topicus


# Manual
# MOO Documentation
# 1.0.0

# Introduction

***by Topicus Finan BV***

*Welcome to the instruction manual of Moo. This manual has separate chapters. Definitions, Scorecard and References.*
*The definitions is a topic where the most important definitions are given.*
*The Scorecard is an explanation of how to set up a Scorecard model with all the different variables.*
*References have the following sub chapters; Functions (Global String, Local String, Global Numeric, Local Numeric and Local Variable), Properties (Data, Restriction, Event, Presentation and Model), Constants and Operators. Here are all the variables explained.*

# Table of Contents


1. Definitions 10

2. Scorecard 12

3. Reference 14

3.1. Functions 14

3.1.1. Global String Functions 14

ActiveScriptName (string function) 14

AfterStr (string function) 15

BeforeStr (string function) 15

Case (string function) 16

Chr (string function) 17

DateStr (string function) 17

DioStatus (string function) 18

EvaluateAsString (string function) 18

FirstLC (string function) 19

FirstUC (string function) 19

If (string function) 20

Initials (string function) 20

OnNoStr (string function) 21

Str (string function) 21

StrField (string function) 22

SubStr (string function) 22

SysVar (string function) 23

TableAsChoices (string function) 23

TableKeyLookup (string function) 24

TableLookup (string function) 24

3.1.2. Local String Functions 25

ActiveTopVarName (string function) 25

ActiveVarName (string function) 25

ColumnHeader (string function) 26

ContextVarName (string function) 26

GetTitle (string function) 26

3.1.3. Global Numeric Functions 27

Abs (numeric function) 27

AddMonth (numeric function) ..... 28
Case (numeric function) ..... 28
CumNormal (numeric function) ..... 29
DateToDay (numeric function) ..... 30
DateToMonth (numeric function) ..... 31
DateToYear (numeric function) ..... 31
DMYtoDate (numeric function) ..... 32
DocumentIndex (numeric function) ..... 32
Exp (numeric function) ..... 33
FesExpression (numeric function) ..... 33
Frequency (numeric function) ..... 34
FileExists (numeric function) ..... 34
FV (numeric function) ..... 35
InvNormal (numeric function) ..... 35
IPMT (numeric function) ..... 36
IsValue (numeric function) ..... 37
Length (numeric function) ..... 37
Licensed (numeric function) ..... 38
Ln (numeric function) ..... 38
Max (numeric function) ..... 39
Min (numeric function) ..... 39
MinMax (numeric function) ..... 40
MatrixLookup (numeric function) ..... 40
Now (numeric function) ..... 41
NPER (numeric function) ..... 42
OnER (numeric function) ..... 42
OnERorNA (numeric function) ..... 43
OnNA (numeric function) ..... 44
OnNeg (numeric function) ..... 44
OnNotPos (numeric function) ..... 45
OnNoValue (numeric function) ..... 46
OnZero (numeric function) ..... 46
OnZeroOrNA (numeric function) ..... 47
PMT (numeric function) ..... 47
Pos (numeric function) ..... 48
PPMT (numeric function) ..... 49
PV (numeric function) ..... 49
Rate (numeric function) ..... 50

Round (numeric function) ..... 51
RoundUp (numeric function) ..... 51
SysVar (numeric function) ..... 52
TableKeyLookup (numeric function) ..... 52
TableLookup (numeric function) ..... 53
ThisMonth (numeric function) ..... 54
ThisQuarter (numeric function) ..... 54
ThisYear (numeric function) ..... 54
**3.1.4.** Local Numeric Functions ..... **55**
Count (numeric function) ..... 55
DataEntered (numeric function) ..... 56
DataAvailable (numeric function) ..... 57
DataCalculated (numeric function) ..... 57
DataEnteredInVar (numeric function) ..... 58
DataFoundInVar (numeric function) ..... 59
DateToT (numeric function) ..... 59
DateToYearNum (numeric function) ..... 60
DocumentIsLocked (numeric function) ..... 60
EnteredValueFoundInT (numeric function) ..... 61
Exists (numeric function) ..... 61
Expand (numeric function) ..... 62
ExpandFraction (numeric function) ..... 63
ExpandLevel (numeric function) ..... 64
ExpandGrowth (numeric function) ..... 64
ExpandOriginalValue (numeric function) ..... 65
FindValueT (numeric function) ..... 66
FirstTinFormulaSet (numeric function) ..... 66
FirstTinPeriod (numeric function) ..... 67
FirstTinSheet (numeric function) ..... 67
FirstTinYear (numeric function) ..... 68
FirstValidT (numeric function) ..... 69
FlowCurrencyFactor (numeric function) ..... 69
ForAll (numeric function) ..... 70
!! GetFrac (numeric function) ..... 70
!! GetPoint (numeric function) ..... 72
GetT (numeric function) ..... 73
GetValue (numeric function) ..... 74
GuessTerm (numeric function) ..... 74

IsReadOnly (numeric function) ... 75
HAvg (numeric function) ... 76
HOvr (numeric function) ... 76
HSum (numeric function) ... 77
HValues (numeric function) ... 77
HYearValues (numeric function) ... 78
HYearOvr (numeric function) ... 78
InHiddenTree (numeric function) ... 79
InputRequired (numeric function) ... 79
IRR (numeric function) ... 80
LastHistYear (numeric function) ... 80
LastTinFormulaSet (numeric function) ... 81
LastTinPeriod (numeric function) ... 82
LastTinSheet (numeric function) ... 82
LastTinYear (numeric function) ... 83
LastSheet (numeric function) ... 83
LastValueT (numeric function) ... 84
MaxT (numeric function) ... 84
MaxValueT (numeric function) ... 85
MidYearNum (numeric function) ... 85
MinValueT (numeric function) ... 86
Mut (numeric function) ... 86
MutCalc (numeric function) ... 87
NPV (numeric function) ... 88
NPV2 (numeric function) ... 88
OnEntered (numeric function) ... 89
OnNotEntered (numeric function) ... 90
PeriodInSheet (numeric function) ... 90
PeriodInT (numeric function) ... 91
RelMut (numeric function) ... 91
ScaleFactor (numeric function) ... 92
SelectDescendants (numeric function) ... 92
Sheet (numeric function) ... 92
T (numeric function) ... 93
TisVisibleInSheet (numeric function) ... 93
TsY (numeric function) ... 94
TsPerYear (numeric function) ... 94
TupleCount (numeric function) ... 95

TupleMax (numeric function) ... 95
TupleMin (numeric function) ... 96
TupleSum (numeric function) ... 96
UltCurrencyFactor (numeric function) ... 97
UltYearNum (numeric function) ... 97
ValueT (numeric function) ... 98
ViewScaleFactor (numeric function) ... 98
Visible (numeric function) ... 99
YearInT (numeric function) ... 99
YearNumToDate (numeric function) ... 100
YearToT (numeric function) ... 100
**3.1.5.** Local Variable Functions ... **101**
GetRelVar (var function) ... 101
**3.2. Properties ... 102**
**3.2.1.** Data properties ... **102**
Datatype ... 102
Frequency ... 103
Aggregation ... 103
Unspecified ... 104
Formula ... 104
Formula_notrend ... 105
Formula_trend ... 106
Flipflop ... 106
Flipflop_trend ... 107
Flipflop_notrend ... 107
InputRequired ... 107
!! Data_options ... 108
Neptuple_size ... 108
**3.2.2.** Restriction properties ... **109**
Decimals_maximum ... 109
Digits ... 109
Pattern ... 110
Length ... 110
Range ... 110
**3.2.3.** Event properties ... **111**
Afterinput ... 111
Afterchange ... 111
**3.2.4.** Presentation properties ... **112**

Displaytype ... 112
Style ... 112
Fixed_decimals ... 113
Title ... 114
Hint ... 114
Blanklines ... 114
Top_separator ... 115
Top_blanklines ... 115
Bottom_separator ... 116
Bottom_blanklines ... 116
Locked ... 117
Visible ... 117
Choices ... 117
Link ... 118
Options_title ... 118
Options_notrend ... 119
Options_trend ... 119
Options ... 120
**3.2.5.** Tuple specific properties ... **120**
Tuple_instance_minimum ... 120
Tuple_instance_maximum ... 121
**3.2.6.** Model properties ... **121**
Version ... 121
**3.3. Constants ... 121**
**3.3.1.** Numeric Constants ... **122**
CreationDate (numeric function) ... 122
ER (numeric constant) ... 122
False (numeric constant) ... 122
NA (numeric constant) ... 123
No (numeric constant) ... 123
Off (numeric constant) ... 123
On (numeric constant) ... 124
PM (numeric constant) ... 124
True (numeric constant) ... 124
Yes (numeric constant) ... 125
Title (formulaset constant) ... 125
NoTrend (formulaset constant) ... 125
Trend (formulaset constant) ... 125

User (formulaset constant) ... 126
Sector (formulaset constant) ... 126
Period constant ... 126
!! Single (numeric constant) ... 127
!! Detail (numeric constant) ... 127
**3.4. Operators ... 127**
**3.4.1.** Numeric Operators ... **127**
IsLarger (numeric operator) ... 127
IsLargerEqual (numeric operator) ... 128
IsSmaller (numeric operator) ... 128
IsSmallerEqual (numeric operator) ... 129
IsEqual (numeric operator) ... 129
IsNotEqual (numeric operator) ... 130
Add (numeric operator) ... 130
Subtract (numeric operator) ... 131
Or (numeric operator) ... 131
Multiply (numeric operator) ... 132
Divide (numeric operator) ... 132
Power (numeric operator) ... 133
And (numeric operator) ... 133
Mod (numeric operator) ... 134
Not (numeric operator) ... 134
Implies (numeric operator) ... 135
**Index** ... 137

# 1 Definitions

**Document**

All data is contained in the document.

**Column**

A column is data of a certain period. This data can be aggregated, making a sum of columns, and this results in a virtual column.

**Tuple**

When there are a lot of the same type items or topics. Instead of making multiple codes you can just copy the same code. This is the definition of Tuple.
A tuple can contain a mixture of other data types.

**Period**

A period is a time period containing columns. A period can consist formulas and consist either historical or forecasted data.

**Variable**

A variable is like a row in excel. Every variable must be declared to use a data type. It also must be given an initial value. Every variable has an unique name and can have one or more children.

**Timeline**

A timeline is containing columns.

**References**

A reference is when Var2 is a reference to Var1, both will be changed when Var1 is edited.
Another example is when Var2 is referenced to Var1, that it will be locked when Var1 is being used.
So two referenced variables will do a certain action.

**Formulas**

An expression using functions and variables to calculate valuables.

**Visibility**

This is a property to change the visibility of a certain variable. If the upper level is hidden, lower levels will be hidden as well.

**Locking**

This is a function to lock a certain value so that users can't edit the value shown in the spreadsheet.

### Aggregration

Aggregration defines the way variables are aggregated from a lower level of detail (e.g. month, quarter) to a higher level of detail (e.g. year).

# 2 Scorecard

A scorecard model is a Finan model that models a scorecard. It can be loaded into the K3 scorecard application.The model needs to have certain conditions. The following document explains what conditions.

# Structure

### Q_ROOT

A normal Finan model has a ROOT variable. A scorecard has under this ROOT variable its own root: Q_ROOT. This is where the scorecard starts; variables that are not being used by Q_ROOT, will not be used in the scorecard (but formulas and reports etc can be). The title of Q_ROOT will not be showed. The hint of Q_ROOT will be shown in all screens as a text. The onecol value of the Q_ROOT variable can either be 0 or 1. 0 means that the scorecard is not completed. 1 means the scorecard is completed.

### Variables under Q_ROOT

Q_ROOT contains children folders (variables with subvariables) and normal variables. Folders are shown as scorecard steps (See paragraph scorecard steps (folder under Q_ROOT). Variables are normally not being used, but there are a few special variables that are being used. These variables will be explained here.

### Q_STATUS

This variable defines the status of the scorecard. 0 means the scorecard is still able to be edited. 1 means the scorecard is in its final stage, which means it can not be edited anymore. Other numbers have to be defined still (this will be used for topics such as the 4-eyes principal).

### Q_STARTED_ON

This variable has as input the date and time of when the scorecard is made.

### Q_STARTED_BY

This variable has as input the username of the user who has made the scorecard.

### Q_FINAL_ON

This variable has as input the date and time of when the scorecard has been made final.

### Q_FINAL_BY

This variable has as input the username of the user who has made the scorecard final.

### Scorecard folders (folders under Q_ROOT)

Under the Q_ROOT are folders. Every single folder stands for a step in the scorecard. The

meaning of a step is a screen in the wizard. The title of the 'step'-variable is shown as title of the step. The hint that has to do with the 'step'-variable is shown as text in this step. The onecol value of a 'step'-variable can either be 0 or 1. The definition of 0 is that the step is not completed yet. 1 means the step has been completed. On the last step there is a button 'finalize'. This button takes care of Q_STATUS variable is getting the value 1 status.

**Warning texts for each step**

There can be a special variable under every 'step'-variable. This variable is named 'step'-variable name + _WARNING. The value of this variable will be shown as warning text in the step. The value of this variable can update itself (due to a formula), and this can be used for texts such as 'not all variables are completed yet'. If the value of the variable is empty, or if the variable does not exist, than there is no warning shown.

**Paragraphs for each step**

For each step there is at least one paragraph defined. There has to be defined at least one paragraph. A paragraph is (just as a scorecard step) a folder. Paragraphs are subfolders in the 'step'-folders. When a paragraph has no title, it will not be showed that way. But be alerted; under a 'step'-folder there has to be a minimum of one defined paragraph. The title of the paragraph variable will be used as title of the paragraph. The hint of the paragraph variable will be used as text under the paragraph title.

**Questions per paragraph**

Under every paragraph are the final questions of the scorecard. A question is a variable. The title of these variables are shown as questions. The possibilities on the answers depends on the type of the variable. The hint of the variable is shown as the 'i' symbol, and the hint appears when the user uses his mouse on the 'i' symbol.

**Scorecard step Q_RESULT**

A special folder under Q_ROOT is Q_RESULT. This is the screen with the results of the scorecard. This certain final step is shown after the scorecard is made final (if the value of Q_STATUS is 1). On this page the results will be shown, and there will be a button with 'create report'.

# 3 Reference

Functions

Properties

Constants

Operators

## 3.1 Functions

Global String Functions

Local String Functions

Global Numeric Functions

Local Numeric Functions

Local Variable Functions

### 3.1.1 Global String Functions

ActiveScriptName, AfterStr, BeforeStr, Case, Chr, DateStr, DioStatus, EvaluateAsString, FirstLC, FirstUC, If, Initials, OnNoStr, Str, StrField, SubStr, SysVar, TableAsChoices, TableKeyLookup, TableLookup

#### 3.1.1.1 ActiveScriptName (string function)

This is a Global String Function.

**Syntax**

```
&ActiveScriptName
```

**Parameters**

No parameters. The result is the name of the active script.

### 3.1.1.2 AfterStr (string function)

This is a Global String Function.

**Syntax**

```
&AfterStr(substring,string,N)
```

**Parameters**

- `substring`: a string expression.
- `string`: a string expression.
- `N`: an optional numeric expression. This third optional parameter is introduced in v2.8.6.
- Result: the part of the string after the Nth substring, see examples. The result is an empty string when the Nth "substring" is not found.

**Notes**

See also BeforeStr (string function) and SubStr (string function).

**Example**

```
&AfterStr("abc,"123abc567abc890")="567"
&AfterStr("abc,"123abc567abc890",2)="890"
&AfterStr("abc,"123abc567abc890",3)=""
```

### 3.1.1.3 BeforeStr (string function)

This is a Global String Function.

**Syntax**

```
&BeforeStr(substring,string,N)
```

**Parameters**

- `substring`: a string expression.
- `string`: a string expression.
- `N`: an optional numeric expression. This third optional parameter is introduced in v2.8.6.
- Result: the part of the string before the Nth substring, see examples. The result is an complete string when the Nth "substring" is not found.

**Notes**

See also AfterStr (string function) and SubStr (string function).

**Example**

```
&BeforeStr("abc,"123abc567abc890")="123"
```

```
&BeforeStr("abc,"123abc567abc890",2)="123abc567"
&BeforeStr("abc,"123abc567abc890",3)="123abc567abc890"
```

### 3.1.1.4 Case (string function)

This is a Global String Function.

**Syntax**

```
&Case(expression,[item list])
```
or
```
&Case(expression,[condition:item|condition:item|..])
```

**Parameters**

- `expression`: numeric expression
- `item list`: list of values that will be searched in order to find the value needed. Entries in the list are separated by the "|"-character.
- `condition`: the condition must be met by expression to get the item specified. A condition may start with ">", "<", ">=", or ><=", see the examples.
- `item`: the resulting item when the condition is met.
- Result: one of the items (string), or an empty string when not found.

**Description**

This function can be used to enter a range of conditions in a script without having to make a complex range of nested if-functions. The list will be searched for the expression. The result will depend on the syntax of the list. See the examples.

**Notes**

The brackets "[..]" were introduced in v3.0.0.32. The old syntax with double quotes is still valid but not recommended. The webclient does not accept double quotes around the item list.
See also StrField (string function), Case (numeric function) and Case (cmd).

**Example**

```
&Case(3,[2:"Amsterdam"|3:"Berlin"|1:"Leer"|6:"Zwolle"])="Berlin"
```
This example returns the second entry in the list. The value of expression is 3, which is the second entry, and this returns the value "Berlin". Note the double quote before the closing parenthesis.

```
&Case(15,[<=10:"Amsterdam"|<=20:"Zwolle"])="Zwolle"
```
In this case each entry in the list contains a range of possible values for expression. The second entry contains the range 11 up to and including 20. This includes the value expression;, so the value will be "Zwolle".

```
&Case(10,[5:"Amsterdam"|"Zwolle"])="Zwolle"
```
In this example the last condition fails. An empty condition means "otherwise".

```
&Case(2,[Amsterdam|Berlin|Paris])="Paris"
```
This example returns the third entry with the value "Paris" in the list, since expression is 2. The index of the items in the list is zero-based, so the first item has index 0. NOTE: This form of the case function (no conditions) is available in version 2.8.

### 3.1.1.5 Chr (string function)

This is a Global String Function.

**Syntax**

```
&Chr(N)
```

**Parameters**

- `N`: a numeric expression.
- Result: a string of one character: the Nth ASCII character.

**Example**

```
&Chr(32)=" "
```

### 3.1.1.6 DateStr (string function)

This is a Global String Function.

**Syntax**

```
&DateStr(date number,format code)
```

**Parameters**

- `date number`: a numeric expression with the Date Number.
- `format code`: an optional numeric expression, see Date Format Code, the default value is 2.
- Result: a string with a formatted date, see examples.

**NOTE**

The second parameter (format code) is not optional in the FES!

**Examples**

```
&DateStr(Now) = "05-04-2007"
&DateStr(Now,6) = "donderdag 5 april 2007"
&DateStr(Now,15) = "05-04-2007 23:43"
```

### 3.1.1.7 DioStatus (string function)

Global string function, created in v2.7.10

**Syntax**

```
'&DioStatus("<dio setting>")
```

**Example**

```
.If &DioStatus("DataFolderBrowseMode")="1"
```

**Parameters**

Depends on implementation of the DioManager. Some parameters are:

- `DataBrowseMode (0/1)`
- `AutoSaveSupported (0/1)`
- `ManagerMode (0/1)`
- `DataFolder`
- `BackupFolder`

**See also**

Application (cmd)

### 3.1.1.8 EvaluateAsString (string function)

This is a Global String Function.

**Syntax**

```
&EvaluateAsString(param)
```

**Parameters**

- `param`: a string expression or a numeric expression.
- Result: the expression evaluated as string.

**Notes**

In windows never needed, this function is automatically generated when a string expression starts with "&" in XML files for the webclient.
See also Str (string function).

### 3.1.1.9 FirstLC (string function)

This is a Global String Function.

**Syntax**

```
&FirstLC(string)
```

**Parameters**

- `string`: a string expression.
- Result: the same string where the first character is converted to lower case.

**Notes**

See also FirstUC (string function), BeforeStr (string function) and SubStr (string function).
NOTE: this function returns the same value (no operation) when the display language is german (Deutsch)!

**Example**

```
&FirstLC("settling Groningen")
```
This will result in "settling Groningen"

```
&"$>Saldo<$ "&FirstLC(&Agro_Info1[0])&" $>per cow<$"
```
This means that the definition of the variable depends on the title column of the variable Agra_Info1. When Agro_Info1 in the title column will be edited by the users into 'milk' then this variable will be 'Saldo milk per cow'.

### 3.1.1.10 FirstUC (string function)

This is a Global String Function.

**Syntax**

```
&FirstUC(string)
```

**Parameters**

- `string`: a string expression.
- Result: the same string where the first character is converted to upper case.

**Notes**

See also FirstLC (string function), BeforeStr (string function) and SubStr (string function).
NOTE: this function returns the same value (no operation) when the display language is german (Deutsch)!

**Example**

&FirstUC("settling Groningen")
This will result in: 'settling Groningen'.

### 3.1.1.11 If (string function)

This is a Global String Function.

**Syntax**

```
&If(expression,string1,string2)
```

**Parameters**

- `expression`: a boolean expression.
- `string1`: a string expression.
- `string2`: a string expression.
- Result: string1 when the boolean expression evaluates to true, otherwise string2

**Notes**

See also OnNoStr (string function).

### 3.1.1.12 Initials (string function)

This is a Global String Function.

**Syntax**

```
&Initials("string")
```

**Parameters**

- `string`: a String expression

**Description**

This function returns the initials of a name.

**Example**

```
&Initials("Jan Klaas")="J.K."
&Initials("klaas")="K."
```

## 3.1.1.13 OnNoStr (string function)

This is a Global String Function introduced in version v3.2.26.

**Syntax**

```
&OnNoStr(string1,string2)
```

**Parameters**

- `string1`: a String expression.
- `string2`: a String expression.
- Result: string1 when this is not an empty string, otherwise string2.

NOTE: An "empty string" includes "", "NA", "..." and ".."!!

**Notes**

See also If (string function).

## 3.1.1.14 Str (string function)

This is a global string function.

**Syntax**

```
&Str(expression , width , decimals , numberformat)
```

**Parameters**

- `expression`: a Numeric expression.
- `width`: the width (Numeric expression), default 0. 0 means that no spaces are added. A special case is a negative width and zero decimals: zero's are added before the number up to the absolute value of width.
- `decimals`: the number of decimals (Numeric expression), default -1. -1 means that FINAN adds as many decimals as needed and possible within the width.
- `numberformat`: default 1. Possible values are:
- 0: FINAN data format, period as decimal separator and no thousands separator.
- 1: Windows data format: current setting of windows
- 2: windows data format but without thousands separator
- 3: comma as decimal separator, no thousands separator

**Description**

This string function converts a numeric value to a string.The FES currently does not support the numberformat parameter and always applies the FINAN data format.

## 3.1.1.15 StrField (string function)

This is a Global String Function.

**Syntax**

```
&StrField("item list",expresssion)
```

**Parameters**

- `item list`: list of string values, separated by the "|"- character.
- `expression`: this is the index in the item list (numeric expression)
- Result: the item in the item list with the given index (string). NOTE: the first item has index 1.

**Description**

This function can be used to select an item in a list.

**Notes**

See also Case (string function).

**Example**

```
&StrField("Amsterdam|Berlin|Paris",2)="Berlin"
```
Note that
```
&Case(2,"Amsterdam|Berlin|Paris")="Paris"
```
The index in the case function is zero based! Note that this simple form of the case function is available in v2.8.

## 3.1.1.16 SubStr (string function)

This is a Global String Function.

**Syntax**

```
&SubStr("string",position,width)
```

**Parameters**

- `string`: a String expression
- `position`: a Numeric expression, the position of the first character to be extracted from the string.
- `width`: a Numeric expression, the number of characters to be extracted.

**Description**

This function can be used to extract a substring from a string.

**Notes**

See also Pos (numeric function), Length (numeric function)

**Example**

```
&SubStr("Amsterdam",3,4)="ster"
```

## 3.1.1.17 SysVar (string function)

This is a global string function introduced in v2.8.4.

**Syntax**

```
&SysVar(string expression)
```

**Parameters**

- `string expression`: a string expression that evaluates to a system variable name.
- Result: the value of the system variable as a string

**Notes**

See also SysVar (numeric function).

## 3.1.1.18 TableAsChoices (string function)

This is a global string function introduced in v2.8.8.

**Syntax**

old syntax:
```
&TableAsChoices(table name)
```

new syntax for webclient introduced in 3.2.33:
```
&TableAsChoices(spreadsheet file name,range name)
```

NOTE: it is possible to use the new syntax in FINAN windows. The first parameter is ignored and the second parameter is the table name.

**Parameters**

- `spreadsheet file name`: a String expression that evaluates to the file name of a spreadsheet.
- `range name`: a String expression that evaluates to the name of a range within the spreadsheet
- `table name`: a String expression that evaluates to the name of a lookup table file (without

extension).
- Result: a list of choices in one string separated by "|"s.

**Description**

Use this string command in the Choices Section of Calculation models (fin).

**Notes**

See also TableLookup (string function), TableLookup (numeric function), TableKeyLookup (string function) and TableKeyLookup (numeric function).

**Example**

See Choice specification (fin)

## 3.1.1.19 TableKeyLookup (string function)

This is a global string function introduced in v2.8.8.

**Syntax**

```
&TableKeyLookup(table name,search string)
```

**Parameters**

- `table name`: a string expression that evaluates to the name of a lookup table file (without extension).
- `search string`: a string expression that evaluates tot a key to be looked up in the table.
- Result: the string value found in the lookup table. Use TableKeyLookup (numeric function) for a numeric result.

**Notes**

See also MatrixLookup (numeric function), TableLookup (numeric function), TableLookup (string function) and TableKeyLookup (numeric function).

**Example**

See Lookup table (fib).

## 3.1.1.20 TableLookup (string function)

This is a global string function introduced in v2.8.8.

**Syntax**

```
&TableLookup(table name,search value)
```

**Parameters**

- `table name`: a string expression that evaluates to the name of a lookup table file (without extension).
- `search value`: a numeric expression that evaluates tot a value to be looked up in the table.
- Result: the string value found in the lookup table. Use TableLookup (numeric function) for a numeric result.

**Notes**

See also MatrixLookup (numeric function), TableLookup (numeric function), TableKeyLookup (string function) and TableKeyLookup (numeric function).

**Example**

See Lookup table (fib).

## 3.1.2 Local String Functions

ActiveTopVarName, ActiveVarName, ColumnHeader, ContextVarName, GetTitle

### 3.1.2.1 ActiveTopVarName (string function)

This is a Local String Function.

**Syntax**

```
&ActiveTopVarName
```

**Parameters**

No parameters. The result is the name of the top variable of the currently active screen in the grid. The "top variable" is the variable at one level higher in the presentation tree.

**Notes**

See also ActiveVarName (string function), ContextVarName (string function).

### 3.1.2.2 ActiveVarName (string function)

This is a Local String Function.

**Syntax**

```
&ActiveVarName
```

**Parameters**

No parameters. The result is the name of the variable at the cell pointer in the grid.

**Notes**

See also ActiveTopVarName (string function), ContextVarName (string function).

### 3.1.2.3 ColumnHeader (string function)

This is a Local String Function.

**Syntax**

```
&ColumnHeader(expr1,expr2)
```

**Parameters**

- `Expr1`: a (Numeric expression) representating an Internal column.
- `Expr2` (optional): a (Numeric expression) representing a internal level of detail.
- result: the caption of the column in the grid.

### 3.1.2.4 ContextVarName (string function)

This is a Local String Function.

**Syntax**

```
&ContextVarName
```

**Parameters**

No parameters. The result is the name of the context variable. The "context variable" is the variable in which context this string function is evaluated.

**Notes**

See also ActiveVarName (string function), ActiveTopVarName (string function).

### 3.1.2.5 GetTitle (string function)

This is a Local String Function.

**Syntax**

```
&GetTitle
```

**Parameters**

- `VarName`: a string expression that evaluates to the name of a variable. A name of a variable without quotes is accepted as well.
- Result: the description (in the title column) of the variable.

**Description**

The result is the description of the variable.

**Notes**

This function is implemented in order to facilitate the webversion of FINAN.

```
 &GetTitle(Sales)
```

is equivalent to

```
 &Sales[0]
```

## 3.1.3 Global Numeric Functions

Abs, AddMonth, Case, CumNormal, DateToDay, DateToMonth, DateToYear, DMYtoDate, DocumentIndex, Exp, FesExpression, Frequency, FileExists, FV, InvNormal, IPMT, IsValue, Length, Licensed, Ln, Max, Min, MinMax, MatrixLookup. Now, NPER, OnER, OnERorNA, OnError, OnNA, OnNeg, OnNotPos, OnNoValue, OnZero, OnZeroOrNA, PMT, Pos, PPMT, PV, Rate, Round, RoundUp, SysVar, TableKeyLookup, TableLookup, ThisMonth, ThisQuarter, ThisYear.

### 3.1.3.1 Abs (numeric function)

This is a global numeric function.

**Syntax**

```
Abs(expression)
```

**Description**

Returns the absolute value of expression.

**Parameters**

- `Expression`: this numeric expression evaluates to a number.
- Result: the absolute value of expression.

**Example**

```
Abs(-100)
```

This example returns 100.

### 3.1.3.2 AddMonth (numeric function)

This is a global numeric function.

**Syntax**

```
AddMonth(expression1,expression2)
```

**Description**

This function adds a number of months to a date.

**Parameters**

- `Expression1`: this numeric expression evaluates to a numeric date.
- `Expression2`: a numeric expression that will be evaluated and rounded to an integer value. This is the number of months that will be added (subtracted when negative) to the date (expression1).
- Result: a numeric date.

**Note**

The result will be the last day of a month when the day of expression1 is the last day of a month.
See also DateToDay, DateToMonth, DateToYear, DMYtoDate, Now.

**Example**

```
AddMonth(DMYtoDate(3,2,2006),11)
```
The result of this example is 3-1-2007.

```
AddMonth(DMYtoDate(28,2,2006),-1)
```
The result of this example is 31-1-2007.

### 3.1.3.3 Case (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Case(expression,[item list])
```
or
```
Case(expression,[condition:item|condition:item|..])
```

**Parameters**

- `Expression`: numeric expression
- `Item list` (only in first syntax): list of values that will be searched in to find the value needed. Entries in the list are separated by the "|"-character.
- `Condition`: the condition must be met by expression to get the item specified. A condition may start with ">", "<", ">=", or >=", see the examples.
- `Item`: the resulting value when the condition is met.
- Result: the value of one of the items, or NA when not found.

**Description**

This function is used to enter a range of conditions in a script without having to make a complex range of nested if-functions. The list will be searched for the first parameter expression. The result will depend on the syntax of the list.

**Notes**

The brackets "[..]" were introduced in v3.0.0.32. The old syntax with double quotes is still valid but not recommended. The webclient does not accept double quotes around the item list.
See also Case (string function), Case (cmd) and ReadFile (cmd).

**Example**

```
Case(3,[100|20|40])=40
```
This example returns the third entry with a value of 40 in the list, since expression is 3.

```
Case(3,[2:10|3:7|1:100|6:10])=7
```
This example returns the second entry in the list. The value of expression is 3, which is the second entry, and this returns the value 7 .

```
Case(15,[<=10:3|<=20:5])=2
```
In this case each entry in the list contains a range of possible values for expression. The second entry contains the range 11 up to and including 20. This includes the value expression=15, so the value will be 5.

```
Case(10,[5:1|2])=2
```
No condition is given in the last item, so this means "otherwise". The resulting value is 2.

## 3.1.3.4 CumNormal (numeric function)

This is a global numeric function.

**Syntax**

```
CumNormal(expression)
```

**Description**

Returns the standard normal cumulative distribution function. The distribution has a mean of 0 (zero) and a standard deviation of one.

**Parameters**

- `Expression`: this numeric expression evaluates to a number.
- Result: Returns the standard normal cumulative distribution function. The distribution has a mean of 0 (zero) and a standard deviation of one. It should give the same result as the NORMSDIST function in Excel.

**Notes**

See also InvNormal (numeric function)

**FINAN Windows Delphi Source code**

```
Result := CumNormal(CalcIt(CalcNode1))
```

**Example**

```
CumNormal(0,7)
```
This example returns 0,7580364218341

## 3.1.3.5 DateToDay (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
DateToDay(expression)
```

**Description**

Returns day (number) of the date value given in the expression.

**Parameters**

- `Expression`: this numeric expression represents a date and evaluates to a Date Number.
- Result: Returns the number of the day (1-31). It should give the same result as the DAY function in Excel.

**Notes**

See also DateToMonth (numeric function), DateToYear (numeric function), DMYtoDate (numeric function).

**Example**

```
DateToDay(DMYToDate(3,5,2009))
```
This example returns 3

### 3.1.3.6 DateToMonth (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
DateToMonth(expression)
```

**Description**

Returns month (number) of the date value given in the expression.

**Parameters**

- `Expression`: this numeric expression represents a date and evaluates to a Date Number.
- Result: Returns the number of the month (1-12). It should give the same result as the MONTH function in Excel.

**Notes**

See also DateToDay (numeric function), DateToYear (numeric function).

**Example**

```
DateToMonth(DMYToDate(3,5,2009))
```
This example returns 5

### 3.1.3.7 DateToYear (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
DateToYear(expression)
```

**Description**

Returns year (number) of the date value given in the expression.

**Parameters**

- `Expression`: this numeric expression represents a date and evaluates to a Date Number.
- Result: Returns the year as a whole number. It should give the same result as the YEAR function in Excel.

**Notes**

See also DateToDay (numeric function), DateToMonth (numeric function), DMYtoDate (numeric function).

**Example**

```
DateToYear(DMYToDate(3,5,2009))
```
This example returns 2009

### 3.1.3.8 DMYtoDate (numeric function)

This is a global numeric function.

**Syntax**

```
DMYtoDate(expression1,expression2,expression3)
```

**Description**

This function calculates the numeric date when Day, Month en Year are given.

**Parameters**

- `Expression1`: this numeric expression should evaluate to a day number (1-31).
- `Expression2`: this numeric expression should evaluate to a month number (1-12).
- `Expression1`: this numeric expression should evaluate to a year number (e.g. 2009).
- Result: a numeric date.

**Note**

See also DateToDay, DateToMonth, DateToYear, Now, AddMonth (numeric function).

**Example**

```
DMYtoDate(3,2,2006)
```
The result of this example is 38751.

### 3.1.3.9 DocumentIndex (numeric function)

This is a global numeric function.

**Syntax**

```
DocumentIndex
```

**Description**

Returns the (zero based) index of the current context document in the project. It is the (zero based) index of the upper tab that shows this document. Usually the DocumentIndex refers to the active upper tab (the visible document in the grid). The Document (cmd) is able to change the context document in a script without changing the active document.

**Notes**

See also Document (cmd), DocumentIndex (sysvar), AfterChangeDocument (event).

## 3.1.3.10 Exp (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Exp(expression)
```

**Description**

Returns e (=2,718281828) raised to the power of a given number.

**Parameters**

- `expression`: evaluates to a numeric value.
- result: e (=2,718281828) raised to the power of this number.

**Links**

See alo: Ln (numeric function).

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if X>LN(MaxReal) then
  Result := ER
else
if IsValue(X) then
  Result := EXP(X)
else
  Result := X;
```

## 3.1.3.11 FesExpression (numeric function)

This is a global numeric function.

**Syntax**

```
FesExpression(expression)
```

**Description**

Use `FesExpression` when an expression is not yet accepted by FINAN. It will be exported to model2java xml file.

**Parameters**

- `string expression`: the expression for model2java
- Result: NA (numeric constant)

## 3.1.3.12 Frequency (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Frequency(variable)
```

**Description**

This function returns the numeric value of a variable. This function has been made for the FES.

**Parameters**

- `variable`: the variable name.
- result: the frequency of the variable. Equals to SINGLE for a onecol and to DETAIL for a non-onecol.

## 3.1.3.13 FileExists (numeric function)

This is a global numeric function.

**Syntax**

```
FileExists(string expression)
```

**Description**

Returns true when the file exists.

**Parameters**

- `string expression`: the file specification (path and file name).
- Result: value 1 (true) or value 0 (false).

## 3.1.3.14 FV (numeric function)

This is a global numeric function.

**Syntax**

```
FV(rate,nper,pmt,pv,prenumerando)
```

**Description**

This function returns the future value of an investment based on periodic, constant payments and a constant interest rate. This function is equivalent to the Excel FV function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate per period.
- `nper` (Numeric expression): the total number of payment periods in an investment.
- `pmt` (Numeric expression): is the payment made each period and cannot change over the life of the investment (annuity).
- `pv` (Numeric expression, optional, default 0: is the present value, or the lump-sum amount that a series of future payments is worth now.
- `prenumerando` (Boolean expression, optional, default false): payments are due at the beginning of a period.
- Result: a number, the future value.

**Notes**

See also:
- Rate, PMT, PV, NPER, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.15 InvNormal (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
InvNormal(expression)
```

**Description**

Returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

**Parameters**

- `Expression`: this numeric expression evaluates to a number. This number should be larger than 0 and smaller than 1 (otherwise result is ER).
- Result: Returns the standard normal inverse cumulative distribution function. It should give the same result as the NORMSINV function in Excel.

**Notes**

See also CumNormal (numeric function)

**Algorithm**

See algorithm

**Example**

```
InvNormal(0,4)
```
This example returns -0,253347103

## 3.1.3.16 IPMT (numeric function)

This is a global numeric function.

**Syntax**

```
IPMT(rate,per,nper,pv,fv)
```

**Description**

This function returns the interest payment for a loan based on constant payments and a constant interest rate. This function is equivalent to the Excel IPMT function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate for the loan.
- `per` (Numeric expression): the period in the range 1 to nper.
- `nper` (Numeric expression): the total number of payments.
- `pv` (Numeric expression): is the present value (the total amount that a series of future payments is worth now).
- `fv` (Numeric expression, optional, default 0): is the future value, or a cash balance you want to attain after the last payment is made.
- Result: a number, the interest payment.

NOTE: Postnumerando is assumed!

**Notes**

See also:
- Rate, PV, FV, NPER, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.17 IsValue (numeric function)

This is a global numeric function.

**Syntax**

```
IsValue(numeric expression)
```

**Description**

Returns true when the parameter is a "normal" value, not NA, ER or PM.

**Parameters**

- `numeric expression`: the value to be tested.
- Result: value 1 (true) or value 0 (false).

**Notes**

See also OnNA, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnER, OnERorNA.

## 3.1.3.18 Length (numeric function)

This is a global numeric function.

**Syntax**

```
Length(string)
```

**Parameters**

- `string`: a String expression.
- Result: the length of the string (number of characters).

**Notes**

See also SubStr (string function), Pos (numeric function)

**Example**

```
   Length("monky")
```
returns 5

## 3.1.3.19 Licensed (numeric function)

This is a global numeric function.

**Syntax**

```
Licensed(string expression)
```

**Description**

Returns true when the characters in the string expression are all included in the current license code. This function was introduced in v2.8.6.1.

**Parameters**

- `string expression`: the characters to be tested. Usually the string expression is a string constant with one character, see example. The characters are converted to upper case.
- Result: value 1 (true) or value 0 (false).

**Notes**

See also LicFeatures (sysvar).

## 3.1.3.20 Ln (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Ln(expression)
```

**Description**

Returns the natural logaritm of a number.

**Parameters**

- `expression`: evaluates to a numeric value.
- result: the natural logaritm.

**Links**

See alo: Exp (numeric function).

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsPos(X) then
  Result := LN(X)
else
if IsValue(X) then
```

```
  Result := ER
else
  Result := X;
```

## 3.1.3.21 Max (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Max(expression1,expression2)
```

**Description**

This function returns the largest value, expression1 or expression2.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: evaluates to the second numeric value.
- result: the largest value.

**Notes**

```
Max(NA,-3) = NA
Max(0,NA) = NA
```

**Links**

See alo: Min (numeric function).

**FINAN Windows Delphi source code**

```
Result := Max(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.1.3.22 Min (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Min(expression1,expression2)
```

**Description**

This function returns the smalles value, expression1 or expression2.

**Parameters**

- `expression1`: evaluates to the first numeric value.

- `expression2`: evaluates to the second numeric value.
- result: the smallest value.

**Notes**

```
Min(3,NA) = NA
Min(0,NA) = NA
Min(-3,NA) = -3
```

**Links**

See alo: Max (numeric function).

**FINAN Windows Delphi source code**

```
Result := Min(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.1.3.23 MinMax (numeric function)

This is a global numeric function introduced in v3.0.0.18.

**Syntax**

```
MinMax(Value,Min,Max,OnNoValue)
```

**Parameters**

- `Value`: a numeric expression.
- `Min`: a numeric expression, the lower limit of Value.
- `Max`: a numeric expression, the upper limit of Value.
- `OnNoValue` (optional): a numeric expression, the result when Value is not a value (see IsValue).
- Result: the Value between Min and Max

**Notes**

See also Min (numeric function), Max (numeric function), OnNoValue (numeric function).

## 3.1.3.24 MatrixLookup (numeric function)

This is a global numeric function introduced in v2.8.8.

**Syntax**

```
MatrixLookup(group name,matrix name,key,search value)
```

**Parameters**

- `group name`: the name of a group matrices (e.g. in an Excel spreadsheet). This parameter

is ignored in the FINAN windows version.

- `matrix name`: a String expression that evaluates to the name of a lookup matrix file (without extension).
- `key`: a String expression that evaluates tot the name of a row in the matrix.
- `search value`: a Numeric expression that evaluates tot a value to be looked up in the matrix. It corresponds to the first column where the value of the column header is smaller or equal to this value. NOTE: an additional column is expected in the matrix for the result value when no match is found. See example matrix file.
- Result: the value of the matrix cell which row title equals to key and column title matches search value.

**Specification of matrix in Excel**

Search Options:

- Exact: the row corresponding to the search value
- ExactOrPrevious: the row corresponding to the search value, if no match; the row closest to the search value where row < searchValue (thus, the previous row)
- ExactOrNext: the row corresponding to the search value, if no match; the row closest to the search value where row > searchValue (thus, the next row)
- Previous:
- Next:

**Notes**

See also TableLookup (numeric function), TableKeyLookup (numeric function).
WARNING: this function has another implementation in the webclient.

**Example**

See Lookup matrix (fmf).

## 3.1.3.25 Now (numeric function)

This is a global numeric function.

**Syntax**

```
Now
```

**Description**

Returns the numeric representation (Date Number) of the current date and time. 39366,62555 is the value for 11-10-2007 15.00.

**Notes**

See also DateToDay, DateToMonth, DateToYear, DMYtoDate, AddMonth.

**Parameters**

There are no parameters
- Result: the current date and time as a floating point value.

## 3.1.3.26 NPER (numeric function)

This is a global numeric function.

**Syntax**

```
NPER(rate,pmt,pv,fv,prenumerando)
```

**Description**

This function returns the number of periods for an investment based on periodic constant payments and a constant interest rate. This function is equivalent to the Excel NPER function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate for the loan.
- `pmt` (Numeric expression): the fixed payment made each period.
- `pv` (Numeric expression): is the present value (the total amount that a series of future payments is worth now).
- `fv` (Numeric expression, optional, default 0): is the future value, or a cash balance you want to attain after the last payment is made.
- `prenumerando` (Boolean expression, optional, default false): payments are due at the beginning of a period.
- Result: a number, the number of payment periods.

**Notes**

See also:
- Rate, PV, FV, PMT, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.27 OnER (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnER(expression1,expression2)
OnError(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to ER. This function is useful if expression1 contains a division. A division by zero results in expression2.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is ER.
- result: the first value or the second value.

**Links**

See also IsValue, OnNA, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnERorNA.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsER(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.28 OnERorNA (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnERorNA(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to ER or NA.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is ER or NA.
- result: the first value or the second value.

**Links**

See also IsValue, OnNA, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnER.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsER(X) or IsNA(X) then
  Result := CalcIt(CalcNode2)
else
```

```
  Result := X;
```

## 3.1.3.29 OnNA (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnNA(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to NA.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is NA.
- result: the first value or the second value.

**Links**

See also IsValue, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnER, OnERorNA.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsNA(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.30 OnNeg (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnNeg(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to a negative value. In many cases the same result can be reached by the Max (numeric function), while Max(-100,0) is equal to OnNeg(-100,0). OnNeg can be be used to force an ER for variables that are not allowed to be negative (for example bookvalues).

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is negative.
- result: the first value or the second value.

**Links**

See also IsValue, OnNA, OnNotPos, OnZero, OnZeroOrNA, OnER, OnERorNA.

**Example**

```
OnNeg(Asset1[T-1]-Asset1Depr,ER)
```

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsNeg(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.31 OnNotPos (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnNotPos(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 does not evaluate to a positive value.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is not positive.
- result: the first value or the second value.

**Links**

See also IsValue, OnNA, OnNeg, OnZero, OnZeroOrNA, OnER, OnERorNA.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if not IsPos(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.32 OnNoValue (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnNoValue(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 is not a value. See IsValue (numeric function).
Introduced in v3.0.0.18.

**Parameters**

- `expression1`: evaluates to a numeric value.
- `expression2`: is only evaluated when the first value is not a valid "value".
- result: the first value or the second value.

**Links**

See alo: IsValue (numeric function), OnERorNA (numeric function), OnER (numeric function), OnNA (numeric function), OnNeg (numeric function), OnZero (numeric function), OnZeroOrNA (numeric function).

## 3.1.3.33 OnZero (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnZero(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to zero or PM.

**Parameters**

- `expression1`: a numeric expression (the first value)
- `expression2`: this numeric expression is only evaluated when the first value is (almost) zero or NA.
- result: the value of the first or the second expression.

**Links**

See also IsValue, OnNA, OnNeg, OnNotPos, OnZeroOrNA, OnER, OnERorNA.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsZero(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.34 OnZeroOrNA (numeric function)

This is a Global Numeric Function.

**Syntax**

```
OnZeroOrNA(expression1,expression2)
```

**Description**

This function returns the value expression2 when expression1 evaluates to zero or PM or NA.

**Parameters**

- `expression1`: evaluates to the first numeric value.
- `expression2`: is only evaluated when the first value is (almost) zero or NA or PM.
- result: the first value or the second value.

**Links**

See also IsValue, OnNA, OnNeg, OnNotPos, OnZero, OnER, OnERorNA.

**FINAN Windows Delphi source code**

```
X := CalcIt(CalcNode1);
if IsZeroOrNA(X) then
  Result := CalcIt(CalcNode2)
else
  Result := X;
```

## 3.1.3.35 PMT (numeric function)

This is a global numeric function.

**Syntax**

```
PMT(rate,nper,pv,fv,prenumerando)
```

**Description**

This function returns the payment (interest and repayment) for a loan based on constant payments and a constant interest rate. This function is equivalent to the Excel PMT function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate for the loan.
- `nper` (Numeric expression): the total number of payments.
- `pv` (Numeric expression): is the present value (the total amount that a series of future payments is worth now).
- `fv` (Numeric expression, optional, default 0): is the future value, or a cash balance you want to attain after the last payment is made.
- `prenumerando` (Boolean expression, optional, default false): payments are due at the beginning of a period.
- Result: a number, the payment (interest and repayment) for a loan based on constant payments and a constant interest rate.

**Notes**

See also:
- Rate, PV, FV, NPER, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.36 Pos (numeric function)

This is a global numeric function.

**Syntax**

```
Pos(sub string,main string)
```

**Parameters**

- `sub string`: a String expression. This is the string to be searched in the main string.
- `main string`: a String expression.
- Result: the character position of the first occurence of sub string in the main string. The result is zero when the sub string is not found.

**Notes**

See also SubStr (string function), Length (numeric function)

**Example**

```
Pos("key","monkey")
```
returns 4 ("key" starts at the 4th position of monkey).

## 3.1.3.37 PPMT (numeric function)

This is a global numeric function.

**Syntax**

```
PPMT(rate,per,nper,pv,fv)
```

**Description**

This function returns the payment ("aflossingsdeel") on the principal for a loan based on constant payments and a constant interest rate. This function is equivalent to the Excel PPMT function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate for the loan.
- `per` (Numeric expression): the period in the range 1 to nper.
- `nper` (Numeric expression): the total number of payments.
- `pv` (Numeric expression): is the present value (the total amount that a series of future payments is worth now).
- `fv` (Numeric expression, optional, default 0: is the future value, or a cash balance you want to attain after the last payment is made.
- Result: a number, the payment.

NOTE: Postnumerando is assumed!

**Notes**

See also:

- Rate, PV, FV, NPER, PMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.38 PV (numeric function)

This is a global numeric function.

**Syntax**

```
PV(rate,nper,pmt,fv,prenumerando)
```

**Description**

This function returns the present value of an investment: the total amount that a series of future payments is worth now. This function is equivalent to the Excel PV function, see Excel help.

**Parameters**

- `rate` (Numeric expression): the interest rate per period.
- `nper` (Numeric expression): the total number of payment periods in an investment.
- `pmt` (Numeric expression): is the payment made each period and cannot change over the life of the investment (annuity).
- `fv` (Numeric expression, optional, default 0): is the future value, or a cash balance you want to attain after the last payment is made.
- `prenumerando` (Boolean expression, optional, default false): payments are due at the beginning of a period.
- Result: a number, the present value.

**Notes**

See also:
- Rate, PMT, FV, NPER, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.39 Rate (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
RATE(nper,pmt,pv,fv,prenumerando)
```

**Description**

Returns the interest rate per period of an annuity. RATE is calculated by iteration and can have zero or more solutions. If the successive results of RATE do not converge to within 0.0000001 after 20 iterations, RATE returns the ER error value. This function is equivalent to the Excel RATE function.

**Parameters**

- `nper` (Numeric expression): the total number of payment periods in an annuity.
- `pmt` (Numeric expression): the payment made each period and cannot change over the life of the annuity. Typically, pmt includes principal and interest but no other fees or taxes. NOTE: this number should be negative (cash outflow).
- `pv` (Numeric expression): is the present value (the total amount that a series of future payments is worth now).
- `fv` (Numeric expression, optional, default 0): is the future value, or a cash balance you want to attain after the last payment is made.
- `prenumerando` (Boolean expression, optional, default false): payments are due at the beginning of a period.

- Result: Returns the interest rate per period of the annuity.

**Notes**

See also:
- PMT, PV, FV, NPER, PPMT, IPMT.
- Spreadsheet Annuity Functions
- Finan model Finan Annuity Functions
- Excel help files

## 3.1.3.40 Round (numeric function)

This is a Global Numeric Function.

**Syntax**

```
Round(value,digits)
```

**Description**

This function rounds the value to the given number of digits (default 0).

**Parameters**

- `value`: a numeric expression
- `digits` (optional, default 0): this numeric expression evaluates to the number of digits.
- result: the rounded value.

**Links**

See alo: RoundUp (numeric function).

**Example**

```
Round(2/3,2) = 0.67
```

**FINAN Windows Delphi source code**

```
N := IntMinMax(RealToInt(CalcIt(CalcNode2)),0,MaxDecimals,0,0,0);
Result := RoundReal(CalcIt(CalcNode1),N);
```

## 3.1.3.41 RoundUp (numeric function)

This is a Global Numeric Function.

**Syntax**

```
RoundUp(value,digits)
```

**Description**

This function rounds the value away from zero to the given number of digits (default 0).

**Parameters**

- `value`: a numeric expression
- `digits` (optional, default 0): this numeric expression evaluates to the number of digits.
- result: the rounded value.

**Links**

See alo: Round (numeric function).

**Example**

```
RoundUp(1/3) = 1
RoundUp(1/3,1) = 0.4
RoundUp(-1/3,1) = -0.4
```

## 3.1.3.42 SysVar (numeric function)

This is a global numeric function introduced in v2.8.4.

**Syntax**

```
SysVar(string expression)
```

**Parameters**

- `string expression`: a string expression that evaluates to a system variable name.
- Result: the value of the system variable as a number

**Description**

Normally system variables (specified in scripts between #...#) are evaluated when the script is executed. Sometimes you need a system variable to be evaluated later, e.g. in a visible formula (see Variable (cmd) or an Action (cmd).

**Notes**

See also SysVar (string function).

## 3.1.3.43 TableKeyLookup (numeric function)

This is a global numeric function introduced in v2.8.8.

**Syntax**

```
TableKeyLookup(table name,search string)
```

**Parameters**

- `table name`: a string expression that evaluates to the name of a lookup table file (without extension).
- `search string`: a string expression that evaluates tot a key to be looked up in the table.
- Result: the numeric value found in the lookup table. Use TableKeyLookup (string function) for a string result.

**Notes**

See also MatrixLookup (numeric function), TableLookup (numeric function), TableLookup (string function) and TableKeyLookup (string function).

**Example**

See Lookup table (fib).

## 3.1.3.44 TableLookup (numeric function)

This is a global numeric function introduced in v2.8.8.

**Syntax**

```
TableLookup(table name,search value,interpolation)
```

**Parameters**

- `table name`: a string expression that evaluates to the name of a lookup table file (without extension).
- `search value`: a numeric expression that evaluates to a value to be looked up in the table.
- `interpolation` (optional): a numeric expression, the result is interpolated within the table when true.
- Result: the numeric value found in the lookup table. Use TableLookup (string function) for a string result. NOTE: the result is NA when the search value is NA, same for ER and PM.

**Notes**

See also MatrixLookup (numeric function), TableKeyLookup (numeric function), TableLookup (string function) and TableKeyLookup (string function).

**Example**

See Lookup table (fib).

## 3.1.3.45 ThisMonth (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
ThisMonth
```

**Description**

Returns the current month number.

**Parameters**

- no parameters.
- Result: Returns the number of the current month (1-12), same as DateToMonth(Now)

**Note**

See ThisQuarter (numeric function), ThisYear (numeric function)

## 3.1.3.46 ThisQuarter (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

```
ThisQuarter
```

**Description**

Returns the current month number.

**Parameters**

- no parameters.
- Result: Returns the number of the current quarter (1-4).

**Note**

See ThisMonth (numeric function), ThisYear (numeric function)

## 3.1.3.47 ThisYear (numeric function)

This is a part of the list of global numeric functions.

**Syntax**

`ThisYear`

**Description**

Returns the current year number.

**Parameters**

- no parameters.
- Result: Returns the number of the current year (e.g. 2009).

**Note**

See ThisMonth (numeric function), ThisQuarter (numeric function)

## 3.1.4 Local Numeric Functions

Count, DataEntered, DataAvailable, DataCalculated, DataEnteredInVar, DataFoundInVar, DateToT, DateToYearNum, DocumentIsLocked, EnteredValueFoundInT, Exists, Expand, ExpandFraction, ExpandLevel, ExpandGrowth, ExpandOriginalValue, FindValueT, FirstTinFormulaSet, FirstTinPeriod, FirstTinSheet, FirstTinYear, FirstValidT, FlowCurrencyFactor, ForAll, GetFrac, GetPoint, GetT, GetValue, GuessTerm, IsReadOnly, HAvg, HOvr, HSum, HValues, HYearValues, HYearOvr, InHiddenTree, InputRequired, IRR, LastHistYear, LastTinFormulaSet, LastTinPeriod, LastTinSheet, LastTinYear, LastSheet, LastValueT, MaxT, MaxValueT, MidYearNum, MinValueT, Mut, MutCalc, NPV, NPV2, OnEntered, OnNotEntered, PeriodInSheet, PeriodInT, RelMut, ScaleFactor, SelectDescendants, Sheet, T, TisVisibleInSheet, TsY, TsPerYear, TupleCount, TupleMax, TupleMin, TupleSum, UltCurrencyFactor, UltYearNum, ValueT, ViewScaleFactor, Visible, YearInT, YearNumToDate, YearToT.

### 3.1.4.1 Count (numeric function)

This is a Local Numeric Function.

**Syntax**

```
Count (QuantificationVarName, Variable List, Expression)
```

**Parameters**

- `QuantificationVarName`: variablename bound to this quantification, can be X, Y, or Z. The quantification variablename is to be used in the expression for referring to the variables from the variable list.
- `Variable List`: a selection of variables.
- `Expression`: the expression that determines whether a variable should be counted.
- Result: 0 or 1.

**Description**

This function returns the amount of variables specified in the variable list that satisfy the given expression. In the case of tuples, it counts every instance of each tupleproperty in the selection of variables for which the expression evaluates to true.

**Notes**

See also SelectDescendants (numeric function), ForAll (numeric function).

**Example**

To determine how many mandatory questions need to be filled in, one can write:

```
Count(X, SelectDescendants(Q_ROOT), InputRequired(X) and Not(DataEntered(X))
```

### 3.1.4.2 DataEntered (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DataEntered (VarName,T,TsY,SubMode)
```

**Parameters**

- `VarName`: name of the variable.
- `T`: a Internal column number (Numeric expression).
- `TsY`: the number of columns per year (Numeric expression).
- SubMode:
- 0: function is true when data was entered in `VarName[T,TsY]` itself.
- 1: function is true when data was entered in `VarName[T,TsY]` or at least at one variable on a lower level below `VarName[T,TsY]`.
- 2: function is true when data was entered in `VarName[T,TsY]` or at least at one non-copy variable on a lower level below `VarName[T,TsY]`.
- 3: DEFAULT: function is true when data was entered in `VarName[T,TsY]` or at least at one variable on a lower level below `VarName[T,TsY]` that is part of the summation of `VarName[T,TsY]`.
- Result: 0 or 1.

**Description**

This is a check for data-entry by the user in a specific cell. It returns the value 1 when data has been entered by the user.

**Notes**

- At submode 1 or 2: function can be true when VarName is a locked variable!
- The old function `OVR(VarName,T,TsY)` is equivialent to `DataEntered(VarName,`

```
T,TsY,3)
```

- Old name OVR is still available, the use of the new name is recommented in order to improve the readability of a model!

This function was previously known as OVR
See also DataEnteredInVar (numeric function), EnteredValueFoundInT (numeric function), DataAvailable (numeric function), DataCalculated (numeric function).

**Example**

```
DataEntered(Turnover,LastTinFormulaset(Trend))
```
returns 1 if the variable Turnover has been changed by the user in the last column of the second formulaset (=forecast).

```
IF(DataEntered(Turnover,LastTinFormulaset(Trend)),"This value is user input.
```
This example shows that DataEntered can be used as a boolean expression in an if-command.

### 3.1.4.3 DataAvailable (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DataAvailable (VarName,T)
```

**Parameters**

- `VarName`: name of the variable
- `T`: optional a Internal column number (Numeric expression).
- Result: 0 or 1.

**Description**

This function checks whether data is available for a specific cell. It returns the value 1 when data is available regardless whether it has been entered by the user or calculated by the FES.

**Notes**

See also DataEnteredInVar (numeric function), EnteredValueFoundInT (numeric function), DataAvailable (numeric_function), DataCalculated (numeric function).

### 3.1.4.4 DataCalculated (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DataCalculated (VarName,T)
```

**Parameters**

- `VarName`: name of the variable
- `T`: optional a Internal column number (Numeric expression).
- Result: 0 or 1.

**Description**

This function checks whether calculated data is available for a specific cell. It returns the value 1 when the data for this cell is calculated by the FES.

**Notes**

See also DataEnteredInVar (numeric function), EnteredValueFoundInT (numeric function), DataAvailable_(numeric_function), DataCalculated (numeric function).

### 3.1.4.5 DataEnteredInVar (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DataEnteredInVar (VarName,SubMode)
```

**Parameters**

- `VarName`: name of the variable
- `SubMode`:
- 0: function is true when data was entered in VarName itself.
- 1: function is true when data was entered in VarName or at least at one variable on a lower level below VarName[T,TsY].
- 2: function is true when data was entered in VarName or at least at one non-copy variable on a lower level below VarName[T,TsY].
- 3: DEFAULT: function is true when data was entered in VarName or at least at one variable on a lower level below VarName that is part of the summation of VarName.
- Result: 0 or 1.

**Description**

This function return true (=1) when the user has entered data in some column or in the description.

**Notes**

See also DataEntered (numeric function), DataFoundInVar (numeric function), EnteredValueFoundInT (numeric function)

### 3.1.4.6 DataFoundInVar (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DataFoundInVar (VarName,SubMode)
```

**Parameters**

- `VarName`: name of the variable
- `SubMode`:
- 0: function is true when data is found in VarName itself.
- 1: function is true when data is found in VarName or at least at one variable on a lower level below VarName[T,TsY].
- 2: function is true when data is found in VarName or at least at one non-copy variable on a lower level below VarName[T,TsY].
- 3: DEFAULT: function is true when data is found in VarName or at least at one variable on a lower level below VarName that is part of the summation of VarName.
- Result: 0 or 1.

**Description**

This function returns true (=1) when data is found in some column. The description is ignored!

**Notes**

See also DataEntered (numeric function), DataEnteredInVar (numeric function) , FindValueT (numeric function)

### 3.1.4.7 DateToT (numeric function)

This is a part of the list of local numeric functions.

**Syntax**

```
DateToT(expression,expression)
```

**Description**

This function returns the internal column of the first column (left to right) that refers to the Date Number in the first expression.

**Parameters**

- `Expression`: this numeric expression represents a date and evaluates to a Date Number.
- `Expression` (default 0): this numeric expression represents a period and evaluates to a Period constant. All internal columns in all periods are searched in that case.

- Result: the first internal column found including this date.

**Notes**

See also GetT (numeric function), DMYtoDate (numeric function)

**Example**

```
DateToT(DMYToDate(3,5,2009),1)
```

### 3.1.4.8 DateToYearNum (numeric function)

This is a part of the list of local numeric functions.

**Syntax**

```
DateToYearNum(expression)
```

**Description**

Returns the Year Number given the Date Number in the expression.

**Parameters**

- `Expression`: this numeric expression represents a date and evaluates to a Date Number.
- Result: Returns the Year Number.

**Notes**

See also YearNumToDate (numeric function)

**Example**

```
DateToYearNum(DMYToDate(3,5,2009))
```
This example returns 2009,338888889

**FINAN Windows Delphi source code**

```
Result := DateToYearNum(CalcIt(CalcNode1))
```

### 3.1.4.9 DocumentIsLocked (numeric function)

This is a Local Numeric Function.

**Syntax**

```
DocumentIsLocked
```

**Parameters**

This function has no parameters.

**Description**

The result is 1 (true) when the document is locked, otherwise 0 (false).

**Note**

See also IsReadOnly (numeric function)

## 3.1.4.10 EnteredValueFoundInT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
EnteredValueFoundInT(expression1,expression2)
```

**Parameters**

- `expression1`: first Internal column number.
- `expression2`: (optional) last Internal column number.
- result: a boolean value (1=true or 0=false)..

**Description**

This function is true (value 1) when at least one entered value is found in the Internal columns.

**Note**

Introduced in (v3.0.0.15).
See also DataEnteredInVar (numeric function), DataEntered (numeric function) , FindValueT (numeric function).

**Example**

formula in Formulas Section of Calculation model:

```
ManualChangesInForecast= EnteredValueFoundInT(FirstTinFormulaSet(Trend,Perio
```

## 3.1.4.11 Exists (numeric function)

This is a Local Numeric Function.

**Syntax**

```
Exists (QuantificationVarName, Variable List, Expression)
```

**Parameters**

- `QuantificationVarName`: variablename bound to this quantification, can be X, Y, or Z.

The quantification variablename is to be used in the expression for referring to the variables from the variable list.

- `Variable List`: a selection of variables.
- `Expression`: the expression that should be evaluated for each variable in the specified variable list.
- Result: 0 or 1.

**Description**

This function evaluates the expression for each variable in the variable list and only returns true if at least one of these evaluations result in true. In the case of tuples, it evaluates the expression for each instance of each tupleproperty in the selection of variables.

**Notes**

See also SelectDescendants (numeric function), Count (numeric function).

**Example**

To check whether the user has provided a value values for at least mandatory variables under Q_ROOT one can write the following:

```
Exists(X, SelectDescendants(Q_ROOT), InputRequired(X) implies DataEntered(X)
```

## 3.1.4.12 Expand (numeric function)

This is a Local Numeric Function.

**Syntax**

```
Expand(growthSource,lookbackMethod,growthMin,growthMax,W1, W2, W3)
```

**Description**

The variable in the formula of which this function is used, will be extrapolated including seasonal patterns.

**Parameters**

- `growthSource`: variable used as the source of the growth. Possible values are:
- none (behaviour of ExpandLevel)
- `<variableName>` (behaviour of ExpandFraction)
- self (behaviour of ExpandGrowth)
- `lookbackMethod` (optional): the method used to look at previous columns, default 0. Possible values are:
- bookyear (for behaviour identical to old ExpandGrowth/ExpandLevel)
- year
- column (for mixes of columns with different durations, i.e. IFRS)
- shift

shift when growthSource=variableName (behaviour of ExpandFraction)

- expression1 (optional): minimum growth (e.g. -10%), default unlimited.
- expression2 (optional): maximum growth (e.g. 10%), default unlimited.
- expression3 (optional): weighting factor growth two years ago (default NA).
- expression4 (optional): weighting factor growth three years ago (default NA).
- result: a number, the extrapolated value.

**Note**

See also the old ExpandLevel (numeric function), ExpandFraction (numeric function).
The weighting factors currently do not work correctly in combination with lookbackMethod column.

## 3.1.4.13 ExpandFraction (numeric function)

This is a Local Numeric Function.

**Syntax**

```
ExpandFraction(variable1,variable2,expression)
```

**Description**

variable1 will be extrapolated following the growth path of variable2.

**Parameters**

- `variable1`: the variable to be extrapolated.
- `variable2`: the reference variable.
- `expression` (optional numeric expression): a time shift in order to prevent circular references, default 0 (no time shift).
- result: a number, the extrapolated value.

**Note**

See also ExpandGrowth (numeric function), ExpandLevel (numeric function).

**Example**

```
.Formula Trend
VariableCosts = ExpandFraction(VariableCosts,Sales)
```

**FINAN Windows Delphi source code**

```
if EnableRecursiveCalc then
  VarNode1.CalculateT(CalcT,true);
Shift := RealToInt(CalcIt(CalcNode3));
Result := ExpandFraction(VarNode1,VarNode2,CalcT,CalcTsY,Shift);
```

## 3.1.4.14 ExpandLevel (numeric function)

This is a Local Numeric Function.

**Syntax**

```
ExpandLevel(variable)
```

**Description**

The variable will be extrapolated excluding growth, but including the season pattern. When using the syntax `ExpandLevel(Self)` the result is similar (but not identical!) to `var[T-TsY]`.

**Parameters**

- `variable`: the variable to be extrapolated.
- `expression1` (optional, default NA): weighting factor of value of two years ago, introduced in v3.0.0.34.
- `expression2` (optional ,default NA): weighting factor of value of three years ago, introduced in v3.0.0.34.
- `boolean expression` (optional, default On): When On: the weighting factors are only applied in the first column of a formula set, introduced in v3.0.0.34.
- result: a number, the extrapolated value.

**Note**

See also ExpandGrowth (numeric function), ExpandFraction (numeric function).

**FINAN Windows Delphi source code**

```
Result := ExpandLevel(VarNode1,CalcT,CalcTsY);
```

## 3.1.4.15 ExpandGrowth (numeric function)

This is a Local Numeric Function.

**Syntax**

```
ExpandGrowth(variable,expression1,expression2,expression3,expression4)
```

**Description**

variable will be extrapolated including seasonal patterns.

**Parameters**

- `variable`: the variable to be extrapolated.
- expression1 (optional): minimum growth (e.g. -10%), default unlimited.
- expression2 (optional): maximum growth (e.g. 10%), default unlimited.

- expression3 (optional): weighting factor growth two years ago (default NA).
- expression4 (optional): weighting factor growth three years ago (default NA).
- result: a number, the extrapolated value.

**Note**

See also ExpandLevel (numeric function), ExpandFraction (numeric function).

**FINAN Windows Delphi source code**

```
Result := ExpandAverageGrowth(VarNode1,
                              OnNA(CalcIt(CalcNode2),MinReal),
                              OnNA(CalcIt(CalcNode3),MaxReal),
                              NA,CalcIt(CalcNode4),CalcIt(CalcNode5),
                              CalcT,CalcTsY);
```

## 3.1.4.16 ExpandOriginalValue (numeric function)

This is a local numeric function

**Syntax**

```
ExpandOriginalValue(CostPriceVarName,InvestmentVarName,DepreciationPeriod)
```

**Description**

This function expands the original value (historical cost price) of an asset when the series of investments and the period are given.

**Parameters**

- `CostPriceVarName`: the name of a variable.
- `InvestmentVarName`: the name of a variable.
- `DepreciationPeriod`: numeric expression, the depreciation period in years (see GuessTerm (numeric function)).
- `Result: a numeric value`: the cost price in the current internal column.

**Notes**

See also GuessTerm (numeric function)

**Example**

```
Asset1CostPrice = ExpandOriginalValue(Asset1CostPrice,Asset1Investment,Asset
```

## 3.1.4.17 FindValueT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FindValueT(variable,expression1,expression2,expression3)
```

**Description**

This function returns the Internal column number of the first or last column that has the given value `(expression1)`.

**Parameters**

- `variable`: the variable name.
- expression1: the value to search for.
- expression2: the first Internal column (optional, default value is 1)
- expression3: the last Internal column (optional, default value is MaxT).
- expression4: evaluation to true reverses the search direction such that the result will be the last column with the given value (optional, default value is false).
- result: an Internal column number.

**Note**

NOTE: No level of detail conversion (Quarter, Month etc.)
See also MaxValueT (numeric function), MinValueT (numeric function), EnteredValueFoundInT (numeric function), DataFoundInVar (numeric function).

## 3.1.4.18 FirstTinFormulaSet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FirstTinFormulaset(expression1,expression2,expression3)
```

**Description**

This function is used to calculate the first T (internal column number) in a formula set within a period. The result is 0 (zero) when no such T exists.

**Parameters**

- `expression1` (optional): formula set number (default is the current formula set).
- `expression2` (optional): period number (default is the current period), the value zero indicates "any period".
- `expression3`: (optional), evaluates to zero (false, default) or any other value (true). When

true: the result is the first T in the first whole year that matches the formula set and period. Introduced in v2.8.7.
- result: an internal column number.

**Note**

See also LastTinFormulaSet (numeric function), Formulaset constants, Period constants.

**Example**

```
FirstTinFormulaSet(Trend,MainPeriod)
```
The function returns the first T where the formulaset is Trend.
```
FirstTinFormulaSet(Sector,0)
```
The function returns the first T where the formulaset is Sector.

## 3.1.4.19 FirstTinPeriod (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FirstTinPeriod(expression)
```

**Description**

This function is used to calculate the first T (internal column number) in a period.

**Parameter**

- `expression`: period number (optional, default is the current period).
- result: an internal column number.

**Note**

See also FirstTinFormulaSet (numeric function), LastTinPeriod (numeric function), Period constants.

**Example**

```
FirstTinPeriod(BudgetPeriod)
```
The function returns the first T in the budget period. Note that the constant BudgetPeriod is model-dependent.

## 3.1.4.20 FirstTinSheet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FirstTinSheet(expression)
```

**Description**

This function is used to find the first T (internal column number) in a sheet.

**Parameters**

- `expression`: the index number of the sheet (default the current sheet).
- result: an internal column number of NA when the sheet does not exist.

**Note**

See also LastTinSheet (numeric function).

**Example**

```
FirstTinSheet(1)
```

## 3.1.4.21 FirstTinYear (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FirstTinYear(expression)
```

**Description**

This function is used to find the first T (internal column number) in the same year as the given T (parameter).

**Parameter**

- `expression`: internal column (optional, default is the current internal column), NOTE: this parameter is NOT a year!
- result: an internal column number.

**Note**

See also LastTinYear (numeric function), FirstTinPeriod (numeric function), FirstTinFormulaSet (numeric function), FirstTinSheet (numeric function).

**Example**

```
VoorBetBel= Max(WinstBel[T-TsY,1]/TsY-WinstBel+If(T=FirstTinYear,NA,VoorBetB
BelastbaarBedrCum= HSum(Self,FirstTinYear,T)
```

## 3.1.4.22 FirstValidT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FirstValidT
```

**Description**

This function is used to find the first valid T (internal column number). This is the first T where `ValCheck` is zero or NA.

**Parameters**

This function has no parameters.

- Result: a number, the first valid internal column number.

## 3.1.4.23 FlowCurrencyFactor (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FlowCurrencyFactor(T)
```

**Parameters**

- `T` (optional): internal column number (numeric expression).
- Result: a number.

**Description**

Multiply a data currency value in this internal column in order to a display currency value for this column. Use this factor for flow variables.

**Notes**

NOT YET IMPLEMENTED. It is a dummy function in order to generate models for the Webclient!
See UltCurrencyFactor (numeric function)

## 3.1.4.24 ForAll (numeric function)

This is a Local Numeric Function.

**Syntax**

```
ForAll (QuantificationVarName, Variable List, Expression, [IncludeTupleDefir
```

**Parameters**

- `QuantificationVarName`: variablename bound to this quantification, can be X, Y, or Z. The quantification variablename is to be used in the expression for referring to the variables from the variable list.
- `Variable List`: a selection of variables.
- `Expression`: the expression that should be evaluated for each variable in the specified variable list.
- `IncludeTupleDefinitions`: optional argument (true or false) that specifies whether the expression also should be evaluated for tupledefinitions (that themselves do not contain data).
- Result: 0 or 1.

**Description**

This function evaluates the expression for each variable in the variable list and only returns true if all these evaluations result in true.In the case of tuples, it evaluates the expression for each instance of each tupleproperty in the selection of variables. If the fourth optional parameter `IncludeTupleDefinitions` is true this function also evaluates the expression for the tupledefinition.

**Notes**

See also SelectDescendants (numeric function), Count (numeric function).

**Example**

To check whether the user has provided values for all mandatory variables under Q_ROOT one can write the following:

```
ForAll(X, SelectDescendants(Q_ROOT), InputRequired(X) implies DataEntered(X)
```

## 3.1.4.25 !! GetFrac (numeric function)

This is a local numeric function

**Syntax**

```
GetFrac(FlowVarName,value)
```

### Description

This function is used for term indicators. The first parameter is a flow variable, for example turnover. The second parameter is a total of assets, debtors for example. In this case the debtorsterm is calculated for a fraction of a year (multiply this with 52 to see the weeks)
For example: The easiest case would be a year and a debtors amount being lower as the year turnover. In that case the solution is simple: the debtors amount divided by the year turnover (Value/FlowVarNameValue). In this case we would not need a separate function. In the GetFrac function corrections are made in case the timebase in column (T) is not a year. Also corrections are made when the debtors are a greater amount than the flow variable (this can happen in a column shown per month or per quarter of the year). In this case the GetFrac function searches in older columns to see what time it takes in the flow variable (turnover) to achieve the amount of debtors. The results will be shown in a fraction of a year (can be greater than 1).

### Parameters

- `FlowVarName`: the name of a flow variable.
- `value`: numeric expression, the value of a balance sheet variable.
- Result: a numeric value: the fraction of a year.

### Notes

See also GetPoint (numeric function)

### Example

```
.Variables
PPPP     X     0              Test model (flipflop with term ratios)

PII      FN    1 1            Sales
PII      BN      1            Inventories: GetPoint(Sales,TermRatio/52)

PPI    1 CN    1 1            TermRatio: GetFrac(Sales,Inventories)*52)

.Formulas NoTrend
TermRatio = GetFrac(Sales,Inventories)*52

.Formulas Trend
Sales = ExpandGrowth(Sales,-10%,10%)
Inventories = IF(TermRatio=NA,ExpandLevel(Inventories),GetPoint(Sales,TermRa
TermRatio = IF(DataEntered(Inventories) or TimeAggregated,GetFrac(Sales,Inve
```

### FINAN Windows Delphi source code

```
Result := GetFracValue(VarNode1,CalcIt(CalcNode2),CalcT,CalcTsY);
```

## 3.1.4.26 !! GetPoint (numeric function)

This is a local numeric function

**Syntax**

`GetPoint(FlowVarName,fraction)`

**Description**

This function is used for term indicators. The first parameter is a flow variable, for example turnover. The second parameter is a debtors term as fraction of a year. In this case GetPoint calculates the debtors amount.
For example: The easiest case would be a year and a year fraction smaller than one. In this case the solution is easy: Turnover times year fraction is the debtors amount. Meaning: In case the year base the turnover is 1000, this results in GetPoint(Turnover,1/2)=500.
In this case we would not need a separate function. In the GetPoint function there are corrections done in case the timebase in the column (TsY) is different than a year, same in case the yearfraction is greater than 1/TsY. In this case the GetPoint function searches in older column to find the right value of the debtors amount.
An easy example: The sales volume is 1000 this year and it was 500 last year. This means GetPoint(Omzet,2)=1500. Or in economical terms: if the debtors term is 104 weeks (2 years) than in this example the value on debtors is 1500 (without VAT corrections). GetPoint assures that all fractions (=value of term number) and all time bases are working properly.

**Parameters**

- `FlowVarName`: the name of a flow variable.
- `fraction`: Numeric expression, the fraction of a year.
- Result: a numeric value: the value of a balance sheet variable.

**Notes**

See also GetFrac (numeric function)

**Example**

```
.Variables
PPPP     X     0              Test model (flipflop with term ratios)

PII      FN    1 1            Sales
PII      BN      1            Inventories: GetPoint(Sales,TermRatio/52)

PPI    1 CN    1 1            TermRatio: GetFrac(Sales,Inventories)*52)

.Formulas NoTrend
TermRatio = GetFrac(Sales,Inventories)*52
```

```
.Formulas Trend
Sales = ExpandGrowth(Sales,-10%,10%)
Inventories = IF(TermRatio=NA,ExpandLevel(Inventories),GetPoint(Sales,TermRa
TermRatio = IF(DataEntered(Inventories) or TimeAggregated,GetFrac(Sales,Inve
```

**FINAN Windows Delphi source code**

```
 Result := GetPointValue(VarNode1,CalcIt(CalcNode2),CalcT,
CalcTsY);
```

## 3.1.4.27 GetT (numeric function)

This is a Local Numeric Function.

**Syntax**

`GetT(expression1,expression2,expression3,expression4)`

**Description**

This function is used to find a T a few columns before or after the base T (internal column number). This function finds the correct T even when the previous year has another internal level of detail or is located in another period.

**Parameters**

- `expression1`: the base T (often the current internal column: T).
- `expression2`: the "shift": distance from the base T, measured in multiples of the TsY of the base T. See the examples below.
- `expression3` (optional): period number. This is default the current period (value 0), including the "history period" when this parameter is set in the Period script command in the model.
- `expression4` (optional): the level of detail TsY. This is default the internal level of detail of the base T. Introduced in v3.0.0.21.
- result: an internal column number.

**Note**

FINAN automatically converts "`var[T-n]`" to "`var[GetT(T,-n)]`" (where n is a number).
For example: `var[T-1]` is interpreted as `var[GetT(T,-1)]`. This conversion only takes place when the string "`[T-`" is found. So "`var[T-1]`" is not the same as "`var[0+T-1]`"!!

See also T (numeric function), TsY (numeric function), Period constants.

**FINAN Windows Delphi source code**

```
T := RealToInt(CalcIt(CalcNode1));
Shift := RealToInt(CalcIt(CalcNode2));
Pd := IntMinMax(RealToInt(CalcIt(CalcNode3)),0,MaxPd,0,0,0);
```

```
Result := GetShiftedT(T,0,Shift,Pd);
```

## 3.1.4.28 GetValue (numeric function)

This is a Local Numeric Function.

**Syntax**

```
GetValue(variable,expression1,expression2,lookbackMethod)
```

**Description**

This function returns the numeric value of a variable. This function has been made for the FES.

**Parameters**

- `variable`: the variable name.
- `expression1`: the Internal column
- `expression2`: the Internal level of detail.
- `lookbackMethod` (optional): the method used to look at previous columns, default 0. Currently this argument is only available in the FES. Possible values are:
- bookyear (old default behaviour)
- backward (new default behaviour)
- column (for mixes of columns with different durations, i.e. IFRS)
- result: a value

**Note**

This function is never required in a calculation model for FINAN Windows. It can be represented as a Cell specification.

**Examples**

- `GetValue(Sales,4,1,Bookyear) is the same as Sales[4,1]`
- `Sales[T-TsY,1] is the same as GetValue(Sales,GetT(T,-TsY),1,Bookyear)`

## 3.1.4.29 GuessTerm (numeric function)

This is a local numeric function

**Syntax**

```
GuessTerm(BookValueVarName,DepreciationVarName,LowerLimit,UpperLimit,Default
```

**Description**

This function guesses the depreciation period in years when only the bookvalue and the depreciation is given.

The magic formula is:
`Result = 2*BookValue/Depreciation + 1`
BookValue and Depreciation of the last whole year in the NoTrend (formulaset constant).
See the spreadsheet "GuessTerm toverformule.xls" for details about the formula.

**Parameters**

- `BookValueVarName`: the name of a book value variable (balance sheet).
- `DepreciationVarName`: the name of a depreciation variable (profit and loss account).
- `LowerLimit`: numeric expression, the minimum value of the result.
- `UpperLimit`: numeric expression, the maximimum value of the result.
- `DefaultValue`: numeric expression, the default value of the result (e.g. when bookvalue or depreciation has no value).
- Result: a numeric value: the depreciation period in years

**Notes**

See also ExpandOriginalValue (numeric function)

**Example**

```
Asset1Term = GuessTerm(Asset1BookValue,Asset1Depreciation,2,8,4)
```

**FINAN Windows Delphi source code**

```
Result := GuessTerm(VarNode1,VarNode2,CalcIt(CalcNode3),CalcIt(CalcNode4),Ca
```

## 3.1.4.30 IsReadOnly (numeric function)

This is a Local Numeric Function.

**Syntax**

```
IsReadOnly
```

**Parameters**

This function has no parameters.

**Description**

The result is 1 (true) when the document is readonly (in the database), otherwise 0 (false).

**Note**

The result is not influenced by the status of the Locked parameter in Document (cmd)!
See DocumentIsLocked (numeric function)

## 3.1.4.31 HAvg (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HAvg(variable,expression1,expression2)
```

**Description**

The average of values of variable will be calculated in a range of internal columns. This calculation ignores the value NA (numeric constant), while the values zero and PM (numeric constant) will be taken into account.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the average number.

**Note**

This function does not recognize differences in the level of detail (year - quarter etc.)!
See also HSum (numeric function), MaxT (numeric function).

## 3.1.4.32 HOvr (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HOvr(variable,expression1,expression2)
```

**Description**

The number of user-entered values of variable will be counted in a range of internal columns.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the number of entered values in this range.

**Note**

See also DataEntered (numeric function), HYearOvr (numeric function), HValues (numeric function), HSum (numeric function).

## 3.1.4.33 HSum (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HSum(variable,expression1,expression2)
```

**Description**

The sum of values of variable will be calculated in a range of internal columns.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the calculated sum.

**Note**

See also HAvg (numeric function), MaxT (numeric function).

## 3.1.4.34 HValues (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HValues(variable,expression1,expression2)
```

**Description**

The number of values (unequal to NA (numeric constant)) of variable will be counted in a range of internal columns.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the number of entered values in this range.

**Note**

See also HYearValues (numeric function), HOvr (numeric function), HSum (numeric function).

## 3.1.4.35 HYearValues (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HYearValues(variable,expression1,expression2)
```

**Description**

The number of values (unequal to NA (numeric constant)) of variable will be counted in a range of internal columns. The only difference with the HValues (numeric function) is that only one value per year will be counted.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the number of entered values in this range.

**Note**

See also HValues (numeric function), HOvr (numeric function), HSum (numeric function).

## 3.1.4.36 HYearOvr (numeric function)

This is a Local Numeric Function.

**Syntax**

```
HYearOvr(variable,expression1,expression2)
```

**Description**

The number of user-entered values of variable will be counted in a range of internal columns. The only difference with the HOvr (numeric function) is that only one entered value per year will be counted.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column.
- `expression2`: the last internal column.
- result: the number of entered values (max one per year counted) in this range.

**Note**

See also DataEntered (numeric function), HOvr (numeric function), HYearValues (numeric

function), HSum (numeric function).

## 3.1.4.37 InHiddenTree (numeric function)

This is a Local Numeric Function.

**Syntax**

```
InHiddenTree(VarName)
```

**Parameters**

- `VarName`: name of the variable
- Result: 0 or 1.

**Description**

This is a check whether the variable is in a hidden tree.

**Notes**

Use Visible (numeric function) in order to check whether the variable itself is visible!!

## 3.1.4.38 InputRequired (numeric function)

This is a Local Numeric Function.

**Syntax**

```
InputRequired(VarName,T,TsY)
```

**Parameters**

- `VarName`: name of the variable
- `T` (optional): a internal column number (numeric expression), default the current column.
- `TsY` (optional): the internal level of detail number (numeric expression), default the TsY of the current column.
- Result: 0 or 1.

**Description**

This is a check whether the user is required to provide a value for a specific cell. It returns the value 1 when data is required.

**Notes**

Third parameter (TsY) not yet available in FES.
See Visible (numeric function), Locked (numeric function), Valid (numeric function)

## 3.1.4.39 IRR (numeric function)

This is a local numeric function

**Syntax**

```
IRR(CashFlowVarName,StartT,EndT)
```

**Description**

This function calculates the internal rate of return of a series of cash flows. The IRR function in Excel is similar.

**Parameters**

- `CashFlowVarName`: the name of a cash flow variable.
- `StartT` (optional, default 1): numeric expression, the first internal column of the cash flow variable to be used in the calculation.
- `EndT` (optional, default the last T in the current period): numeric expression, the last internal column of the cash flow variable to be used in the calculation.
- Result: a numeric value: the internal rate of return.

**Notes**

A time line is created between StartT and EndT, so FirstT and LastT don't need to be member of the same period (since v3.1.25)!
See also NPV (numeric function), NPV2 (numeric function)

**FINAN Windows Delphi source code**

```
Result := IRR(VarNode1,RealToInt(CalcIt(CalcNode2)),RealToInt(CalcIt(CalcNoc
```

## 3.1.4.40 LastHistYear (numeric function)

This is a Local Numeric Function.

**Syntax**

```
LastHistYear
```

**Description**

This function restores last functional year.

**Parameters**

This function has no parameters.

- Result: a number, the last whole year in the first formula set of the first period.

**Note**

See also YearInT (numeric function).

## 3.1.4.41 LastTinFormulaSet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
LastTinFormulaset(expression1,expression2,expression3)
```

**Description**

This function is used to calculate the last T (Internal column number) in a Formula set within a Period. The result is 0 (zero) when no such T is found.

**Parameters**

- `expression1`: formula set number (optional, default is the current formula set).
- `expression2`: period number (optional, default is the current period), the value zero indicates "any period".
- `expression3`: (optional), evaluates to zero (false, default) or any other value (true). When true: the result is the last T in the last whole year that matches the formula set and period.
- Result: an Internal column number.

**Note**

See also FirstTinFormulaSet (numeric function), FirstTinPeriod (numeric function), Formulaset constants, Period constants.The FES does not support the third parameter!

**Example**

`LastTinFormulaSet(NoTrend,MainPeriod)`
The function returns the last T where the formulaset is NoTrend within the MainPeriod.

`LastTinFormulaSet(NoTrend,0)`
The function returns the last T where the formulaset is NoTrend.

`LastTinFormulaSet(NoTrend,MainPeriod,1)`
The function returns the last T in the last whole year where the formulaset is NoTrend within the MainPeriod.

## 3.1.4.42 LastTinPeriod (numeric function)

This is a Local Numeric Function.

**Syntax**

`LastTinPeriod(expression)`

**Description**

This function is used to calculate the last T (internal column number) in a period.

**Parameter**

- expression (optional): period number (default is the current period).
- result: a number, the number of the last internal column in the period.

**Note**

See also FirstTinPeriod (numeric function), LastTinFormulaSet (numeric function), Period constants.

**Example**

`LastTinPeriod(MainPeriod)`
The function returns the last T in the main period. Note that the constant MainPeriod is model-dependent.

## 3.1.4.43 LastTinSheet (numeric function)

This is a Local Numeric Function.

**Syntax**

`LastTinSheet(expression)`

**Description**

This function is used to find the last T (internal column number) in a sheet.

**Parameters**

- `numeric expression`: the index number of the sheet (default the current sheet).
- result: an internal column number of NA when the sheet does not exist.

**Note**

See also FirstTinSheet (numeric function).

**Example**

```
LastTinSheet(1)
```

## 3.1.4.44 LastTinYear (numeric function)

This is a Local Numeric Function.

**Syntax**

```
LastTinYear(expression)
```

**Description**

This function is used to find the last T (internal column number) in the same year as the given T (parameter).

**Parameter**

- `expression`: internal column (optional, default is the current internal column), NOTE: this parameter is NOT a year!
- result: an internal column number.

**Note**

See also FirstTinYear (numeric function), LastTinPeriod (numeric function), LastTinFormulaSet (numeric function), LastTinSheet (numeric function).

## 3.1.4.45 LastSheet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
LastSheet
```

**Description**

Returns the index number of the last sheet.

**Notes**

The first sheet has index 1. In the case where no sheet is specified, the grid will display all columns and the sheet index is 0.
See also Sheet (cmd), Sheet (numeric function), Sheet (sysvar), LastSheet (sysvar).

## 3.1.4.46 LastValueT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
LastValueT(variable,expression1,expression2)
```

**Description**

This function returns the internal column number of the last column that has a value (unequal to NA) between the given two internal columns.

**Parameters**

- `variable`: the variable name.
- `numeric expression1`: the first internal column (optional, default value is 1)
- `numeric expression2`: the last internal column (optional, default value is MaxT).
- result: an internal column number.

**Note**

See also FindValueT (numeric function), MaxValueT (numeric function), MinValueT (numeric function).

## 3.1.4.47 MaxT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
MaxT
```

**Description**

This function is used to find the last T (internal column number). This last T is part of the last period. There is one list of T's numbered from one to MaxT. MaxT is the last T within the last period. MaxT increases when the internal level of detail (year, quarter, month...) increases. Note that the external level of detail (as presented on the screen) is not related to MaxT.

**Parameters**

This function has no parameters.

- Result: a number, the last internal column number.

**Note**

See also LastTinPeriod (numeric function), LastTinFormulaSet (numeric function).

## 3.1.4.48 MaxValueT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
MaxValueT(variable,expression1,expression2)
```

**Description**

This function returns the internal column number of the column that has the largest value between the given two internal columns.

**Parameters**

- `variable`: the variable name.
- `expression1`: the first internal column (optional, default value is 1)
- `expression2`: the last internal column (optional, default value is MaxT).
- result: an internal column number.

**Note**

See also FindValueT (numeric function), MinValueT (numeric function).

## 3.1.4.49 MidYearNum (numeric function)

This is a part of the list of local numeric functions.

**Syntax**

```
MidYearNum(T,TsY)
```

**Description**

Returns the Year Number with fraction in the middle of a specified column.

**Parameters**

- `T`: a numeric expression representing an internal column.
- `TsY` (optional, default the TsY of column T): the internal level of detail.
- Result: The Year Number in the middle of the time period represented by T and TsY

**Note**

This function can be useful for discounting flow-values.
See also UltYearNum (numeric function), DateToYearNum (numeric function)

**FINAN Windows Delphi source code**

```
if CalcNode1=nil
then
  Result := MidYearNum(CalcT,CalcTsY)
else
begin
  T := RealToInt(CalcIt(CalcNode1));
  TsY := IntMinMax(RealToInt(CalcIt(CalcNode2)),0,MaxMaxTsY,0,0,0);
  if (T<1) or (T>MaxT) then
    Result := NA
   else
     Result := MidYearNum(T,TsY);
end;
```

## 3.1.4.50 MinValueT (numeric function)

This is a Local Numeric Function.

**Syntax**

`MinValueT(variable,expression1,expression2)`

**Description**

This function returns the internal column number of the column that has the smallest value (unequal to NA and PM) between the given two internal columns.

**Parameters**

- `variable`: the variable name.
- `numeric expression1`: the first internal column (optional, default value is 1)
- `numeric expression2`: the last internal column (optional, default value is MaxT).
- result: an internal column number.

**Note**

See also FindValueT (numeric function), MaxValueT (numeric function).

## 3.1.4.51 Mut (numeric function)

This is a Local Numeric Function, introduced in v2.7.22

**Syntax**

`Mut(variable)`

**Description**

This function is used to calculate the change relative to the previous Internal column.

**Parameter**

- `variable`: the name of a variable.
- result: the mutation between the value in the current Internal column and the previous Internal column.

**Notes**

See also Aggregation code with character "M". This is a better solution but not yet possible in webclient.

**FINAN Windows Delphi source code**

```
if MutCalc(CalcT,CalcTsY) then
begin
if EnableRecursiveCalc then
  VarNode1.CalculateT(CalcT,true);
Result := VarNode1.GetValue(CalcT,CalcTsY,Scaled,Cum,agPointMut);
end;
```

## 3.1.4.52 MutCalc (numeric function)

This is a Local Numeric Function.

**Syntax**

```
MutCalc(expression)
```

**Parameters**

- `expression` (optional): the internal column number T (default the current T).
This function is often used without parameter.

**Description**

This function results to `1 (=true)` or `NA (=false)`. The result is true when it is allowed to calculate a mutation (a cash flow between two balance values) at this column. MutCalc is true for most T's except the first T of a period when the first year in that period is not the year in which the organization was founded ("`.MutCalcInFirstCol Yes`" in the data script).

**Note**

See also the HistoryPeriod parameter of Period (cmd).

## 3.1.4.53 NPV (numeric function)

This is a local numeric function

**Syntax**

```
NPV(Rate,CashFlowVarName,StartT,EndT,ValuationDate)
```

**Description**

This function calculates the net present value of a series of cash flows. The NPV function in Excel is similar.

**Parameters**

- `Rate`: a numeric expression, this is the discount rate.
- `CashFlowVarName`: the name of a cash flow variable.
- `StartT` (optional, default 1): numeric expression, the first internal column of the cash flow variable to be used in the calculation.
- `EndT` (optional, default the last T in the current period): numeric expression, the last internal column of the cash flow variable to be used in the calculation.
- `ValuationDate` (optional, default the middle of whole year before StartT): numeric expression, a Date Number.
- Result: a numeric value: net present value.

**Notes**

A time line is created between StartT and EndT, so FirstT and LastT don't need to be member of the same period (since v3.1.25)!

See also IRR (numeric function), NPV2 (numeric function)

**FINAN Windows Delphi source code**

```
Result := NPVclassic(CalcIt(CalcNode1),VarNode2,RealToInt(CalcIt(CalcNode3)
```

## 3.1.4.54 NPV2 (numeric function)

This is a local numeric function

**Syntax**

```
NPV2(RateVarName,CashFlowVarName,StartT,EndT,ValuationDate)
```

**Description**

This function calculates the net present value of a series of cash flows. The difference with the NPV function is that the discount rate is a time series (variable), not a constant.

**Parameters**

- `RateVarName`: the name of a variable with interest rates.
- `CashFlowVarName`: the name of a cash flow variable.
- `StartT` (optional, default 1): numeric expression, the first internal column of the cash flow variable to be used in the calculation.
- `EndT` (optional, default the last T in the current period): numeric expression, the last internal column of the cash flow variable to be used in the calculation.
- `ValuationDate` (optional, default the middle of whole year before StartT): numeric expression, a Date Number.
- Result: a numeric value: net present value.

**Notes**

See also IRR (numeric function), NPV (numeric function)

**FINAN Windows Delphi source code**

```
Result := NPVwithVariableRate(VarNode1,VarNode2,RealToInt(CalcIt(CalcNode3)
```

## 3.1.4.55 OnEntered (numeric function)

This is a Local Numeric Function. Introduced in v3.2.35

**Syntax**

```
OnEntered(variable,expression)
```

**Description**

This function returns the value expression when variable is entered in current column context.

**Parameters**

- `variable`
- `expression`: is only evaluated when the cell is entered.
- result: the value of the variable when the variable is not entered in the current column context, otherwise the expression.

**Links**

See also IsValue, OnNotEntered, OnNA, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnERorNA, OnER.

## 3.1.4.56 OnNotEntered (numeric function)

This is a Local Numeric Function. Introduced in v3.2.35

**Syntax**

```
OnNotEntered(variable,expression)
```

**Description**

This function returns the value expression when variable is not entered in the current column context.

**Parameters**

- `variable`
- `expression`: is only evaluated when the cell is entered.
- result: the value of the variable when the variable is entered in the current column context, otherwise the expression.

**Links**

See also IsValue, OnEntered, OnNA, OnNeg, OnNotPos, OnZero, OnZeroOrNA, OnERorNA, OnER.

## 3.1.4.57 PeriodInSheet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
PeriodInSheet(expression)
```

**Description**

This function is used to find the most important period in a sheet.

**Parameters**

- `numeric expression` (optional): the index number of the sheet (default the current sheet).
- result: an period number or NA when the period does not exist.

**Note**

See also PeriodInT (numeric function), Sheet (cmd) (Period option).

WARNING: using this function in formulas of the calculation model is very tricky. PeriodInSheet results the period number of the current visible sheet, the formulas are not

automatically recalculated when the user or a script select another sheet! PeriodInT(T) might be an alternative.

## 3.1.4.58 PeriodInT (numeric function)

This is a Local Numeric Function.

**Syntax**

`PeriodInT(expression)`

**Description**

This function is used to find the period number at an internal column T.

**Parameters**

- `numeric expression` (optional): an internal column number (default the internal column that is calculated now).
- result: an period number or NA when the internal column is out of range.

**Note**

See also PeriodInSheet (numeric function)

## 3.1.4.59 RelMut (numeric function)

This is a Local Numeric Function, introduced in v3.0.0.18

**Syntax**

`RelMut(variable)`

**Description**

This function is used to calculate the relative change in a variable.

**Parameter**

- `variable`: the name of a variable.
- result: the relative mutation between the value in the current internal column and the previous internal column.

**Notes**

See also Mut (numeric function)

This function prevents the situation that with a copy-variable in grow-mode the calculation goes on. For example with an indicative rating.

## 3.1.4.60 ScaleFactor (numeric function)

This is a Local Numeric Function.

**Syntax**

```
ScaleFactor
```

**Description**

Its result is almost always the value 1 because most internal calculations are performed in the data scale and data currency, not in the view scale and view currency as in version 2.4.
See also Scale (cmd), ScriptScale (cmd), Decimals (cmd), Scale (sysvar), ViewScaleFactor (numeric function)

## 3.1.4.61 SelectDescendants (numeric function)

This is a Local Numeric Function.

**Syntax**

```
SelectDescendants (VarName, [StopBeforeVarName])
```

**Parameters**

- `VarName`: name of the variable
- `StopBeforeVarName`: optional parameter

**Description**

Returns a list containing all children, grandchildren, etc.. of the specified variable.

**Notes**

See also ForAll (numeric function), Count (numeric function).

## 3.1.4.62 Sheet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
Sheet
```

**Description**

Returns the index number of the currently visible sheet.

**Parameters**

none

**Notes**

The first sheet has index 1. In the case where no sheet is specified, the grid will display all columns and the sheet index is 0.
See also Sheet (cmd), LastSheet (numeric function), Sheet (sysvar), LastSheet (sysvar).

## 3.1.4.63 T (numeric function)

This is a Local Numeric Function.

**Syntax**

T

**Parameters**

This function has no parameters.

**Description**

This function is used to find the current T (internal column number).

**Note**

See also GetT (numeric function), TsY (numeric function), MaxT (numeric function).
!!!!! Use ValueT(T) in FES formulas!!!!!!

## 3.1.4.64 TisVisibleInSheet (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TisVisibleInSheet(expression1,expression2)
```

**Parameters**

- `expression1`: an internal column number.
- `expression2` (optional): a sheet number, default the current sheet.
- result: a boolean value (1=true or 0=false)..

**Description**

This function is true (value 1) when the T (internal column number) is visible in the given sheet.

**Note**

Introduced in (v3.0.0.15).
See Visible (numeric function)

## 3.1.4.65 TsY (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TsY(expression)
```

**Description**

This function is used to find the current internal level of detail.

**Parameters**

- `expression` (optional): the internal column number T. (default the current T).
- Result: the internal level of detail. This result can have the following values:
- 1 (years)
- 2 (half years)
- 4 (quarters)
- 12 (months)
- 13 (4-week periods)
- 52 (weeks)

This function is often used without parameter.

**Note**

See also T (numeric function), MaxT (numeric function).

## 3.1.4.66 TsPerYear (numeric function)

Same as TsY

**Syntax**

```
TsPerYear(expression)
```

**Description**

This function is used to find the current internal level of detail.

**Parameters**

- `expression` (optional): the internal column number T. (default the current T).
- Result: the internal level of detail. This result can have the following values:
- 1 (years)
- 2 (half years)
- 4 (quarters)
- 12 (months)
- 13 (4-week periods)
- 52 (weeks)

This function is often used without parameter.

**Note**

See also T (numeric function), MaxT (numeric function).

## 3.1.4.67 TupleCount (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TupleCount(variable)
```

**Parameters**

- `variable`: the name of a tuple property variable
- result: the amount of tuple instances created for this tuple property. NOTE: result is always NA in FINAN windows.

**Notes**

See also TupleSum (numeric function), TupleMin (numeric function), TupleMax (numeric function)

## 3.1.4.68 TupleMax (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TupleMax(variable)
```

**Parameters**

- `variable`: the name of a tuple property variable
- result: the highest value of this property for all instances of the tuple definition to which the property belongs. NOTE: result is always NA in FINAN windows.

**Notes**

See also TupleCount (numeric function), TupleMin (numeric function), TupleSum (numeric function)

## 3.1.4.69 TupleMin (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TupleMin(variable)
```

**Parameters**

- `variable`: the name of a tuple property variable
- result: the smallest value of this property for all instances of the tuple definition to which the property belongs. NOTE: result is always NA in FINAN windows.

**Notes**

See also TupleCount (numeric function), TupleSum (numeric function), TupleMax (numeric function)

## 3.1.4.70 TupleSum (numeric function)

This is a Local Numeric Function.

**Syntax**

```
TupleSum(variable)
```

**Parameters**

- `variable`: the name of a tuple property variable
- result: the calculated sum of this property for all instances of the tuple definition to which the property belongs. NOTE: result is always NA in FINAN windows.

**Notes**

See also TupleCount (numeric function), TupleMin (numeric function), TupleMax (numeric function)

## 3.1.4.71 UltCurrencyFactor (numeric function)

This is a Local Numeric Function.

**Syntax**

```
FlowCurrencyFactor(T)
```

**Parameters**

- `T` (optional): internal column number (numeric expression).
- Result: a number.

**Description**

Returns the ULTIMO exchange rate between the base and the view currency for the current year (column)
Multiply a data currency value in this internal column in order to a display currency value for this column. Use this factor for balance variables.

**Notes**

NOT YET IMPLEMENTED. It is a dummy function in order to generate models for the Webclient!
See FlowCurrencyFactor (numeric function)

## 3.1.4.72 UltYearNum (numeric function)

This is a part of the list of local numeric functions.

**Syntax**

```
UltYearNum(T,TsY)
```

**Description**

Returns the Year Number with fraction at the end of a specified column.

**Parameters**

- `T`: a numeric expression representing an internal column.
- `TsY` (optional, default the TsY of column T): the internal level of detail.
- Result: The Year Number at the end of the time period represented by T and TsY

**Note**

This function can be useful for discounting flow-values.
See also MidYearNum (numeric function), DateToYearNum (numeric function)

**FINAN Windows Delphi source code**

```
if CalcNode1=nil
then
  Result := UltYearNum(CalcT,CalcTsY)
else
begin
  T := RealToInt(CalcIt(CalcNode1));
  TsY := IntMinMax(RealToInt(CalcIt(CalcNode2)),0,MaxMaxTsY,0,0,0);
  if (T<1) or (T>MaxT) then
    Result := NA
   else
     Result := UltYearNum(T,TsY);
end;
```

## 3.1.4.73 ValueT (numeric function)

This is a Local Numeric Function.

**Syntax**

`ValueT(expression)`

**Description**

This is a dummy function, the result of the function is the value of the expression. We need this function in de FES where T cannot be used (T is a column object in the FES). In the FES this function converts an internal column object to an integer value, the internal column number.

**Parameters**

- `expression`
- Result: a the value of the parameter.

**Notes**

See also T (numeric function)

## 3.1.4.74 ViewScaleFactor (numeric function)

This is a Local Numeric Function.

**Syntax**

`ViewScaleFactor`

**Description**

The result is the scale as displayed in the grid (1, 1000 or 1000000)
See also Scale (cmd), ScriptScale (cmd), Decimals (cmd), Scale (sysvar), ScaleFactor (numeric function)

## 3.1.4.75 Visible (numeric function)

This is a Local Numeric Function.

**Syntax**

```
Visible(VarName,T,TsY)
```

**Parameters**

- `VarName`: name of the variable
- `T` (optional): a internal column number (numeric expression), default the current column.
- `TsY` (optional): the internal level of detail number (numeric expression), default the TsY of the current column.
- Result: 0 or 1.

**Description**

This is a check whether the cell is visible. It returns the value 1 when the cell is visible.

**Notes**

Second parameter (T) and third parameter (TsY) not yet available in FES.
Use InHiddenTree (numeric function) in order to check whether the variable is visible in the tree!!
See InHiddenTree (numeric function), InputRequired (numeric function), Locked (numeric function), Valid (numeric function)

## 3.1.4.76 YearInT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
YearInT(expression)
```

**Description**

This function is used to find the year (e.g. 2009) at an internal column. Returns the last year when two or more years are defined within an internal column (not yet possible in FINAN Windows).

**Parameters**

- `expression` (optional): an internal column number (default the current internal column: T).
- Result: a four digit year number.

**Note**

Result is NA when the expression evaluates to a value smaller than 1 or larger than MaxT
See also LastHistYear (numeric function).

## 3.1.4.77 YearNumToDate (numeric function)

This is a part of the list of local numeric functions.

**Syntax**

```
YearNumToDate(expression)
```

**Description**

Returns the Date Number given the Year Number in the expression.

**Parameters**

- `Expression`: this numeric expression represents a Year Number.
- Result: Returns the Date Number.

**Notes**

See also DateToYearNum (numeric function)

**Example**

```
YearNumToDate(2009.5)
```
This example returns 39995 (July 1st 2009)

**FINAN Windows Delphi source code**

```
Result := DateToYearNum(CalcIt(CalcNode1))
```

## 3.1.4.78 YearToT (numeric function)

This is a Local Numeric Function.

**Syntax**

```
YearToT(year,period)
```

**Description**

This function is used to find the first internal column with a given year in a period.

**Parameters**

- `Year` (numeric expression): a year (e.g. 2009)
- `period` (optional numeric expression, default 0): Value 0 means that FINAN searches in all periods starting with the first period.
- Result: an internal column number.

### 3.1.5 Local Variable Functions

These are functions that are giving a variable.
GetRelVar

#### 3.1.5.1 GetRelVar (var function)

This is a Local Variable Function.

**Syntax**

```
GetRelVar(VarName,NavigationPath)
```

**Parameters**

- `VarName`: name of the variable
- `NavigationPath`: String expression: the relative path starting with the variable in the first parameter. The following characters in the path are definined:
- up
- down
- prev
- next
- first
- last
- close prev (stops at blank line)
- close next (stops at blank line)
- close first (stops at blank line)
- close last (stops at blank line)
- original variable (when this variable is a copy-variable)
- related variable
- ,(comma): when not found, start again here

**Description**

The result of this function is a variable relative to `VarName` following the NaviagionPath. This function is useful in visibility-formulas.

## 3.2 Properties

Data properties

Restriction properties

Event properties

Presentation properties

Tuple specific properties

Model properties

### 3.2.1 Data properties

property datatype, property frequency, property aggregation, property unspecified, property formula, property formula_notrend, property formula_trend,
property flipflop, property flipflop_trend, property flipflop_notrend, property inputRequired,
property data_options, property neptuple_size

#### 3.2.1.1 Datatype

This is a Data property.

**Syntax**

```
EnumItem(none, string, number, currency);
```

**Description**

This is a property within the variable to know what can be written. This is either a string; number; currency.

**Notes**

String; this is a line of text.
Number; Double. This is used for calculations. It is the easier variant and calculates fast.
Currency; BigDecimal. This is used for calculations, most likely used for currency calculations due to the precision of its calculations.

*The advantage of BigDecimal over Double is that the various of numbers are more precise. But it is slower and more difficult to program algorithms.

**Example**

```
datatype : number;
```

### 3.2.1.2 Frequency

This is a Data property.

**Syntax**

```
EnumItem(none, document, timeline, column);
```

**Values**

- default;
- document; variable will be shown in a document
- timeline; variable will be shown in a timeline
- column; variable will be shown in a column

**Description**

This property shows where the variable is shown.

**Example**

```
frequency : document;
```

### 3.2.1.3 Aggregation

This is a Data property.

**Syntax**

```
EnumItem(none, balance, flow, average, index, maximum, minimum);
```

**Values**

- Default. one-column variable, no aggregation.
- balance sheet variable: take last column when aggregated
- flow: flow-variable (add when aggregated), this is the normal code to be used in variables in the Profit and Loss account.
- average, calculate the average when aggregated (e.g. number of personnel, square meters, etc.)
- index: use this code for index variable (e.g. price index)
- max: largest value of aggrated figures (can be useful for ValCheck)
- min: smallest value of aggrated figures

**Description**

This code defines the way variables are aggregated from a lower level of detail (e.g. month, quarter) to a higher level of detail (e.g. year).

**Notes**

- NOTE: to be used at a copy-variable, the original variable should be a balance sheet variable. See alo Mut (numeric function)

**Example**

```
aggregation : flow;


aggregation : balance;
```

### 3.2.1.4 Unspecified

This is a Data property.

**Syntax**

```
EnumItem(default, preferred, previous, never);
```

**Values**

- default; divided by ratio
- preferred; if other columns are empty, this columns is preferred before others.
- previous; compares to previous column.
- never; never use this column

**Description**

The meaning of unspecified means the object is not clear.

**Example**

```
unspecified : previous;
```

### 3.2.1.5 Formula

This is a Data property.

**Syntax**

```
Expression;
```

**Values**

- Column 1 Title
- Column 2 Notrend (History)
- Column 3 Trend (Forecast)
- Column 4 User
- unprotected
- protected
- hidden, underlying level is hidden as well.
- data is hidden, but underlying level can be reached. It is a node.

**Description**

This formula set is for both notrend and trend.

**Example**

```
formula : protected;
```

### 3.2.1.6 Formula_notrend

This is a Data property.

**Syntax**

```
Expression;
```

**Values**

- Also see Formula.

**Description**

NoTrend is one of the Formulaset constants. It is a predefined numeric constants with the value 1.
This formula set is used for historical data.

**Notes**

Also see Formula.

**Example**

```
formula : protected;
```

### 3.2.1.7 Formula_trend

This is a Data property.

**Syntax**

```
Expression;
```

**Values**

- Also see Formula.

**Description**

Trend is one of the Formulaset constants. It is a predefined numeric constants with the value 2. This formula set is used for forecast data.

**Notes**

Also see Formula.

**Example**

```
formula : protected;
```

### 3.2.1.8 Flipflop

This is a Data property.

**Syntax**

```
Expression; //VariableArg
```

**Description**

This property is used to connect variables. If Var1 has input, Var2 will be ignored and vice versa.

**Example**

```
flipflop: turnover; //Var2
              Costs; //Var1
```

### 3.2.1.9 Flipflop_trend

This is a Data property.

**Syntax**

```
Expression; //VariableArg
```

**Description**

This formula set is used for forecast data.

**Notes**

Also see flipflop.

**Example**

```
flipflop_trend: turnover; //Var2
                    Costs; //Var1
```

### 3.2.1.10 Flipflop_notrend

This is a Data property.

**Syntax**

```
Expression; //VariableArg
```

**Description**

This formula set is used for historical data.

**Notes**

Also see flipflop.

**Example**

```
flipflop_notrend: turnover; //Var2
                      Costs; //Var1
```

### 3.2.1.11 InputRequired

This is a Data property.

**Syntax**

```
Expression;
```

**Parameters**

- default situation: no input required.
- 'true': input is required for this variable. This will be checked when going to a higher level or when the datafile is closed. If no input, a error-message will occur

**Description**

This property is used to let the user know if there is input required. True or false.

**Example**

```
inputRequired : true;
```

## 3.2.1.12 !! Data_options

This is a Data property.

**Syntax**

```
EnumList(saveCalculatedValue, hideFirstNoTrendColumn, calculateAggregation,
```

**Parameters**

- saveCalculatedValue;
- hideFirstNoTrendColumn;
- calculateAggregation;
- unscalable;
- generateNepTuples;

**Description**

This property is to give data more options. Can be used as;
`unscalable`; no scale given.
`hideFirstNoTrendColumn`; hides the first no trend column.

**Example**

```
data_options : unscalable;
```

## 3.2.1.13 Neptuple_size

This is a Data property.

**Syntax**

```
Integer;
```

**Description**

Simulates tuple size. Tuple is a code to make exact copies of historical data.

**Example**

```
neptuple_size : Integer;
```

### 3.2.2 Restriction properties

property pattern, property length, property range, property digits, property decimals_maximum

#### 3.2.2.1 Decimals_maximum

This is a restriction property.

**Syntax**

```
Integer;
```

**Description**

This code determines the maximum number of decimal places.

**Example**

```
decimals_maximum : 2.0;
```

#### 3.2.2.2 Digits

This is a restriction property.

**Syntax**

```
Integer;
```

**Description**

This property is used to give acces to put digits before the comma. This is variable from 0-9.

**Example**

```
digits : integer;
```

### 3.2.2.3 Pattern

This is a restriction property.

**Syntax**

```
String;
```

**Description**

This property is used to make patterns. An ObjectName is either a property list pattern or a property value pattern or both.
Can use regular to restrict input.

**Notes**

An ObjectName can be written as a String with the following elements in order:

- The domain.
- A colon (:).
- A key property list as defined below

### 3.2.2.4 Length

This is a restriction property.

**Syntax**

```
Restriction;
```

**Description**

This code determines the length restrictions of a text.

**Example**

```
 length : 10;
```

### 3.2.2.5 Range

This is a restriction property.

**Syntax**

```
Restriction;
```

**Description**

This code determines the digits range. For example the restriction is; > 0 and < 10

**Example**

```
range : >0,<10;
```

### 3.2.3 Event properties

property afterinput, property afterchange

#### 3.2.3.1 Afterinput

This is an event property.

**Syntax**

```
Expression;
```

**Description**

This event is fired after user puts input.

**Example**

```
afterinput : Expression;
```

#### 3.2.3.2 Afterchange

This is an event property.

**Syntax**

```
Expression;
```

**Description**

This event is fired after user puts change.

**Example**

```
afterchange : Expression;
```

## 3.2.4 Presentation properties

property displaytype, property style, property fixed_decimals, property title, property hint, property blanklines, property top_separator, property top_blanklines, property bottom_separator, property bottom_blanklines, property locked, property visible, property choices, property link, property options_title, property options_notrend, property options_trend, property options

### 3.2.4.1 Displaytype

This is a presentation property.

**Syntax**

```
EnumItem(default, procent, currency, date, memo, select, multiselect, checkb
```

**Parameters**

- Default. no special presentation format.
- Procent. Show the variable as a percentage.
- Currency. Show the variable as valuta.
- Date. Display the value of the variable as a date, an optional code in fixed_decimals is the format code. Not possible in combination with a flow variable.
- Memo. Memoscreen. Memos per period are possible.
- Select. Only one selection possible.
- Multiselect. Can make a multiple selection
- Checkbox. A checkbox will be displayed
- Percentage. Show the variable as percentages.
- Choices. A Choice variable, specify the options in the Choices section.

**Description**

This property is used for how this column is shown. This can be currency; memo; e.g.

**Example**

```
displaytype : memo;
```

### 3.2.4.2 Style

This is a presentation property.

**Syntax**

```
String;
```

**Parameters**

- alfanumeric variable (string)

**Description**

This property is to give a style to the variable.

**Example**

```
style : String;
```

### 3.2.4.3 Fixed_decimals

This is a presentation property.

**Syntax**

```
Integer;
```

**Parameters**

- Number of decimals set by user in menu (menu View-Decimals).
- 0..9: Fixed number of decimal places, independent from menu or toolbar settings. This makes the variable also "unscaled" (independend from scale settings). This can be used for ratios, number of personnel, square meters etc.

**Description**

This property is to choose how many decimal places are shown.

**Notes**

NOTE: the default number of decimals is 1 when the variable has a percent format ("procent", "Growth view" or "Integer variable" in displaytype).
NOTE: "(x € 1,-)" will be added in description of variable in grid when variable is currency and not scalable.

**Example**

```
fixed_decimals : 2.0;
```

### 3.2.4.4 Title

This is a presentation property.

**Syntax**

```
Expression;
```

**Description**

Title is one of the Formulaset constants. It is a predefined numeric constants with the value 0. This formula set is used for the title column.

**Example**

```
title : "Column visible";
```

### 3.2.4.5 Hint

This is a presentation property.

**Syntax**

```
String;
```

**Parameters**

- alfanumeric variable (string)

**Description**

The Hints Section starts with the command Hints (cmd).
The section body contains one or more lines. Each line starts with the word `Hint` followed up by`': " text ";'`
See also Variable (cmd) with Hint parameter.

**Example**

```
Hint : "example text for the given hint.";
```

### 3.2.4.6 Blanklines

This is a presentation property.

**Syntax**

```
Expression;
```

**Description**

This property is to show a blank line in between of a text.

**Example**

```
blanklines : 1.0;
```

### 3.2.4.7 Top_separator

This is a presentation property.

**Syntax**

```
Integer;
```

**Parameters**

- No line
- A single line can be drawn above the variable on screen.
- A double line can be drawn above the variable on screen.

**Description**

This property is to show a separator on the top of a text.

**Example**

```
top_separator : 0.0;
```

### 3.2.4.8 Top_blanklines

This is a presentation property.

**Syntax**

```
Integer;
```

**Parameters**

- No empty lines
- '0'..'9': This number of lines will be empty above the variable on the screen.

**Description**

This property is to show a blank line on the top of a text.

**Example**

```
top_blanklines : 1.0;
```

### 3.2.4.9 Bottom_separator

This is a presentation property.

**Syntax**

```
Integer;
```

**Parameters**

- No line
- A single line can be drawn underneath the variable on the screen.
- A double line can be drawn underneath the variable on the screen.

**Description**

This property is to show a separator on the bottom of a text.

**Example**

```
bottom_separator : 0.0;
```

### 3.2.4.10 Bottom_blanklines

This is a presentation property.

**Syntax**

```
Integer;
```

**Parameters**

- No empty lines
- '0'..'9': This number of lines will be empty under the variable on the screen.

**Description**

This property is to show a blank line on the bottom of a text.

**Example**

```
bottom_blanklines : 0.0;
```

### 3.2.4.11 Locked

This is a presentation property.

**Syntax**

```
Expression;
```

**Description**

This property is to have a certain variable blocked for the users input. For example; the total sum of several numbers is pre-calculated. There is no need for changing that number itself.

**Example**

```
locked : On;
```

### 3.2.4.12 Visible

This is a presentation property.

**Syntax**

```
Expression;
```

**Description**

This column determines whether a variable will be visible in the grid.

**Example**

```
visible : 1.0;
```
Variable is visible

```
visible : 0.0;
```
Variable is invisible

### 3.2.4.13 Choices

This is a presentation property.

**Syntax**

```
Expression; //and templates
```

**Description**

This function can be used to select an item in a list.

**Notes**

The specification of possible variaties in a variable can be found in the example given.
Also: Lookup table (fib), TableAsChoices (string function)

**Example**

```
choices : "0:Incomplete.|1:Complete.";
```
User can choose either 0 or 1.

```
choices : "0:Active|1:Final";
```
User can choose either 0 or 1.

```
choices : "dd/MM/yyyy";
```
User has date as input

## 3.2.4.14 Link

This is a presentation property.

**Syntax**

```
Expression; //VariableArg
```

**Description**

This is a property to link to other variables.

**Example**

```
link : Expression; //Var2
link: Turnover; //Var2
```

## 3.2.4.15 Options_title

This is a presentation property.

**Syntax**

```
EnumList(locked, invisible);
```

**Parameters**

- locked; locks the variable title
- invisible; makes the variable title invisible

**Description**

This property is to have listed options in the title.

**Example**

```
options_title : locked;
```

## 3.2.4.16 Options_notrend

This is a presentation property.

**Syntax**

```
EnumList(locked, invisible);
```

**Parameters**

- locked; locks the variable notrend
- invisible; makes the variable notrend invisible

**Description**

This property is to have listed options for historical data.

**Example**

```
options_notrend : locked;);
```

## 3.2.4.17 Options_trend

This is a presentation property.

**Syntax**

```
EnumList(locked, invisible);
```

**Parameters**

- locked; locks the variable trend
- invisible; makes the variable trend invisible

**Description**

This property is to have options listed in forecasted data.

**Example**

```
options_trend : locked;);
```

### 3.2.4.18 Options

This is a presentation property.

**Syntax**

```
EnumList(locked, invisible);
```

**Parameters**

- locked; locks the variable
- invisible; makes the variable invisible

**Description**

This property is to have options listed.

**Example**

```
options : locked;
```

## 3.2.5 Tuple specific properties

property tuple_instance_minimum, property tuple_instance_maximum

### 3.2.5.1 Tuple_instance_minimum

This is a turple specific property.

**Syntax**

```
Integer;
```

**Description**

Tuple is a code to make exact copies of historical data. This property gives more specific data. With Tuple_instance_minimum the number given will return as minimum amount of tuples.

**Example**

```
tuple_instance_minimum : 3;
```
This will return the minimum of 3 tuples

### 3.2.5.2 Tuple_instance_maximum

This is a turple specific property.

**Syntax**

```
Integer;
```

**Description**

Tuple is a code to make exact copies of historical data. This property gives more specific data. With Tuple_instance_maximum the number given will return as maximum amount of tuples.

**Example**

```
tuple_instance_maximum : 3;
```
This will return the maximum of 3 tuples.

## 3.2.6 Model properties

property version

### 3.2.6.1 Version

This is a model property.

**Syntax**

```
Integer;
```

**Description**

This property shows what version the model is.

**Example**

```
version : "00.00.001.108";
```

## 3.3 Constants

CreationDate, ER, False, NA, No, Off, On, PM, True, Yes, Title, NoTrend, Trend, User, Sector, Period constants, Single, Detail.

## 3.3.1 Numeric Constants

### 3.3.1.1 CreationDate (numeric function)

**Description**

This function results in the date the datafile has been created, expressed as a serial number of days to date from 30-12-1899.

### 3.3.1.2 ER (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant ER is an error code. For example a division by zero results in the value ER. This error code is displayed as "ER" on the screen and in reports.

**Notes**

Most numeric functions and operators are aware of this special constant. A formula should return ER where one of its parts evaluates to ER.
See also NA (numeric constant), PM (numeric constant), ER (pascal).

### 3.3.1.3 False (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant False has the value 0 (zero).

**Notes**

See also boolean expression, True (numeric constant), No (numeric constant), Off (numeric constant).

### 3.3.1.4 NA (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant NA ("not available"). This is a code that will be displayed as ".." on screen and in the reports. In formulas it is represented by "NA".

**Notes**

Most numeric functions and operators are aware of this special constant. For example: a formula should return NA when a number is multiplied with NA.
See also ER (numeric constant), PM (numeric constant), NA (pascal).

### 3.3.1.5 No (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant No has the value 0 (zero).

**Notes**

See also boolean expression, Yes (numeric constant), False (numeric constant), Off (numeric constant)

### 3.3.1.6 Off (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant Off has the value 0 (zero).

**Notes**

See also boolean expression, On (numeric constant), No (numeric constant), False (numeric constant).

### 3.3.1.7 On (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant On has the value 1 (one).

**Notes**

See also boolean expression, Off (numeric constant), Yes (numeric constant), True (numeric constant).

### 3.3.1.8 PM (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant PM ("pro memory"). This is a code that will be displayed as "pm" on screen and in the reports. In formulas it is represented by "PM".

**Notes**

Most numeric functions and operators are aware of this special constant. For example: a formula should return PM when two variables with the value PM are added.
See also ER (numeric constant), NA (numeric constant), PM (pascal).

### 3.3.1.9 True (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant True has the value 1 (one).

**Notes**

See also boolean expression, False (numeric constant), Yes (numeric constant), On (numeric constant).

### 3.3.1.10 Yes (numeric constant)

This is a part of the list of Constants.

**Description**

The numeric constant Yes has the value 1 (one).

**Notes**

See also boolean expression, No (numeric constant), True (numeric constant), On (numeric constant).

### 3.3.1.11 Title (formulaset constant)

This is a part of the list of Constants.

**Description**

Title is one of the Formulaset constants. It is a predefined numeric constants with the value 0.
This formula set is used for the title column.

### 3.3.1.12 NoTrend (formulaset constant)

This is a part of the list of Constants.

**Description**

NoTrend is one of the Formulaset constants. It is a predefined numeric constants with the value 1.
This formula set is used for historical data.

### 3.3.1.13 Trend (formulaset constant)

This is a part of the list of Constants.

**Description**

Trend is one of the Formulaset constants. It is a predefined numeric constants with the value 2.
This formula set is used for forecast data.

## 3.3.1.14 User (formulaset constant)

This is a part of the list of Constants.

**Description**

User is one of the Formulaset constants. It is a predefined numeric constants with the value 3. This formula set is often used for budget data or for the "BNK" column ("Balans na kredietverlening").

## 3.3.1.15 Sector (formulaset constant)

This is a part of the list of Constants.

**Description**

Sector is one of the Formulaset constants. It is a predefined numeric constants with the value 4. This formula set is used for benchmark data.

## 3.3.1.16 Period constant

This is a part of the list of Constants.
Period constants are not predefined. De constants are defined in the Period (cmd) in the model. The names of the periods evaluate to numbers in the order of creation of the period.

**Example**

In the V03 model the periods are defined in this order:

- `MainPeriod=1`
- `Forecast2Period=2`
- `Forecast3Period=3`
- `Forecast4Period=4`
- `BudgetPeriod=5`
- `SectorPeriod=6`
- `BNKPeriod=6`

**Notes**

See also Formulaset_constants. (Notrend, Trend, User, Sector)
The formula set constant Title evaluates to 0 (zero), this constant can also be used to indicate period zero (the title column).

#### 3.3.1.17 !! Single (numeric constant)

#### 3.3.1.18 !! Detail (numeric constant)

## 3.4 Operators

> (IsLarger), >= (IsLargerEqual), < (IsSmaller), <= (IsSmallerEqual), = (IsEqual), <> (IsNotEqual), + (Add), - (Subtract), or, * (Multiply), / (Divide), ^ (Power), and, mod, not, implies.

### 3.4.1 Numeric Operators

#### 3.4.1.1 IsLarger (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 > expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

### 3.4.1.2 IsLargerEqual (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 >= expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := ORD(IsLargerOrEqual(CalcIt(CalcNode1),CalcIt(CalcNode2)));
```

### 3.4.1.3 IsSmaller (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 < expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number, value 0 (false) or 1 (true).

**FINAN Windows Delphi source code**

```
Result := ORD(IsSmaller(CalcIt(CalcNode1),CalcIt(CalcNode2)));
```

### 3.4.1.4 IsSmallerEqual (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 <= expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number, value 0 (false) or 1 (true).

**FINAN Windows Delphi source code**

```
Result := ORD(IsSmallerOrEqual(CalcIt(CalcNode1),CalcIt(CalcNode2)));
```

### 3.4.1.5 IsEqual (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 = expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := ORD(IsEqual(CalcIt(CalcNode1),CalcIt(CalcNode2)));
```

### 3.4.1.6 IsNotEqual (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 <> expression2)
```

**Description**

This function performs a logical operation.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := ORD(not IsEqual(CalcIt(CalcNode1),CalcIt(CalcNode2)));
```

### 3.4.1.7 Add (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 + expression2)
```

**Description**

This function performs an addition.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := Add(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

### 3.4.1.8 Subtract (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 - expression2)
```

**Description**

This function performs a subtraction.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := Subtract(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

### 3.4.1.9 Or (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 or expression2)
```

**Description**

This function performs a logical or.

**Parameters**

- `Expression1`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- `Expression2`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- Result: a number, value 0 (false) or 1 (true).

**Notes**

See also And (numeric operator), Not (numeric operator), Implies (numeric operator).

**Example**

```
(5.23 or 0)
```
The result of this expression is 1 (true).

**FINAN Windows Delphi source code**

```
 Result := ORD(IsTrue(CalcIt(CalcNode1)) or IsTrue(CalcIt
(CalcNode2)));
```

## 3.4.1.10 Multiply (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 * expression2)
```

**Description**

This function performs a multiplication.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := Multiply(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.4.1.11 Divide (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 / expression2)
```

**Description**

This function performs a division.

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := Divide(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.4.1.12 Power (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 ^ expression2)
```

**Description**

This function raises expression1 to the power of expression2

**Parameters**

- `Expression1`: this expression evaluates to a number.
- `Expression2`: this expression evaluates to a number.
- Result: a number.

**FINAN Windows Delphi source code**

```
Result := Power(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.4.1.13 And (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 and expression2)
```

**Description**

This function performs a logical and.

**Parameters**

- `Expression1`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- `Expression2`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- Result: a number, value 0 (false) and 1 (true).

**Notes**

See also Or (numeric operator), not (numeric operator).

**Example**

```
(5.23 and 0)
```
The result of this expression is 0 (false).

**FINAN Windows Delphi source code**

```
 Result := ORD(IsTrue(CalcIt(CalcNode1)) and IsTrue(CalcIt
(CalcNode2)));
```

## 3.4.1.14 Mod (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 mod expression2)
```

**Description**

This function performs an integer mod (modulo) operation.

**Parameters**

- `Expression1`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- `Expression2`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- Result: a number, value 0 (false) and 1 (true).

**Example**

```
(13 mod 7)
```
The result of this expression is 6.
```
(13,3 mod 7,1)
```
The result of this expression is 6,2.

**FINAN Windows Delphi source code**

```
 Result := ModValue(CalcIt(CalcNode1),CalcIt(CalcNode2));
```

## 3.4.1.15 Not (numeric operator)

This is a part of the list of operators.

**Syntax**

```
not expression
```

**Description**

This function performs a logical not.

**Parameters**

- Expression: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- Result: a number, value 0 (false) and 1 (true).

**Notes**

See also Or (numeric operator), And (numeric operator), Implies (numeric operator).

**Example**

```
(not 5.23)
```
The result of this expression is 0 (false).

**FINAN Windows Delphi source code**

```
Result := ORD(not IsTrue(CalcIt(CalcNode1)));
```

## 3.4.1.16 Implies (numeric operator)

This is a part of the list of operators.

**Syntax**

```
(expression1 implies expression2)
```

**Description**

This function performs logical implication. Logical implication results in false if the antecedent expression1 if true and the consequence expression2 is false. In all other cases logical implication results in true.

**Parameters**

- `Expression1`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- `Expression2`: this expression evaluates to a number. This number will be converted to 0 (false, when the number equals zero) or 1 (true, when the number does not equal zero).
- Result: a number, value 0 (false) and 1 (true).

**Notes**

See also Or (numeric operator), Not (numeric operator), And (numeric operator).

**Example**

```
(5.23 implies 0)
```
The result of this expression is 0 (false).

**FES Java source code**

```
return (value.boolValue() && !value2.boolValue()) ? FinanMath.FALSE : Finar
```

# Index

## - A -


Abs: 27
ActiveScriptName: 14
ActiveTopVarName: 25
ActiveVarName: 25
Add: 130
AddMonth: 28
Afterchange: 111
Afterinput: 111
AfterStr: 15
Aggregration: 103
And: 133


## - B -


BeforeStr: 15
Blanklines: 114
Bottom_blanklines: 116
Bottom_separator: 116


## - C -


Case: 16, 28
Choices: 117
Chr: 17
ColumnHeader: 26
Constants: 121
ContextVarName: 26
Count: 55
CreationDate: 122
CumNormal: 29


## - D -


Data Properties: 102
Data_options: 108
DataAvailable: 57
DataCalculated: 57
DataEntered: 56
DataEnteredInVar: 58
DataFoundInVar: 59
Datatype: 102
DateStr: 17
DateToDay: 30
DateToMonth: 31
DateToT: 59
DateToYear: 31
DateToYearNum: 60
Decimals_maximum: 109
Definitions: 10
Detail: 127
Digits: 109
DioStatus: 18
Displaytype: 112
Divide: 132
DMYtoDate: 32
DocumentIndex: 32
DocumentIsLocked: 60


## - E -


EnteredValueFoundInT: 61
ER: 122
EvaluateAsString: 18
Event Properties: 111
Exists: 61
Exp: 33
Expand: 62
ExpandFranction: 63
ExpandGrowth: 64
ExpandLevel: 64
ExpandOriginalValue: 65


## - F -


False: 122
FesExpression: 33
FileExists: 34
FindValueT: 66
FirstLC: 19
FirstTinFormulaSet: 66
FirstTinPeriod: 67
FirstTinSheet: 67
FirstTinYear: 68
FirstUC: 19
FirstValidT: 69

Fixed_decimals: 113
Flipflop: 106
Flipflop_notrend: 107
Flipflop_trend: 107
FlowCurrencyFactor: 69
ForAll: 70
Formula: 104
Formula_notrend: 105
Formula_trend: 106
Frequency: 34, 103
Functions: 14
FV: 35

## - G -

GetFrac: 70
GetPoint: 72
GetRelVar: 101
GetT: 73
GetTitle: 26
GetValue: 74
Global Numeric Functions: 27
Global String Functions: 14
GuessTerm: 74

## - H -

HAvg: 76
Hint: 114
HOvr: 76
HSum: 77
HValues: 77
HYearOvr: 78
HYearValues: 78

## - I -

If: 20
Implies: 135
InHiddenTree: 79
Initials: 20
InputRequired: 79, 107
InvNormal: 35
IPMT: 36
IRR: 80
IsEqual: 129
IsLarger: 127
IsLargerEqual: 128
IsNotEqual: 130
IsReadOnly: 75
IsSmaller: 128
IsSmallerEqual: 129
IsValue: 37

## - L -

LastHistYear: 80
LastSheet: 83
LastTinFormulaSet: 81
LastTinPeriod: 82
LastTinSheet: 82
LastTinYear: 83
LastValueT: 84
Length: 37, 110
Licensed: 38
Link: 118
Ln: 38
Local Numeric Functions: 55
Local String Functions: 25
Local Variable Functions: 101
Locked: 117

## - M -

MatrixLookup: 40
Max: 39
MaxT: 84
MaxValueT: 85
MidYearNum: 85
Min: 39
MinMax: 40
MinValueT: 86
Mod: 134
Model Properties: 121
Multiply: 132
Mut: 86
MutCalc: 87

## - N -

NA: 123
Neptuple_size: 108

No: 123
Not: 134
NoTrend: 125
Now: 41
NPER: 42
NPV: 88
NPV2: 88

## - O -

Off: 123
On: 124
OnEntered: 89
OnER: 42
OnERorNA: 43
OnNA: 44
OnNeg: 44
OnNoStr: 21
OnNotEntered: 90
OnNotPos: 45
OnNoValue: 46
OnZero: 46
OnZeroOrNA: 47
Operators: 127
Options: 120
Options_notrend: 119
Options_title: 118
Options_trend: 119
Or: 131

## - P -

Pattern: 110
Period Constant: 126
PeriodInSheet: 90
PeriodInT: 91
PM: 124
PMT: 47
Pos: 48
Power: 133
PPMT: 49
Presentation properties: 112
Properties: 102
PV: 49

## - R -

Range: 110
Rate: 50
Reference: 14
RelMut: 91
Restriction Properties: 109
Round: 51
RoundUp: 51

## - S -

ScaleFactor: 92
Scorecard: 12
Sector: 126
SelectDescendants: 92
Sheet: 92
Single: 127
Str: 21
StrField: 22
Style: 112
SubStr: 22
Subtract: 131
SysVar: 23, 52

## - T -

T: 93
TableAsChoices: 23
TableKeyLookup: 24, 52
TableLookup: 24, 53
ThisMonth: 54
ThisQuarter: 54
ThisYear: 54
TisVisibleInSheet: 93
Title: 114, 125
Top_blanklines: 115
Top_separator: 115
Trend: 125
True: 124
TsPerYear: 94
TsY: 94
Tuple Specific Properties: 120
Tuple_instance_maximum: 121
Tuple_instance_minimum: 120

TupleCount: 95
TupleMax: 95
TupleMin: 96
TupleSum: 96

# - U -

UltCurrencyFactor: 97
UltYearNum: 97
Unspecified: 104
User: 126

# - V -

ValueT: 98
Version: 121
ViewScaleFactor: 98
Visible: 99, 117

# - Y -

YearInT: 99
YearNumToDate: 100
YearToT: 100
Yes: 125

# MOO Documentation