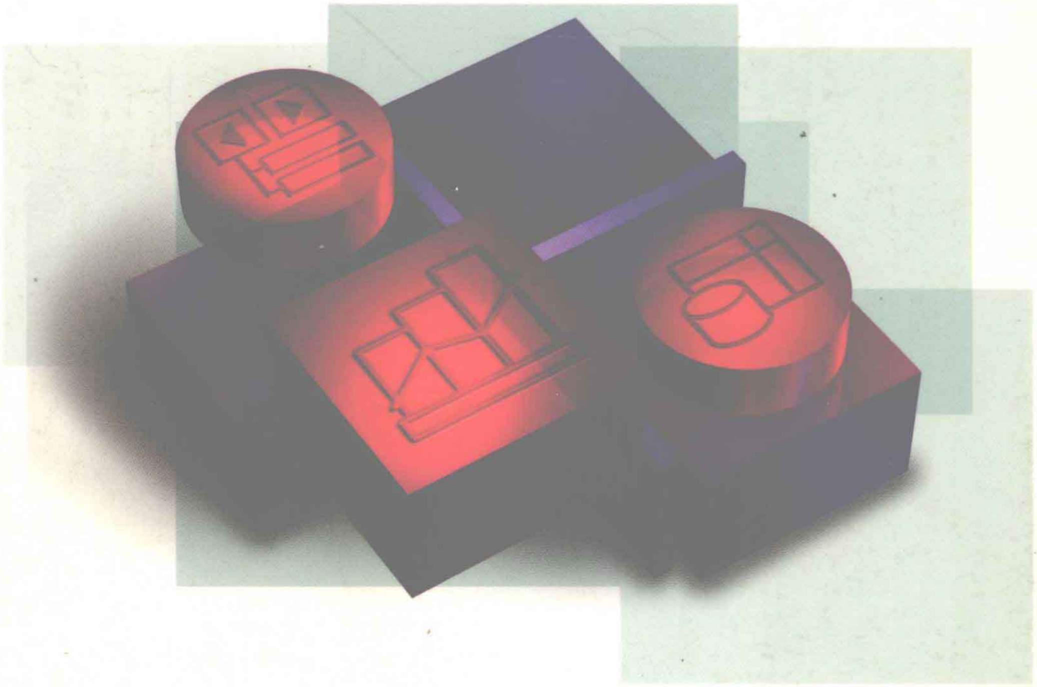
Designed for
Microsoft®
Windows NT®
Windows® 95

Microsoft Visual Basic 5.0
Reference Library



Microsoft® Visual Basic® 5.0
Language Reference

Microsoft® Press

Microsoft Visual Basic 5.0
Reference Library

Microsoft® Visual Basic® 5.0 Language Reference

Microsoft® Press

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Visual Basic 5.0 Language Reference
p. cm.
ISBN 1-57231-507-5
1. Microsoft Visual Basic. 2. BASIC (Computer program language).
I. Microsoft Corporation.
QA76.73.B3M557 1997
005.26'8--dc21 97-3524
CIP

Printed and bound in the United States of America.

3 4 5 6 7 8 9 WCWC 2 1 0 9 8

Distributed in Canada by ITP Nelson, a division of Thomson Canada Limited.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at mspress.microsoft.com.

Acquisitions Editor: Eric Stroo
Project Editor: Maureen Williams Zimmerman

Introduction

This guide is an alphabetic reference for the Visual Basic Programming System. The guide includes an A—Z reference listing objects, functions, statements, methods, properties, and events for the Visual Basic language. Several appendixes provide information on the ANSI character set, data types, operators, and derived math functions.

The information in this guide is the best available at the time of publication and is essentially identical to the information contained in the Visual Basic online Help. In some cases, more up-to-date or complete information may be available in online Help.

Document Conventions

Visual Basic documentation uses the following typographic conventions.

Convention	Description
Sub, **If**, **ChDir**, **Print**, **True**, **Debug**	Words in bold with initial letter capitalized indicate language-specific keywords.
setup	Words you are instructed to type appear in bold.
object, *varname*, *arglist*	Italic, lowercase letters indicate placeholders for information you supply.
pathname, ***filenumber***	Bold, italic, and lowercase letters indicate placeholders for arguments where you can use either positional or named-argument syntax.
[*expressionlist*]	In syntax, items inside brackets are optional.
{**While** | **Until**}	In syntax, braces and a vertical bar indicate a mandatory choice between two or more items. You must choose one of the items unless all of the items are also enclosed in brackets. For example: [{**This** | **OrThat**}]
ESC, ENTER	Words in small capital letters indicate key names and key sequences.
ALT+F1, CTRL+R	A plus sign (+) between key names indicates a combination of keys. For example, ALT+F1 means hold down the ALT key while pressing the F1 key.

Code Conventions

The following code conventions are used:

Sample Code	Description
`MyString = "Hello, world!"`	This font is used for code, variables, and error message text.
`' This is a comment.`	An apostrophe (') introduces code comments.

`MyVar = "This is an " _`
`& "example" _`
`& " of how to continue code."`
	A space and an underscore (_) continue a line of code.

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 53 BOTHELL, WA

POSTAGE WILL BE PAID BY ADDRESSEE

MICROSOFT PRESS REGISTRATION
MICROSOFT® VISUAL BASIC® 5.0
LANGUAGE REFERENCE
PO BOX 3019
BOTHELL WA 98041-9946

Contents

Introduction v
A–Z Reference 1
Appendix A ANSI Character Set 1577
Appendix B Data Types 1579
Appendix C Operators 1591
Appendix D Derived Math Functions 1593

A-Z Reference

& Operator

Used to force string concatenation of two expressions.

Syntax

result = *expression1* **&** *expression2*

The **&** operator syntax has these parts:

Part	Description
result	Required; any **String** or **Variant** variable.
expression1	Required; any expression.
expression2	Required; any expression.

Remarks

If an *expression* is not a string, it is converted to a **String** variant. The data type of *result* is **String** if both expressions are string expressions; otherwise, *result* is a **String** variant. If both expressions are **Null**, *result* is **Null**. However, if only one *expression* is **Null**, that expression is treated as a zero-length string ("") when concatenated with the other expression. Any expression that is **Empty** is also treated as a zero-length string.

See Also

Operator Summary, **Operator** Precedence, **Concatenation** Operators

Example

This example uses the **&** operator to force string concatenation.

```
Dim MyStr
MyStr = "Hello" & " World"              ' Returns "Hello World".
MyStr = "Check " & 123 & " Check"       ' Returns "Check 123 Check".
```

- Operator

Used to find the difference between two numbers or to indicate the negative value of a numeric expression.

Syntax 1

result = *number1*–*number2*

Syntax 2

–*number*

The **–** operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
number	Required; any numeric expression.
number1	Required; any numeric expression.
number2	Required; any numeric expression.

Remarks

In Syntax 1, the **–** operator is the arithmetic subtraction operator used to find the difference between two numbers. In Syntax 2, the **–** operator is used as the unary negation operator to indicate the negative value of an expression.

The data type of *result* is usually the same as that of the most precise expression. The order of precision, from least to most precise, is **Byte**, **Integer**, **Long**, **Single**, **Double**, **Currency**, and **Decimal**. The following are exceptions to this order:

If	Then *result* is
Subtraction involves a **Single** and a **Long**,	converted to a **Double**.
The data type of *result* is a **Long**, **Single**, or **Date** variant that overflows its legal range,	converted to a **Variant** containing a **Double**.
The data type of *result* is a **Byte** variant that overflows its legal range,	converted to an **Integer** variant.
The data type of *result* is an **Integer** variant that overflows its legal range,	converted to a **Long** variant.
Subtraction involves a **Date** and any other data type,	a **Date**.
Subtraction involves two **Date** expressions,	a **Double**.

Note The order of precision used by addition and subtraction is not the same as the order of precision used by multiplication.

One or both expressions are **Null** expressions, *result* is **Null**. If an expression is **Empty**, it is treated as 0.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators

Example

This example uses the–operator to calculate the difference between two numbers.

```
Dim MyResult
MyResult = 4 - 2            ' Returns 2.
MyResult = 459.35 - 334.90 ' Returns 124.45.
```

* Operator

Used to multiply two numbers.

Syntax

result = *number1***number2*

The * operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
number1	Required; any numeric expression.
number2	Required; any numeric expression.

Remarks

The data type of *result* is usually the same as that of the most precise expression. The order of precision, from least to most precise, is **Byte**, **Integer**, **Long**, **Single**, **Currency**, **Double**, and **Decimal**. The following are exceptions to this order:

If	Then *result* is
Multiplication involves a **Single** and a **Long**,	converted to a **Double**.
The data type of *result* is a **Long**, **Single**, or **Date** variant that overflows its legal range,	converted to a **Variant** containing a **Double**.
The data type of *result* is a **Byte** variant that overflows its legal range,	converted to an **Integer** variant.
the data type of *result* is an **Integer** variant that overflows its legal range,	converted to a **Long** variant.

If one or both expressions are **Null** expressions, *result* is **Null**. If an expression is **Empty**, it is treated as 0.

Note The order of precision used by multiplication is not the same as the order of precision used by addition and subtraction.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators

Example

This example uses the * operator to multiply two numbers.

```
Dim MyValue
MyValue = 2 * 2' Returns 4.
MyValue = 459.35 * 334.90  ' Returns 153836.315.
```

/ Operator

Used to divide two numbers and return a floating-point result.

Syntax

result = *number1*/*number2*

The **/** operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
number1	Required; any numeric expression.
number2	Required; any numeric expression.

Remarks

The data type of *result* is usually a **Double** or a **Double** variant. The following are exceptions to this rule:

If	Then *result* is
Both expressions are **Byte**, **Integer**, or **Single** expressions,	a **Single** unless it overflows its legal range; in which case, an error occurs.
Both expressions are **Byte**, **Integer**, or **Single** variants,	a **Single** variant unless it overflows its legal range; in which case, *result* is a **Variant** containing a **Double**.
Division involves a **Decimal** and any other data type,	a **Decimal** data type.

One or both expressions are **Null** expressions, *result* is **Null**. Any expression that is **Empty** is treated as 0.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators

Example

This example uses the **/** operator to perform floating-point division.

```
Dim MyValue
MyValue = 10 / 4   ' Returns 2.5.
MyValue = 10 / 3   ' Returns 3.333333.
```

\ Operator

Used to divide two numbers and return an integer result.

Syntax

result = *number1\number2*

The **** operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
number1	Required; any numeric expression.
number2	Required; any numeric expression.

Remarks

Before division is performed, the numeric expressions are rounded to **Byte**, **Integer**, or **Long** expressions.

Usually, the data type of *result* is a **Byte**, **Byte** variant, **Integer**, **Integer** variant, **Long**, or **Long** variant, regardless of whether *result* is a whole number. Any fractional portion is truncated. However, if any expression is **Null**, *result* is **Null**. Any expression that is **Empty** is treated as 0.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators

Example

This example uses the **** operator to perform integer division.

```
Dim MyValue
MyValue = 11 \ 4      ' Returns 2.
MyValue = 9 \ 3       ' Returns 3.
MyValue = 100 \ 3     ' Returns 33.
```

^ Operator

Used to raise a number to the power of an exponent.

Syntax

result = *number*^*exponent*

The **^** operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
number	Required; any numeric expression.
exponent	Required; any numeric expression.

Remarks

A *number* can be negative only if *exponent* is an integer value. When more than one exponentiation is performed in a single expression, the ^ operator is evaluated as it is encountered from left to right.

Usually, the data type of *result* is a **Double** or a **Variant** containing a **Double**. However, if either *number* or *exponent* is a **Null** expression, *result* is **Null**.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators

Example

This example uses the ^ operator to raise a number to the power of an exponent.

```
Dim MyValue
MyValue = 2 ^ 2' Returns 4.
MyValue = 3 ^ 3 ^ 3  ' Returns 19683.
MyValue = (-5) ^ 3' Returns -125.
```

+ Operator

Used to sum two numbers.

Syntax

result = *expression1*+*expression2*

The + operator syntax has these parts:

Part	Description
result	Required; any numeric variable.
expression1	Required; any expression.
expression2	Required; any expression.

Remarks

When you use the + operator, you may not be able to determine whether addition or string concatenation will occur. Use the **&** operator for concatenation to eliminate ambiguity and provide self-documenting code.

If at least one expression is not a **Variant**, the following rules apply:

If	Then
Both expressions are numeric data types (**Byte**, **Boolean**, **Integer**, **Long**, **Single**, **Double**, **Date**, **Currency**, or **Decimal**)	Add.
Both expressions are **String**	Concatenate.
One expression is a numeric data type and the other is any **Variant** except **Null**	Add.

(continued)

If	**Then**
One expression is a **String** and the other is any **Variant** except **Null**	Concatenate.
One expression is an **Empty Variant**	Return the remaining expression unchanged as *result*.
One expression is a numeric data type and the other is a **String**	A `Type mismatch` error occurs.
Either expression is **Null**	*result* is **Null**.

If both expressions are **Variant** expressions, the following rules apply:

If	**Then**
Both **Variant** expressions are numeric	Add.
Both **Variant** expressions are strings	Concatenate.
One **Variant** expression is numeric and the other is a string	Add.

For simple arithmetic addition involving only expressions of numeric data types, the data type of *result* is usually the same as that of the most precise expression. The order of precision, from least to most precise, is **Byte**, **Integer**, **Long**, **Single**, **Double**, **Currency**, and **Decimal**. The following are exceptions to this order:

If	**Then *result* is**
A **Single** and a **Long** are added,	a **Double**.
The data type of *result* is a **Long**, **Single**, or **Date** variant that overflows its legal range,	converted to a **Double** variant.
The data type of *result* is a **Byte** variant that overflows its legal range,	converted to an **Integer** variant.
The data type of *result* is an **Integer** variant that overflows its legal range,	converted to a **Long** variant.
A **Date** is added to any data type,	a **Date**.

If one or both expressions are **Null** expressions, *result* is **Null**. If both expressions are **Empty**, *result* is an **Integer**. However, if only one expression is **Empty**, the other expression is returned unchanged as *result*.

Note The order of precision used by addition and subtraction is not the same as the order of precision used by multiplication.

See Also

Operator Summary, **Operator** Precedence, **Arithmetic** Operators, **Concatenation** Operators

Example

This example uses the **+** operator to sum numbers. The **+** operator can also be used to concatenate strings. However, to eliminate ambiguity, you should use the **&** operator instead. If the components of an expression created with the **+** operator include both strings and numerics, the arithmetic result is assigned. If the components are exclusively strings, the strings are concatenated.

```
Dim MyNumber, Var1, Var2
MyNumber = 2 + 2  ' Returns 4.
MyNumber = 4257.04 + 98112 ' Returns 102369.04.

Var1 = "34": Var2 = 6' Initialize mixed variables.
MyNumber = Var1 + Var2  ' Returns 40.

Var1 = "34": Var2 = "6"  ' Initialize variables with strings.
MyNumber = Var1 + Var2  ' Returns "346" (string concatenation).
```

AboutBox Method

Displays the **About** box for the control.

Applies To

DBGrid Control, **MSChart** Control

Syntax

object.**AboutBox**

The *object* placeholder represents an object expression that evaluates to an object in the **Applies To** list.

Remarks

This is the same as clicking **About** in the **Properties** window.

Abs Function

Returns a value of the same type that is passed to it specifying the absolute value of a number.

Syntax

Abs(*number***)**

The required *number* argument can be any valid numeric expression. If *number* contains **Null**, **Null** is returned; if it is an uninitialized variable, zero is returned.

Remarks

The absolute value of a number is its unsigned magnitude. For example, `ABS(-1)` and `ABS(1)` both return `1`.