Examples of manipulating text values in forms, reports, and data access pages

The following table lists examples of expressions that you can use in calculated controls on forms, reports, and data access pages.

Expression	Description
="N/A"	Displays N/A.
=[FirstName] & " " & [LastName]	Displays the values of the FirstName and LastName fields separated by a space.
=Left([ProductName], 1)	Uses the Left function to display the first character of the value of the ProductName field.
=Right([AssetCode], 2)	Uses the **Right** function to display the last 2 characters of the value of the AssetCode field.
=Trim([Address])	Uses the **Trim** function to display the value of the Address field, removing any leading or trailing spaces.
=IIf(IsNull([Region]), [City] & " " & [PostalCode], [City] & " " & [Region] & " " & [PostalCode])	Uses the **IIf** function to display the values of the City and PostalCode fields if Region is **Null**; otherwise, it displays the values of the City, Region, and PostalCode fields, separated by spaces.

Notes

- In the **ControlSource** property of a calculated control, precede the expression with the **=** operator. On a data access page, you can omit the = operator, and type an alias instead; for example, type **FullName: [FirstName] & " " & [LastName]**.
- When you set the **Name** property of a calculated control in a form or report, or set the **ID** property of a calculated control in a data access page, make sure to use a unique name. Don't use the name or ID of one of the controls that you used in the expression.
- In an expression on a form or report, you can use the name of a control or field in the underlying record source. In an expression on a data access page, you can use only the name of a field that's in the data definition of that page.

Examples of expressions for page numbers

The following table lists examples of page number expressions you can use in form or report Design view and the results that you see in other views.

Expression	Result
=[Page]	1, 2, 3
="Page " & [Page]	Page 1, Page 2, Page 3

="Page " & [Page] & " of " & [Pages]	Page 1 of 3, Page 2 of 3, Page 3 of 3
=[Page] & " of " & [Pages] & " Pages"	1 of 3 Pages, 2 of 3 Pages, 3 of 3 Pages
=[Page] & "/"& [Pages] & " Pages"	1/3 Pages, 2/3 Pages, 3/3 Pages
=[Country] & " - " & [Page]	UK - 1, UK - 2, UK - 3
=Format([Page], "000")	001, 002, 003

Examples of performing arithmetic operations in forms, reports, and data access pages

The following table lists examples of expressions that you can use in calculated controls on forms, reports, and data access pages.

If you use this expression	Microsoft Access displays
=[Subtotal]+[Freight]	The sum of the values of the Subtotal and Freight fields.
=[RequiredDate]-[ShippedDate]	The difference between the values of the RequiredDate and ShippedDate fields.
=[Price]*1.06	The product of the value of the Price field and 1.06 (adds 6 percent to the Price value).
=[Quantity]*[Price]	The product of the values of the Quantity and Price fields.
=[EmployeeTotal]/[CountryTotal]	The quotient of the values of the EmployeeTotal and CountryTotal fields.

Notes

- In the **ControlSource** property of a calculated control, precede the expression with the **=** operator. On a data access page, you can omit the = operator, and type an alias instead; for example, type **ExtendedPrice: [Quantity]*[Price]**.
- When you set the **Name** property of a calculated control in a form or report, or set the **ID** property of a calculated control on a data access page, make sure to use a unique name. Don't use the name or ID of one of the controls that you used in the expression.
- In an expression on a form or report, you can use the name of a control or the name of a field in the underlying record source. In an expression on a data access page, you can use only the name of a field that's in the data definition of the page.
- When you use an arithmetic operator (**+**, **-**, *****, **/**) in an expression, and the value of one of the controls in the expression is **Null**, the result of the entire expression will be **Null**. On a form or report, if some records in one of the controls that you used in the expression might have a **Null** value, you can convert the **Null** value to zero by using the **Nz** function; for example:

=Nz([Subtotal])+Nz([Freight])

For more information on the **Nz** function, click .

Examples of referring to values on forms and reports

The following table lists examples of expressions you can use in calculated controls on forms.

If you use this expression	Microsoft Access displays
=Forms![Orders]![OrderID]	The value of the OrderID control on the Orders form.
=Forms![Orders]![Orders Subform]![OrderSubtotal]	The value of the OrderSubtotal control on the Orders Subform on the Orders form.
=Forms![Orders]![Orders Subform]![ProductID].Column(2)	The value of the third column in ProductID, a multiple-column list box on the Orders Subform on the Orders form. (0 refers to the first column, 1 refers to the second, and so on.)
=Forms![Orders]![Orders Subform]![Price]*1.06	The product of the value of the Price control on the Orders Subform on the Orders form and 1.06 (adds 6 percent to the value of the Price control).
=Parent![OrderID]	The value of the OrderID control on the main or parent form of the current subform.

The following table lists examples of expressions you can use in calculated controls on reports.

If you use this expression	Microsoft Access displays
=Reports![Invoice]![OrderID]	The value of the OrderID control on the Invoice report.
=Reports![Summary]![Summary Subreport]![SalesTotal]	The value of the SalesTotal control on the Summary Subreport on the Summary report.
=Parent![OrderID]	The value of the OrderID control on the main or parent report of the current subreport.

Notes

- In a calculated control, precede the expression with the = operator.
- When you set the **Name** property of a calculated control, make sure you use a unique name. Don't use the name of one of the controls you used in the expression.

Examples of using aggregate functions in forms and reports

The following table lists examples of expressions you can use in calculated controls on forms and reports.

Expression	Description
=Avg([Freight])	Uses the **Avg** function to display the average of the values of the Freight control.
=Count([OrderID])	Uses the **Count** function to display the number of records in the OrderID control.
=Sum([Sales])	Uses the **Sum** function to display the sum of the values of the Sales control.
=Sum([Quantity]*[Price])	Uses the **Sum** function to display the sum of the product of the values of the Quantity and Price controls.
=[Sales]/Sum([Sales])*100	Displays the percentage of sales, determined by dividing the value of the Sales control by the sum of all the values of the Sales control. **Note** If the control's **Format** property is set to **Percent**, don't include the *100.

Notes

- In a calculated control, precede the expression with the = operator.
- When you set the **Name** property of a calculated control, make sure you use a unique name. Don't use the name of one of the controls you used in the expression.
- You can't use the name of a control in an expression that uses an aggregate function; you must use only field names from a table, query, or SQL statement. For information about using aggregate functions with

  calculated controls, click .

Examples of using domain aggregate functions on forms and reports

The following table lists examples of expressions you can use in calculated controls on forms and reports.

Expression	Description
=DLookup("[ContactName]", "[Suppliers]", "[SupplierID] = Forms![SupplierID]")	Uses the **DLookup** function to display the value of the ContactName field in the Suppliers table where the value of the SupplierID field in the table matches the value of the SupplierID control on the active

	form.
=DLookup("[ContactName]", "[Suppliers]", "[SupplierID] = Forms![New Suppliers]![SupplierID]")	Uses the **DLookup** function to display the value of the ContactName field in the Suppliers table where the value of the SupplierID field in the table matches the value of the SupplierID control on the New Suppliers form.
=DSum("[OrderAmount]", "[Orders]", "[CustomerID] = 'RATTC'")	Uses the **DSum** function to display the sum total of values of the OrderAmount field in the Orders table where the CustomerID is RATTC.

Notes

- In a calculated control, precede the expression with the = operator.
- When you set the **Name** property of a calculated control, make sure you use a unique name. Don't use the name of one of the controls you used in the expression.
- You can't use the name of a control in an expression that uses a domain aggregate function; you must use only field names from a table, query, or SQL statement. For information about using domain aggregate functions with calculated controls, click .

Examples of manipulating and calculating dates on forms, reports, and data access pages

The following table lists examples of expressions that you can use in calculated controls on forms, reports, and data access pages.

Expression	Description
=Date()	Uses the **Date** function to display the current date in the form of *mm-dd-yy*, where *mm* is the month (1 through 12), *dd* is the day (1 through 31), and *yy* is the last two digits of the year (1980 through 2099).
=Format(Now(), "ww")	Uses the **Format** function to display the number of the week of the year the current date represents, where *ww* is 1 through 53.
=DatePart("yyyy", [OrderDate])	Uses the **DatePart** function to display the four-digit year of the value of the OrderDate field.
=DateAdd("y", -10, [PromisedDate])	Uses the **DateAdd** function to display a date that is 10 days before the value of the PromisedDate field.
=DateDiff("d", [OrderDate], [ShippedDate])	Uses the **DateDiff** function to display the variance in days between the values of the OrderDate and ShippedDate fields.

Notes

- In the **ControlSource** property of a calculated control, precede the expression with the **=** operator. On a data access page, you can omit the = operator, and type an alias instead; for example, type **WeekNumber: Format(Now(), "ww")**.
- When you set the **Name** property of a calculated control in a form or report, or set the **ID** property of a calculated control in a data access page, make sure that you use a unique name. Don't use the name or ID of one of the controls you used in the expression.
- In an expression on a form or report, you can use the name of a control or the name of a field in the underlying record source. In an expression on a data access page, you can use only the name of a field that's in the data definition of the page.

Examples of returning one of two values on forms, reports, and data access pages

The following table lists examples of expressions that you can use in calculated controls on forms, reports, and data access pages.

Expression	Description
=IIf([Confirmed] = "Yes", "Order Confirmed", "Order Not Confirmed")	Uses the **IIf** function to display the message "Order Confirmed" if the value of the Confirmed field is Yes; otherwise, it displays the message "Order Not Confirmed."
=IIf(IsNull([Country]), " ", [Country])	Uses the **IIf** function to display an empty string if the value of the Country field is **Null**; otherwise, it displays the value of the Country control.
=IIf(IsNull([Region]), [City] & " " & [PostalCode], [City] & " " & [Region] & " " & [PostalCode])	Uses the **IIf** function to display the values of the City and PostalCode fields if Region is **Null**; otherwise, it displays the values of the City, Region, and PostalCode fields.
=IIf(IsNull([RequiredDate] - [ShippedDate]), "Check for a missing date", [RequiredDate] - [ShippedDate])	Uses the **IIf** function to display the message "Check for a missing date" if the result of subtracting ShippedDate from RequiredDate is **Null**; otherwise, it displays the difference between the values of the RequiredDate and ShippedDate fields.

Notes

- In the **ControlSource** property of a calculated control, precede the expression with the **=** operator. On a data access page, you can omit the = operator, and type an alias instead; for example, type **DisplayCountry: IIf(IsNull([Country]), " ", [Country])**.

- When you set the **Name** property of a calculated control in a form or report, or set the **ID** property of a calculated control in a data access page, make sure to use a unique name. Don't use the name or ID of one of the controls that you used in the expression.

- In an expression on a form or report, you can use the name of a control or the name of a field in the underlying record source. In an expression on a data access page, you can use only the name of a field that's in the data definition of the page.

What is an expression?

Expressions are a fundamental part of many Microsoft Access operations. An expression is a combination of symbols — identifiers, operators, and values — that produces a result.

For example, you can use the following expression in a control on a form or report to display the sum of the values in the Subtotal and Freight controls:

= [Subtotal] + [Freight]

Here are a few common examples of operations where you use expressions:

- Setting a property that defines a calculated control, establishes a validation rule, or sets a default field value.
- Entering a criteria expression, creating a calculated field, or updating records in a query or filter.
- Setting a condition for carrying out an action or series of actions in a macro, or specifying arguments for many actions.
- Specifying arguments for many functions, statements, and methods in Visual Basic for Applications procedures.
- Editing an SQL query in SQL view of the Query window, or using an SQL statement in a property setting or argument.

For information on creating an expression, click .

To see examples of expressions, click .

Create an expression without using the Expression Builder

You create an expression by combining identifiers, operators, and values to produce the result you want. For example, the following expression increases the value displayed in the Freight control on the Orders form by 10 percent:

= [Forms]![Orders]![Freight] * 1.1

In this expression:

- Forms![Orders]![Freight] is an identifier that refers to the value of the Freight control on the Orders form.
- * is the multiplication operator.
- 1.1 is the value by which Microsoft Access multiplies the value of the Freight control.

Depending on the result you want, you can combine identifiers, operators, and values in a variety of ways. You use expressions to combine strings of text, add or multiply numeric values, call functions, refer to objects and their values, and perform many other operations.

Some expressions produce a true or false result. For example, if you enter an expression in the **Condition** column of a macro, Microsoft Access carries out the specified action only when the expression evaluates to true. The following expression is true only if UK is the value displayed in the Country control on the Employees form:

Forms![Employees]![Country] = "UK"

For more information on using identifiers in expressions, click .

For more information on operators, click .

For more information on using values in expressions, click .

Refer to an object or its properties in expressions

You use an identifier in an expression to refer to an object or its properties. For example, you can refer to an open form, an open report, a control on an open form or report, or any properties of the form, report, or control. The following identifier refers to the **Visible** property of a control:

Reports![Invoice]![ShipName].Visible

The full identifier for an object or property shows the relationship between items in the identifier. In this identifier:

- Reports refers to the collection of open reports in the database. Microsoft Access automatically creates **Forms** and **Reports** collections for each database. The **Forms** collection is made up of all open forms, while the **Reports** collection is made up of all open reports.
- [Invoice] refers to the Invoice report in the **Reports** collection.
- [ShipName] refers to the ShipName control on the Invoice report.
- Visible refers to the **Visible** property of the ShipName control.

It's often a good idea to refer to an object or property using its full identifier. In some cases, a full identifier is required. For example, to refer to a control on a form or report that isn't the current form or report, you must type its full identifier. The following expression displays the sum of the values in the Subtotal and Freight controls on the Orders form in a control on a different form:

= Forms![Orders]![Subtotal] + Forms![Orders]![Freight]

However, in some circumstances you can refer to a control or its properties without specifying a full identifier:

- If you're referring to a control on the current form or report, you don't have to specify the form or report identifier. For example, to display the sum of the values in the Subtotal and Freight controls in a different control on the same form, set the **ControlSource** property of the control to:

  = [Subtotal] + [Freight]

- If you're referring to a control on a subform or subreport, you don't have to specify the full identifier for the form or report using the **Form** or **Report** property. For example, you can use the following identifier to refer to the Quantity control on the Orders Subform subform:

  Forms![Orders]![Orders Subform]![Quantity]

  The full identifier for the Quantity control would be:

  Forms![Orders]![Orders Subform].Form![Quantity]

- In a macro or action argument, you don't have to specify the identifier for the form or report from which the macro is run. For example, if you set an event property on a form to the name of a macro, you can refer to controls on the form in the macro's **Condition** column or action arguments without specifying the form's identifier.

- In a Visual Basic for Applications procedure, you can use the **Me** keyword rather than the full identifier to refer to a control on the current form or report. For example, to assign the sum of the values in the Subtotal and Freight controls on a form to the variable OrderTotal in one of the form's event procedures, add the following statement to the event procedure:

  ```
  OrderTotal = Me![Subtotal] + Me![Freight]
  ```

Notes

- When you run a macro or Visual Basic for Applications code containing an expression that refers to a form or report, the form or report must be open.

- In a Visual Basic procedure, you can refer to an object by enclosing its name in parentheses and double (") quotation marks instead of using the ! operator. The parentheses syntax is required if you want to use a variable in an identifier. For example, the following identifiers are equivalent:
- Forms![Orders]![OrderDate]

  Forms("Orders")("OrderDate")

For information on using the ! and . (dot) operators in expressions, click .

For more information on referring to a form, report, subform, or subreport in an expression, click .

For more information on referring to the value of a control or property in an expression, click .

Use values in expressions

You can specify a value in an expression by using a literal value, a constant, a function, or an identifier:

- A literal value represents a value such as a number, string, or date that Microsoft Access evaluates exactly as written. "New York," 100, and #1-Jan-94# are examples of literal values. Dates are enclosed in number signs (#), and strings in double (") quotation marks.
- A constant represents a value that doesn't change. **True**, **False**, and **Null** are examples of constants that are defined automatically by Microsoft Access. You can also define your own constants in Visual Basic for Applications that you can use in Visual Basic procedures.
- A function returns a value based on the results of a calculation or other operation. Microsoft Access includes many built-in functions; for example:
    - The **Date** function returns the current date.
    - The **Sum** function returns the sum of a set of field values.
    - The **DLookup** function returns a specific field value.

  You can also create your own functions using Visual Basic.
- An identifier refers to the value of a field, control, or property. For example, the following identifier refers to the value of the **DefaultValue** property of the OrderDate control on the Orders form:

  Forms![Orders]![OrderDate].DefaultValue

You can combine the value of a field, control, or property with a literal string by using the **&** (concatenation) operator. For example, the following expression combines the literal string "[CategoryID] = " with the value of the CategoryID control on the Products form:

"[CategoryID] = " & Forms![Products]![CategoryID]

In some circumstances — for example, in a domain aggregate function such as **DLookup** — the value of the field, control, or property must appear in single (') or double (") quotation marks. The easiest way to accomplish this is to add a single quotation mark to the literal string, and then combine the expression with another literal string made up of a single quotation mark after the field, control, or property value, as follows:

"[CategoryID] = ' " & Forms![Products]![CategoryID] & " ' "

Existing Microsoft Access applications may use the vertical bar operators (**| |**) in place of an opening and closing combination of double quotation marks and **&** (concatenation) operators, as follows:

"[CategoryID] = '|Forms![Products]![CategoryID]|' "

However, the use of vertical bars is not recommended because they can produce unexpected results in some circumstances.

If you want an expression to produce a string that is enclosed in double quotation marks, you can either enclose the nested string in single quotation marks or three sets of double quotation marks. For example, the following expressions are equivalent:

Forms![Contacts]![City].DefaultValue = ' "Paris" '

Forms![Contacts]![City].DefaultValue = " " "Paris" " "

For more information on constants, click .

For more information on creating your own functions, click .

For more information on using identifiers in expressions, click .