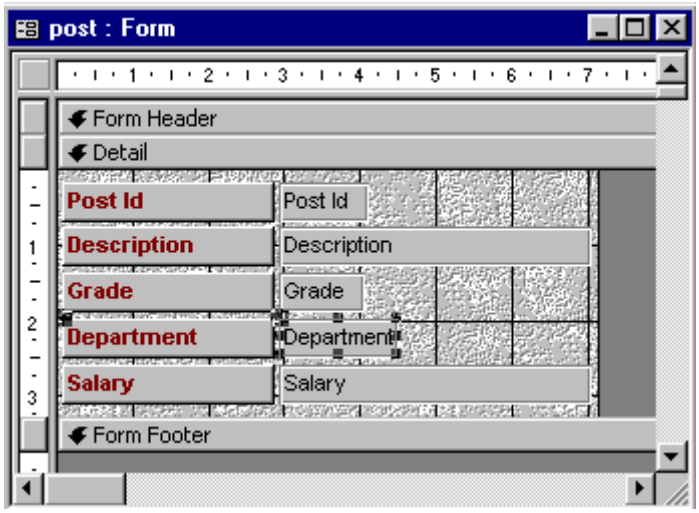
You can associate a macro with a control on a form or report using the **Event** properties of the control. An event is an action, such as a mouse click or a change in value that can initiate a response. The macro runs automatically when the specified event involving that control occurs.

Many events involve the control having the **focus. Focus** means that the control can receive data from mouse clicks or keyboard actions. For example, field boxes, toggle buttons, and radio buttons can have the focus, since they can respond to data entry from the keyboard or mouse. Only one control can have the focus at a time. The **Event** properties of a text box control are listed in the following table:

Event Property	Action is initiated:
Before Update	before the data in a control is updated
After Update	after the data in a control is updated
On Change	when the data in a control is updated
On Enter	when the control first receives the focus from another control on the same form
On Exit	when the control loses the focus to another control on the same form
On Got Focus	when the control gets the focus from another form or control
On Lost Focus	when the control loses the focus to another control or form
On Click	by clicking and releasing the primary mouse button on the control
On Dbl Click	by clicking and releasing the primary mouse button twice on the control
On Mouse Down	by clicking the mouse button while the mouse pointer is on a control
On Mouse Move	by moving the mouse pointer over a control
On Mouse Up	by releasing the mouse button while the mouse pointer is over the control
On Key Down	by pressing any key on the keyboard when a control has the focus or is using a SendKeys macro
On Key Up	by releasing a key or immediately after running the SendKeys macro
On Key Press	by pressing and releasing any key on the keyboard when on a control that has the focus or when using a SendKeys macro

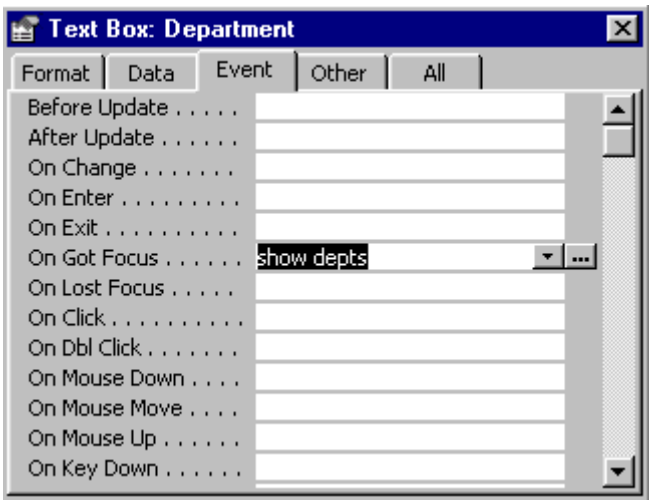
How To Assign A Macro To A Control:-

1. Open the Post form in **Design** view.



2. Select the appropriate control, in this case Department

3. Click the **Properties** button  .

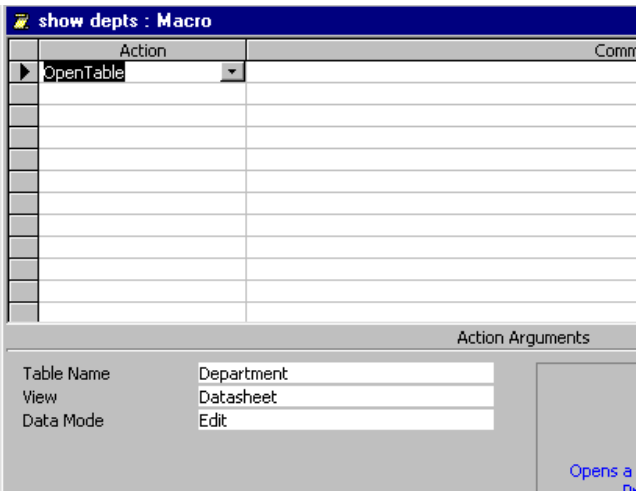


4. Select the **Event** tab.

5. Select the desired property, in this case **On Got Focus**

6. Select the desired macro, in this case **show depts.** This macro simply opens a table showing departments names and their Id's.

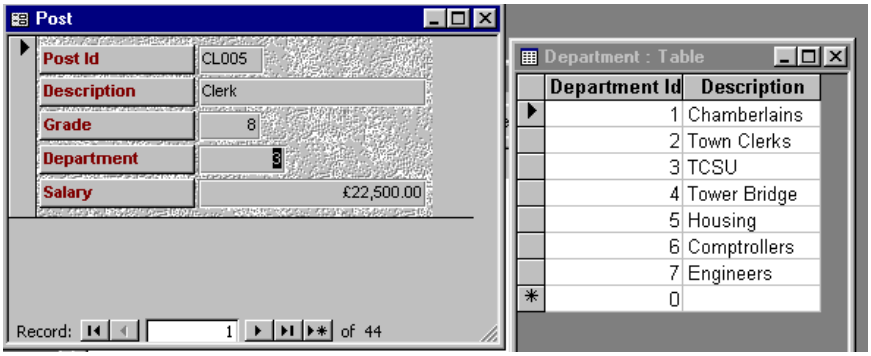
7. The macro is shown below.



8. Click the **Close** button to close the property sheet.

9. Close the form, saving changes.

10. From the database window, select the form Post and tab or press enter to move around the fields. When you reach the field Department, the macro will execute and the table Department will be displayed. This is shown below.

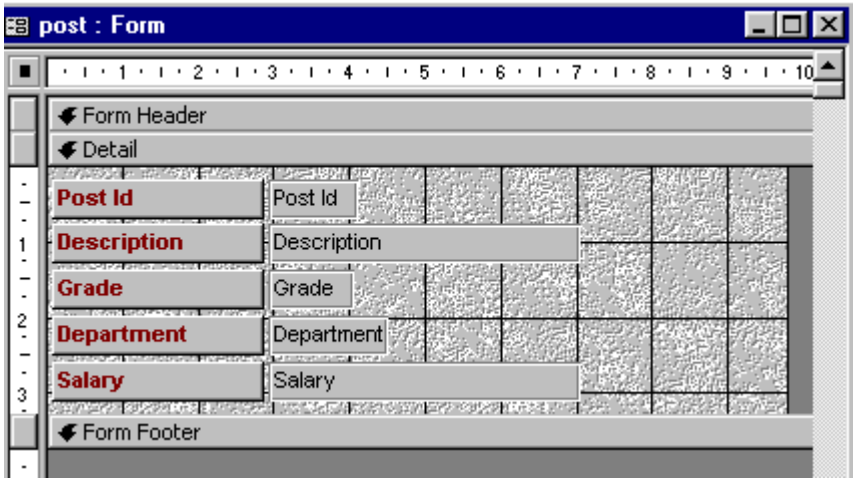


CREATING A COMMAND BUTTON AND ASSIGNING A MACRO

Command buttons are controls on Forms or Reports to which you can assign actions. They are often used to run macros in a form or report. You can place descriptive text or a picture on a command button so that the user can easily identify the purpose of the button.

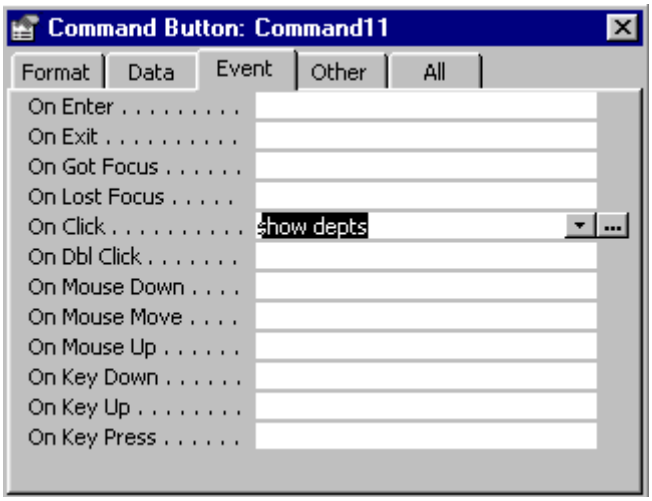
There are two ways to use a Command Button. The first technique involves creating the button and then assigning the macro to the button. This can be a macro you have written. The second technique involves the use of the Control Button Wizard, where you have to select from a set of predefined actions. This is described under 'Using the Command Button Wizard'.

1. You create a command button in Design view using the Command Button tool in the toolbox. After you create the command button on the form, you set the properties, such as the event property to initiate the action and the text or picture to appear on the button.
2. To create a command button using this method, you must disable the Control Wizards button on the toolbar.
3. Open the desired form in **Design** view.



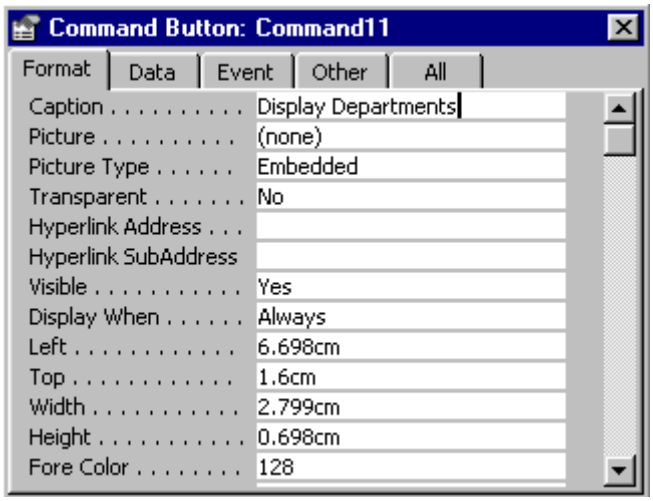
4. Display the toolbox.
5. Click the **Control Wizards** button  to disable it, as necessary.
6. Click the **Command Button** tool  in the toolbox.
7. Click in the desired location in the form or report where you want to create the command button. Draw out the area for the button.
8. Click the **Properties** button  .
9. Select the **Event** tab.

10. Select the desired property, **On Click**.



11. Select the desired macro, **show depts**

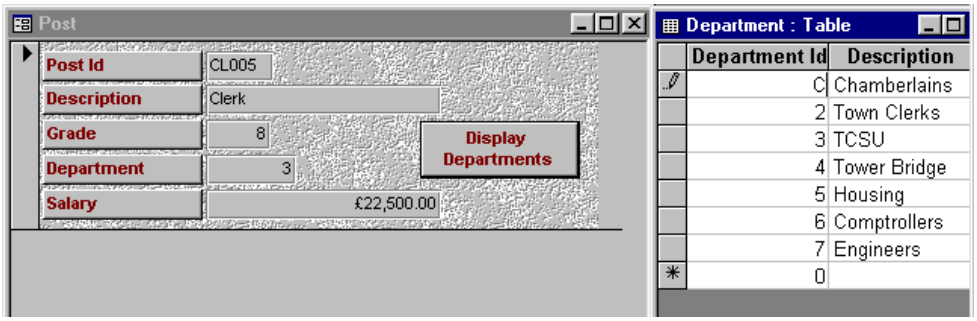
12. Select the **Format** tab.



13. Select the text in the **Caption** property box to add text to the button, or use the **Picture** property box, to select from a list of pictures.

14. Click the **Close** button to close the property sheet.

15. Close the Form, saving changes. In the database window, open the form and click the button, **Display Departments.**



USING THE COMMAND BUTTON WIZARD

You can use the Command Button Wizard to create command buttons. The wizard assists you in assigning actions to a button. You should review these actions before you create macros because many of them have already been defined.

The Command Button Wizard guides you through the process of creating a command button. The first dialog box provides a list of categories, many of which appear on the **Standard** toolbar. You can create a custom application in which the Access menus and toolbars are hidden and command buttons are added for the actions you want the user to be able to perform.

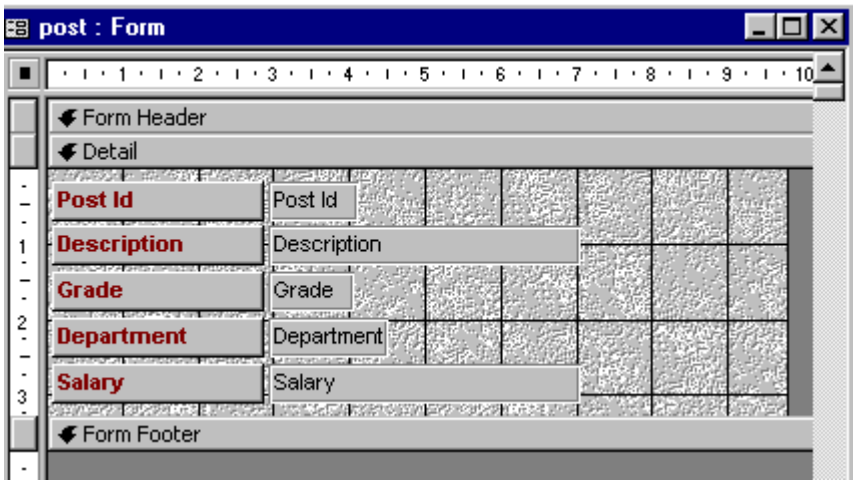
The categories available when you use the Command Button Wizard and the actions associated with each are listed in the following table:

Categories	**Actions**
Record Navigation	Find Next, Find Record, Go to First Record, Go to Last Record, Go to Next Record, Go to Previous Record
Record Operations	Add New Record, Delete Record, Duplicate Record, Print Record, Save Record, Undo Record
Form Operations	Apply Form Filter, Close Form, Edit Form Filter, Open Form, Print a Form, Print Current Form, Refresh Form Data
Report Operations	Mail Report, Preview Report, Print Report, Send Report to File
Application	Quit Application, Run Application, Run MS Excel, Run MS Word, Run Notepad
Miscellaneous	AutoDial, Print Table, Run Macro, Run Query

Once a category and an action have been selected, you can assign text or a picture to the command button. You should then assign a descriptive name to the button to identify it.

Note: The **Control Wizards** button in the toolbox must be enabled in order to use the Command Button Wizard.

1. Open the desired form in **Design** view.



2. Display the toolbox.

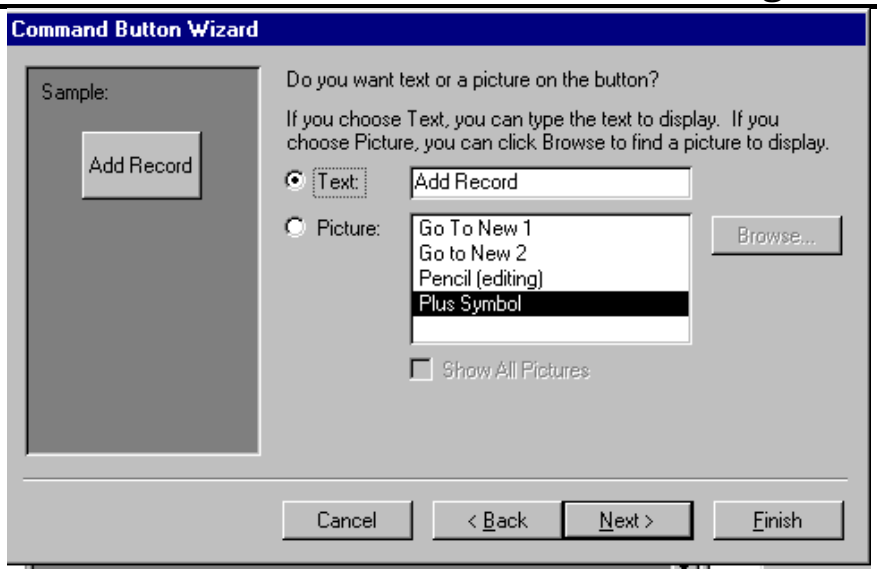
3. Click the **Control Wizards** button  to enable it, as necessary.

4. Click the **Command Button** tool  in the toolbox. Draw the button in the required position. The Command Button Wizard dialogue box appears.

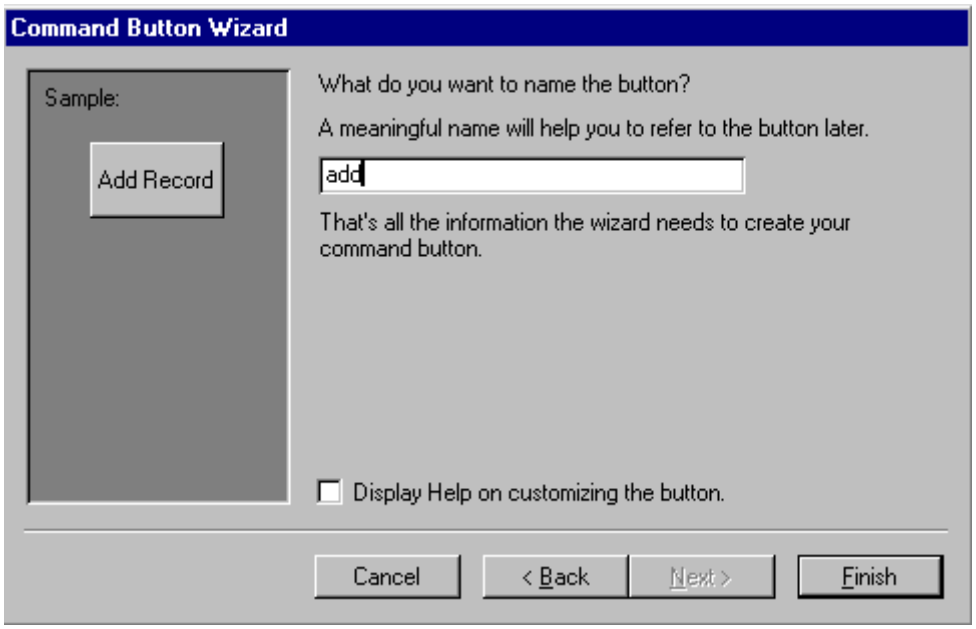
5. Select a category from the **Categories** list box, in this case Record Operations.

6. Select an action from the **Actions** list box, in this case Add New Record.

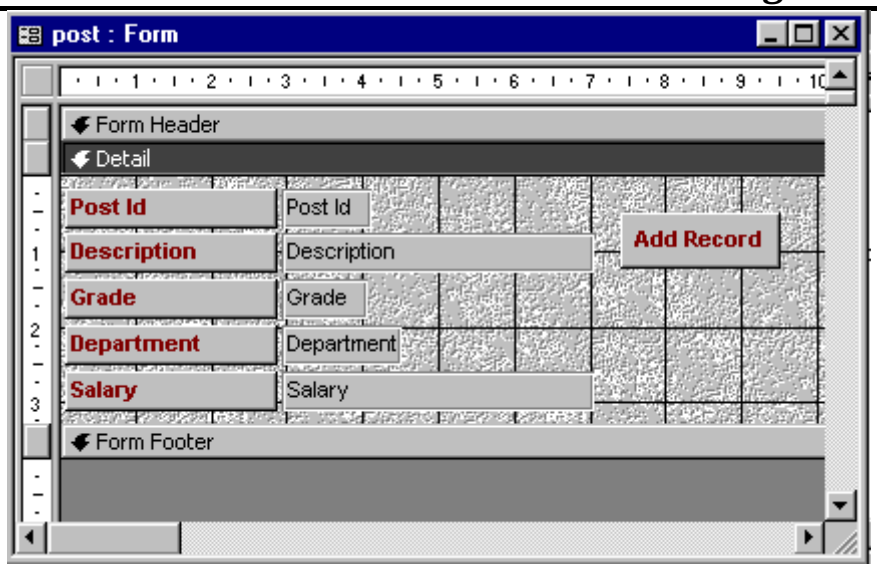
7. Select **Next**.



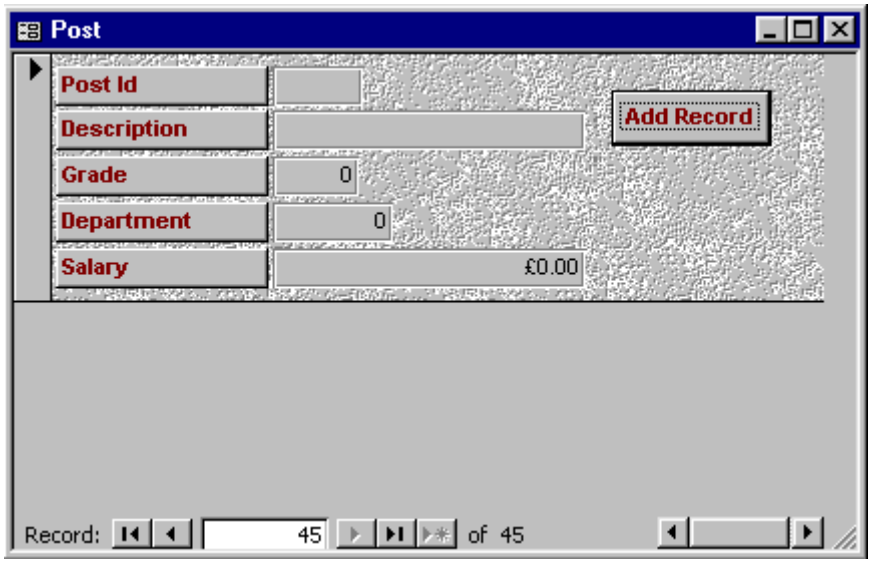
8. Select the **Text** or **Picture** option.
9. Select the text in the text box, as necessary or type new text, or select a picture.
10. Select **Next**.



11. Type a descriptive name to identify the button in the text box.
12. Select Finish. The form is displayed in Form Design View.



13. Close the Form and save changes. In the database Window select the Form and Open. Click the button.



ADDING A CONDITION TO A MACRO

You can add a condition argument to a macro. An argument acts very much like a filter. Just as a filter displays only the records that meet the condition, the argument only performs the macro when the condition is met.

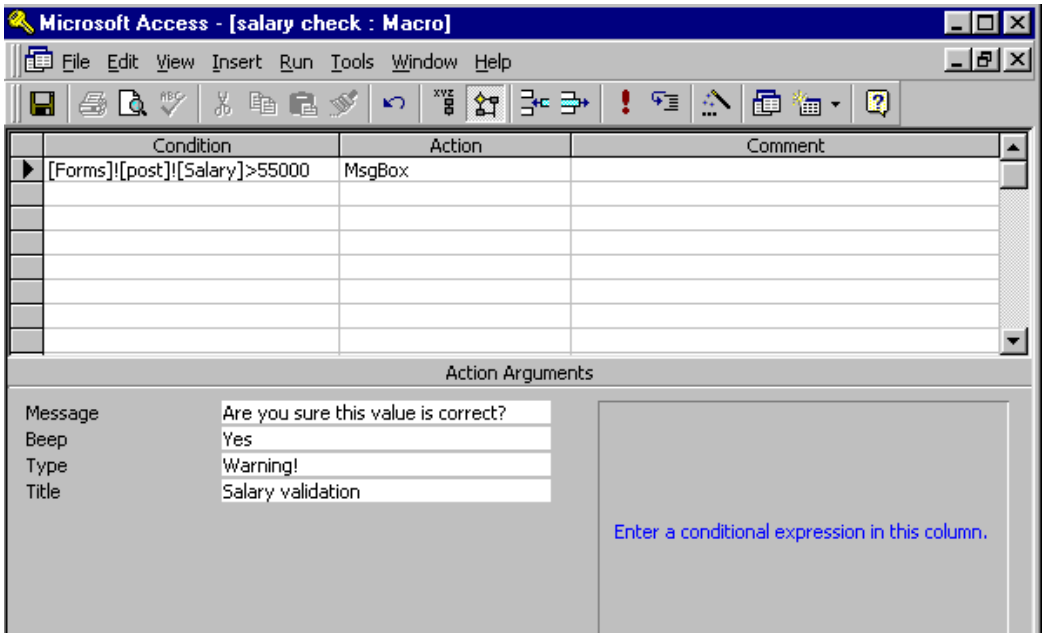
Before you can add a condition to a macro, you must display the **Condition** column in the Macro Design window. The **Condition** column appears to the left of the **Action** column in the upper pane of the Macro Design window. You can type a condition or use the Expression Builder dialog box to create the expression.

If you type an expression, there are certain rules you must follow when referring to controls in tables, queries, forms, and reports. All references must be separated with an exclamation point. Additional rules for the database objects and controls are listed in the following table:

Controls	Rules
Controls in tables	Enclose the name of the table and the name of the control in square brackets and separate them with an exclamation point. For example, [Orders]![Customer ID] refers to the Customer ID field in the Orders table.
Controls in queries	Enclose the name of the query and the name of the control in square brackets and separate them with an exclamation point. For example, [Order Items]![Item Number] refers to the Item Number field in the Order Items query.
Controls in forms	Enclose the name of the form and the name of the control in square brackets and separate them with an exclamation point. Indicate that you are referring to a form by beginning the statement with the word Forms. For example, Forms![Customers]![Customer ID] refers to the Customer ID field in the Customers form.
Controls in reports	Enclose the name of the report and the name of the control in square brackets and separate them with an exclamation point. Indicate that you are referring to a report by beginning the statement with the word Reports. For example, Reports![Customer Sales]![Contact Name] refers to the Contact Name field in the Customer Sales report.

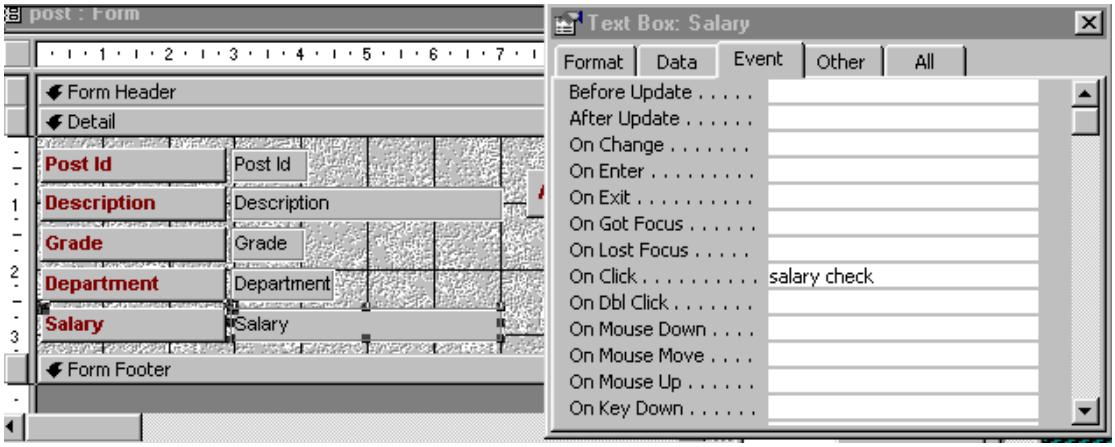
Since you enter the condition in the **Condition** column in the Macro Design window, you do not need to include the word **IF** in the statement. Access assumes the statement is a condition. If the condition is true, Access performs the corresponding action in that row. If the condition is false, Access does not perform the action.

1. Open the desired macro in the Macro Design window, or create a new macro. In this case we are to create a macro that checks the value entered into the field Salary. If the value is greater than 55000, the macro will activate.
2. Once you display the **Condition** column, you can open the Expression Builder dialog box by clicking the right mouse button in a field in the **Condition** column, by clicking in the **Action Arguments** pane and selecting the **Build** command from the shortcut menu, or by clicking the **Build** button on the **Macro Design** toolbar. 
3. Alternatively type the condition in the field.
4. Choose an appropriate action and arguments.
5. Save the macro and close the Window.

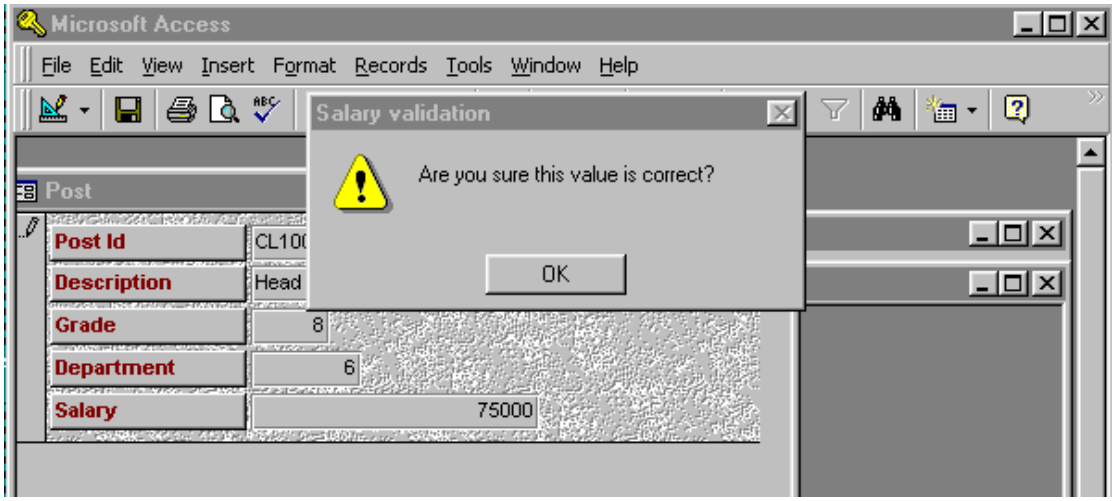


TO ASSIGN THE MACRO,

1. Open the form in Design View, select the control, such as Salary and using the property field, assign the macro.



2. Using the Form and typing in a value such as 75000 will cause the macro to execute, and the warning messgae to be displayed.



Note:	If the condition is true, you can have Access perform more than one action by entering an ellipsis (...) in the **Condition** column in the Macro Design window. Access performs the action on the same row as the condition and all rows thereafter with an ellipsis.

CREATING A GROUP MACRO

You can place multiple macros on the same macro sheet. This option, called grouping macros, has several advantages. The most obvious advantage is that it reduces the number of macros that appear in the window. More importantly, it allows you to organise your macros by grouping related macros together onto a single macro sheet. For example, if you created several macros that are related to one form, you can group all of them together on one macro sheet.

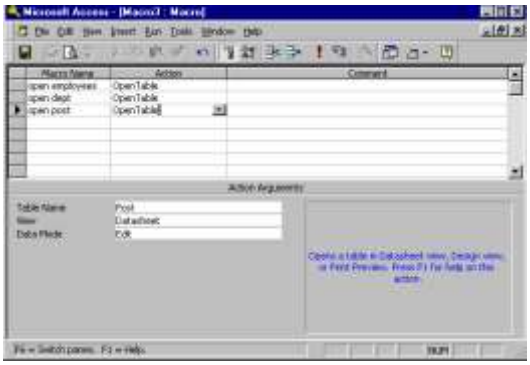
Before you can create a group macro, you need to display the **Macro Name** column in the Macro Design window. The **Macro Name** column appears to the left of the **Action** or **Condition** column, if it is displayed. You can type the macro name directly into the appropriate field in the **Macro Name** column.

Like any other macro, a group macro must be saved with a name. You should make sure that the group macro name is descriptive enough to indicate its contents. This helps other users know what macros are contained in the group macro. For example, if all the macros relate to the **Orders** form, you could name the group macro **Order Form Macros**.

When macros are grouped, you can still refer to individual macros. To refer to an individual macro in a group, you can use its group and macro name in the format **Groupname.Macroname**.

Note:	If you have already created macros, you can cut and paste them into a new group macros sheet or edit an existing macro sheet.

1. Open the Macro Design window.
2. Click the **Macro Names** button  .
3. Select the first empty field in the **Macro Name** column.
4. Type a name for the macro.
5. Select the corresponding field in the **Action** column next to the macro name.
6. Select the list.
7. Select the desired action.
8. Select an argument for the action.
9. Select the argument list.
10. Select the desired argument.



CREATING AN AUTOEXEC MACRO

While you can select a command to run a macro, you can also have a macro that runs automatically when a database is opened. This type of macro is called an Autoexec macro because it is automatically executed when the database is opened. This feature was available for Access 2.0. There is also a new feature available in Access 97 via the Tools Startup menu.

You can use the Startup dialog box instead of or in addition to an AutoExec macro. An AutoExec macro runs after the Startup options have taken effect; therefore, you should avoid any actions in an AutoExec macro that change the effect of the Startup option settings. For example, if you specify a form in the Display Form box in the Startup dialog box, and you also use the OpenForm action in an AutoExec macro, Microsoft Access first displays the form specified in the Startup dialog box, then immediately displays the form specified in the OpenForm action.

You can create an Autoexec macro to automate or simplify the database for other users. For example, assume that you want someone in your office to add customer orders on a specific form. To ensure that they use the correct form, you could create an Autoexec macro that opens the form for them.

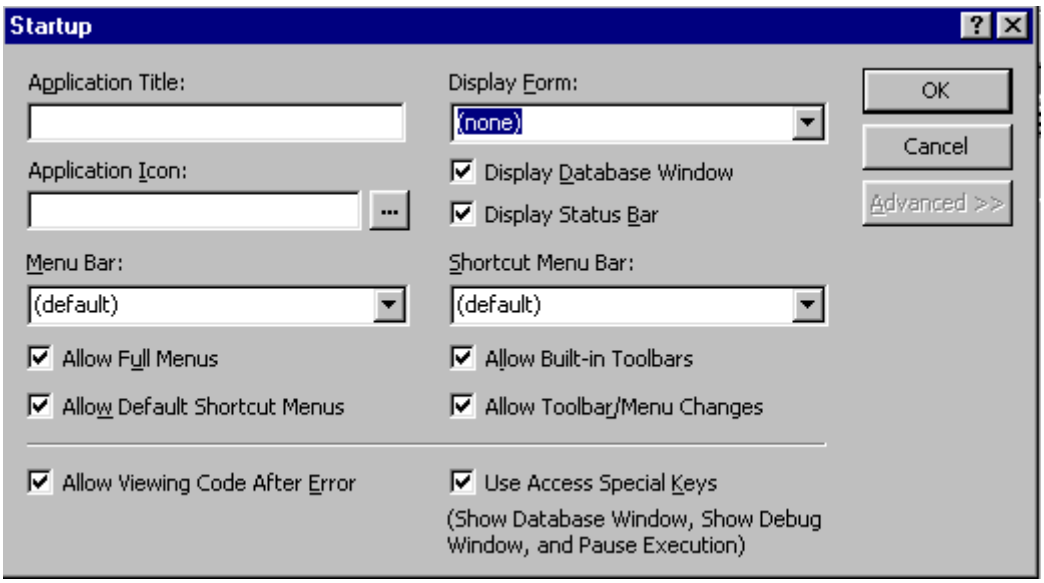
You can create an Autoexec macro in two ways. You can rename an existing macro Autoexec, or you can create a new macro and save it with the name Autoexec. Since the Autoexec macro is stored with the database, each database can have its own Autoexec macro. However, you can only have one Autoexec macro per database.

Note:	You can bypass the Autoexec macro by holding the [Shift] key while you open a database.

1. Select the **Macros** tab.
2. Click the right mouse button on the macro you want to rename.
3. Select the **Rename** command.
4. Type Autoexec.
5. Press **[Enter]**.

TOOLS.. STARTUP

This can be used in addition or as an alternative to the Autoexec macro.



Startup properties affect how your database application appears when it's opened. For example, Startup properties enable you to customise the application's title bar, menus, toolbars, and Startup form. This is a more powerful feature than the Autoexec macro.

EXERCISE THREE

Objectives Practice using a macro

Prep Open ***Train12*.**

Instructions

1. Rename the **open tables** macro to **Autoexec**. Make sure that you press **[Enter]** after you rename it. Close the database and reopen it to test the macro. Notice that the two form windows open. Close the form windows.
2. The **date check** macro opens a message box when a value is entered into the **Start date** field of an employee. Attach the date check macro to the **Start Date** text box control on the **Employees** form. Use the **On Exit** event. Close and save the form.
3. Open the **Employees** form. For the first employee, change the **Start date** value. The message box will appear.
4. Add a condition to the **date check** macro so that the macro only runs when the start date is greater than today's date. Use the **Date** function.
5. Close the Macro Design window and save the form.
6. Open the **Employees** form. For the first employee, change the **Start date** value to a date in the future. Notice that a message box opens. Close the message box. Close the form window
7. Create a command button on the **Senior staff** form, without using the wizard. Place it in the lower left corner of the form. Attach the **senior staff** macro to the command button so that it runs when the mouse button is clicked. Enter the text **Create new senior staff table** using the **Caption** property box. Resize the button to display all the text, if necessary.
8. Close and save the **Senior staff** form **.** Open the **Senior staff** form and test the **Create new senior staff table** button. This macro runs a make table query to create a new table called seniors with just the higher graded staff. Reply yes to any prompts. Close the form. Check the existence of the new table.
9. Create a command button on the **Senior staff** form using the Command Button Wizard. Place it in the lower right corner of the form. Select the Report Operations in the **Categories** list box and Preview Report in the **Actions** list box. Default the name of the report to **Senior names**. Use the **Magnifying glass** picture for the button. Name the button, **Preview senior names**. Close and save the form.
10. Open the **Senior staff** form and test the new button. Close the report and the form.

EXERCISE FOUR

Objectives Create a group macro

Preparation Display the Macro Name column

Instructions

1. Enter the following names, actions, and arguments in the Macro Design window:

Macro Name	**Action**	**Argument**
Open the form	OpenForm	Form Name**: names and job titles**
Print the Report	OpenReport	Report Name: **names and job titles** View: **Print Preview**

2. Save the macro as **Employee Macros**.
3. Close the Macro Design window.
4. Open the Form **Employees Information System** in design view. Attach the two above macros to the appropriate buttons. Save and close the form. Open the form and test the buttons.

NOTES

NOTES