

PowerTrax® Pro

Formula Reference



Version 2.01



Results Through Focused Technology...

PowerTrax[®] Pro

Formula Reference



Version 2.01

© 1996-2000 Soft Solutions, Inc. All Rights Reserved.

Published World-Wide by Soft Solutions, Inc.

Soft Solutions, Inc.

2900 Chamblee Tucker Road

Building 12, Suite 200

Atlanta, GA 30341 USA



Results Through Focused Technology...

Table of Contents

Introduction	1
Constants	2
String Constants	2
Date Constants	2
Time Constants	3
Numeric Constants	3
Functions	4
String Functions	4
Character Reference Symbols	4
Length	5
Substring	6
Position	6
Change String	7
Insert String	7
Delete String	8
Replace String	9
Lowercase	10
Uppercase	10
.CapFirst	10
Date Functions	11
Current date	11
Date	11
Day of	12
Month of	12
Year of	12
Time Functions	13
Current time	13
Time	13
Time string	14
Mathematical Functions	14
Abs	15
Dec	15
Int	15

iv Table of Contents

Num	16
Num	16
Random	17
Round	17
Trunc	18
Statistical Functions	18
Using a Field	19
Average	19
Max	20
Min	20
Sum	21
Sum squares	21
Std deviation	21
Variance	22
Logical Functions	22
True	22
False	22
Not	23

Formula Reference

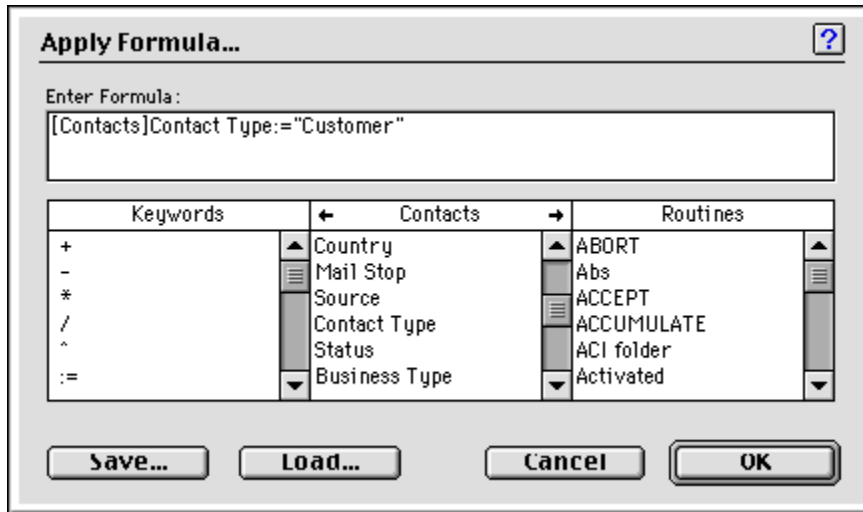
Introduction

The “Apply Formula” feature is available within PowerTrax Pro to help you efficiently update groups of database records. The Apply Formula feature allows you to literally “apply” formulas or calculations on database fields for any number of database records and for any purpose.



This feature is effective at helping you save time by modifying your database information in an endless variety of ways. This feature can also place some or all of your data at risk when used improperly. Because of this, the Apply Formula feature is protected from general availability by requiring the Administrator’s password.

This documentation provides syntax requirements, explanations of common characters and symbols used in Constants and Functions and examples of how the various functions might be applied to your database. The examples shown contain text strings. You may also use the field names within your database (e.g. FirstName, LastName, etc.).



2 Formula Reference

Constants

A constant is an expression that has a fixed value. Constants can be of four data types:

- string
- numeric
- date
- time

String Constants

A string constant is enclosed in double, straight quotation marks ("...").

Here are some examples of string constants:

```
"Add Records"  
"No records found."  
"Invoice"
```

An empty string is specified by two quotation marks with nothing between them ("").

Date Constants

A date constant is enclosed in exclamation marks (!...!).

A date is ordered month/day/year, with a slash (/) setting off each part.

Here are some examples of date constants:

```
!1/1/76!  
!4/4/04!  
!12/25/89!
```

An empty date is specified by !00/00/00!.

A two-digit year is assumed to be in the 1900's.

Time Constants

A time constant is enclosed in time symbols (†...†). (Press the Option-t key combination to get the time symbols.)

A time is ordered hour:minute:second, with a colon (:) setting off each part.

Times are specified in 24-hour format.

If the minute or second is omitted, it is assumed to be zero. For example, †1† is equal to †1:00† which is equal to †01:00:00†.

Here are some examples of time constants:

```
†01:00:00†
†01:01:00†
†13:01:59†
```

Numeric Constants

A numeric constant is written as a real number.

Here are some examples of numeric constants:

```
27
123.76
.0076
```

Negative numbers are specified with the negation symbol (-):

```
-27
-123.76
-.0076
```

Numbers can be specified with scientific notation, using an e, followed optionally by the negation symbol for a negative exponent, and completed with the exponent:

```
2.7e1
1.2376e+2
7.6e-3
```

4 Formula Reference

Functions

The functions in this section are grouped in the following categories:

- String
- Date
- Time
- Mathematical
- Statistical
- Logical

String Functions

Length	Insert String	Uppercase
Substring	Delete String	String
Position	Replace String	ASCII
Change String	Lower Case	Char

This section describes functions that work on strings. It also describes the character reference symbols. None of the string functions alters the string expressions used as parameters.

Character Reference Symbols

string ≤ position ≥ → String (1 character)

Table of Parameters

Parameter	Type	Description
string	String	String whose character to return
position	Number	Position of character to return

The character reference symbols (≤...≥) are used to refer to a single character within string. The position of the character, position, is specified between the character reference symbols.

The character at position is returned.

To create the ≤ character, depress the option key and the "less than" (<) character concurrently. To create the ≥ character, depress the option key and the "greater than" (>) character concurrently.

The \rightarrow character shown throughout this documentation refers to the result that will be given as a result of the calculation. When reviewing commands/functions in this documentation which give examples containing the \rightarrow character, the command/function shown on the left portion of the example should appear on the right side of an `_assignment` statement. Alternatively, the function may be imbedded within a larger formula. The first "Length" example (also shown later in the section) indicates that a number is returned when using the `Length (string)` function. The next example is an actual usage which places the Length of "5" in the field "Result".

`Length (string) \rightarrow Number`

`Result := Length ("Topaz")`

If the character reference symbols appear on the left side of the assignment operator, a character is assigned to the referenced position in the string. For example, the following line sets the first character of Name to uppercase:

`Name<1> = Uppercase (Name<1>)`

If you refer to characters that are beyond the length of the string, the results are undefined.

The following lines show the use of the character reference symbols.

<code>Result := "abcd" 3</code>	Result gets "c"
<code>Result := Last Name 1</code>	Result gets first character in Last Name
<code>Result := City \$i</code>	Result gets the \$i'th character in City

Length (string) \rightarrow Number

Length Parameters

Parameter	Type	Description
string	String	String whose length to return

Length is used to find the length of string. Length returns the number of characters that are in the string.

The following example illustrates the use of Length. The results are assigned to the variable Result. The comments describe what Result is set to.

<code>Result := Length ("Topaz")</code>	Result gets 5
<code>Result := Length ("Citizen")</code>	Result gets 7

6 Formula Reference

Substring (source, first char, {number of chars}) → String

Substring Parameters

Parameter	Type	Description
source	String	String from which to get substring
first char	Number	Position of first character
number of chars	Number	Number of characters to get

Substring returns the portion of source defined by first char and number of chars. The first char parameter points to the first character in the string to return, and number of chars specifies how many characters to return.

If the sum of first char and number of chars exceeds 32,767, the results are undefined.

If first char plus number of chars is greater than the number of characters in the string, or if number of chars is not specified, substring returns the last character(s) in the string, starting with the character specified by first char. If first char is greater than the number of characters in the string, substring returns an empty string ("").

The following example illustrates the use of Substring. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Substring ("08/04/62"; 4; 2)      Result gets "04"  
Result := Substring ("Emergency"; 1; 6)    Result gets "Emerge"  
Result := Substring (var; 2)               Result gets all characters except  
                                           the first
```

Position (find, string) → Number

Position Parameters

Parameter	Type	Description
find	String	String to find
string	String	String in which to search

Position returns the position of the first occurrence of find in string.

If position fails to find the string find, it returns a zero (0). If position finds an occurrence of find, it returns the position of the first character of the occurrence in string. If you ask for the position of an empty string within an empty string, Position returns one (1).

The following example illustrates the use of Position. The results are assigned to the variable result. The comments describe what Result is set to.

Result := **Position** ("II"; "Willow") Result gets 3
 Result := **Position** (var1; var2) Result gets the first occurrence of var1 in var2

Change String (source, what, where) —> String

Change String Parameters

Parameter	Type	Description
source	String	Original String
what	String	New Characters
where	Number	Where to start the changes

Change string changes a group of characters in source and returns the resulting string. Change string overlays source, with the characters in what, starting at the character described by where.

If what is an empty string (""), change string returns source unchanged. Change string always returns a string of the same length as source.

Change string is different from Insert string in that it overwrites characters instead of inserting them.

The following example illustrates the use of Change string. The results are assigned to the variable Result. The comments describe what result is set to.

Result := **Change string** ("Macintosh SE"; "II"; 11) Result gets "Macintosh II"
 Result := **Change string** ("Acme"; "CME"; 2) Result gets "ACME"
 Result := **Change string** ("November"; "Dec", 1) Result gets "December"

Insert String (source, what, where) —> String

Insert String Parameters

Parameter	Type	Description
source	String	String into which to insert
what	String	String to insert
where	Number	Where to insert

8 Formula Reference

Insert string inserts a string into source and returns the resulting string. Insert string inserts the string what before the character described by where.

If what is an empty string (""), Insert string returns source unchanged.

If where is greater than the length of source, then what is appended to source. If where is less than zero (0), then what is inserted in front of source.

Insert string is different from Change string in that it inserts characters instead of overwriting them.

The following example illustrates the use of Insert string. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Insert string ("The tree"; " green"; 4)   Result gets "The green tree"  
Result := Insert string ("Shut"; "o"; 3)           Result gets "Shout"  
Result := Insert string ("Indention"; "ta"; 6)     Result gets "Indentation"
```

Delete String (source, where, number of chars) —> String

Delete String Parameters

Parameter	Type	Description
source	String	String from which to delete
where	Number	First character to delete
number of chars	Number	Number of characters to delete

Delete string deletes number of chars from source, starting at where, and returns the resulting string. Delete string does not modify source.

Delete string returns the same string as source in a number of cases:

- if source is an empty string
- if where is zero (0) or less than zero
- if where is greater than the length of source
- if number of chars is zero (0)

If where plus number of chars is equal to or greater than the length of source, the characters are deleted to the end of source.

If number of chars is negative, the characters that would have been deleted are inserted.

The following example illustrates the use of Delete string. The results are assigned to the variable result. The comments describe what Result is set to.

```
Result := Delete string ("Lamborghini"; 6; 5)    Result gets "Lambo"
Result := Delete string ("Indentation", 6; 2)   Result gets "Indention"
Result := Delete string (var; 3; 32000)         Result gets the first two characters
                                                of var
```

Replace String (source, old string, new string, {how many}) —> String

Replace String Parameters

Parameter	Type	Description
source	String	Original string
old string	String	Characters to replace
new string	String	String to replace with
how many	Number	How many times to replace

Replace string replaces every occurrence of old string in source with new string.

If new string is an empty string (""), Replace string deletes each occurrence of old string in source.

If how many is specified, Replace string will replace only the number of occurrences of old string specified, starting at the first character of source.

If old string is an empty string, Replace string returns an empty string.

The following example illustrates the use of Replace string. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Replace string ("Willow"; " ll"; "d")  Result gets "Widow"
Result := Replace string ("Shout"; "o "; "")    Result gets "Shut"
Result := Replace string (var; Char (9); ",")   Replaces all tabs in var with commas
```

10 Formula Reference

Lowercase (string) → String

Lowercase Parameters

Parameter	Type	Description
string	String	String to convert to lowercase

Lowercase takes string and returns the string with all alphabetic characters in lowercase, the original string is not affected. Lowercase affects only the characters A through Z.

Result := **Lowercase** ("SMITH") Result gets "smith"

Uppercase (string) → String

Uppercase Parameters

Parameter	Type	Description
string	String	String to convert to uppercase

Uppercase takes string and returns the string with all alphabetic characters in uppercase. The original string is not affected. Uppercase affects only the characters a through z.

Result := **uppercase** ("smith") Result gets "SMITH"

.CapFirst (string) → String

.CapFirst Parameters

Parameter	Type	Description
string	String	Convert first character to capital

.CapFirst converts the first character to the capitalized equivalent of the same letter. Also works on any characters preceded by a blank (See example).

The following example is a function called .CapFirst, which capitalizes the first character of the string passed to it. For example

Name := **.CapFirst** ("john") would set Name = "John".

The .CapFirst routine also capitalizes each letter in string which is preceded by a blank.

Date Functions

Current date	Day number	Month of
Date	Day of	Year of

This section describes date functions.

Current date () → Date

Current date returns the current date as kept by the Mac OS system clock.

The following example displays an alert box with the current date in it.

[Contacts]birthdate := **Current date**

Date (date string) → Date

Date Parameters

Parameter	Type	Description
date string	String	String representing the date to be returned

Date evaluates date string and returns a date.

The date string parameter must follow the normal rules for the format of a date. The date must be in the order MM/DD/YY (month, day, year) for United States systems. The month and day may be one or two digits. The year may be two or four digits. If the year is two digits, the Date adds 19 to the beginning of the year. The following characters are valid date separators; slash (/), space, period (.), and hyphen (-).

If date string is invalid, the date is returned undefined. Date does not evaluate an alphabetic date like “Jan 1, 1990.”

The following example prompts the user for a date, using a request box. The string the user enters is converted a date and stored in the ReqDate variable.

ReqDate := **Date** (01/01/93) ReqDate, a date field, is set to 01/01/93.

12 Formula Reference

Day of (date) → Description

Day of Parameters

Parameter	Type	Description
date	Date	Date for which to return the day

Day of returns the day of the month of date.

The following example illustrates the use of Day of. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Day of (!12/25/88!)    Result gets 25
Result := Day of (Current date)  Result gets the day of the current date
```

Month of (date) → Number

Month of Parameters

Parameter	Type	Description
date	Date	Date for which to return the month

Month of return a number indicating the month of date.

The following example illustrates the use of Month of. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Month of (!12/25/88!)  Result gets 12
Result := Month of (Current date) Result gets the month of the current date
```

Year of (date) → Number

Year of Parameters

Parameter	Type	Description
date	Date	Date for which to return the year

Year of returns a number indicating the year of date.

The following example illustrates the use of Year of. The results are assigned to the variable Result. The comments describe what Result is set to.

```
Result := Year of (!12/25/88!)    Result gets 1988
Result := Year of (Current date)  Result gets the year of the current date
```

Time Functions

Current time Time Time string

This section describes time functions. Times can be treated like a number when performing calculations. You can use the following statements to calculate the hours, minutes, and seconds of a time:

Hours := Time Var \ 3600	Returns the number of hours
Minutes := Time Var \ 60 % 60	Returns the number of minutes
Seconds := Time Var % 60	Returns the number of seconds

Current time () → Time

Current time returns the current time from the Mac OS system clock. The current time is always between †00:00:00† and †23:59:59†, inclusive. String can be used to convert the time into a string.

The following example shows a method you can use to time the length of an operation. In the example, LongOperation is a global procedure that needs to be timed.

It Took := Current time	Save the start time
LongOperation	Perform the operation to be timed
It Took := Current time – It Took	Calculate how long it took
Notes := ("The operation took" + String (It Took; 4))	Assign how long it took into notes field

Time (time string) → Time

Time Parameters

Parameter	Type	Description
time string	String	Time for which to return the number of seconds

Time returns the time specified by time string. The time string parameter must follow the HH:MM:SS format and be in the 24-hour time.

The following example displays an alert box with the message, "1:00 P.M. = 13 hours 0 minute."

ALERT ("1:00 P.M. = "+ **String** (**Time** ("13:00:00"); 4))

14 Formula Reference

Time string (seconds) → String

Time String Parameters

Parameter	Type	Description
seconds	Number	Seconds from midnight

Time string takes seconds, the number of seconds since midnight, and returns the time as a string in 24-hour format. The string is in the HH:MM:SS format.

If you go beyond the number of seconds in day (86,400), Time string continues to add hours, minutes, and seconds. For example, Time string (86401) returns 24:00:01.

String is different from time. The String command formats a number as a number and the Time command formats it as a time.

The following example displays an alert box with the message, "46800 seconds is 13:00:00."

```
ALERT ("46800 seconds is "+ Time string (46800))
```

Mathematical Functions

Abs	Int	Random
Dec	Log	Round
Exp	Num	Trunc

This section describes the standard math functions. Each of these functions returns a numeric value.

Note

Your database uses SANE (the Standard Apple Numeric Environment) for all calculations and with all numeric functions. The accuracy of numeric operations and functions is therefore dependent on SANE. SANE packages are also used for different hardware configurations. The use of different SANE packages may cause slightly different results for the same operations.

Abs (number) → Number

Abs Parameters

Parameter	Type	Description
number	Number	Number of which to return the absolute value

Abs returns the absolute (unsigned, positive) value of number.

If number is negative, it is returned as a positive. If number is positive, it is unchanged.

The following example returns the absolute value of -10.3 , which is 10.3 .

vVector := **Abs** (-10.3) vVector gets 10.3

Dec (number) → Number

Dec Parameters

Parameter	Type	Description
number	Number	Number of which to return the decimal point

Dec returns the decimal (fractional) part of number. The value returned is always a positive or zero.

The following example takes a monetary value expressed as a real number, and extracts the dollar part and cents part. If Amount were 7.31 , then Dollars would be set to 7 and Cents would be set to 31 .

Dollars := **Int** (Amount) Get the Dollars
 Cents := **Dec** (Amount) * 100 Get the fractional part

Int (number) → Number

Int Parameters

Parameter	Type	Description
number	Number	Number of which to return the integer portion

Int returns the integer portion of number without rounding. Int truncates a negative number toward zero.

16 Formula Reference

The following example illustrates how `Int` works for both a positive and a negative number. Note that the decimal portion of the number is removed.

```
x := Int (123.4)      x gets 123
y := Int (-123.4)    y gets -123
```

Num (string) → Number

Num Parameters

Parameter	Type	Description
string	String	String to be converted to a number

Num (Boolean) → Number (0 or 1)

Num (Boolean) Parameters

Parameter	Type	Description
Boolean	Boolean	Boolean value to be converted to 0 or 1

The `Num` function has two forms.

The first form of `Num` converts string into a numeric value. If string consists only of one or more alphabetic characters, `Num` returns a zero. If string includes alphabetic characters mixed in with numeric characters, `Num` ignores the alphabetic characters. Thus, `Num` transforms the string "a1b2c3" into the number 123.

There are three reserved characters that `Num` treats specially. They are the period (`.`), the hyphen (`—`), and e (or E). They are interpreted as numeric format characters.

The period is interpreted as a decimal place and must appear embedded in a numeric string.

The hyphen causes the number or an exponent to be negative. The hyphen must appear before any numeric characters or after the e for an exponent. If a hyphen is embedded in a numeric string, all numbers to the right are ignored, so `Num ("123—456")` returns 123.

The e or E causes any numeric characters to its right to be interpreted as an exponent. The e must be embedded in a numeric string. Thus, `Num ("123e—2")` returns 1.23.

The second form of `Num` evaluates `Boolean` and returns 0 or 1. If `Boolean` is `FALSE`, `Num` returns 0. If `Boolean` is `TRUE`, `Num` returns 1.

The following example illustrates how Num works when passed a numeric argument. Each line assigns a number to the Result variable. The comments describe the results.

```
Result := Num ("ABCD")           Result gets 0
Result := Num ("A1B2C3")        Result gets 123
Result := Num ("123")           Result gets 123
Result := Num ("123.4")         Result gets 123.4
Result := Num ("—123")          Result gets —123
Result := Num ("—123e2")        Result gets —12300
```

In the following example, Num of the customer debits returns wither 0 or 1. Using the asterisk (*) as a string repetition operator, the customer comment is then stored in a field called Risk within a file called Client.

```
'If client owes less than 1000, a good risk.
'If client owes more than 1000, a bad risk.
[Client]Risk:=("Good" * Num ([Client]Debt<1000))+("Bad" * Num ([Client]Debt>=
1000))
```

Random () —> Number

Random returns a random integer value between 0 and 32,767 (inclusive).

To define a range of integers, use this formula:

Random % (End — Start + 1) + Start

Start is the first number in the range, and End the last.

The following example assigns a random integer between 10 and 30 to the Result variable.

```
Result := Random % 21 + 10
```

Round (number, places) —> Number

Round Parameters

Parameter	Type	Description
number	Number	Number to be rounded
places	Number	Number of decimal places to round to

Round returns number rounded to the number of decimal places given by places.

If places is positive, number is rounded to places decimal places. If places is negative, number is rounded on the left of the decimal point.

18 Formula Reference

If the digit following places is 5 through 9, Round rounds toward positive infinity for a positive number, and toward negative infinity for negative number. If the digit following places is 0 through 4, Round rounds toward zero.

The following example illustrates how Round works with different arguments. Each line assigns a number to the Result variable. The comments describe the results.

```
Result := Round (16.857;2)      Result gets 16.86
Result := Round (32345.67; -3)  Result gets 32000
Result := Round (29.8725; 3)   Result gets 29.873
Result := Round (-1.5; 0)      Result gets -2
```

Trunc (number, places) → Number

Trunc Parameters

Parameter	Type	Description
number	Number	The number to truncate
places	Number	Number of decimal places to truncate to

Trunc returns number with its decimal part truncated by the number of decimals specified by places. Trunc always truncates toward negative infinity.

If places is positive, number is truncated to places decimal places. If places is negative, number is truncated on the left of the decimal point.

The following example illustrates how Trunc works with different arguments. Each line assigns a number to the Result variable. The comments describe the results.

```
Result := Trunc (216.897; 1)    Result gets 216.8
Result := Trunc (216.897; -1)   Result gets 210
Result := Trunc (-216.897; 1)  Result gets -216.9
Result := Trunc (-216.897; -1) Result gets -220
```

Statistical Functions

Average	Sum	Std deviation
Max	Sum squares	Variance
Min		

These functions perform calculations on a series of values. The values for the Average, Max, Min, and Sum functions can be fields from a selection of records, or they can be sub-records. The values for the Sum squares, Std deviation, and Variance functions can be fields when used within the Quick Report generator. For more details on these functions, please read the chapter on Quick Reports in your *PowerTrax Pro User's Guide*.

These functions work on numeric data only. Each of these functions returns a numeric value. These statistical functions are used primarily in the "Totals" row within the Quick Report generator.

Using a Field

When Average, Max, Min, or Sum is used on a field, it must load each record in the current selection to calculate the result. If there are many records, this process may take some time.

When these functions are used in a Quick Report, they behave differently than at other times. This is because the report itself must load each record.

When you use these functions in a Quick Report, the values that are returned are meaningful only in a footer or break and at break level 0. This means that they are meaningful only at the end of a report, after all the records have been processed. You typically use the functions in a script for a nonenterable area that is included in the B0 Break area. The script assigns the value returned to the variable associated with the area.

Average (series) —> Number

Average Parameters

Parameter	Type	Description
series	Field or subfield	Data for which to return the average

Average returns the arithmetic mean (average) of series.

The following example sets a variable that is in the B0 Break area of a Quick Report

```
vAverage := Average ([Employees] Sales)
```

The following example finds the average age of an employee's children from subfile data.

```
vAvg Age := Average ([Employees] Children'Age)
```

20 Formula Reference

Max (series) —> Number

Max Parameters

Parameter	Type	Description
series	Field or subfield	Data for which to return the maximum value

Max returns the maximum value in series.

The following example is a script for a variable, vMax, placed in the break 0 portion of the layout. The variable is printed at the end of the report. The script assigns the maximum value of the field to the variable, which is then printed in the last break of the report.

```
vMax := Max ([People]Age)
```

The following example finds the maximum sales of an employee, and displays the result in an alert box. The sales amounts are stored in a subfield, [Employees]Sales'Dollars

```
Alert ("The maximum sale was "+ String (Max ([Employees]Sales'Dollars))
```

Min (series) —> Number

Min Parameters

Parameter	Type	Description
series	Field or subfield	Data for which to return the minimum value

Min returns the minimum value in series.

The following example is a script for a variable, vMin, placed in the break 0 portion of the layout. The variable is printed at the end of the report. The script assigns the minimum value of the field to the variable, which is then printed in the last break of the report.

```
vMin := Min ([People]Age)
```

The following example finds the minimum sales of an employee, and displays the result in an alert box. The sales amounts are stored in a subfield, [Employees]Sales'Dollars.

```
Alert ("the minimum sale was "+ String (Min ([Employees]Sales'Dollars))
```

Sum (series) → Number

Sum Parameters

Parameter	Type	Description
series	Field or subfield	Data for which to return the sum

Sum returns the sum (total of all values) for series.

The following example is a script for a variable, vTotal, placed in a Quick Report. The script assigns the sum of all the lines in an invoice to the variable. The invoice lines are stored in a subfile called [Invoice]Lines.

```
vTotal := Sum ([Invoice]Lines'Line Total)
```

Sum squares (series) → Number

Sum squares Parameters

Parameter	Type	Description
series	Subfield or field	Data for which to return the sum of squares

Sum squares returns the sum of squares of series. You can only use a field with this function when printing a Quick Report.

The following example is a script for a variable called Squares. The script assigns the sum of squares for a data series to Squares.

```
Squares := Sum squares ([File]Data'Series)
```

Std deviation (series) → Number

Std deviation Parameters

Parameter	Type	Description
series	Subfield or field	Data for which to return the standard deviation

Std deviation returns the standard deviation of series. You can only use a field with this function when printing a Quick Report.

The following example is a script for a variable called Deviate. The script assigns standard deviation for a data series to Deviate.

```
Deviate := Std deviation ([File]Data'Series)
```

22 Formula Reference

Variance (series) → Number

Variance Parameters

Parameter	Type	Description
series	Subfield or field	Data for which to return the variance

Variance returns the variance for series. You can only use a field with this function when printing a Quick Report.

The following example is a script for a variable called Var. The script assigns the sum of squares for a data series to Var.

Var := **Variance** ([File]Data'Series)

Logical Functions

True False Not

This section describes the logical functions.

True () → Boolean (TRUE)

True returns the Boolean value TRUE.

The following example sets the variable, My Var, to TRUE.

My Var := **True**

False () → Boolean (FALSE)

False returns the Boolean value FALSE.

The following example sets the variable, My Var, to FALSE.

My Var := **False**

Not (Boolean) —> Boolean

Not Parameters

Parameter	Type	Description
Boolean	Boolean	Boolean value to negate

The Not function returns the negation of Boolean, changing a TRUE to FALSE or a FALSE to TRUE.

The following example first assigns TRUE to a variable. The example then changes the variable's value to FALSE, and then back to TRUE.

```
Result := True           'Result is set to TRUE
Result := Not (Result)    'Result is set to FALSE
Result := Not (Result)    'Result is set to TRUE
```

